

GitLab Pipeline Script 語法



我想要理解gitlab pipeline job script->、|用法，為了實驗修改gitlab pipeline
，你能教我嗎，或是你能給更簡單的範例：



當然可以！我們來一步步解釋 GitLab CI/CD 中 `script:` 區塊裡常見的兩種 shell 語法：

你想了解的是：

- `>`（單箭頭）
- `|`（管線符）

GitLab `.gitlab-ci.yml` 的基本背景

在 GitLab CI 裡，每個 job 都會執行一段 shell script，寫法通常長這樣：

```
job_name:
  stage: build
  script:
    - echo "Hello World"
    - ls -la
```

這裡 `script:` 後的每一行指令，其實就像你在 bash 裡打的一樣。

`>` 與 `|` 的差異 (YAML 語法, 非 shell)

其實這兩個符號是 **YAML** 裡的語法, 不是 shell 的喔!

例	結果
行文字, 保留換行符號	`script:`
<code>cript: >` cho A` cho B`</code>	等同於: <code>`bash
echo A echo B`</code>

簡單範例：比較 `|` 與 `>`

 用 `|`：每一行是單獨一行 **shell** 指令

```
my_job:
  stage: test
  script: |
    echo "第一行"
    echo "第二行"
```

會執行：

```
echo "第一行"
echo "第二行"
```

⚠ 用 `>` : 多行合成一行執行

```
my_job:
  stage: test
  script: >
    echo "第一行"
    echo "第二行"
```

實際等同於這樣執行：

```
echo "第一行" echo "第二行"
```

這會報錯，因為 `echo` 不該接另一個 `echo`。

💡 建議

在 `script:` 中幾乎 **一定要用 `|`**，因為你是想寫「一行一行 shell 指令」，保留換行才不會錯。

🧪 更實用的範例

```
build_app:
  stage: build
  image: alpine:latest
  before_script:
    - apk add --no-cache curl
  script: |
    echo "✅ 建立目錄"
    mkdir -p output/

    echo "📄 下載檔案"
```

```
curl -o output/data.txt https://example.com/data.txt
```

```
echo "📦 完成"
```

這就會逐行執行這些 shell 指令，非常清楚！

如果你有一段實驗中使用的 ``.gitlab-ci.yml`` 想優化，我可以幫你一起看 👍
要不要貼一下來分析看看？



```
job_name:
  stage: build
  script:
    - echo "Hello World"
    - ls -la

build_app:
  stage: build
  image: alpine:latest
  before_script:
    - apk add --no-cache curl
  script: |差別
```



你這段 YAML 裡有兩個 job，我來幫你逐一比較這兩個 job 的差別，並著重說明你問的 `script:` 區塊中 `-``（列表）與 `|``（文字區塊）的使用差異：

兩段 GitLab CI job 的比較

✅ 第一段：使用 ` ` 表示「每一行都是獨立的指令」

```
job_name:
  stage: build
  script:
    - echo "Hello World"
    - ls -la
```

這裡的 **script** 是 **YAML** 的 **list**（陣列）格式。每一行用 ` ` 開頭，表示這是「一條 shell 指令」：

等效於：

```
echo "Hello World"
ls -la
```

這是 GitLab 最常見、最推薦的 `script:` 寫法。

✅ 第二段：使用 `|` 表示「整個 script 是一段文字，會當成單一 shell script 執行」

```
build_app:
  stage: build
  image: alpine:latest
  before_script:
    - apk add --no-cache curl
  script: |
    echo "第一行"
    echo "第二行"
```

這裡的 `script:` 是一段 **YAML** 的多行文字（**Literal Block Scalar**），因為用了 `|`。

GitLab 會把這整段文字合併為 **單一 script file** 再執行，所以它也可以跑，但要小心縮排和格式。

等效執行內容會是：

```
echo "第一行"
echo "第二行"
```

✓ ` ` vs `|` 的實務差異與建議

優點	缺點
清楚、容易閱讀、容易除錯	行數多的話略長
所有指令都寫在一起	格式接近你本地 shell script 可用變數、流程控制等

🔍 補充範例（| 寫法支援 if、for 等 bash 控制流）

```
script: |
  echo "開始"
  for i in 1 2 3; do
    echo "第 $i 次"
  done
  echo "結束"
```

這就是 `|` 比 `-` 靈活的地方：你可以寫整段邏輯流程。

✅ 實戰建議

- 如果你只需要單行指令，或單純列出動作 → 用 `-`（推薦）
- 如果你需要流程控制（如 if/for/while）、區塊邏輯 → 可用 `|`

如果你願意，我可以幫你將兩段改成 **完全等價的版本**（例如把 `|` 改成 `-` 寫法，或反過來），你要試試嗎？



```
script: |
  echo "開始"
  for i in 1 2 3; do
    echo "第 $i 次"
  done
  echo "結束"
```

改成 - 寫法



好的！你這段是使用 `|` 的 **文字區塊寫法**，裡面包含了 `for` 迴圈與多行邏輯：

```
script: |
  echo "開始"
  for i in 1 2 3; do
    echo "第 $i 次"
```

```
done
echo "結束"
```

要改成 ``-`` (YAML list) 寫法的話，每個 ``-`` 都要是一行合法的 Shell 指令。但 **bash** 的迴圈通常不能拆成多個獨立命令列出來執行，所以有兩種做法：

✅ 方式一：整個區塊包進單一 ``bash -c`` 中

```
script:
  - echo "開始"
  - bash -c 'for i in 1 2 3; do echo "第 $i 次"; done'
  - echo "結束"
```

這裡的 ``bash -c '...'`` 是一次執行整個區塊，效果跟 ``|`` 寫法一模一樣。

✅ 方式二：直接把邏輯展開（只適合簡單範例）

```
script:
  - echo "開始"
  - echo "第 1 次"
  - echo "第 2 次"
  - echo "第 3 次"
  - echo "結束"
```

這種方式雖然不能跑迴圈，但在簡單情境下也能達到同樣結果。

總結對照

適用情境

可以像寫 bash script 一樣自由排程式邏輯

支援區塊邏輯 想要清楚列出每一步，仍有邏輯需求

不需要邏輯控制，只是列步驟

如果你想要進一步練習 for/if 搭配 GitLab pipeline，想不想我幫你設計一個小小任務流程來練習看看？👨🏻💻