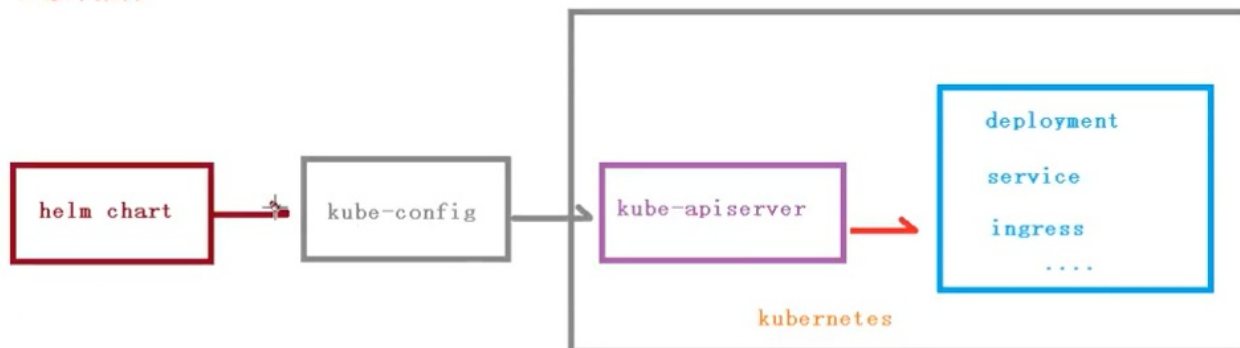


# helm 概述

V3版本架构



## Helm

### 1. Helm 引入

1. 之前方式部署應用的基本過程：
  - 編寫 YAML 文件：
    - Deployment
    - Service
    - Ingress
2. 如果使用之前方式部署單一應用，少數服務的應用，比較合適：
  - 例如部署微服務項目，可能有幾十個服務，每個服務都有一套 YAML 文件，需要維護大量 YAML 文件，版本管理特別不方便。

### 2. 使用 Helm 可以解決哪些問題？

1. 使用 Helm 可以把這些 YAML 作為一個整體管理。
2. 實現 YAML 高效複用。
3. 使用 Helm 應用級別的版本管理。

### 3. Helm 介紹

Helm 是一個 Kubernetes 的包管理工具，類似於 Linux 下的包管理器，如 yum/apt 等，可以很方便地將之前打包好的 YAML 文件部署到 Kubernetes 上。

### 4.Helm 有 3 個重要概念：

1. **Helm**：  
一個命令行客戶端工具，主要用於 Kubernetes 應用 Chart 的創建、打包、發布和管理。
2. **Chart**：  
應用描述，一系列用於描述 Kubernetes 資源相關文件的集合。
3. **Release**：  
基於 Chart 的部署實體。一個 Chart 被 Helm 運行後將會生成對應的一個 Release；它會在 Kubernetes 中創建出真實運行的資源對象。

### 5.Helm 在 2019 年發布 V3 版本，和之前版本相比有變化

1. V3 版本刪除 Tiller，架構變化。
2. Release 可以在不同命名空間重用。
3. 將 Chart 推送到 Docker 倉庫中。

以下是圖片內容的文字辨識及翻譯成繁體中文：

# 1. Helm 安裝

## 1. 下載 Helm 安裝壓縮文件，上傳到 Linux 系統中

```
[root@k8smaster ~]# tar zxvf helm-v3.0.0-linux-amd64.tar.gz
linux-amd64/helm
linux-amd64/README.md
linux-amd64/LICENSE
```

## 2. 解壓 Helm 壓縮文件，將解壓後的 Helm 目錄複製到 `/usr/bin` 目錄下

```
[root@k8smaster linux-amd64]# mv helm /usr/bin
[root@k8smaster linux-amd64]# helm
```

# 2. 配置 Helm 倉庫

## (1) 添加倉庫

```
helm repo add 倉庫名稱 倉庫地址
```

示例：

```
[root@k8smaster linux-amd64]# helm repo add stable http://mirror.azure.cn/kubernetes/charts
"stable" has been added to your repositories
```

檢查倉庫列表：

```
[root@k8smaster linux-amd64]# helm repo list
NAME    URL
stable  http://mirror.azure.cn/kubernetes/charts
```

## (2) 更新倉庫地址

```
[root@k8smaster linux-amd64]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "aliyun" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. Happy Helming!
```

## (3) 刪除倉庫

```
[root@k8smaster linux-amd64]# helm repo remove aliyun
"aliyun" has been removed from your repositories
```

# 說明：

- 使用 `helm repo remove <倉庫名稱>` 可以刪除已添加的 Helm 倉庫。
- 這樣可以清理不再需要的倉庫條目，保持倉庫列表整潔。

以下是圖片內容的文字辨識及翻譯成繁體中文：

```
[root@k8smaster linux-amd64]# helm search repo weave
NAME          CHART VERSION  APP VERSION  DESCRIPTION
```

```
stable/weave-cloud 0.3.7      1.4.0      Weave Cloud is a add-on...
stable/weave-scope 1.1.10     1.12.0     A Helm chart for the We...
```

```
[root@k8smaster linux-amd64]# helm install ui stable/weave-scope
NAME: ui
LAST DEPLOYED: Mon Sep 7 14:22:47 2020
NAMESPACE: default
STATUS: deployed
```

```
[root@k8smaster linux-amd64]# helm list
NAME      NAMESPACE  REVISION  UPDATED                               STATUS
ui        default    1         2020-09-07 14:22:47.383427149 +0800 CST deployed
```

```
[root@k8smaster linux-amd64]# helm status ui
NAME: ui
LAST DEPLOYED: Mon Sep 7 14:22:47 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

#### 1. 搜索 Helm 倉庫中的 Weave 應用

- 使用命令 `helm search repo weave`，顯示兩個可用的應用：
  - **stable/weave-cloud**：版本 0.3.7，應用版本 1.4.0。
  - **stable/weave-scope**：版本 1.1.10，應用版本 1.12.0。

#### 2. 安裝 Weave Scope

- 使用命令 `helm install ui stable/weave-scope` 將 Weave Scope 安裝到 Kubernetes 中，並命名為 **ui**。

#### 3. 查看已安裝的 Helm 應用列表

- 使用命令 `helm list`，顯示應用 **ui** 已成功部署，位於命名空間 **default**。

#### 4. 查看應用狀態

- 使用命令 `helm status ui`，顯示應用名稱 **ui**，狀態為 **deployed**，部署版本為 **1**。

以下是圖片內容的文字辨識及翻譯成繁體中文：

## 如何自己創建 Chart

### 1. 使用命令創建 Chart

```
helm create chart名稱
```

示例：

```
[root@k8smaster ~]# helm create mychart
Creating mychart
[root@k8smaster mychart]# ls
charts Chart.yaml templates values.yaml
```

- **Chart.yaml**：當前 Chart 屬性配置信息。
- **templates**：編寫 YAML 文件放到這個目錄中。
- **values.yaml**：YAML 文件可以使用全局變量。

### 2. 在 templates 文件夾創建兩個 YAML 文件

- **deployment.yaml**
- **service.yaml**

示例：

```
[root@k8smaster templates]# ls
deployment.yaml service.yaml
```

## 說明：

1. **Helm create** 命令自動生成 Chart 的目錄結構和必要文件，方便進一步編輯。

2. **templates 文件夾** 是放置 Kubernetes 配置文件（如 Deployment、Service 等 YAML 文件）的地方。
3. **values.yaml** 文件允許定義變量，提供動態和靈活的配置支持。

以下是圖片內容的文字辨識及翻譯成繁體中文：

## 3. 安裝 mychart

```
[root@k8smaster ~]# helm install web1 mychart/  
NAME: web1  
LAST DEPLOYED: Mon Sep 7 15:03:17 2020  
NAMESPACE: default  
STATUS: deployed  
REVISION: 1  
TEST SUITE: None
```

### 說明：

1. **命令解釋：**
  - `helm install`：用於安裝 Chart。
  - `web1`：為此次安裝命名為 `web1`。
  - `mychart/`：指定需要安裝的 Chart 目錄。
2. **輸出信息：**
  - **NAME**：安裝的應用名稱，這裡是 `web1`。
  - **LAST DEPLOYED**：最後部署的時間。
  - **NAMESPACE**：部署的命名空間，默認為 `default`。
  - **STATUS**：應用部署狀態，這裡是 `deployed`（已部署）。
  - **REVISION**：部署的版本號，這裡是 `1`。
  - **TEST SUITE**：測試套件信息，這裡是 `None`（沒有測試）。

## 4. 應用升級

```
helm upgrade chart名稱
```

示例：

```
[root@k8smaster ~]# helm upgrade web1 mychart/  
Release "web1" has been upgraded. Happy Helming!
```

### 說明：

1. **命令解釋：**
  - `helm upgrade`：用於升級已部署的 Chart 應用。
  - `web1`：需要升級的應用名稱。
  - `mychart/`：指定升級所基於的 Chart 目錄。
2. **輸出信息：**
  - **Release "web1" has been upgraded**：表示應用 `web1` 已成功升級。

以下是圖片內容的文字辨識及翻譯成繁體中文：

## 實現 YAML 高效複用

- 通過傳遞參數，動態渲染模板，YAML 內容動態傳入參數生成。

```
[root@k8smaster mychart]# ls  
charts Chart.yaml templates values.yaml
```

- 在 Chart 中有 `values.yaml` 文件，定義 YAML 文件全局變量。

## 操作步驟：

1. 在 `values.yaml` 定義變量和值。
2. 在具體 YAML 文件中，獲取定義變量值。

## 說明：

- YAML 文件大體有幾個地方不同的參數：
  - **image**
  - **tag**
  - **label**
  - **port**
  - **replicas**

## 說明：

使用 `values.yaml` 提高了 Chart 的靈活性，可以輕鬆實現模板的複用和參數化，方便不同環境配置。

🕒 修訂版本 #2

★ 由 treeman 建立於 20 🕒@🕒🕒 2025 13:17:43

✎ 由 treeman 更新於 20 🕒@🕒🕒 2025 18:16:16