

## Contents

- Helm
  - Helm concept
  - Helm Commands

## Helm

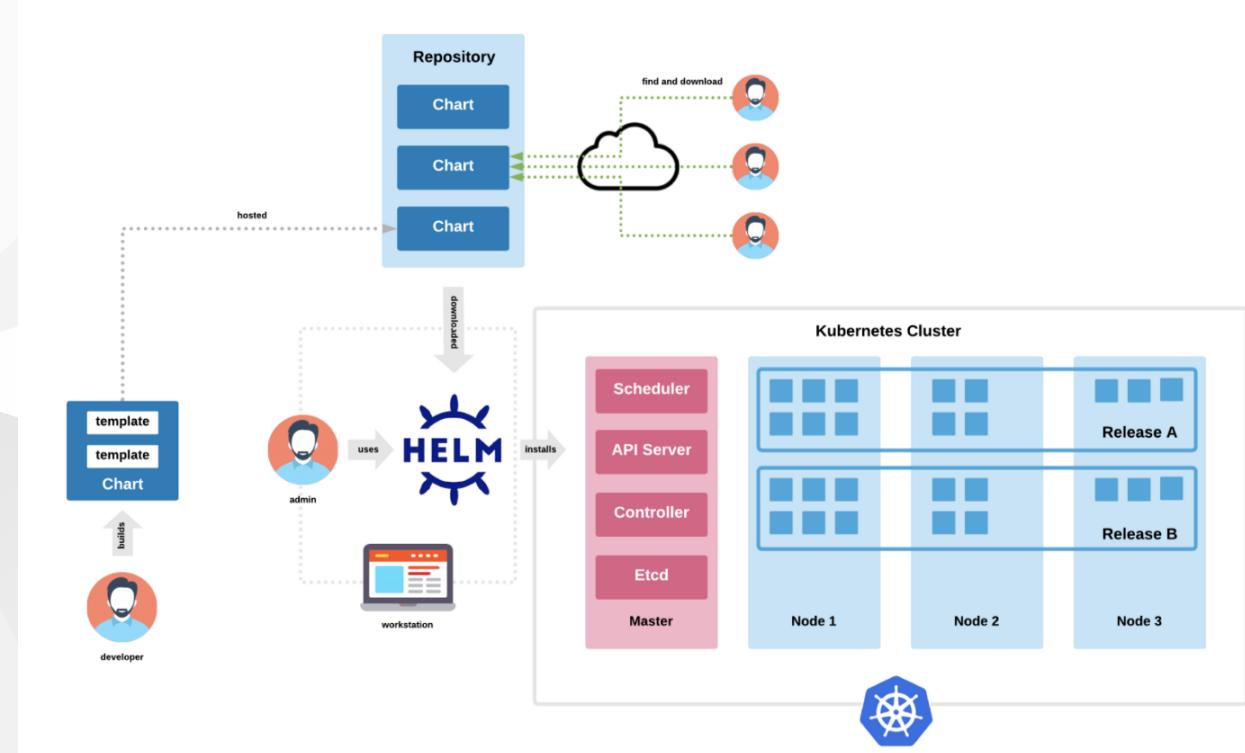
Helm은 Kuberentes **Package Managing Tool**입니다. node.js의 npm과 유사하게 Kuberentes의 Package를 배포 가능하게 하는 Tool이라고 생각하시면 됩니다.

**Chart** 라고 부르는 Package Format을 사용하게되는데, 이 Chart는 **Kubernetes Resource**들을 정의하는 파일들의 집합입니다.

Helm을 통해 다음과 같은 일을 할 수 있습니다.

- 새로운 Chart를 생성할 수 있습니다.
- Chart들을 통해 Chart 아카이브(tgz) 파일을 생성할 수 있습니다.
- Chart들이 저장된 Repositories와 상호작용(가져오기 등)을 할 수 있습니다.
- Chart들을 Kubernetes Cluster에 설치/제거 할 수 있습니다.
- Helm을 통해 설치된 Chart들에 대한 Release Cycle을 관리할 수 있습니다.

# Helm



Helm은 다음과 같은 3가지 중요한 Concept을 가지고 있습니다.

- **Chart** : Helm의 package이며, 이 package에는 Kubernetes Resource들을 담고 있음
- **Repository** : Chart(Kubernetes Package) 저장소
- **Release** : Kubernetes Cluster에서 구동되는 Chart의 Instance

## Charts

Chart는 디렉토리 내부에 파일들로 구성됩니다. 디렉토리 이름이 Chart의 이름이 됩니다.  
예를 들어, WordPress라는 Chart는 `wordpress/` 디렉토리에 저장이 됩니다.

디렉토리 내에는 아래와 같은 구조로 파일들이 존재 합니다.

```
wordpress/
  Chart.yaml      # A YAML file containing information about the chart
  LICENSE        # OPTIONAL: A plain text file containing the license for the chart
  README.md       # OPTIONAL: A human-readable README file
  values.yaml     # The default configuration values for this chart
  values.schema.json # OPTIONAL: A JSON Schema for imposing a structure on the values.yaml file
  charts/          # A directory containing any charts upon which this chart depends.
  crds/            # Custom Resource Definitions
  templates/        # A directory of templates that, when combined with values,
                    # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

Helm은 `charts/` 와 `templates/` 디렉토리와, 지정된 파일명을 사용하도록 하고 있습니다.

## Chart.yaml 파일

Chart.yaml 파일은 Chart에 필수적인 파일입니다. 다음과 같은 Field를 포함하고 있습니다.

```
apiVersion: The chart API version (required)
name: The name of the chart (required)
version: A SemVer 2 version (required)
kubeVersion: A SemVer range of compatible Kubernetes versions (optional)
description: A single-sentence description of this project (optional)
type: The type of the chart (optional)
keywords:
  - A list of keywords about this project (optional)
home: The URL of this projects home page (optional)
sources:
  - A list of URLs to source code for this project (optional)
dependencies: # A list of the chart requirements (optional)
  - name: The name of the chart (nginx)
    version: The version of the chart ("1.2.3")
    repository: (optional) The repository URL ("https://example.com/charts") or alias ("@repo-name")
    condition: (optional) A yaml path that resolves to a boolean, used for enabling/disabling charts (e.g. subchart1.enabled )
    tags: # (optional)
      - Tags can be used to group charts for enabling/disabling together
import-values: # (optional)
  - ImportValues holds the mapping of source values to parent key to be imported. Each item can be a string or pair of child/parent sublist items.
alias: (optional) Alias to be used for the chart. Useful when you have to add the same chart multiple times
```

## 앞장에서 계속

```
maintainers: # (optional)
  - name: The maintainers name (required for each maintainer)
    email: The maintainers email (optional for each maintainer)
    url: A URL for the maintainer (optional for each maintainer)
icon: A URL to an SVG or PNG image to be used as an icon (optional).
appVersion: The version of the app that this contains (optional). Needn't be SemVer. Quotes recommended.
deprecated: Whether this chart is deprecated (optional, boolean)
annotations:
  example: A list of annotations keyed by name (optional).
```

Helm 3를 사용하는 경우 `apiVersion` 은 `v2`를 이어야 합니다. (Helm 2는 `v1`)  
`name`은 Chart의 이름이며, `version`은 Chart의 version을 의미합니다.

## Template 파일

Helm Chart의 Template들은 Go template language로 작성되어 있으며, 다양한 template function들을 사용할 수 있습니다.

모든 Template 파일들은 Chart의 `template/` 디렉토리에 저장되고, Helm이 Chart를 Rendering할 때, 해당 디렉토리 내의 모든 파일들이 template engine으로 전달됩니다.

Template이 사용하는 Value들은 두가지 방법으로 제공될 수 있습니다.

- Chart 개발자가 `values.yaml` 파일을 chart 내에 포함시켜 제공, 이 파일은 default value를 포함하고 있습니다.
- Chart 사용자가 별도의 `yaml` 파일을 사용, `helm install` 명령을 통해 사용합니다.

만약 사용자가 custom value를 사용하는 경우, 이 value는 chart 내에 있는 `values.yaml`의 value를 override 합니다.

template의 예는 아래와 같습니다.

```
{%raw%}apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "sample.fullname" . }}
  labels:
    {{- include "sample.labels" . | nindent 4 --}}
spec:
  {{- if not .Values.autoscaling.enabled --}}
  replicas: {{ .Values.replicaCount }}
  {{- end --}}
  selector:
    matchLabels:
      {{- include "sample.selectorLabels" . | nindent 6 --}}
template:
  metadata:
    {{- with .Values.podAnnotations --}}
  annotations:
    {{- toYaml . | nindent 8 --}}
  {{- end --}}
  labels:
    {{- include "sample.selectorLabels" . | nindent 8 }}}
```

## 앞장에서 계속

```
spec:  
  {{- with .Values.imagePullSecrets }}  
  imagePullSecrets:  
    {{- toYaml . | nindent 8 }}  
  {{- end }}  
  serviceAccountName: {{ include "sample.serviceAccountName" . }}  
  securityContext:  
    {{- toYaml .Values.podSecurityContext | nindent 8 }}  
  containers:  
    - name: {{ .Chart.Name }}  
      securityContext:  
        {{- toYaml .Values.securityContext | nindent 12 }}  
      image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"  
      imagePullPolicy: {{ .Values.image.pullPolicy }}  
      ports:  
        - name: http  
          containerPort: 80  
          protocol: TCP  
      livenessProbe:  
        httpGet:  
          path: /  
          port: http  
      readinessProbe:  
        httpGet:  
          path: /  
          port: http
```

## 앞장에서 계속

```
resources:  
  {{- toYaml .Values.resources | nindent 12 }}  
{{- with .Values.nodeSelector }}  
nodeSelector:  
  {{- toYaml . | nindent 8 }}  
{{- end }}  
{{- with .Values.affinity }}  
affinity:  
  {{- toYaml . | nindent 8 }}  
{{- end }}  
{{- with .Values.tolerations }}  
tolerations:  
  {{- toYaml . | nindent 8 }}  
{{- end }}{{%endraw%}}
```

## values.yaml 파일

template 파일에서 보았듯이, template에 필요한 value 들은 **values.yaml** 파일에서 제공될 수 있습니다. Chart에 포함되어 있는 values.yaml 파일의 value 들은 Chart를 설치하는데 기본값으로 사용됩니다.

values.yaml 파일의 예는 다음과 같습니다.

```
# Default values for sample.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

replicaCount: 1

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

imagePullSecrets: []
nameOverride: ""
fullnameOverride: ""

serviceAccount:
  # Specifies whether a service account should be created
  create: true
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name: ""
```

## 앞장에서 계속

```
podAnnotations: {}

podSecurityContext: {}
  # fsGroup: 2000

securityContext: {}
  # capabilities:
  #   drop:
  #     - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

service:
  type: ClusterIP
  port: 80

ingress:
  enabled: false
  className: ""
  annotations: {}
    # kubernetes.io/ingress.class: nginx
    # kubernetes.io/tls-acme: "true"
hosts:
  - host: chart-example.local
    paths:
      - path: /
        pathType: ImplementationSpecific
tls: []
  #   - secretName: chart-example-tls
  #     hosts:
  #       - chart-example.local
```

## 앞장에서 계속

```
resources: {}
  # We usually recommend not to specify default resources and to leave this as a conscious
  # choice for the user. This also increases chances charts run on environments with little
  # resources, such as Minikube. If you do want to specify resources, uncomment the following
  # lines, adjust them as necessary, and remove the curly braces after 'resources:'.
  # limits:
  #   cpu: 100m
  #   memory: 128Mi
  # requests:
  #   cpu: 100m
  #   memory: 128Mi

autoscaling:
  enabled: false
  minReplicas: 1
  maxReplicas: 100
  targetCPUUtilizationPercentage: 80
  # targetMemoryUtilizationPercentage: 80

nodeSelector: {}

tolerations: []

affinity: {}
```

기본값이 아닌 별도로 지정하고 싶은 value들이 있다면, 별도의 yaml 파일을 작성하여 Chart를 설치할 때 사용할 수 있습니다. 예를 들어, 아래와 같이 myval.yaml 파일을 작성하여 Chart를 설치할 때 사용하면,

```
replicaCount: 2
service:
  type: NodePort
  port: 80
```

실제로 Chart 가 설치될때, replica는 1개가 아닌 2개로, service는 ClusterIP Type이 아닌 NodePort Type으로 생성이 됩니다.

values.yaml 로 정의된 value들은 template 에서는 `.Values` object를 통해 접근 할 수 있습니다.  
상기 예에서 service type은 `.Values.service.type`으로 접근 할 수 있습니다.

Chart에 포함되는 values.yaml 파일의 이름은 변경할 수 없으며, helm 명령과 함께 지정할 수 있는 별도의 yaml 파일명은 어떤 것 이든 가능합니다.

## Helm Commands

helm은 helm cli를 통해 사용 가능하며, 만약 설치되어 있지 않다면, 공식 release binary (아래 링크)를 설치하여 사용 가능합니다.

[Releases · helm/helm · GitHub](#)

사용하는 OS에 맞는 압축된 binary를 다운받아서 적절한 위치에 압축해제 후 사용하시면 됩니다.

### helm search

chart를 검색하기 위한 command이며, 두가지 소스 유형을 검색할 수 있습니다.

- `helm search hub`는 여러 저장소들에 있는 helm chart를 포함하는 [Artifact Hub](#)에서 검색합니다.
- `helm search repo`는 `helm repo add`를 사용하여 로컬 helm client에 추가된 저장소에서 검색합니다.

URL	CHART VERSION	APP VERSION	DESCRIPTION
<a href="https://artifacthub.io/packages/helm/kube-wordpress">https://artifacthub.io/packages/helm/kube-wordp...</a>	0.1.0	1.1	this is my wordpress package
<a href="https://artifacthub.io/packages/helm/bitnami/wordpress">https://artifacthub.io/packages/helm/bitnami/wo...</a>	13.1.15	5.9.3	WordPress is the world's most popular blogging ...
<a href="https://artifacthub.io/packages/helm/bitnami-akashic/wordpress">https://artifacthub.io/packages/helm/bitnami-aka...</a>	13.1.12	5.9.2	WordPress is the world's most popular blogging ...
<a href="https://artifacthub.io/packages/helm/riftbit/wordpress">https://artifacthub.io/packages/helm/riftbit/wo...</a>	12.1.16	5.8.1	Web publishing platform for building blogs and ...
<a href="https://artifacthub.io/packages/helm/sikalabs/wordpress">https://artifacthub.io/packages/helm/sikalabs/w...</a>	0.2.0		Simple Wordpress

## Docker & Kubernetes - 15. Helm

helm search repo는 사전에 추가된 repository로 부터 검색을 하여, repository 추가는 아래와 같이 합니다.

(아래 예는, Open Source S/W 들을 쉽게 구성하고 설치할 수 있도록 해주는 VMWare의 Bitnami Repository를 사용하는 예입니다.)

```
$ helm repo add bitnami https://charts.bitnami.com/bitnami  
"bitnami" has been added to your repositories
```

```
$ helm search repo wordpress  
NAME          CHART VERSION  APP VERSION  DESCRIPTION  
bitnami/wordpress  13.1.15      5.9.3        WordPress is the world's most popular blogging ...  
bitnami/wordpress-intel 0.1.29      5.9.3        WordPress for Intel is the most popular bloggin...
```

search 통해 설치하고자 하는 package를 찾았다면, helm install을 통해 설치할 수 있습니다.

## helm install

chart를 설치하는 방법은 helm install을 사용하는 것이고, 사용 방법은 `helm install 'release명' 'chart명'`을 통해 chart가 설치됩니다.

```
$ helm install my-wordpress bitnami/wordpress
NAME: my-wordpress
LAST DEPLOYED: Mon Apr 11 11:31:55 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: wordpress
CHART VERSION: 13.1.15
APP VERSION: 5.9.3
```

\*\* Please be patient while the chart is being deployed \*\*

Your WordPress site can be accessed through the following DNS name from within your cluster:

my-wordpress.default.svc.cluster.local (port 80)

To access your WordPress site from outside the cluster follow the steps below:

## 앞장에서 계속

1. Get the WordPress URL by running these commands:

NOTE: It may take a few minutes **for** the LoadBalancer IP to be available.

Watch the status with: `'kubectl get svc --namespace default -w my-wordpress'`

```
export SERVICE_IP=$(kubectl get svc --namespace default my-wordpress --include "{{ range (index .status.loadBalancer.ingress 0) }}{{ . }}{{ end }}")  
echo "WordPress URL: http://$SERVICE_IP/"  
echo "WordPress Admin URL: http://$SERVICE_IP/admin"
```

2. Open a browser and access WordPress using the obtained URL.

3. Login with the following credentials below to see your blog:

```
echo Username: user  
echo Password: $(kubectl get secret --namespace default my-wordpress -o jsonpath=".data.wordpress-password" | base64 --decode)
```

## Docker & Kubernetes - 15. Helm

helm install을 통해 chart를 설치하면, Kubernetes Resource가 모두 생성될 때까지 기다리는 것이 아니라, 바로 Deployed라는 결과가 출력됩니다.

`kubectl get all` 명령을 통해 생성된 Resource들을 확인해 보면 다음과 같습니다.

```
$ kubectl get all
NAME                         READY   STATUS    RESTARTS   AGE
pod/my-wordpress-ff8559cd-hrnm8  0/1     Pending   0          5m36s
pod/my-wordpress-mariadb-0       0/1     Pending   0          5m36s

NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)           AGE
service/kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP          93d
service/my-wordpress LoadBalancer 10.96.65.217   <pending>    80:32482/TCP,443:31884/TCP  5m36s
service/my-wordpress-mariadb ClusterIP  10.97.227.246  <none>        3306/TCP         5m36s

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-wordpress  0/1     1            0          5m36s

NAME              DESIRED  CURRENT  READY   AGE
replicaset.apps/my-wordpress-ff8559cd  1        1        0      5m36s

NAME            READY   AGE
statefulset.apps/my-wordpress-mariadb  0/1     5m36s
```

my-wordpress라는 이름으로, **service**, **deployment**, **statefulset** 등이 생성된 것을 확인할 수 있습니다.

## chart customizing

chart를 customizing 하는 방법은, 크게 value 만 수정하는 방법과, chart 자체를 customizing 하는 방법이 있습니다.

### value 변경

앞에서는 별도의 value를 지정하지 않고 설치를 하였기 때문에, chart 내에 포함되어 있는 values.yaml 파일의 내용이 사용되었습니다.

chart에 포함된 values.yaml 파일의 내용을 보는 방법은, `helm show values "chart명"`입니다.

default value들을 확인한 뒤에 특정 value를 변경하여 배포하고자 한다면, 다음과 같은 두가지 방법을 사용할 수 있습니다.

- `--values` (또는 `-f`) : 별도의 value 파일을 생성하고 해당 파일의 경로와 이름을 지정 (예, `--values myval.yaml`)
- `--set` : command line 상에서 value를 지정 (예, `--set wordpressUsername=user,memcached.enabled=false`)

## chart 변경

chart를 수정하여 설치하려면, repository로 부터 chart를 다운받아서 수정을 하고 수정된 chart를 가지고 helm 설치를 할 수 있습니다.

chart를 다운받기 위해서는 pull 명령을 사용하며 --untar 옵션을 사용하면 chart를 다운받은 후 압축도 풀어줍니다.

```
$ helm pull --untar bitnami/wordpress
$ cd wordpress
$ ls -la
total 152
drwxr-xr-x 5 hojoon hojoon 4096 Apr 12 13:12 .
drwxrwxr-x 5 hojoon hojoon 4096 Apr 12 13:12 ..
-rw-r--r-- 1 hojoon hojoon 388 Apr 12 13:12 Chart.lock
drwxr-xr-x 5 hojoon hojoon 4096 Apr 12 13:12 charts
-rw-r--r-- 1 hojoon hojoon 1025 Apr 12 13:12 Chart.yaml
drwxr-xr-x 2 hojoon hojoon 4096 Apr 12 13:12 ci
-rw-r--r-- 1 hojoon hojoon 333 Apr 12 13:12 .helmignore
-rw-r--r-- 1 hojoon hojoon 65131 Apr 12 13:12 README.md
drwxr-xr-x 2 hojoon hojoon 4096 Apr 12 13:12 templates
-rw-r--r-- 1 hojoon hojoon 5706 Apr 12 13:12 values.schema.json
-rw-r--r-- 1 hojoon hojoon 45757 Apr 12 13:12 values.yaml
```

수정이 필요한 부분을 직접 수정한 뒤에, chart를 설치하는 방법은 아래와 같습니다.

```
$ helm install my-wordpress .
```

chart가 있는 디렉토리를 지정하고 (위의 예는 현재 디렉토리), install 명령을 사용하면 됩니다.

value 들이 반영된 manifest를 확인해 보고자 한다면, `--dry-run` 옵션을 통해 install 해 보면됩니다.

## helm uninstall

설치된 chart (release)를 삭제하는 방법은 `helm uninstall 'release명'`을 사용하는 것입니다.

```
$ helm uninstall my-wordpress  
release "my-wordpress" uninstalled
```

helm install을 통해 설치된 모든 resource가 삭제되는 것을 알 수 있습니다.

## helm upgrade

설치된 chart (release)를 업그레이드 하는 방법은 `helm upgrade` 를 사용하는 것입니다.

변경이 있는 resource 들에 대하여, Deployment 같은 경우는, deployment strategies 에 따라, rolling upgrade 혹은 recreate 되며, 다른 Resource 들은 patch 됩니다.

일반적으로 Docker image가 변경되었을때, 주로 upgrade를 하며, 일부 설정이 변경되는 경우에도 upgrade를 수행하게 됩니다.

## value 변경

특정 value로 업그레이드 하고자 한다면, 다음과 같은 두가지 방법을 사용할 수 있습니다. (install 시와 동일)

- `--values` (또는 `-f`) : 별도의 value 파일을 생성하고 해당 파일의 경로와 이름을 지정 (예, `--values myval.yaml`)
- `--set` : command line 상에서 value를 지정 (예, `--set image.tag=5.9.3-debian-10-r4`)

## helm create

나만의 helm chart를 만들고자 한다면, `helm create 'chart명'` 명령을 통해 새로운 chart를 생성할 수 있습니다.

이 명령을 사용하면, chart에 사용되는 파일들과 디렉토리를 '`'chart명'` 디렉토리 아래 생성해 줍니다.

```
$ helm create mychart
Creating mychart

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

생성된 chart를 원하는대로 수정하면 됩니다.

## helm lint

작성된 chart를 오류가 있는지 검증하려면, `helm lint 'chart명'` 명령을 사용하면 됩니다.

```
$ helm lint mychart
==> Linting mychart
[INFO] Chart.yaml: icon is recommended

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

만약 오류가 있다면 아래와 같이 오류에 대한 정보를 표시해 줍니다.  
(아래는 deployment.yaml에서 deployment name을 지우고 lint를 한 결과입니다.)

```
$ helm lint mychart
==> Linting mychart
[INFO] Chart.yaml: icon is recommended
[WARNING] templates/deployment.yaml: object name does not conform to Kubernetes naming requirements: "":
  metadata.name: Invalid value: "": a lowercase RFC 1123 subdomain must consist of lower case alphanumeric
    characters, '-' or '.', and must start and end with an alphanumeric character (e.g. 'example.com',
    regex used for validation is '[a-z0-9]([-a-z0-9]*[a-z0-9])?(\\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*')

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

## helm package

chart 디렉토리를 아카이브(압축) 합니다.

아카이브된 파일명은, Chart.yaml 파일의 `'name_value'-'version_value'.tgz` 으로 생성됩니다.

```
$ helm package ./mychart
Successfully packaged chart and saved it to: /home/hojoon/helm/mychart-0.1.0.tgz
```

## ▶ Hands-on : 15\_Helm

## Summary

- Helm
  - Helm concept
    - Chart : Chart.yaml / Template / values.yaml
    - Repository
    - Release
  - Helm Commands
    - helm search
    - helm install
    - helm uninstall
    - helm upgrade
    - helm create
    - helm lint
    - helm template
    - helm package