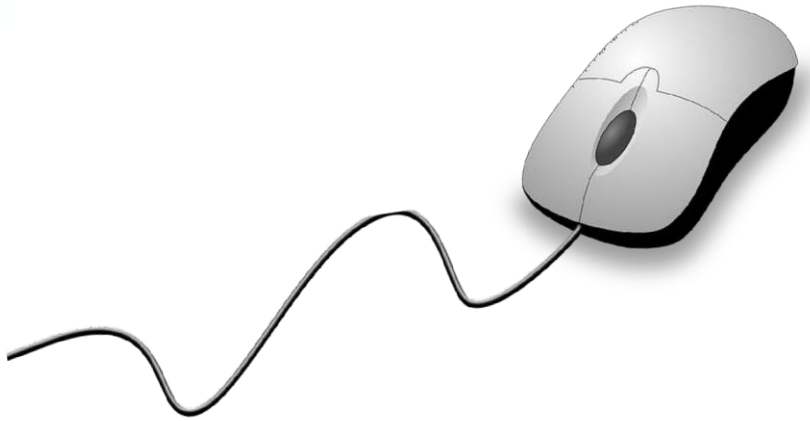
공개SW솔루션설치&활용가이드

미들웨어 > 클라우드 서비스



제대로 배워보자

How to Use Open Source Software

Open Source Software Installation & Application Guide

오픈소스 소프트웨어 통합지원센터

Open Source Software Support Center

CONTENTS

1. 개요

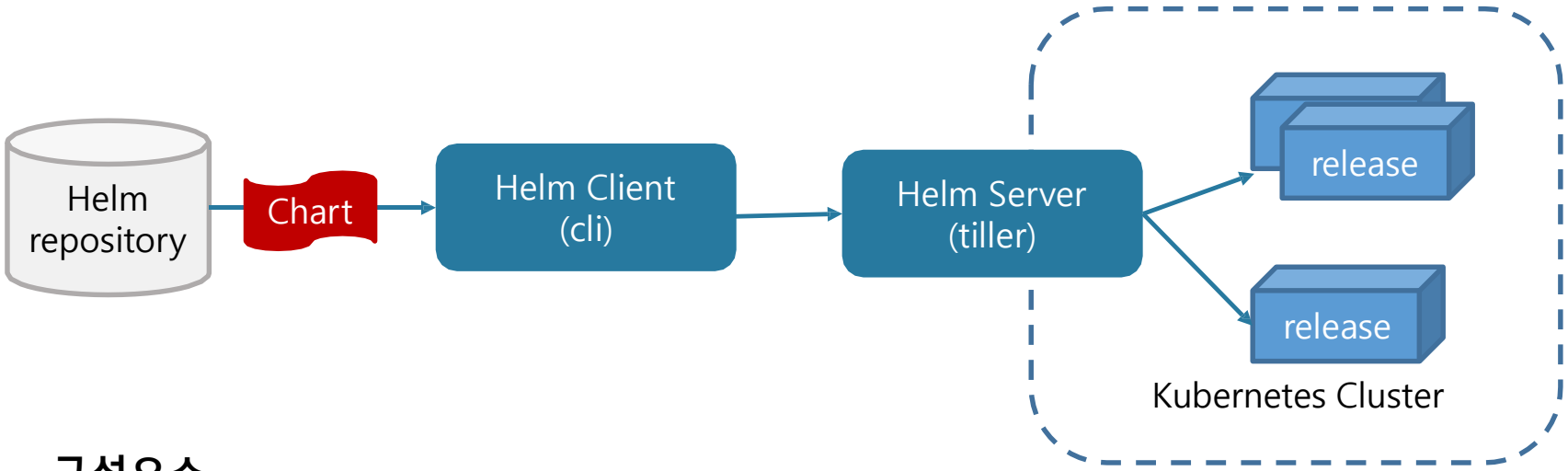
소개	• Kubernetes 환경에서 사용되는 패키지 매니저 •deis라는 회사에서 시작하여 Google에서 진행되던 GCS Deployment Manager 프로젝트가 합해진 프로젝트		
주요기능	• Kubernetes의 응용 프로그램을 정의, 설치 및 업그레이드를 코드로 관리/제공		
대분류	• 미들웨어	**소분류**	• 클라우드 서비스
라이선스형태	• Apache License v2.0	**사전설치 솔루션**	• kubernetes
		버전	• 2.16.1 (2019년 11월 기준)
특징	• 반복 가능한 응용프로그램의 설치 제공 • 어플리케이션의 손쉬운 롤아웃/롤백 기능 제공 • 배포되는 어플리케이션의 릴리즈 관리		
개발회사/커뮤니티	• Cloud Native Computing Foundation		
공식 홈페이지	• https://v2.helm.sh		

2. 기능요약

- helm 주요 기능

복잡한 어플리케이션 배포 관리	• Kubernetes 오케스트레이션된 응용프로그램의 배포는 매우 복잡할 수 있습니다. •Kubernetes 환경에서 helm 차트는 복잡한 응용프로그램의 배포를 코드로 관리 하여 자동으로 배포할 수 있도록 합니다. • 응용프로그램의 빠른 배포를 통하여 다양한 테스트 환경 배포 및 운영 환경 배포 시간을 줄여 개발에 집중 하도록 합니다.
Hooks (응용 프로그램 생명 주기 관리)	•Kubernetes 환경에서 helm 차트로 설치, 업그레이드,삭제 그리고 롤백과 같은 응용프로그램 생명주기의 개입할 수 있는 기능을 Hook을 통하여 제공 합니다.
릴리즈 관리	• Helm으로 배포된 응용프로그램은 하나의 릴리즈로 불립니다. 해당 릴리즈는 배포된 응용프로그램의 버전 관리를 가능하도록 합니다.
재사용성	• 코드로 작성된 helm 차트는 다른환경에서 재사용하여 배포 관리할 수 있습니다.

3. 기본구성



구성요소

- **Helm Chart :** Kubernetes에서 리소스를 만들기 위한 템플릿 화 된 yaml 형식의 파일입니다.
- **Helm (Chart) Repository :** Helm Repository는 해당 리포지토리에 있는 모든 차트의 모든 메타데이터를 포함하는 저장소 입니다. 상황에 따라서, Public Repository를 사용 하거나 내부에 Private Repository를 구성할 수 있습니다.
- **Helm Client(cli) :** 외부의 저장소에서 Chart를 가져 오거나, gRPC로 Helm Server 와 통신하여 요청을 하는 역할을 합니다.
- **Helm Server(tiller) :** Helm Client의 요청을 처리하기 위하여 대기하며, 요청이 있을 경우 Kuberernetes에 Chart를 설치하고 릴리즈를 관리 합니다.

4. 실행환경

1. **OS**
   - CentOS Linux release 7.7.1908 (Core) 환경 (총 3대)

2. **사전 설치 솔루션**
   - Kubernetes v1.15.3

3. **Helm package**
   - helm-v2.16.1-linux-amd64

5. 설치 및 실행

5-1. Client 설치

- Helm 바이너리 파일을 다운로드 하여 등록되어 있는 PATH에 이동하여 실행 합니다.
- Client 버전은 다운로드 받은 버전으로 출력되는 것을 확인할 수 있습니다.
- 아직 Helm Server가 없기 때문에 아래와 같은 에러를 확인할 수 있습니다.

```
$ wget https://get.helm.sh/helm-v2.16.1-linux-amd64.tar.gz
$ tar xvzf helm-v2.16.1-linux-amd64.tar.gz
$ mv linux-amd64/helm  /usr/local/bin/helm
$ helm version
Client: &version.Version{SemVer:"v2.16.1", GitCommit:"bbdfe5e7803a12bbdf97e94cd847859890cf4050", GitTreeState:"clean"}
Error: could not find tiller
```

5. 설치 및 실행

5-2. Helm Server(Tiller) 설치

- Helm Server인 Tiller는 일반적으로 Kubernetes 클러스터 내부에서 실행 됩니다.
- 그렇기 때문에 설치하기 위해서는 Helm Server를 위한 Role을 설정 해야 합니다. RBAC기능을 사용하여 Role을 설정하는 경우, 올바른 역할과 권한을 사용하여 Tiller에 대한 서비스 계정을 생성하여 리소스에 액세스 해야 합니다.
- 아래와 같이 RBAC 기능을 위한 매니패스트 파일을 작성합니다.

```
$ cat rbac-config.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: tiller
    namespace: kube-system
```

5. 설치 및 실행

5-2. Helm Server(Tiller) 설치

- 아래와 같이 RBAC 기능을 위한 작성한 매니패스트 파일을 Kubernetes 클러스터에 적용합니다..

```
$ kubectl create -f rbac-config.yaml  serviceaccount/ti
ller created  clusterrolebinding.rbac.authorization.k8s.io/
tiller created
```

- Kubernetes 에서 생성된 serviceaccount와 clusterrolebinding을 확인할 수 있습니다.

```
$ kubectl get serviceaccount -n kube-system
...
tiller                               1         43s
...

$ kubectl get clusterrolebinding
...
tiller                                                 2m23s
...
```

5. 설치 및 실행

5-2. Helm Server(Tiller) 설치

- 앞에서 설정한 tiller service account 를 이용하여 helm을 초기화 합니다.
- 초기화 과정에서 kubernetes에 tiller가 배포 됩니다.
- **--history-max**를 지정하지 않으면 history 제한없이 증가 합니다.

```
$ helm init --service-account tiller --history-max 200
Creating /root/.helm
Creating /root/.helm/repository  Creat
ing /root/.helm/repository/cache  Cre
ating /root/.helm/repository/local  Cre
ating /root/.helm/plugins
Creating /root/.helm/starters  Crea
ting /root/.helm/cache/archive
Creating /root/.helm/repository/repositories.yaml
Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /root/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users' policy.
To prevent this, run `helm init` with the --tiller-tls-verify flag.
For more information on securing your installation see: https://docs.helm.sh/using_helm/#securing-your-helm-installation
```

5. 설치 및 실행

5-3. Helm Repository관리

- *"helm repo "* 명령을 사용하여 Helm Repository를 관리할 수 있습니다.
- 기본적으로 아래와 같이 stable repository와 client의 local repository를 확인할 수 있습니다.

```
$ helm repo list
NAME            URL
stable          https://kubernetes-charts.storage.googleapis.com
local           http://127.0.0.1:8879/charts
```

- *"helm repo update"* 명령을 사용하여 외부 Helm Repository에 있는 정보와 동기화 합니다.

```
$     helm repo update
Hang tight while we grab the latest from your chart repositories...
...Skip local chart repository
...Successfully got an update from the "stable" chart repository
Update Complete.
```

5. 설치 및 실행

5-3. Helm Repository관리

- *"helm repo add [REPOSITORY NAME] [REPOSITORY HOST]"* 명령을 사용하여 Helm Repository를 추가할 수 있습니다.

```
$ helm repo add dev https://example.com/dev-charts

$ helm repo list
NAME            URL
stable          https://kubernetes-charts.storage.googleapis.com
local           http://127.0.0.1:8879/charts
dev             https://example.com/dev-charts
```

6. 기능 소개

6-1. Helm Chart를 이용한 응용프로그램 배포

- "helm search" 명령을 이용하여 Helm Repository를 통하여 배포 가능한 Helm Chart를 검색할 수 있습니다.

```
$ helm search mariadb
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/mariadb         7.0.1           10.3.20         Fast, reliable, scalable, and easy to use open-source rel...
bitnami/mariadb-cluster 1.0.1           10.2.14         Chart to create a Highly available MariaDB cluster
bitnami/mariadb-galera  0.5.2           10.3.20         MariaDB Galera is a multi-master database cluster solutio...
stable/mariadb          7.0.1           10.3.20         Fast, reliable, scalable, and easy to use open-source rel...
...
```

6. 기능 소개

6-1. Helm Chart를 이용한 응용프로그램 배포

- *"helm install [CHART NAME]"* 명령을 이용하여 Helm Repository를 통하여 Helm Chart를 배포합니다. 배포가 되면 간단한 설명과 배포된 리소스들을 확인할 수 있습니다.

```
$     helm install stable/mariadb
NAME:  wishing-garfish
LAST DEPLOYED: Tue Nov 19 17:17:21  2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/ConfigMap
...
To connect to your database:

  1. Run a pod that you can use as a client:

      kubectl run wishing-garfish-mariadb-client --rm --tty -i --restart='Never' --image docker.io/bitnami/mariadb:10.3.20-debian-9-r0 --
namespace default --command -- bash

  2. To connect to master service (read/write):

      mysql -h wishing-garfish-mariadb.default.svc.cluster.local -uroot -p my_database

  3. To connect to slave service (read-only):

      mysql -h wishing-garfish-mariadb-slave.default.svc.cluster.local -uroot -p my_database

To upgrade this helm chart:

  1. Obtain the password as described on the 'Administrator credentials' section and set the 'rootUser.password' parameter as shown  below:

      ROOT_PASSWORD=$(kubectl get secret --namespace default wishing-garfish-mariadb -o jsonpath="{.data.mariadb-root-password}" |
base64 --decode)
      helm upgrade wishing-garfish stable/mariadb --set rootUser.password=$ROOT_PASSWORD
```

6. 기능 소개

6-1. Helm Chart를 이용한 응용프로그램 배포

- *"helm ls"*명령을 이용하여 배포된 Release를 확인할 수 있습니다.

```
$ helm ls
NAME            REVISION    UPDATED                     STATUS      CHART          APP VERSION  NAMESPACE
wishing-garfish 1           Tue Nov 19 17:17:21 2019    DEPLOYED    mariadb-7.0.1  10.3.20       default√
```

- Helm으로 배포된 응용프로그램의 Pod를 Kubernetes에서 확인할 수 있습니다.
- 이때, Helm의 이름은 지정하지 않으면 자동 생성되며, 생성된 Helm의 이름을 기반으로 Pod의 이름이 자동 생성 됩니다.

```
$ kubectl get pod
NAME                                READY  STATUS            RESTARTS  AGE
wishing-garfish-mariadb-master-0    1/1    Running           0         85s
wishing-garfish-mariadb-slave-0     0/1    Running           0         86s
```

- *"helm delete [HELM RELEASE NAME] "* 명령을 사용하여 배포된 응용프로그램을 삭제 합니다.

```
$ helm delete  wishing-garfish
release "wishing-garfish" deleted
```

6. 기능 소개

6-2. Helm Chart생성하기

- Helm Chart를 직접 생성하여 배포하려는 응용프로그램의 릴리즈 관리를 할 수 있습니다.
- *"helm create"* 명령어를 사용하여 custom한 helm chart를 생성할 수 있다. 아래와 같은 트리구조로 sample chart가 생성 됩니다.

```
$ helm create test-chart
Creating test-chart
$ tree test-chart
cy-test/
├── charts
├── Chart.yaml
├── templates
│   ├── deployment.yaml
│   ├── _helpers.tpl
│   ├── ingress.yaml
│   ├── NOTES.txt
│   ├── serviceaccount.yaml
│   ├── service.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```

6. 기능 소개

6-2. Helm Chart생성하기

- 생성하면 위와 같은 트리 구조로 파일이 생성되며, 각 파일과 디렉토리의 역할은 아래와 같습니다.
  - **charts/** : 해당 디렉토리에 종속성을 가지고 있는 helm chart를 저장 합니다. 만약에 웹 서비스를 실행하는 helm chart에서 설치 시 mysql helm chartr가 필요하다면 별도의 dependency설정을 진행하고 해당디렉토리의 helm chart를 호출 하게 됩니다.
  - **templates/** : 실제 배포에 필요한 yaml 파일이 저장되어 있다. 각 yaml 파일은 템플릿화 되어 지정한 변수에 따라서 release를 생성할수 있도록 재사용성을 제공 하고 있습니다.
  - **deployment.yaml** : kubernetes deployment 형태로 배포되기 위해 사용 되는 yaml 파일
  - **ingress.yaml**: kubernetes ingress 형태로 배포되기 위해 사용 되는 yaml파일
  - **service.yaml**: kubernetes service 형태로 배포되기 위해 사용 되는 yaml파일
  - **NOTES.txt**: 배포 후 사용자에게 제공되는 사용법이나, 구조 등이 설명되어 있는 txt파일로 대부분 서버스 접속 방법이나, 로그인 정보등을 추가 합니다.
  - **values.yaml** : 템플릿화 되어있는 chart의 변수(기본값)를 정의 합니다.
  - **Chart.yaml** : Chart에 대한 정보가 포함되어 있는 yaml파일
  - **README.md** : 사람이 읽을 수 있는 README 파일

6. 기능 소개

6-2. Helm Chart생성하기

- **Chart.yaml**파일은 해당 Helm Chart의 기본적인 정보와 이름 버전등을 기록하고 있다. 명시된 version은 helm repository에서 버전별로 관리 되고 명시 됩니다.
- version의 형식은 Semantic Versioning 2.0.0(https://semver.org) 의 형식을 따르고 있습니다.

```
apiVersion: v1   a
ppVersion: "1.0"
description: A Helm chart for Kubernetes
name: test-chart
version: 0.1.0
```

- **value.yaml**에는 템플릿화된 templates/ 디렉토리 하위의 yaml파일들에 대하여 변수를 정의 합니다.
- 만약 valeue.yaml파일에 replicaCount 이라는 변수로 선언이 되어 있다면 temlates/ 하위 yaml 파일에서 "{{ .Values.replicaCount }}" 형태로 호출하여 사용할 수 있습니다.

```
...
replicaCount: 1

image:
  repository: nginx
  tag: stable
  pullPolicy: IfNotPresent

imagePullSecrets: []
...
```

6. 기능 소개

6-2. Helm Chart생성하기

- 또한, 계층적 구조를 가진 변수에 대하여 계층적으로 호출할 수 있다. 아래와 같이 선언된 변수에 대하여 호출할 때는 *"{{ .Values.image.repository }}"* 형식으로 호출할 수 있습니다.

```
...
image:
  repository: nginx
...
```

- **deployment.yaml**파일에는 실제 kubernetes에 deployment로 배포되는 yaml파일들이 정의 되어 있습니다.
- 이때, 위에서 설정한 values.yaml파일에 정의된 변수들을 호출하여 재사용성 가능하도록 템플릿화 되어 있습니다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "test-chart.fullname" . }}
  labels:
{{ include "test-chart.labels" . | indent 4 }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app.kubernetes.io/name: {{ include "test-chart.name" . }}
      app.kubernetes.io/instance: {{ .Release.Name }}
  template:
    metadata:
...
```

6. 기능 소개

6-2. Helm Chart생성하기

- 이제 생성된 Chart 를 test-chart-release이름으로 설치 합니다.

```
$ helm install ./test-chart --name test-chart-release
NAME:  test-chart-release
LAST DEPLOYED: Wed Nov 20 11:33:02  2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Deployment
NAME                READY UP-TO-DATE AVAILABLE  AGE
test-chart-release  0/1     1            0           <invalid>

==> v1/Pod(related)
NAME                                READY  STATUS             RESTARTS  AGE
test-chart-release-6d87f576d4-swgjm  0/1     ContainerCreating  0         <invalid>

==> v1/Service
NAME                TYPE        CLUSTER-IP EXTERNAL-IP PORT(S)  AGE
test-chart-release  ClusterIP   10.233.4.50   <none>        80/TCP <invalid>

==> v1/ServiceAccount
NAME                SECRETS  AGE
test-chart-release  1        <invalid>


NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=test-chart,app.kubernetes.io/instance=test-chart-
release" -o jsonpath="{.items[0].metadata.name}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl port-forward $POD_NAME 8080:80 ...
```

6. 기능 소개

6-2. Helm Chart생성하기

- *"helm ls"* 명령을 사용하여 배포된 Release정보를 확인 합니다. 배포에 사용된 Chart의 버전과 이름 그리고 배포된 Revision을 확인할 수 있습니다.

```
$ helm ls test-chart-release
NAME                REVISION    UPDATED                     STATUS      CHART               APP VERSION NAMESPACE
test-chart-release 1           Wed Nov 20 11:33:02 2019    DEPLOYED    test-chart-0.1.0    1.0         default
Helm
```

6. 기능 소개

6-3. Helm Release Upgrade / Rollback

- 배포된 Release의 변경사항이 있거나, 문제가 생긴 버전에 대하여 이전 버전 혹은 지정한 버전으로 돌릴 수 있습니다.
- 기본적으로 배포된 pod는 1개로 values.yaml에서 replicaCount가 1로 정의 되어 있기 때문에 1개의 Pod를 확인할 수 있습니다.

```
$ kubectl get pod | grep test-chart-release
test-chart-release-6d87f576d4-8x4gc          1/1     Running  0          4m46s
```

- values.yaml 파일을 수정 하여 replicaCount를 2로 수정하여 다시 배포시 2개의 Pod가 실행할 수 있도록 합니다.

```
...
replicaCount: 2
...
```

6. 기능 소개

6-3. Helm Release Upgrade / Rollback

- Chart.yaml 을 수정 하여 Chart의 버전을 0.1.0에서 0.2.0으로 변경합니다.

```
apiVersion: v1   a
ppVersion: "1.0"
description: A Helm chart for Kubernetes
name: test-chart
version: 0.2.0
```

- 이제 변경 Chart를 가지고 upgrade를 진행 하여 본다. 의도한 것 처럼 Pod의 개수가 증가된 것을 확인할 수 있습니다.

```
$ helm upgrade test-chart-release ./test-chart
$ kubectl get pod | grep test-chart-release
test-chart-release-6d87f576d4-8x4gc          1/1     Running   0          56s
test-chart-release-6d87f576d4-b9sq4          1/1     Running   0          29s
```

6. 기능 소개

6-3. Helm Release Upgrade / Rollback

- 그리고 "helm ls" 명령을 사용하면 변경된 Chart의 버전과 Revision이 증가된것을 볼수 있다. "helm history " 명령을 사용하면 이전 Revision 번호에 따른 정보를 확인할 수 있습니다.

```
$ helm ls test-chart-release
NAME                REVISION    UPDATED                     STATUS      CHART               APP VERSION NAMESPACE
test-chart-release  2           Sat Nov 23 15:15:18 2019    DEPLOYED    test-chart-0.2.0    1.0         default

$ helm history test-chart-release
REVISION    UPDATED                     STATUS      CHART               APP VERSION DESCRIPTION
1           Sat Nov 23 15:14:50 2019    SUPERSEDED  test-chart-0.1.0    1.0         Install complete
2           Sat Nov 23 15:15:18 2019    DEPLOYED    test-chart-0.2.0    1.0         Upgrade complete
```

- 만약, 새로 배포된 Release에 문제가 생겨서 이전버전으로 되돌아 간다면 rollback 명령을 실행하여 돌아갈 수 있습니 다. 최초 배포 Revision인 1로 돌아갔기 때문에 Pod의 개수가 1개로 줄게 됩니다.

```
$ helm rollback test-chart-release 1
Rollback was a success.
$ kubectl get pod | grep test-chart-release
test-chart-release-6d87f576d4-8x4gc          1/1     Running  0          5m3s
$ helm ls test-chart-release
NAME                REVISION    UPDATED                     STATUS      CHART               APP VERSION NAMESPACE
test-chart-release  3           Sat Nov 23 15:19:21 2019    DEPLOYED    test-chart-0.1.0    1.0         default
$ helm history test-chart-release
REVISION    UPDATED                     STATUS      CHART               APP VERSION DESCRIPTION
1           Sat Nov 23 15:14:50 2019    SUPERSEDED  test-chart-0.1.0    1.0         Install complete
2           Sat Nov 23 15:15:18 2019    SUPERSEDED  test-chart-0.2.0    1.0         Upgrade complete
3           Sat Nov 23 15:19:21 2019    DEPLOYED    test-chart-0.1.0    1.0         Rollback to 1
```

7. FAQ

Q Helm으로 설치할때 특정 시점에 명령어를 사용 할 수 있나요 ?

A

metadata.annotations."helm.sh/hook" 태그를 추가 하여 해당 리소스를 인스톨 되는 도중 특정 시점에 명령을 수행할 수 있습니다. 상황에 따라서 인스톨 전 후 등을 지정하여 수행할 수 있습니다.

Q Private Helm Repository를 제공하나요 ?

A

Chartmuseum 이나 Nexus3 (OSS) 으로 Private Helm Repository를 구성할 수 있습니다. 앞에 이야기 드린 방법은 폐쇠망에서 설치가 가능한 솔루션입니다. 인터넷이 안되는 구성에서도 Helm Chart를 Repository를 통하여 관리할 수 있습니다.

Open Source Software Installation & Application Guide