10

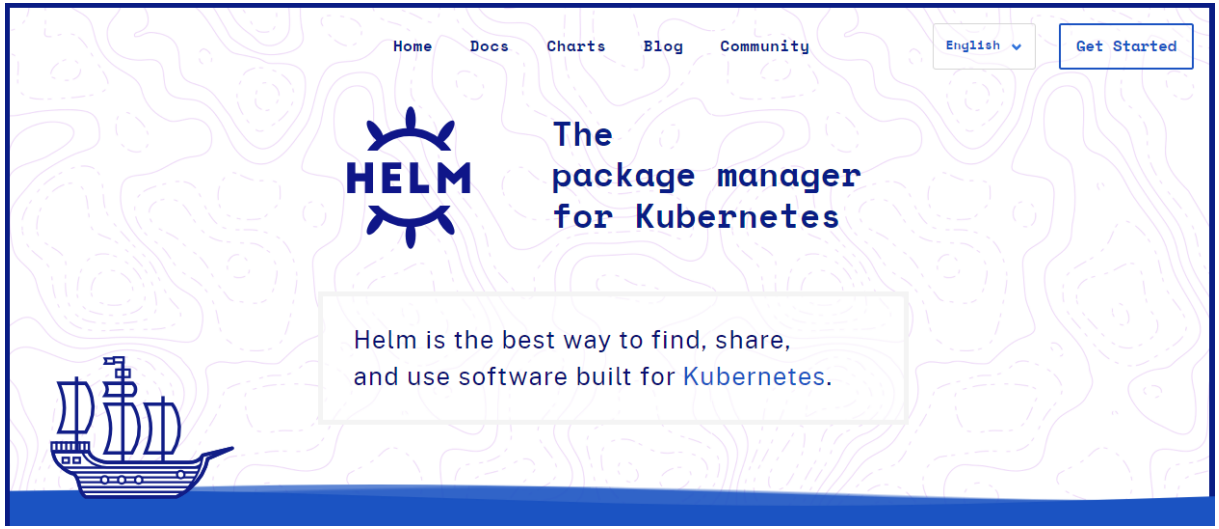
Helm 패키지 관리자

1. Helm이란?
2. Helm의 구성요소
3. helm CLI 명령어
4. helm chart 검토
5. helm chart 테스트

1. Helm이란?

Helm

- https://helm.sh/ko
- k8s를 위한 패키지 매니저
- k8s 기반으로 만들어진 소프트웨어를 검색, 공유, 사용하기 위한 가장 좋은 방법을 제공함
  - 이미 k8s를 학습할 때 nginx-ingress-controller를 설치하기 위해 사용해보았음
- 설치 방법 : https://helm.sh/ko/docs/intro/install/



1. Helm이란?

Ⅰ Helm이 필요한 이유

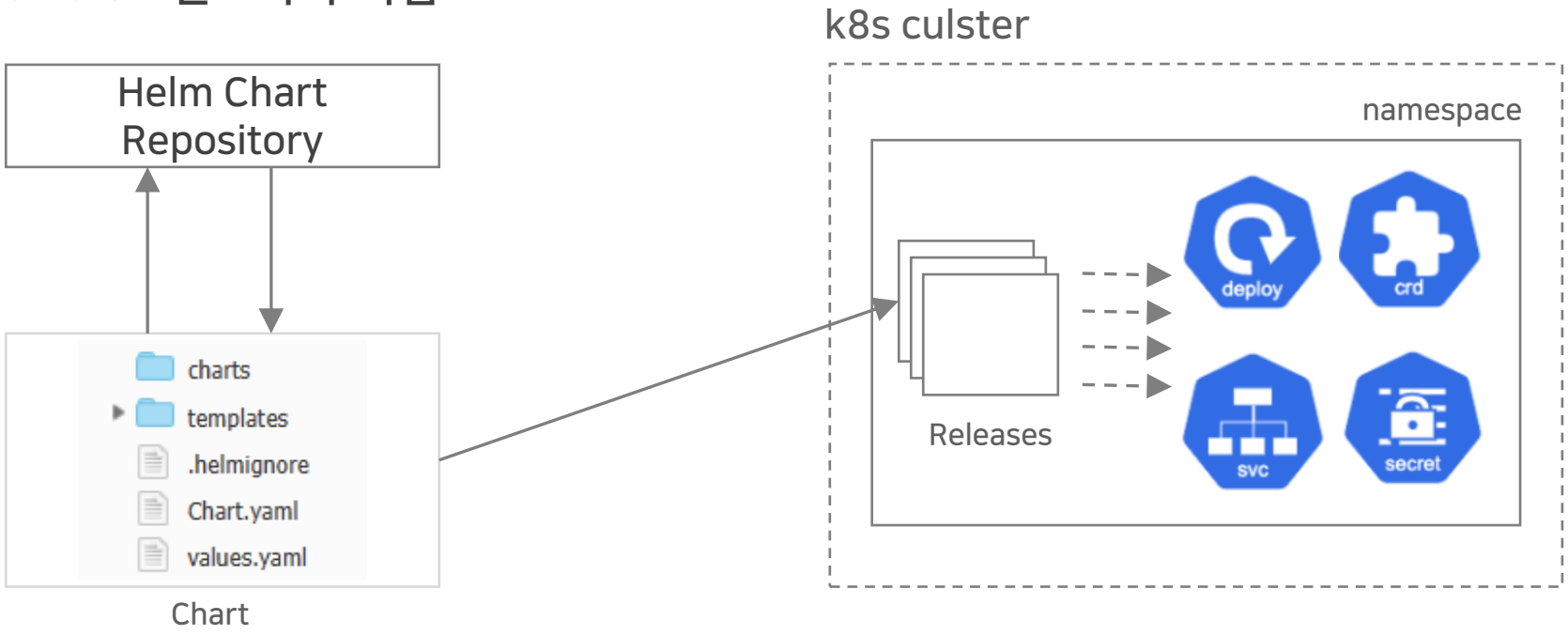
Ⅰ 간단한 마이크로 서비스 기반 애플리케이션 예시

- 배포대상 구성요소
  - 백엔드 MS 3개 --> deployment 3 + clusterip 타입 service 3개
  - 프론트엔드 1개 --> deployment 1 + loadbalancer 타입 service 1개
  - 데이터베이스 연결을 위한 secret 1개
- Helm을 사용하지 않는 경우
  - 배포를 위해서 총 9개의 yaml 파일에 대해 kubectl apply 실행해야 함
  - 패키지 단위로 변경한 후 업그레이드 어려움
  - 특정 시점의 버전으로 롤백 어려움
- Helm을 사용하는 경우
  - Helm 패키지를 구성한 후 한번의 helm install 명령어로 설치
  - Helm 패키지 내부의 템플릿 하나가 변경되더라도 패키지 단위로 업그레이드
  - 설치 또는 업그레이드 후 부여되는 Revision 번호를 이용해 특정 시점으로 롤백할 수 있음

2. Helm의 구성요소

I helm 구성요소

- Chart : 헬름 패키지. k8s cluster에서 애플리케이션, 서비스 등을 실행할 때 필요한 리소스 정의 포함
- Repository : 차트를 모아두고 공유하는 저장소
- Release : 차트의 인스턴스. 차트로 애플리케이션을 설치할 때마다 릴리스 생성. revision 번호가 주어짐



3. helm CLI 명령어

l helm search hub [차트명]

- 공개적으로 사용할 수 있는 차트들을 검색해줌
- 리포지토리 URL도 함께 조회하려면 --list-repo-url 옵션 추가

```
user00:~/environment $ helm search hub fluent-bit --list-repo-url
URL                                             CHART VERSION   APP VERSION     DESCRIPTION                                       REPO URL
https://artifacthub.io/packages/helm/aws/aws-fo...      0.1.27          2.31.11         A Helm chart to deploy aws-for-fluent-bit project       https://aws.github.io/eks-charts
https://artifacthub.io/packages/helm/fluent/flu...      0.34.1          2.1.6           Fast and lightweight log processor and forwarde...      https://fluent.github.io/helm-charts
https://artifacthub.io/packages/helm/stevehipwe...      0.3.0           2.1.6           Helm chart for Fluent Bit running as a collecto...      oci://ghcr.io/stevehipwell/helm-charts/fluent-b...
https://artifacthub.io/packages/helm/stevehipwe...      0.6.0           2.1.6           Helm chart for Fluent Bit running as an aggrega...      oci://ghcr.io/stevehipwell/helm-charts/fluent-b...
```

l helm repo list

- 현재 로컬에 등록된 리포지토리 목록 조회

l helm repo add [repo명] [repo URL]

l helm repo remove [repo명]

l helm search repo [chart명]

```
user00:~/environment $ helm repo add eks-charts https://aws.github.io/eks-charts
"eks-charts" has been added to your repositories
user00:~/environment $ helm search repo fluent-bit
NAME                            CHART VERSION   APP VERSION     DESCRIPTION
eks-charts/aws-for-fluent-bit   0.1.27          2.31.11         A Helm chart to deploy aws-for-fluent-bit project
```

3. helm CLI 명령어

l helm install [release명] [chart명]

- 설치된 후에는 revision 번호가 주어짐

```
user00:~/environment $ helm install fluent-bit eks-charts/aws-for-fluent-bit
NAME: fluent-bit
LAST DEPLOYED: Tue Jul  4 05:02:48 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
fluent-bit has been installed or updated. To check the status of pods, run:

kubectl get pods -n default
```

- 압축 파일, URL로도 설치 가능
  - helm install fluent-bit aws-for-fluent-bit-0.1.27.tgz
  - helm install fluent-bit https://sample.com/charts/aws-for-fluent-bit-0.1.27.tgz

l helm status [release명]

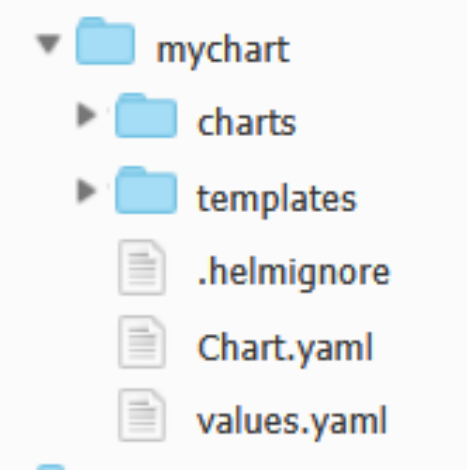
- 릴리스 상태 추적 및 구성 정보 확인

l helm pull [차트명]

- 차트를 압축하여 다운로드

3. helm CLI 명령어

- **helm upgrade [release명] [chart명]**
  - 새로운 revision 번호가 주어짐
- **helm rollback [release명] [특정 revision 번호]**
  - 해당 revision 번호 시점으로 롤백
- **helm list**
  - install된 릴리스 목록 조회
- **helm uninstall [release 명]**
  - helm uninstall fluent-bit
- **helm create [chart 명]**
  - 새로운 차트 생성
  - 예) helm create mychart



4. helm chart 검토

I 이전 페이지에서 생성한 차트 검토

- Chart.yaml
  - 차트의 기본 설정 정보 포함. 필수 파일

```yaml
apiVersion: v2
name: mychart
description: A Helm chart for Kubernetes
type: application
version: 0.1.0
appVersion: "1.16.0"
```

```
apiVersion: 차트 API 버전 (필수)
name: 차트명 (필수)
version: SemVer 2 버전 (필수)
kubeVersion: 호환되는 쿠버네티스 버전의 SemVer 범위 (선택)
description: 이 프로젝트에 대한 간략한 설명 (선택)
type: 차트 타입 (선택)
keywords:
  - 이 프로젝트에 대한 키워드 리스트 (선택)
home: 프로젝트 홈페이지의 URL (선택)
sources:
  - 이 프로젝트의 소스코드 URL 리스트 (선택)
dependencies: # 차트 필요조건들의 리스트 (optional)
  - name: 차트명 (nginx)
    version: 차트의 버전 ("1.2.3")
    repository: 저장소 URL ("https://example.com/charts") 또는 ("@repo-name")
    condition: (선택) 차트들의 활성/비활성을 결정하는 boolean 값을 만드는 yaml 경로 (예시: subchart1.enabled)
    tags: # (선택)
      - 활성화 / 비활성을 함께하기 위해 차트들을 그룹화 할 수 있는 태그들
    enabled: (선택) 차트가 로드될수 있는지 결정하는 boolean
    import-values: # (선택)
      - ImportValues 는 가져올 상위 키에 대한 소스 값의 맵핑을 보유한다. 각 항목은 문자열이거나 하위 / 상위 하위 목록 항목 쌍일 수 있다.
    alias: (선택) 차트에 대한 별명으로 사용된다. 같은 차트를 여러번 추가해야할때 유용하다.
maintainers: # (선택)
  - name: maintainer들의 이름 (각 maintainer마다 필수)
    email: maintainer들의 email (각 maintainer마다 선택)
    url: maintainer에 대한 URL (각 maintainer마다 선택)
icon: 아이콘으로 사용될 SVG나 PNG 이미지 URL (선택)
appVersion: 이 앱의 버전 (선택). SemVer인 필요는 없다.
deprecated: 차트의 deprecated 여부 (선택, boolean)
annotations:
  example: 키로 매핑된 주석들의 리스트 (선택).
```

4. helm chart 검토

I deployment.yaml 의 37행과 values.yaml 의 9~14행을 비교함

```
image: "{{ .Values.image.repository }}:{{ .Values.image.tag |
default .Chart.AppVersion }}"


image:
   repository: nginx
   pullPolicy: IfNotPresent
   # Overrides the image tag whose default is the chart appVersion.
   tag: ""
```

- values.yaml은 여러 리소스에서 사용하는 설정 정보를 한군데서 관리함
- 리소스 yaml 파일에서는 {{ }}과 같은 Go 템플릿을 사용하여 참조

4. helm chart 검토

I 템플릿 built-in 객체

- Release : Release 자체를 서술함
  - Release.Name
  - Release.Revision
- Values : values.yaml 파일의 설정
- Chart : Chart.yaml 파일의 정보
- 예시
  - {{ .Values.replicaCount }}

```
{{- define "testapp.fullname" -}}
{{- if .Values.fullnameOverride }}
{{- .Values.fullnameOverride | trunc 63 | trimSuffix "-" }}
{{- else }}
{{- $name := default .Chart.Name .Values.nameOverride }}
{{- if contains $name .Release.Name }}
{{- .Release.Name | trunc 63 | trimSuffix "-" }}
{{- else }}
{{- printf "%s-%s" .Release.Name $name | trunc 63 |
trimSuffix "-" }}
{{- end }}
{{- end }}
{{- end }}
```

I 정의된 템플릿 호출

- {{ include "testapp.fullname" . }}
- 템플릿 정의는 templates/_helpers.tpl 파일에 define 으로 정의함.
- _helpers.tpl 파일에서 사용하는 템플릿 함수 목록
  - https://helm.sh/ko/docs/chart_template_guide/function_list/

5. helm chart 테스트

l nodeapp으로 변경한 후 install

- Values.yaml에서 다음 내용 변경

```
# 9~14행의 repositor와 tag 변경
image:
   repository: stepanowon/nodeapp
   tag: "1.0.0"
# 57행의 port를 8080으로 변경
   port: 8080
```

- helm install myapp mychart

l 설치된 앱 확인

- kubectl get pods
- kubectl get svc

```
stepanowon:~/environment/helm $ kubectl get pods
NAME                             READY   STATUS    RESTARTS   AGE
myapp-mychart-5c9d5d54fc-6tx4r   1/1     Running   0          44s
stepanowon:~/environment/helm $ kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes      ClusterIP   10.100.0.1      <none>        443/TCP    8m30s
myapp-mychart   ClusterIP   10.100.74.132   <none>        8080/TCP   48s
```

5. helm chart 테스트

I helm 패키지 변경

- values.yaml에서 replicaCount를 1에서 2로 변경
- helm upgrade myapp mychart 실행 후 Revision 확인

```
stepanowon:~/environment/helm $ helm  upgrade  myapp  mychart
Release "myapp" has been upgraded. Happy Helming!
NAME: myapp
LAST DEPLOYED: Tue Aug 12 07:41:34 2025
NAMESPACE: default
STATUS: deployed
REVISION: 2
NOTES:
1. Get the application URL by running these commands:
     NOTE: It may take a few minutes for the LoadBalancer IP to be available.
           You can watch its status by running 'kubectl get --namespace default svc -w myapp-mychart'
  export SERVICE_IP=$(kubectl get svc --namespace default myapp-mychart --template "{{ range (index .sta
tus.loadBalancer.ingress 0) }}{{.}}{{ end }}")
  echo http://$SERVICE_IP:8080
```

5. helm chart 테스트

I kubectl get pods 명령어로 실행중인 pod가 2개인지 확인

```
stepanowon:~/environment/helm $ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
myapp-mychart-5c9d5d54fc-6tx4r    1/1     Running   0          3m43s
myapp-mychart-5c9d5d54fc-xc45r    1/1     Running   0          53s
```

I Revision 1로 롤백 시도

- helm rollback myapp 1
  - 실행중인 pod가 하나인지 확인

```
stepanowon:~/environment/helm $ helm rollback myapp 1
Rollback was a success! Happy Helming!
stepanowon:~/environment/helm $ kubectl get pods
NAME                              READY   STATUS        RESTARTS   AGE
myapp-mychart-5c9d5d54fc-6tx4r    1/1     Running       0          7m54s
myapp-mychart-5c9d5d54fc-xc45r    1/1     Terminating   0          5m4s
```

I 배포된 릴리스 삭제

- helm uninstall myapp

6. Chart Repository

I chart repository 란?

- index.yaml과 패키지화된 Chart를 저장하는 HTTP 서버

I chart repository 구성 방법

- 다양한 저장소 활용 가능
  - HTTP 를 지원하는 저장소 : github page, Google Cloud Storage, AWS S3
  - 리포지토리 지원 전용 서비스 : nexus, harbor
- 자체적인 인증 기능 없음
  - 저장소가 지원하는 인증 기능 활용
- 디렉토리 구조

```
charts/
    |
    |- index.yaml
    |- mychart-0.1.0.tgz
    |- mychart-0.1.1.tgz
```

6.1 github을 이용한 repository 구성

I github에 리포지토리 생성

- name : test-charts
- public 선택

I git clone : cloud9에서

```
mkdir -p ~/environment/helm
cd ~/environment/helm
git clone https://github.com/깃헙사용자명/test-charts
```

I helm 차트 생성, 패키징, 인덱싱 : Cloud9에서

```
# charts 디렉토리 생성
cd test-charts
# 차트 생성
helm create mychart
# 차트 패키징, 인덱스 생성
helm package mychart
helm repo index . --url https://깃헙사용자명.github.io/test-charts
```

6.1 github을 이용한 repository 구성

I git push 수행 : Cloud9에서

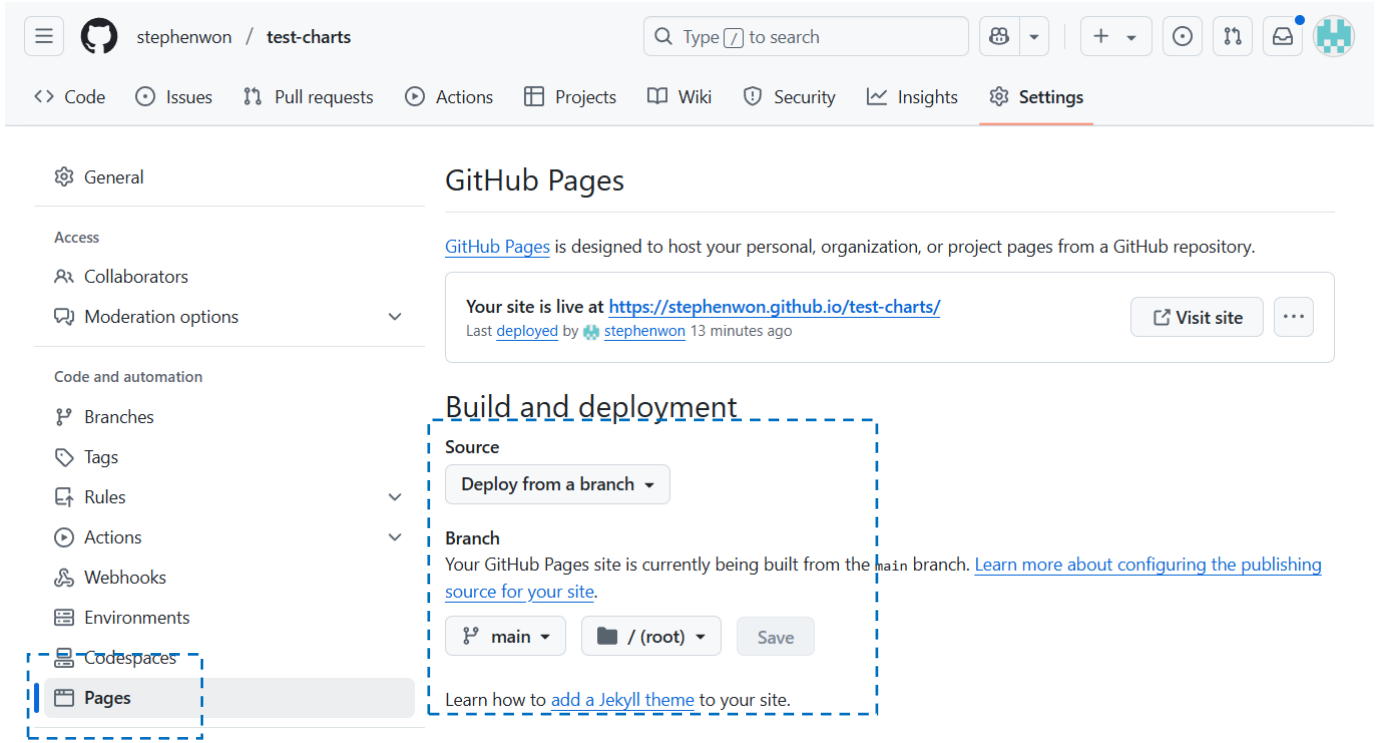
```
cd ~/environment/helm/test-charts
git add .
git commit -m 'initial commit'
git push
```

6.2 github page 구성

I test-charts 리포지토리의 settings-pages로 이동 후 설정

- Source : Deploy from a branch
- Branch : main, /(root)
- 'save' 버튼 클릭

생성된 URL : https://깃헙사용자명.github.io/test-charts/charts



6.3 helm repository 등록, 테스트

I helm 리포지토리 등록 : Cloud9에서

- helm 도구 설정
  - https://helm.sh/ko/docs/intro/install/
- 리포지토리 등록, 업데이트

```
helm repo add test-charts https://깃헙사용자명.github.io/test-charts
helm repo update
```

I helm을 이용해 release 테스트

```
# release명을 myapp으로 설치
helm install myapp test-charts/mychart

# 설치된 pods 확인 : 1 pod
kubectl get pods
```

6.4 차트 변경 후 적용

I 차트 정보 수정, 패키징 후 github에 업데이트

```
# mychart의 values.yaml과 Chart.yaml 편집
# -- values.yaml 에서 replicaCount를 1에서 2로 수정
# -- Chart.yaml 에서 version을 0.1.0에서 0.1.1로 수정

# 패키징, 인덱싱
cd ~/environment/helm/test-charts
helm package mychart
helm repo index . --url https://깃헙사용자명.github.io/test-charts

# git hub에 업데이트
git add .
git commit -m '0.1.1 : replica 2'
git push
```

6.4 차트 변경 후 적용

I helm 을 이용해 배포된 release 업그레이드

```
# github page가 갱신되는데 2-3분 정도가 필요하므로 조금 대기했다가 수행할 것
# helm 리포지토리 업데이트, 업그레이드
helm repo update
helm upgrade myapp test-charts/mychart

# replicaCount 적용 여부 확인 -> pod 갯수 확인 : 1-->2
kubectl get pods
```

I 리소스 정리 : 배포된 release 삭제

- helm uninstall myapp