

Helm: k8s package manager

Kubernetes study week #7

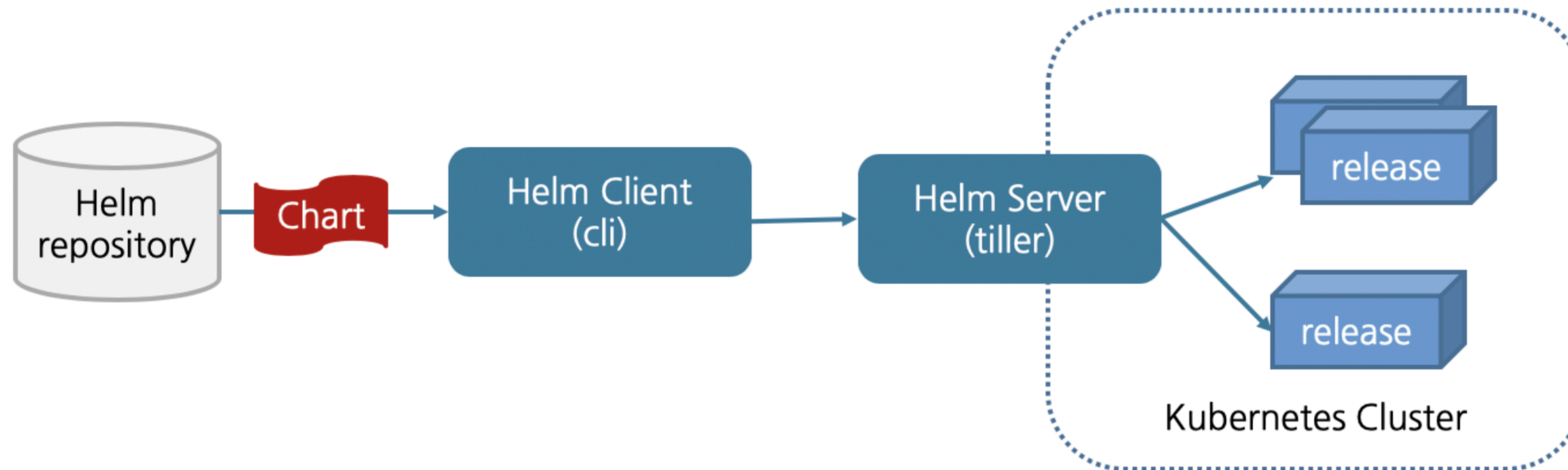
Helm

- helm: 쿠버네티스의 패키지 매니저
- k8s 애플리케이션을 배포할 때, 어떻게 배포할까?
 - > helm을 이용하면 pod, service, deployment 등 애플리케이션에서 필요한 모든 것을 **패키지 형태로 묶어 배포할 수 있다.**



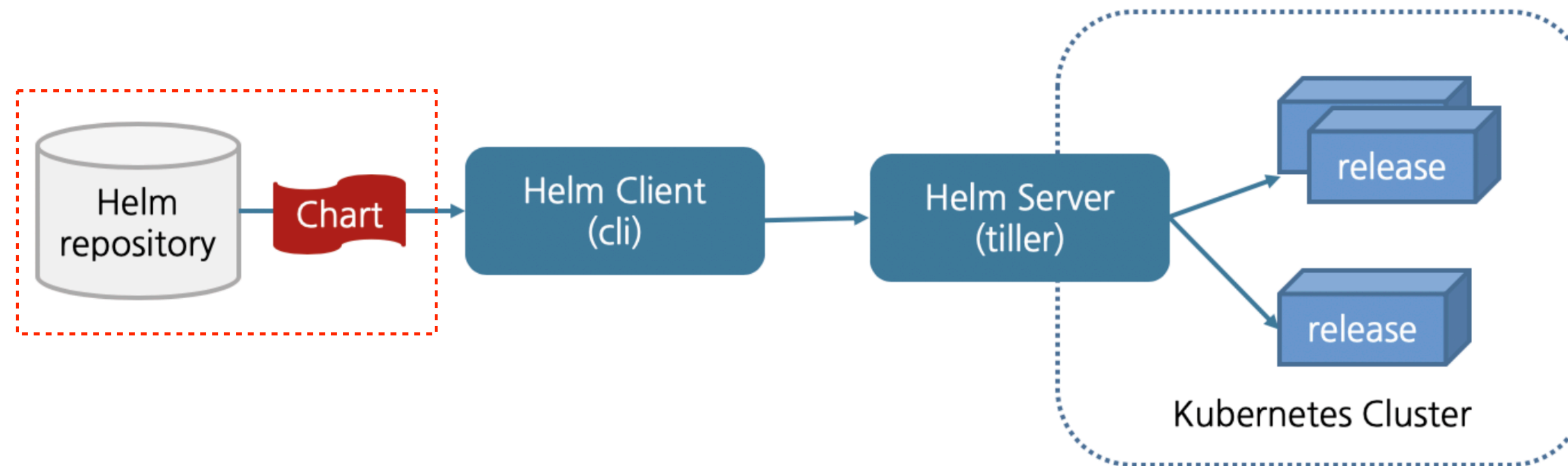
kubernetes

Helm 기본 구성



- Helm Chart: Kubernetes에서 리소스를 만들기 위해 템플릿화 된 yaml 형식의 파일
- Helm (Chart) Repository: 해당 repo.에 있는 모든 차트의 모든 metadata를 포함하는 저장소.
상황에 따라서, Public Repository를 사용 하거나 내부에 Private Repository를 구성할 수 있다.
- Helm Client (cli): 외부의 저장소에서 Chart를 가져 오거나, gRPC로 Helm Server 와 통신하여 요청을 하는 역할
- Helm Server (tiller): Helm Client의 요청을 처리하기 위하여 대기하며,
요청이 있을 경우 Kubernetes에 Chart를 설치하고 릴리즈를 관리한다.

Create helm chart



- **Chart.yaml:** chart의 이름, 버전, 등 chart에 대한 정보가 포함되어 있는 yaml 파일
- **charts/:** 해당 디렉토리에 종속성을 가지는 helm chart를 저장한다. 기본적으로는 비어 있다.
- **templates/:** chart의 뼈대가 되는 쿠버네티스 리소스들이 들어있는 폴더
 - **NOTES.txt:** 배포 후 사용자에게 제공되는 사용법이나, 구조등이 설명되어있는 txt 파일
 - **deployment.yaml:** kubernetes deployment 형태로 배포되기 위해 사용되는 yaml 파일
 - **ingress.yaml:** kubernetes ingress 형태로 배포되기 위해 사용되는 yaml 파일
 - **service.yaml:** kubernetes service 형태로 배포되기 위해 사용되는 yaml 파일
- **values.yaml:** 사용자가 정의하는 설정값을 가진 yaml 파일

```
$ helm create mychart
```

```
$ tree mychart
```

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

Chart.yaml / value.yaml

```
$ helm create mychart
```

```
$ tree mychart
```

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

- **Chart.yaml**: chart의 이름, 버전, 등 chart에 대한 정보가 포함되어 있는 yaml 파일
- **charts/**: 해당 디렉토리에 종속성을 가지는 helm chart를 저장한다. 기본적으로는 비어 있다.
- **templates/**: chart의 뼈대가 되는 쿠버네티스 리소스들이 들어있는 폴더
 - **NOTES.txt**: 차트의 “도움말”. Install 시 사용자에게 보여진다.
 - **deployment.yaml**: kubernetes deployment 형태로 배포되기 위해 사용되는 yaml 파일
 - **ingress.yaml**: kubernetes ingress 형태로 배포되기 위해 사용되는 yaml 파일
 - **service.yaml**: kubernetes service 형태로 배포되기 위해 사용되는 yaml 파일
- **values.yaml**: 사용자가 정의하는 설정값을 가진 yaml 파일

```
user@master:~/mychart$ cat Chart.yaml
```

```
apiVersion: v2
name: mychart #helm 차트의 이름
description: A Helm chart for Kubernetes
type: application
version: 0.1.0 #helm 차트의 버전
appVersion: 1.16.0 #helm 차트로 배포되는 app의 버전
```

```
user@master:~/mychart$ cat values.yaml
```

```
replicaCount: 1
image:
  repository: nginx
  pullPolicy: IfNotPresent
  tag: ""
imagePullSecrets: []
nameOverride: ""
fullnameOverride: ""
...
```

Templates 내 리소스 파일

```
$ helm create mychart
```

```
$ tree mychart
```

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

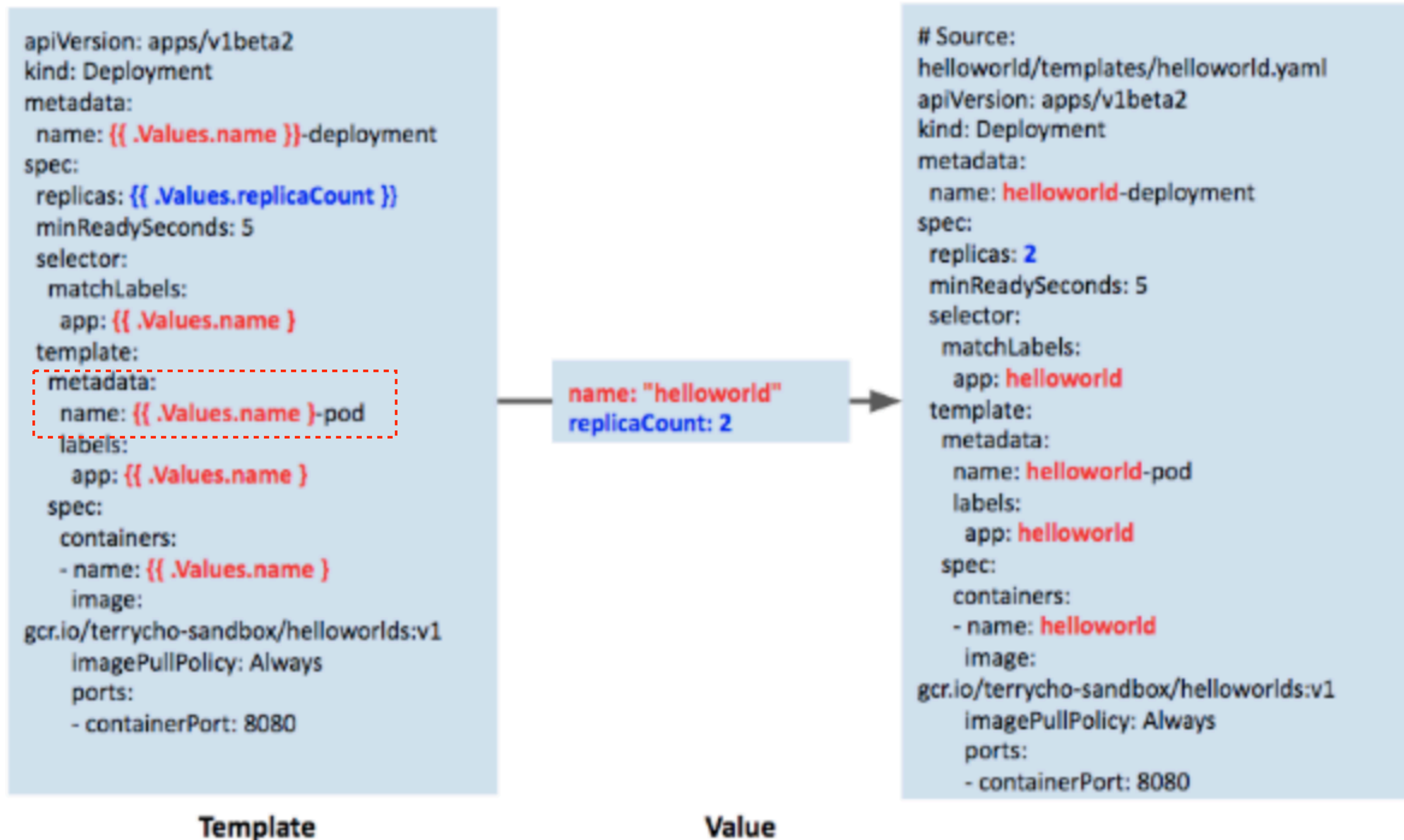
```
user@master:~/mychart/templates$ cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ include "mychart.fullname" . }}
  labels:
    {{- include "mychart.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: http
      protocol: TCP
      name: http
  selector:
    {{- include "mychart.selectorLabels" . | nindent 4 }}
```

```
user@master:~/mychart/templates$ cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "mychart.fullname" . }}
  labels:
    {{- include "mychart.labels" . | nindent 4 }}
spec:
  {{- if not .Values.autoscaling.enabled }}
  replicas: {{ .Values.replicaCount }}
  {{- end }}
  selector:
    matchLabels:
      {{- include "mychart.selectorLabels" . | nindent 6 }}
  ...
```

“mychart.fullname” 는 어디있죠?!

-> helpers.tpl 에 저장!

Templates 과 Value



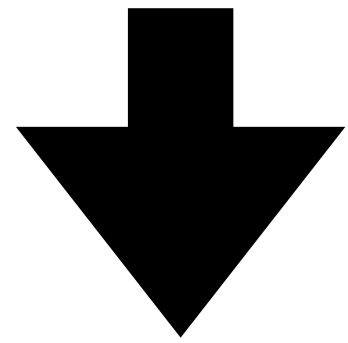
같은 chart 로 두 개 이상의 App. 을
생성하고 싶을 때 문제는? (ex. mySQL)

->

동일한 Value.yaml 파일을 사용해서
Chart 생성 시 metadata 내 name이
같다는 문제가 발생합니다.

name 에는 Release를 사용하자

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: {{ .Values.name }}-deployment
```



```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: {{ .Release.Name }}
```

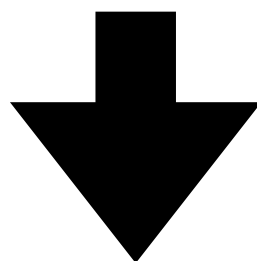
**Name 은 value.yaml 보다는
Release.name 을 참고하도록 하는 것을 권장한다!**

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: {{ .Chart.Name }}-{{ .Release.Name }}
```


Install helm chart

```
user@master:~/mychart$ helm install foo .

NAME: foo
LAST DEPLOYED: Sun Jul 11 08:25:51 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these
commands:
...
```



```
user@master:~/mychart$ helm list
```

NAME	NAMESPACE	REVISION	UPDATED	STATUS	CHART	APP VERSION
foo	default	1	2021-07-11 08:25:51.954827853 +0000 UTC	deployed	mychart-0.1.0	1.16.0

```
user@master:~$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
foo-mychart-54c7666456-rwxvj	1/1	Running	0	34m

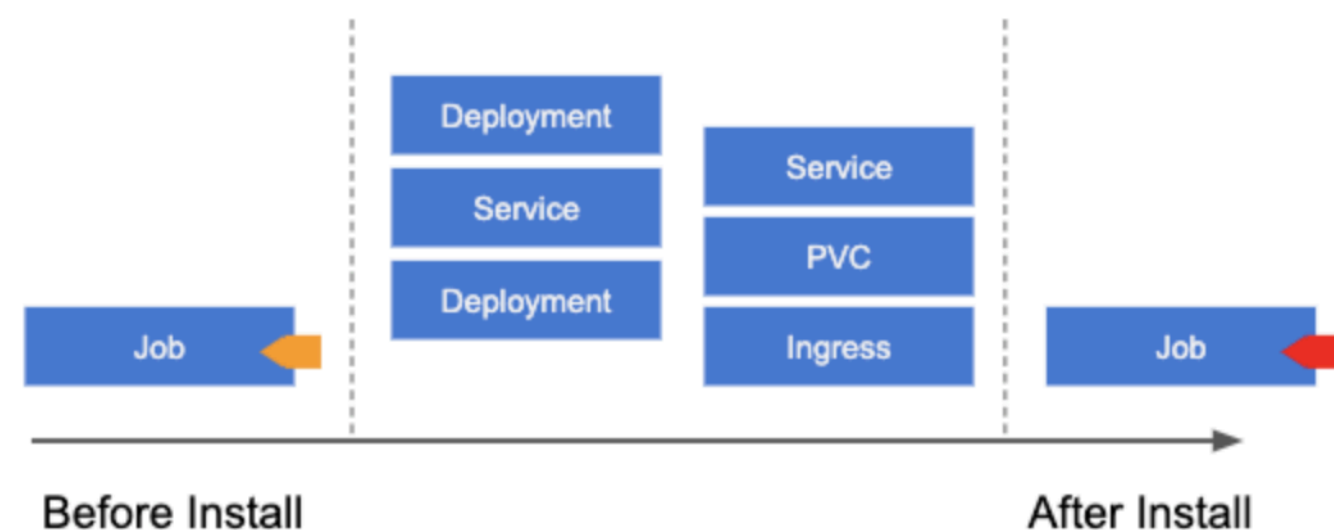
```
user@master:~$ kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
foo-mychart	NodePort	10.43.71.125	<none>	8888:32179/TCP	36m

Helm hook

- Hook은 차트의 라이프 사이클 중 동작의 순서 등을 지정해줄 수 있도록 해주는 기능이다
- MySQL을 차트로 설치한 후에 MySQL에 테이블을 생성하고 데이터를 로딩하거나, 차트로 Pod를 설치하기전에 Configmap이나 Secret 의 값을 세팅해놓는 것과 같은 작업을 예를 들 수 있다.
- 리소스를 정의한 yaml 파일에서 metadata.annotations 에 “helm.sh/hook” 를 지정해주면 된다.

```
apiVersion: ...  
kind: ....  
metadata:  
  annotations:  
    "helm.sh/hook": "pre-install"
```



- pre-install: 리소스를 설치하기전에 실행된다. (정확히 이야기 하면 리소스를 설치하기 위한 템플릿이 렌더링 된 후에, 렌더링된 템플릿으로 리소스를 설치하기 전에 실행된다.)
- post-install: Helm 차트에 의해서 리소스들이 모두 설치 된 후에 실행된다.
- pre-delete: 릴리즈의 리소스를 삭제할때, 삭제 전에 실행된다.
- post-delete: 릴리즈의 리소스를 삭제한 후에, 실행된다.
- pre-upgrade: 릴리즈를 업그레이드 하기 전, (템플릿이 렌더링 된 후에) 리소스가 생성되기 바로 전에 실행된다.
- post-upgrade: 릴리즈 업그레이드가 끝난 후에, 실행된다.
- pre-rollback: 릴리즈를 기존 버전으로 롤백 할때 실행된다. (템플릿이 렌더링 된 후에)
- post-rollback: 릴리즈에 대한 롤백이 완료된 후에 실행된다.
- crd-install: CRD 리소스를 인스톨하는데, 다른 Hook이나 기타 모든 다른 태스크 보다 우선적으로 실행된다. 이 Hook 은 다른 리소스에서 이 CRD를 참조하여사용할때 주로 사용된다.
- test-success: “helm test” 명령을 실행할때 수행되는데, 그 결과가 성공 (return code == 0)일때만 실행된다.
- test-failure: “helm test” 명령을 실행할때 수행되는데, 그 결과가 실패 (return code != 0)일때만 실행된다.

- helm.sh/hook-weight: n -> 동일한 hook 이 여러개일 때 n 이 작을수록 먼저 실행된다. (n은 음수, 양수 모두 가능)
- helm.sh/hook-delete-policy: hook-succeeded -> hook이 실행된 후에 리소스 삭제

Helm chart 조회 / 렌더링 / 업그레이드

- **Helm 차트 조회**

- Helm list

- **Helm chart 렌더링**

- Helm template foo ./mychart > foo.output.yaml

- **Helm chart upgrade**

- Helm upgrade foo ./mychart

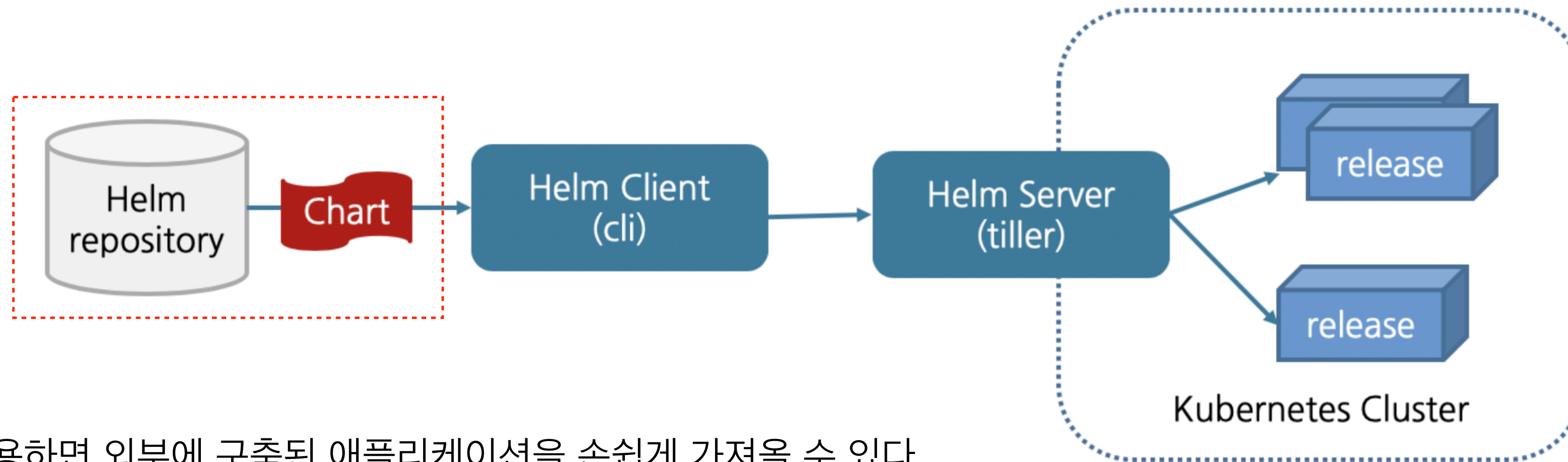
```
user@master:~/mychart$ helm upgrade foo .  
Release "foo" has been upgraded. Happy Helming!  
NAME: foo  
LAST DEPLOYED: Sun Jul 11 09:27:43 2021  
NAMESPACE: default  
STATUS: deployed  
REVISION: 2
```

- **Helm rollback**

- Helm rollback foo 1

```
user@master:~/mychart$ helm rollback foo 1  
Rollback was a success! Happy Helming!
```

원격 Repository



- helm을 사용하면 외부에 구축된 애플리케이션을 손쉽게 가져올 수 있다.
- Helm repository: 여러 chart를 한 곳에 묶어서 보관해놓은 저장소
- **원격 repository 추가**
 - Helm repo add stable <https://kubernetes-charts.storage.googleapis.com>
 - Helm repo update

- **Helm repo list**

```
user@master:~/mychart$ helm repo list
NAME      URL
stable    https://charts.helm.sh/stable
common    https://charts.helm.sh/incubator
```

외부 Chart 설치

- 외부 **Chart** 바로 설치

- Helm install wp stable/wordpress —version 9.0.3 —set service.port=8080 —namespace default

- 외부 **chart**를 로컬 디렉토리로 다운로드해서 설치

- Helm fetch —untar stable/wordpress —version 9.0.3
- Vim wordpress/values.yaml
- Helm install wp-fetch ./wordpress

CKA (Certificated Kubernetes Administrator)

Certified Kubernetes Administrator (CKA)

The Certified Kubernetes Administrator (CKA) program provides assurance that CKAs have the skills, knowledge, and competency to perform the responsibilities of Kubernetes administrators.

Who Is It For


This certification is for Kubernetes administrators, cloud administrators and other IT professionals who manage Kubernetes instances.

About This Certification

CKA was created by The Linux Foundation and the Cloud Native Computing Foundation (CNCF) as a part of their ongoing effort to help develop the Kubernetes ecosystem. The exam is an online, proctored, [read more](#)

What It Demonstrates

A certified K8s administrator has demonstrated the ability to do basic installation as well as configuring and managing production-grade Kubernetes clusters. They will have an understanding of key concepts [read more](#)



CERTIFIED kubernetes ADMINISTRATOR

\$375
Exam only

Enroll Today
[Get a Quote](#)

Add the Companion Course [Kubernetes Fundamentals \(LES258\)](#) and save \$100

\$575
Course + Exam






Buy Bundle
[Get a Quote](#)

100% Money Back Guarantee

Includes

Domains & Competencies

[Expand All](#)

	Storage	10%	+
	Troubleshooting	30%	+
	Workloads & Scheduling	15%	+
	Cluster Architecture, Installation & Configuration	25%	+
	Services & Networking	20%	+

[See Less](#)

- CKA 자격증
- 응시료 375달러 -> 할인 시 319달러
- 구매 후 1년 내 2번 응시 기회
 - 한 번에 3시간 진행
 - 온라인 / GSAT 처럼 응시환경 체크
 - Kubernetes doc. 참고 가능

**Certified
Kubernetes
Administrator (CKA)**

Coupon Code ✕
SCOFFER15

\$318.75 ✕

Total:

\$318.75

CKA (Certificated Kubernetes Administrator)

제가 개인적으로 생각하기에 크게 4가지 유형으로 나뉘는 것 같습니다.

- 오브젝트 조회
- 오브젝트 생성
- 트러블슈팅
- 노드 추가

오브젝트 조회같은 경우는 주관식 문제라고 봐도 무방합니다. label이 name=soonbee 인 pod들의 이름을 적으라던가 현재 클러스터에서 available 상태인 노드의 개수를 적으라던가 등등의 문제들입니다. 특정 경로에 있는 txt 파일에 답안을 적으라고 합니다. 보통 배점은 1~3점 정도 입니다.

오브젝트 생성은 조건을 주고 생성하라고 합니다. 오브젝트의 이름과 사용할 이미지등을 지정하고 생성하라고 하거나 Service와 Pod를 생성해서 연결시키거나 Configmap을 생성해서 그 값을 Pod의 env로 사용하기 등등 다양한 문제들이 있습니다. 배점은 2~4점 정도 입니다..

트러블슈팅은 시간이 조금 걸립니다. 문제형식은 ‘노드 등 뭔가가 Fail 상태인데 고쳐주세요’ 입니다. log 등 찍어보면서 원인을 찾고 해결하면 됩니다. 이러한 유형의 문제를 풀 때 tmux 사용하시면 유용합니다. 금방 고칠때도 있고 엄청 오래걸리기도 합니다. 배점은 4~7점정도 되는 것 같습니다.

노드 추가 문제는 1문제씩 꼭 나오는 것 같아요. 배점은 7점입니다. 노드를 클러스터에 새로 추가하라는 건데, 저는 그냥 스킵했습니다. 오래 걸리기도 하고 목표는 100점이 아니라 합격이었으니까요. 문제가 엄청 길어요. 보는 순간 이거구나 싶을꺼예요.

이 외에도 DNS 등의 문제도 나옵니다.