

Helm 渲染錯誤排查



我的deployment一直出現helm轉yaml失敗，找不到是哪段出錯



你遇到 Helm 轉 YAML 失敗的問題，要找出是哪段出錯，我可以協助你一步步排查。請你提供以下幾項資訊，我可以更精確地幫你定位問題：

請提供的資訊

1. 錯誤訊息全文（含錯誤行數）

Helm 通常會明確指出是哪一行、哪個變數或哪個模板渲染出錯。

2. 使用的 **Helm** 指令

例如：

```
helm install myapp ./mychart -f values.yaml  
helm template ./mychart -f values.yaml
```

3. 你用的 **values.yaml**（至少出錯那段）

4. 出錯的 **template**（例如 `deployment.yaml`）

常見錯誤類型

錯誤類型	說明
<code>`template: ...: executing "...": nil pointer evaluating...`</code>	<code>`.Values.xxx`</code> 沒有定義或拼錯了
<code>`YAML parse error: ... found character that cannot start any token`</code>	缺縮排、冒號後漏空格
<code>`mapping values are not allowed in this context`</code>	<code>`:`</code> 之後沒有加空格，或用 <code>`{{ }}`</code> 時沒有注意縮排與縮格格式
<code>`expected "<string>", got ...`</code>	模板語法錯誤，可能是 <code>`if`</code> 或 <code>`range`</code> 區塊沒 properly 關閉

你也可以這樣排查

1. 試著跑 ``helm template`` 把渲染後 **YAML** 印出來看

```
helm template myapp ./mychart -f values.yaml
```

可以看到是哪一段渲染不成功，這會很直觀地讓你發現錯誤。

2. 用 ``--debug`` 印出更多細節

```
helm install myapp ./mychart -f values.yaml --debug --dry-run
```

3. 單獨檢查某個 **values.yaml** 是否格式錯誤

可以用線上 [YAML Validator](#) 或用 ``yamllint``：

```
yamllint values.yaml
```

範例：縮排錯誤常見陷阱

錯誤的寫法：

```
containers:
- name: myapp
  image: {{ .Values.image.repository }}:{{ .Values.image.tag }}
  ports:
    - containerPort: 80
```

正確的應該是：

```
containers:
- name: myapp
  image: {{ .Values.image.repository }}:{{ .Values.image.tag }}
  ports:
    - containerPort: 80
```

你可以先貼上出錯訊息或 values.yaml / templates 的出錯段落，我來幫你找是哪裡出錯。



YAML parse error:



好的，你遇到的是 **YAML parse error**，這通常表示 **Helm** 成功渲染了模板，但結果不是一份合法的 **YAML** 檔案，所以最常見的原因是：

常見 YAML parse error 原因

問題類型	說明	範例
縮排錯誤	YAML 是靠縮排表示結構，縮排錯會造成解析錯誤	<code>`key: value`</code> 卻寫成 <code>`key: value`</code>
冒號後沒空格	YAML 的 key-value 要有空格	<code>`key:value`</code> ❌ → <code>`key: value`</code> ✅
模板語法產出錯誤	Helm 的 <code>`{{ }}`</code> 渲染出一段不合法的內容	<code>`replicas: {{ .Values.replicas }}`</code> 若 <code>` .Values.replicas`</code> 是字串而非數字會錯
多行字串未加	<code>`或`>`</code>	多行字串要用 <code>`</code>
list 冒號後直接接東西	YAML list 的項目要在 <code>`-`</code> 之後換行或空格	<code>`-name: test`</code> ❌ → <code>`- name: test`</code> ✅

偵錯步驟

1. 執行下面這行命令來取得完整錯誤行與內容：

```
helm template myapp ./mychart -f values.yaml
```

2. 如果還是只出現 ``YAML parse error``，請加上 debug 模式跑：

```
helm install myapp ./mychart -f values.yaml --dry-run --debug
```

它會印出：

- 哪個檔案出錯 (e.g. ``templates/deployment.yaml``)
- 錯在哪一行 (e.g. ``line 12: found character that cannot start any token``)

3. 看錯誤行的上下文，確定 **YAML** 結構正確，例如：

```
env:
  - name: ENV_VAR
    value: {{ .Values.envVar }}
```

假設 `.Values.envVar` 是布林值 `true`，會變成：

```
value: true
```

✅ 合法 YAML。

但如果是未加引號的字串：

```
value: {{ .Values.envVar }}
```

而 `.Values.envVar` 是 `some:thing` → ❌ 非法 YAML，因為 `:` 被誤認為新的 key。

✅ 解法建議

1. 所有字串類型建議加引號：

```
value: "{{ .Values.envVar }}"
```

2. 檢查所有 list 要以 `-` 開頭並對齊縮排：

```
ports:
  - containerPort: 80
```

3. 多行字串使用 `|``：

```
config: |  
  line1  
  line2
```

如果你方便的話

請貼上以下任何一項，我可以直接幫你指出錯誤：

- ``templates/deployment.yaml`` 的一小段（出錯那段上下 5 行）
- ``values.yaml`` 的一小段（你有改動過的）
- ``helm install --dry-run --debug`` 的錯誤輸出

這樣我可以更快幫你定位問題 🍷



現在還不能用helm



了解，如果你目前 **還不能執行 Helm**（例如還沒安裝好或環境無法跑 **Helm**），但你懷疑是 Helm template 內容有錯，那我們可以用「**人工預覽與驗證模板**」的方式來檢查錯誤：

手動檢查 Helm Chart 是否會產出無效 YAML 的方法

你可以提供以下檔案內容，我幫你人工檢查：

1. ``templates/deployment.yaml``（或你懷疑出錯的 template）

2. `values.yaml` (或你自己有修改的 key-value)
3. 你使用的 Helm chart 結構也可以簡單貼一下，例如：

```
mychart/  
├── Chart.yaml  
├── values.yaml  
└── templates/  
    └── deployment.yaml
```

我會幫你 **手動代入變數 + 檢查語法正確性**，模擬 Helm 渲染出 YAML 的樣子，確認哪裡出錯。

如果你願意先自查，也可以照這樣處理

步驟一：找出所有 `{{ .Values.xxx }}` 的地方

→ 確認你在 `values.yaml` 中真的有定義該變數，格式對不對（是否為空、格式錯誤、類型錯）

步驟二：手動代入內容後貼到線上 **YAML Linter 驗證**

→ 例如貼到：<https://www.yamllint.com/>

範例模擬：

deployment.yaml 範例：

```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: {{ .Values.name }}  
spec:  
  replicas: {{ .Values.replicas }}  
  template:  
    spec:
```

```
containers:
  - name: app
    image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
    ports:
      - containerPort: 80
```

values.yaml 範例：

```
name: my-app
replicas: 3
image:
  repository: nginx
  tag: latest
```

→ 渲染後為：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  template:
    spec:
      containers:
        - name: app
          image: "nginx:latest"
          ports:
            - containerPort: 80
```

這樣就是合法 YAML 

下一步

請你貼出以下任一項，我可以立刻幫你分析是哪裡錯：

- `templates/deployment.yaml` 中的一段
- `values.yaml` 中你有加過的設定
- 你想要的變數值（我來幫你手動代入）

這樣即使你現在 Helm 執行不了，我也能幫你找出錯誤。



ubuntu怎麼才能使用helm



要在 Ubuntu 上使用 Helm，你只需要按照下面幾個步驟安裝與驗證，約 3 分鐘內就能完成：

✓ Ubuntu 安裝 Helm 步驟

🏆 方法一：使用官方安裝指令（建議）

1. 打開終端機，執行以下指令：

```
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
```

這會自動下載並安裝最新的 Helm 3。

2. 驗證安裝成功：

```
helm version
```

若看到類似下面輸出表示成功：

```
version.BuildInfo{Version:"v3.14.0", ...}
```

方法二：透過 apt 安裝（通常版本較舊，不推薦）

```
sudo apt update  
sudo apt install helm -y
```

同樣可以用 `helm version` 檢查是否安裝成功。

範例驗證：渲染範本為 YAML

假設你已經在 `mychart/` 目錄中：

```
helm template myrelease ./mychart -f values.yaml
```

這會把渲染後的 YAML 輸出到終端機。

常見問題排查

問題	解法
<code>`helm: command not found`</code>	可能沒加到 <code>`\$PATH`</code> ，請重新開一個終端機或手動加路徑
權限不足安裝	確保你有 <code>`sudo`</code> 權限或在 <code>root</code> 下執行
版本過舊	建議用 <code>curl</code> 官方指令安裝最新版，不建議用 <code>apt</code>

需要我幫你一步步安裝的話，也可以告訴我你目前的狀態（有沒裝 `curl`、版本、環境），我可以即時指導。是否要我直接幫你確認目前系統是否已具備 Helm 執行環境？

你是否喜歡這種個性？   