Quản lý package trên Kubernetes

By Minh Monmen

Nội dung

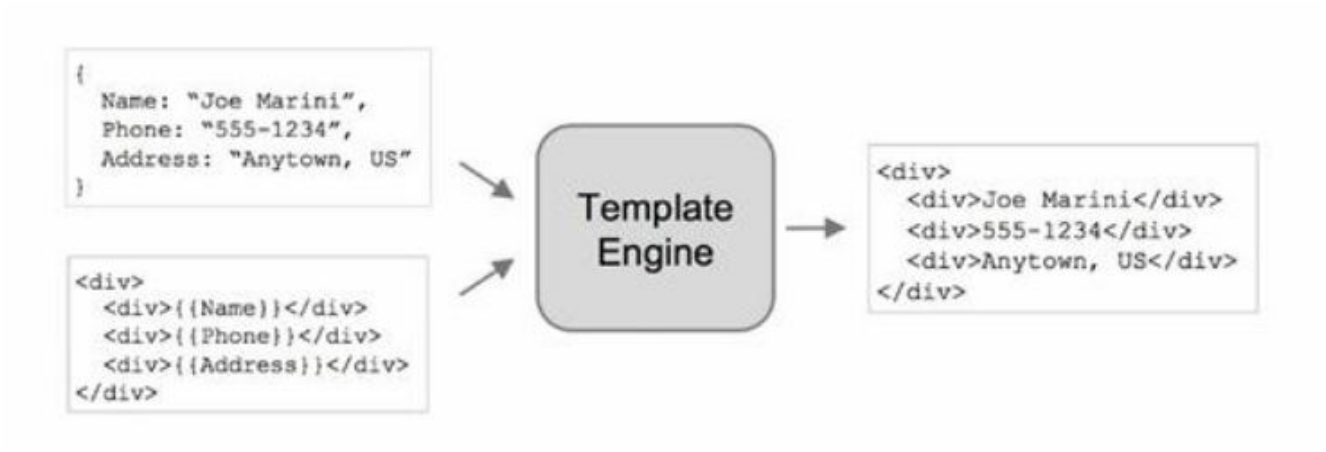
- Yaml template và package manager helm
- Kiến trúc và cấu trúc helm chart
- Sử dụng helm chart
- Một số hàm cơ bản thường dùng
- Đóng gói ứng dụng nâng cao
- Thực hành 1: Đóng gói ứng dụng vào helm chart
- Thực hành 2: Triển khai app bằng helm chart

Yaml config

- Declarative configuration
- Infrastructure as code
- Tất cả mọi resource của K8S đều định nghĩa bằng yaml
- State cũng biểu diễn qua yaml
- Rõ ràng NHƯNG khó quản lý
- Dễ hiểu NHƯNG khó dùng lại

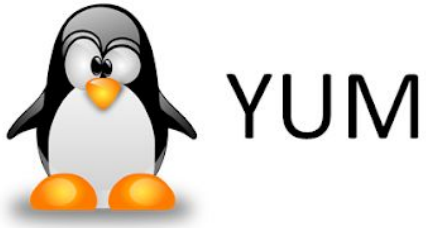
Template

- Dùng nhiều file config tương tự nhau
- Deploy 1 app lên nhiều môi trường
- Template + Values = Config file



Package manager

- Dùng lại 1 thư viện / app nhiều lần
- Đóng gói và triển khai "cả cục"

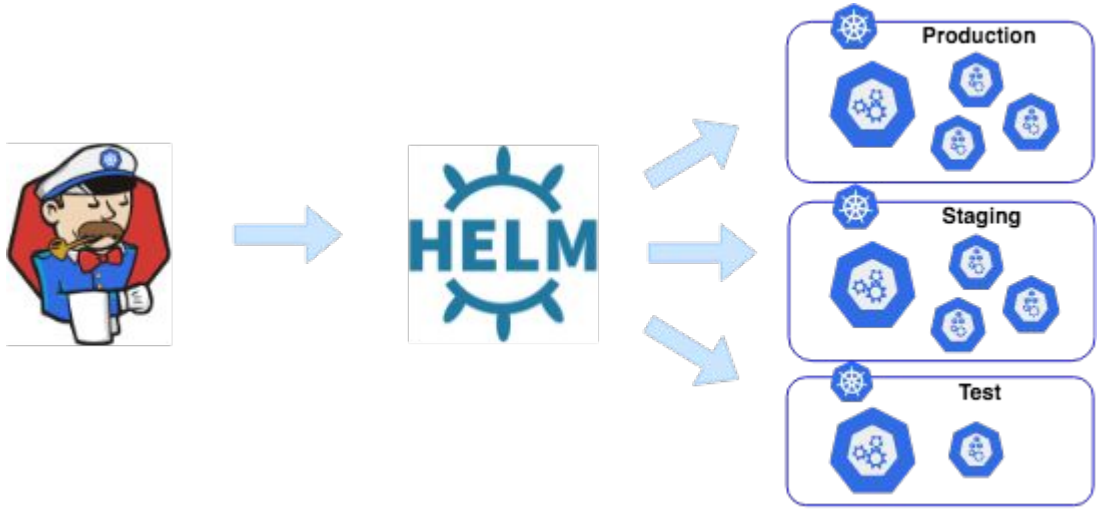


Các hiểu nhầm hay gặp

- Thuật ngữ và công cụ
  - Templating engine
  - Patching system
  - Package manager
- Templating engine: Jsonnet, Helm template
- Patching system: Kustomize
- Package manager: Helm

Helm

- Package manager cho K8S
- Chart (yaml template) + Values = Deployment



Helm concept

- **Chart**: chứa yaml template
- **Values**: chứa giá trị cụ thể để thay vào template
- **Release**: Chart sau khi deploy lên K8S tạo thành 1 Release
- **Repository**: Nơi publish helm chart
- Giống concept docker image, config, container, docker hub

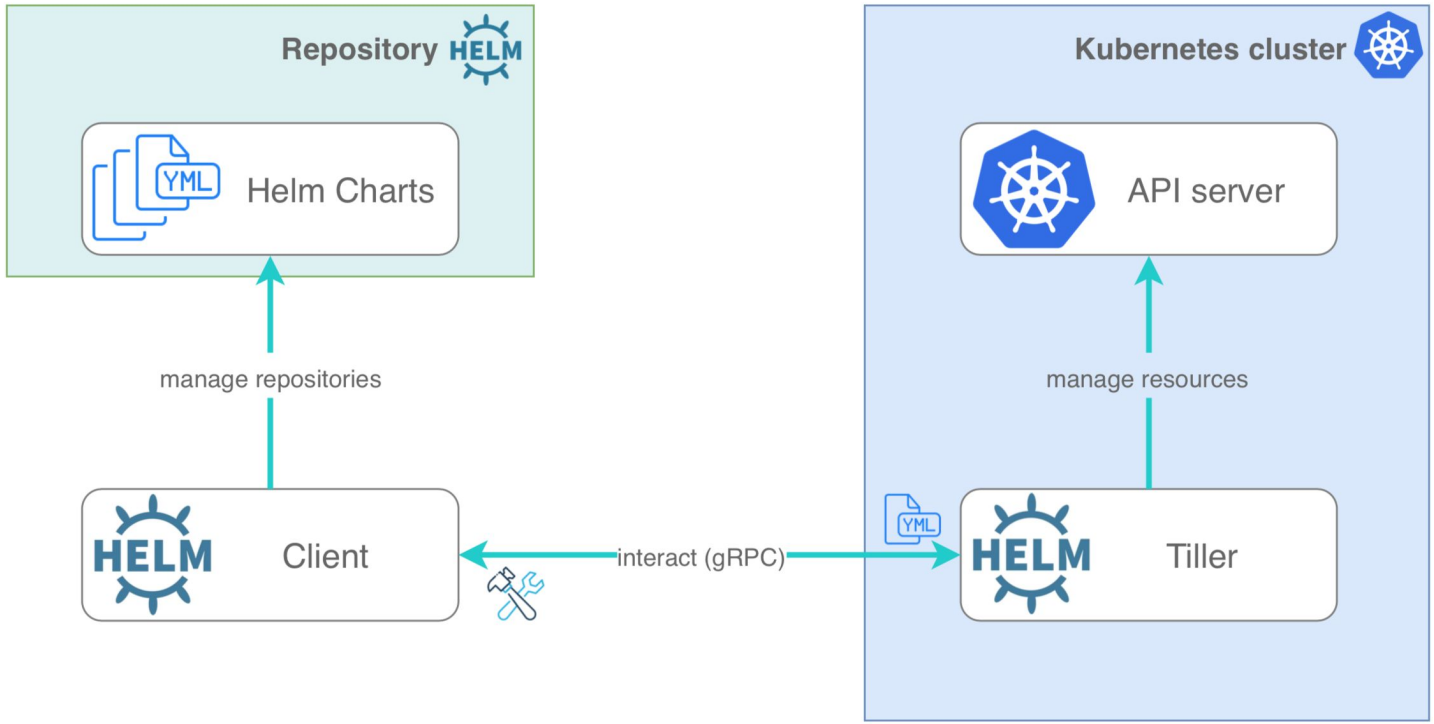
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-deployment
  labels:
    app: todo-v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: todo-v1

  # Pod template
  template:
    metadata:
      name: todo-v1-web
      labels:
        app: todo-v1
    spec:
      containers:
        - name: web
          image: some-image
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
          env:
            - name: SOME_ENV
              value: "hello"
```

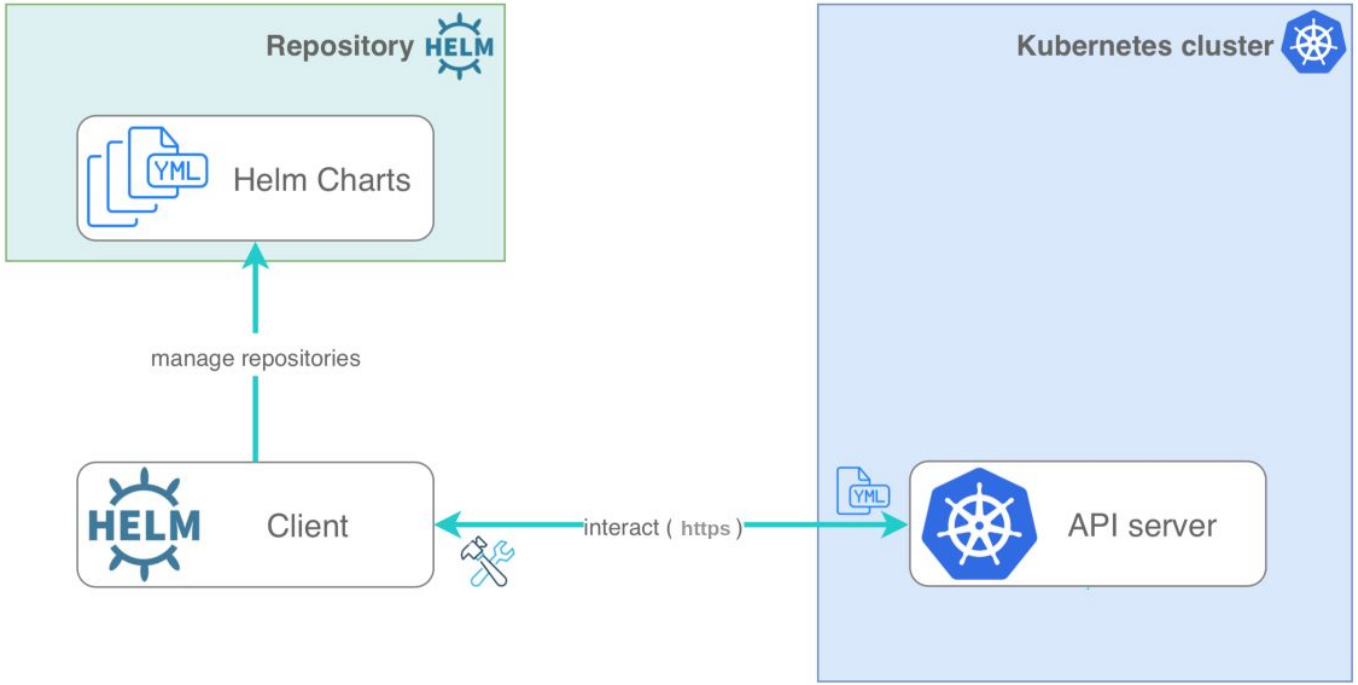
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name | quote }}
  labels:
    app: {{ .Release.Name | quote }}
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: {{ .Release.Name | quote }}

  # Pod template
  template:
    metadata:
      name: {{ .Release.Name }}-web
      labels:
        app: {{ .Release.Name | quote }}
    spec:
      containers:
        - name: web
          image: {{ .Values.image }}
          imagePullPolicy: {{ .Values.imagePullPolicy }}
          ports:
            - containerPort: {{ .Values.containerPort }}
          env:
            {{- range $key, $val := .Values.env }}
            - name: {{ $key | quote}}
              value: {{ $val | quote }}
            {{- end}}
```

Kiến trúc Helm 2



Kiến trúc Helm 3



Cấu trúc 1 helm chart

$ helm create mychart

```
mychart
├── Chart.yaml   # Information about your chart, metadata, version and dependency
├── charts       # Charts that this chart depends on
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── deployment.yaml
│   ├── ingress.yaml
│   ├── service.yaml
│   ├── serviceaccount.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml  # The default values for your templates
```

File Chart.yaml

- Chứa thông tin metadata của package

```yaml
apiVersion: v2
name: demo-chart
description: A Helm chart for Kubernetes

type: application

version: 0.1.0

appVersion: 1.16.0
```

File trong thư mục templates

_helpers.tpl

```
{{/* vim: set filetype=mustache: */}}
{{/*
Expand the name of the chart.
*/}}
{{- define "demo-chart.name" -}}
{{- default .Chart.Name .Values.nameOverride | trunc 63 | trimSuffix "-" -}}
{{- end -}}

{{/*
Create a default fully qualified app name.
We truncate at 63 chars because some Kubernetes name fields are limited to this (by the DNS naming spec).
If release name contains chart name it will be used as a full name.
*/}}
{{- define "demo-chart.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 -}}
```

File trong thư mục templates

- Các file .yaml chứa thông tin resource cần tạo
- Có thể là các file raw yaml, không cần phải có template

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name | quote }}
  labels:
    app: {{ .Release.Name | quote }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: {{ .Release.Name | quote }}

  # Pod template
  template:
    metadata:
      name: {{ .Release.Name }}-web
      labels:
        app: {{ .Release.Name | quote }}
    spec:
      containers:
        - name: web
          image: {{ .Values.image.repository }}
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          ports:
            - containerPort: {{ .Values.containerPort }}
          env:
            {{- range $key, $val := .Values.env }}
            - name: {{ $key | quote}}
              value: {{ $val | quote }}
            {{- end}}
```

File values.yaml

- Chứa thông tin giá trị config cần thay thế
- File values.yaml trong thư mục chart CHỨA CÁC GIÁ TRỊ MẶC ĐỊNH

```yaml
replicaCount: 10

image:
  repository: minhpq331/demo-deployment:v1.0
  pullPolicy: Always

containerPort: 3000

env:
  PORT: "3000"
  MESSAGE: "Hello from the other side"
```

Thứ tự ghi đè yaml value

- **values.yaml trong chart**: Chứa các giá trị mặc định
- **values-<custom>.yaml:** helm... -f values-<custom>.yaml
  - Nếu có nhiều file trong -f thì theo thứ tự sau > trước
- từng value lẻ khi chạy: helm... --set some.thing=value
  - Nếu có nhiều value trong --set thì cũng sau > trước

Kết hợp template + value

```yaml
replicaCount: 10

image:
  repository: minhpq331/demo-deployment:v1.0
  pullPolicy: Always

containerPort: 3000

env:
  PORT: "3000"
  MESSAGE: "Hello from the other side"
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name | quote }}
  labels:
    app: {{ .Release.Name | quote }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: {{ .Release.Name | quote }}

  # Pod template
  template:
    metadata:
      name: {{ .Release.Name }}-web
      labels:
        app: {{ .Release.Name | quote }}
    spec:
      containers:
        - name: web
          image: {{ .Values.image.repository }}
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          ports:
            - containerPort: {{ .Values.containerPort }}
          env:
            {{- range $key, $val := .Values.env }}
            - name: {{ $key | quote}}
              value: {{ $val | quote }}
            {{- end}}
```

Sử dụng helm trong thực tế

- **Cài đặt ứng dụng của bên thứ 3:** Nginx ingress controller, EFK, Prometheus Operator,...
  - Hiểu giá trị mặc định trong values.yaml
  - Đọc chart template khi muốn custom thêm
- **Triển khai ứng dụng của bản thân:** Application, service,...
  - Xây dựng working manifest trước (raw yaml)
  - Thay thế dần các giá trị trong raw manifest ra file values.yaml
  - Dùng giữa các môi trường bằng cách thêm custom values

1 số cú pháp thường dùng

- Cặp dấu {{ }} để bắt đầu sử dụng template và thay thế giá trị
- Cặp dấu {{- -}} nhưng có dấu sẽ loại bỏ khoảng trắng trước và sau (phải rất chú ý với case này)
- with: Tạo 1 scope nhỏ để đỡ lặp lại
- if else: Rẽ nhánh
- range: Vòng lặp
- https://github.com/minhpq331/demo-helm-chart

1 số hàm thường dùng

- default: thay value hoặc giá trị mặc định
- quote: đóng gói giá trị bằng nháy kép
- upper, lower: biến đổi case của chuỗi
- toYaml: include 1 đoạn value dạng yaml
- indent, nindent: lùi lề vào bao nhiêu space (nindent có thêm 1 new line ở đầu)
- include: include 1 template trong _helpers.tpl

Các câu lệnh tương tác với helm 3

- helm ls
- helm install -n <ns> <release-name> <chart-name>
- helm uninstall -n <ns> <release-name>
- **helm upgrade -n <ns> --install -f <custom-values.yaml> --set <key.prop>=<value> <release-name> <chart-name>**
- helm repo add <name> <url>
- helm ... --dry-run --debug

Đóng gói và triển khai ứng dụng nâng cao

- Xây dựng docker image: **1 container - 1 process**
  - 1 image - 1 binary
  - 1 image - n binary
- Xây dựng helm chart: **1 chart - 1 source repo**
  - 1 chart - 1 app
  - 1 chart - n app

Đóng gói và triển khai ứng dụng nâng cao

- Dựa trên kiến trúc ứng dụng để đóng gói
  - API
  - Worker
  - Socket
  - Cronjob
- Quan hệ các thành phần, source code chung,...
- Nắm rõ về yêu cầu concurrency của từng thành phần

Usecase: Multiple version control

- Sử dụng khi chạy song song nhiều version (v1.1, v1.2, v1.3,...)
- Application code KHÔNG quản lý version
- Code mỗi version 1 branch và build với tag khác nhau
- Clone version mới với values khác (image tag khác)
  - helm upgrade -i -f values-v2.yaml app-v2 ./app-chart
- Thay đổi ingress:
  - **/v1/users ~> /users** (release v1)
  - **/v2/users ~> /users** (release v2)

Usecase: Migration

- Sử dụng khi cần deploy 1 phần code mới để chạy script migrate
- Clone version mới với environment y hệt nhưng khác release
  - helm upgrade -i -f values-migration.yaml app-migration ./app-chart
- Chạy migration script trên version mới
- Rolling update v1 ~> v2
- Delete migration release

Usecase: Traffic shadowing

- Sử dụng khi muốn clone traffic, đặt debug trên code...
- Clone version mới với environment y hệt nhưng khác release
  - helm upgrade -i -f values-v2.yaml app-v2 ./app-chart
- Enable traffic mirror trên nginx ingress controller
- Test traffic thật với version mới
- https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#mirror

Usecase: Distributed Monolith

- Sử dụng khi:
  - Ứng dụng có các thành phần có mức depend vào bên thứ 3 khác nhau
  - Chia tách theo mức quan trọng
- Ví dụ:
  - Tách request theo method: GET, POST ~> 2 release khác nhau
  - Tách request theo path dựa vào độ quan trọng ~> release khác nhau
  - Tách request theo path dựa vào kết nối DB, kết nối tới svc khác,...

Thực hành 1: Cài đặt nginx ingress controller

- https://github.com/minhpq331/demo-helm-chart
- Trong README.md có chi tiết các function và cú pháp
- Deploy nginx ingress controller lên namespace của mình
- Trở lại bài thực hành 3 của buổi trước, expose ứng dụng ra ingress

Thực hành 2: Triển khai app bằng helm chart

- https://github.com/minhpq331/demo-helm-chart
- Sử dụng file deployment và service trong bài trước và deploy bằng helm chart.

Tài liệu tham khảo

- https://blog.ropnop.com/attacking-default-installs-of-helm-on-kubernetes/

Thanks for watching!