COSCUP x UbuCon Asia 2026

Git 繁體中文翻譯流程治理實戰

Git 繁體中文翻譯流程治理實戰

從個人翻譯專案，到能送進 Git 上游的協作流程。

共筆

即時字幕

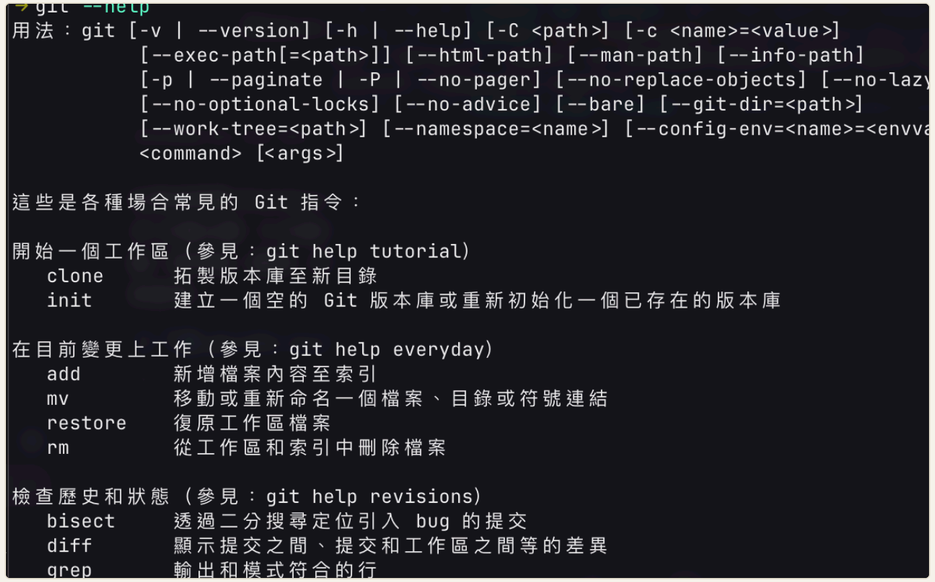
Design By COSCUP 2025 Design Team CC-BY 4.0

開場提問

先猜一下：Git 的繁體中文，哪一年出現？

Git 的繁體中文翻譯，第一次進入正式 release 是哪一年？

- 2017
- 2019
- 2021
- 2013



先看一行就好：Git 自己輸出的訊息。

開場答案

繁體中文在 Git v2.24 正式出現

2019 年 11 月

Git v2.24



繁體中文翻譯正式進入 Git。

第一版的做法

從既有翻譯出發，校對後送進上游

既有 zh_CN.po

↓

繁化姬的進階轉換設定

↓

我逐句檢查，Franklin 校對 Git 情境

↓

確認後，送 patch 到 Git 上游

```
31  msgstr ""
32  "Project-Id-Version: Git\n"
33  "Report-Msgid-Bugs-To: Git Mailing List <git@vger.kernel.org>\n"
34  "POT-Creation-Date: 2026-06-26 12:08+0800\n"
35  "PO-Revision-Date: 2026-06-28 14:21+0800\n"
36  "Last-Translator: Lumynous <lumynou5.tw@gmail.com>\n"
37  "Language-Team: Chinese (Traditional Han script) <https://weblate.slat.org/"
38  "projects/git-po/git-cli/zh_Hant/>\n"
39  "Language: zh_TW\n"
40  "MIME-Version: 1.0\n"
41  "Content-Type: text/plain; charset=UTF-8\n"
42  "Content-Transfer-Encoding: 8bit\n"
43  "Plural-Forms: nplurals=1; plural=0;\n"
44  "X-Generator: Poedit 3.9\n"
45  "X-ZhConverter: 繁化姬 dict-f4bc617e-r910 @ 2019/11/16 20:23:12 | https://"
46  "zhconvert.org\n"
47
48  #: add-interactive.c
49  #, c-format
50  msgid "Huh (%s)?"
```

檔頭留下了 2019 年轉換流程的 credit。

讓更多人能參與

貢獻不該從學會 Git l10n 開始

譯者

翻譯、討論、送出

→

維護流程

同步、整理、提交

直接修改 PO / 1

想改一句話，先走完這三步

01
找到入口

繁體中文團隊的儲存庫

→

02
準備工具

下載並學會 PO 編輯器

→

03
理解狀態

分辨現在可翻的字串

找到入口後，流程還沒有結束

04
找到共識

散在別處的詞彙表

→

05
完成提交

學會提交修改

→

06
等待基準

等維護者同步上游

人一多，直接改 PO 就會卡住

五個人各自貢獻，合併卻不是五次簡單相加

譯者 A

補新的字串

譯者 B

改同一個詞

譯者 C、D、E

也各自送了修改

同一份 PO，不只裝翻譯

```
#: builtin/clone.c:102
"Last-Translator: …"
msgstr "加入未追蹤項目"
```

自動生成的行數與 header 也會變；合併一個 PR，其餘 PR 常得重新解衝突。

翻譯不是一次性工作

Git 每一個版本，都可能帶來新的字串與新的語境

Git v2.x

新增訊息

→

Git v2.x+1

改寫舊訊息

→

Git v2.x+2

再新增訊息

> 版本一直往前，翻譯也得一直跟上。

人力有限，不能只靠一個人

要長期維護，就得讓更多人能安全地一起做

一個人承擔

翻譯、校閱、追版本

工作會隨上游迭代持續累積。

→

真正的風險

很快 burnout

社群很小，停下來就沒有下一個人接手。

→

需要的改變

降低參與門檻

讓人可以先貢獻，再逐步理解流程。

讓不熟悉 Git 流程的人，也能先開始翻譯



原文、翻譯輸入框、提交者與時間，都在同一個工作空間。

打開 Weblate

↓

找到字串

↓

翻譯、討論、送出

不用先安裝 PO 編輯器，也不用直接修改同一份檔案；入口先縮短，再把後面的流程接住。

Weblate 的協作機制

把詞彙表、討論與審核放進同一個流程

協作記憶

共筆詞彙表

將常用譯法與定義集中維護。

→

協作過程

翻譯討論

提案、理由與上下文都留在字串旁。

→

最後責任

審核員 approval

由熟悉 Git 情境的人做最終裁決。

新的摩擦出現了

Weblate 的 commit，不能直接當成 git-l10n 的提交

Weblate 的優點

每一次貢獻都留下來

細碎的 commit 很適合協作與回看過程。

→

git-l10n 的要求

一份可審閱的正式提交

要能 squash、留下共同作者，並正確處理 Signed-off-by。

→

中間流程

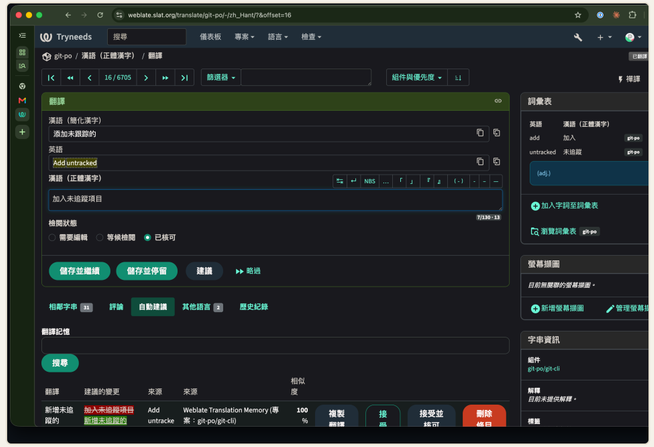
維護者整理

> 降低貢獻門檻，不能用犧牲上游規範來交換。

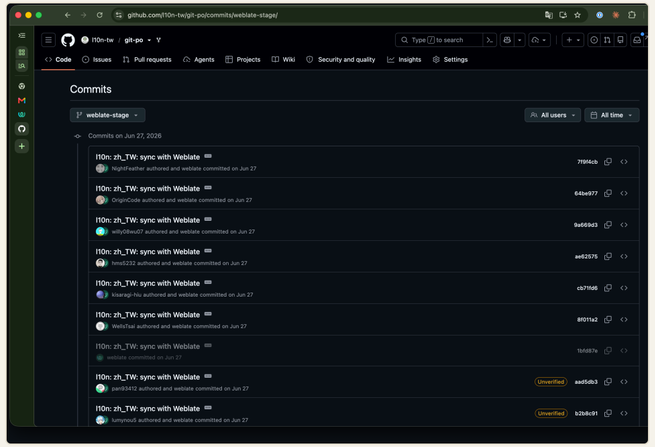
真實案例

Weblate 到正式提交

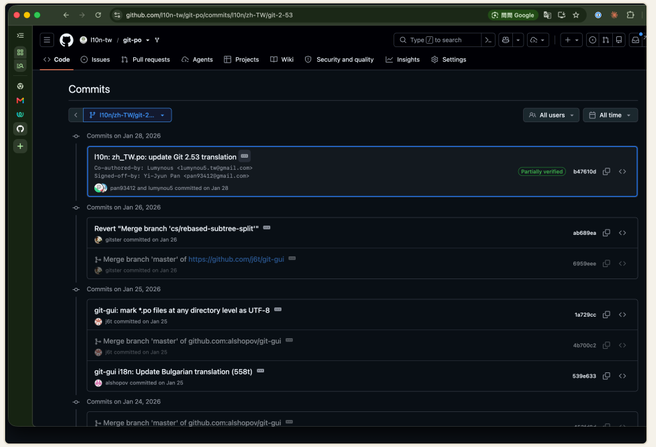
① Weblate 貢獻



② weblate-stage



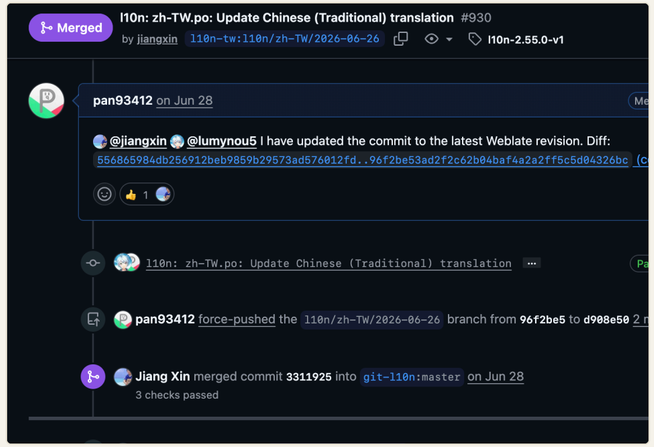
③ 正式 commit



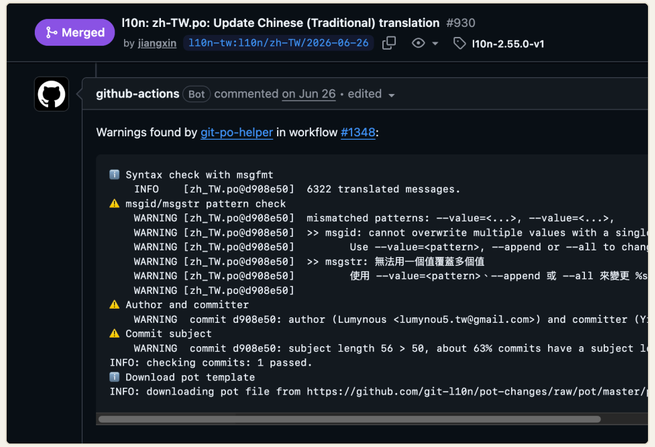
真實案例

正式提交到 Git release

④ 協調者 PR



⑤ git-po-helper



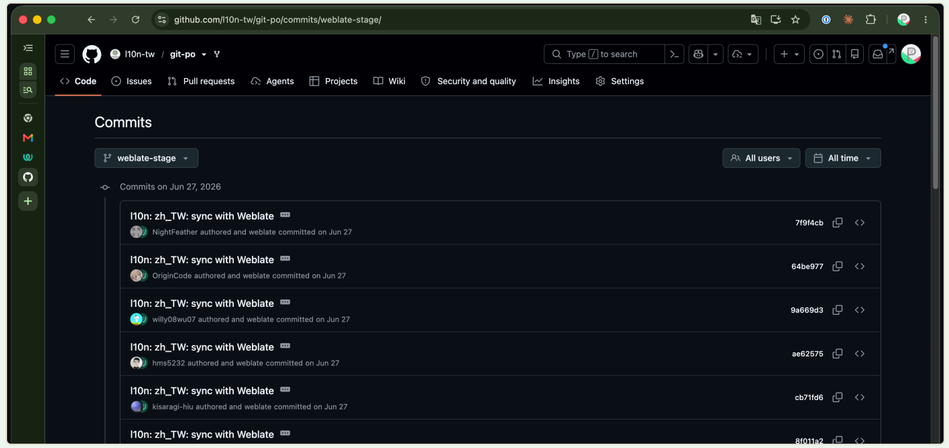
⑥ Git release



先看一次實際流程，再回來看它的結構

同一批變更，需要兩種歷史

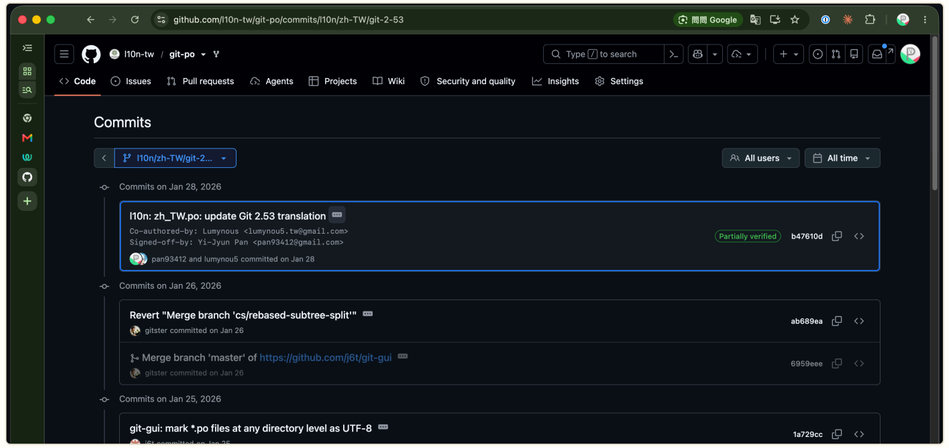
協作歷史 · weblate-stage



細碎修改、討論與後續調整，都留在這裡。

它是工作區，不必先把歷史整理得漂亮。

正式歷史 · git-l10n



送上游的是整理過、可審閱的正式結果。

維護者整理提交、保留作者，並為它負責。

把兩種歷史接起來

三個位置，三種工作

同步基準

weblate-base

跟上游同步，讓 Weblate 從正確狀態開始工作。

→

協作過程

weblate-stage

保存大家怎麼一起改，不必先追求漂亮歷史。

→

正式提交

l10n/zh_TW/yy-mm-dd

維護者整理、保留作者、跑檢查，再送協調者 PR。

正式提交不是把 Weblate 的修改直接搬過來

維護者送上游前，要過三關

01

翻譯內容

比對原文、確認訊息情境，也查有沒有漏翻。

02

貢獻歸屬

squash 成提交，但用 `Co-authored-by` 保留每位作者。

03

送上游

檢查格式與 PO lint，再用 `Signed-off-by` 承擔責任。

協調者把版本窗口變成一輪可交付的工作

協調者其實是在追著版本跑



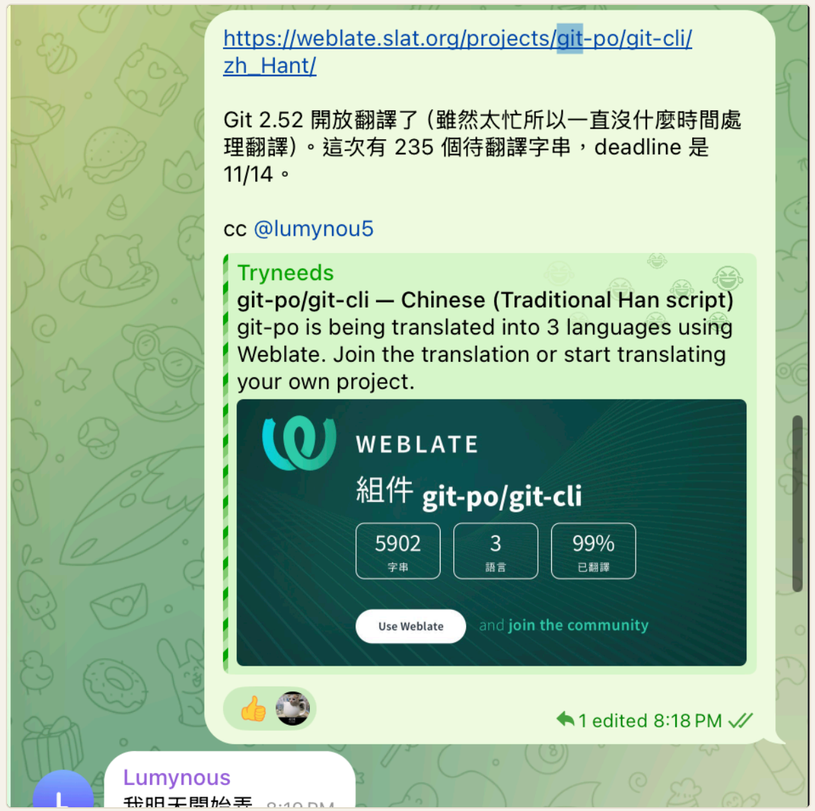
一輪窗口：125 條字串，約兩到三週。

窗口裡要記得的事

1. 更新 POT，開啟翻譯窗口
2. 邀請譯者，整理 weblate-stage
3. 追候選版本與新的字串
4. 重新 msgmerge，更新基準

協調者不只收 PR

也要主動邀請大家進來翻譯



開窗口時：說明待翻字串與 deadline。



完成後：感謝貢獻，並邀請繼續校對與修正。

工具處理機械工作，人處理責任

自動化是替維護者省力，不是替人做決策

值得交給腳本

- 同步 POT 和 PO
- 建立工作分支、收集貢獻者資料
- 產生 squash commit 的範本
- 執行格式與 lint 檢查

仍由維護者判斷

- 翻譯在 Git 情境裡是否合理
- 哪些變更應該一起送出
- 共同作者與 Signed-off-by 是否正確
- 這一輪能否負責任地送上游

回到治理與人

Weblate 沒有讓角色消失

翻譯者

不用再追著 Git 的流程跑，先把翻譯做出來。

→

維護者

同步上游、追版本、整理歷史、保留歸屬、處理檢查。

→

協調者

開窗口、邀請人、收 PR，讓協作接得起來。

> 入口變簡單了，治理工作還在。



把這條路再縮短一點

有人想幫忙翻譯 → Weblate → 維護者整理 → Git release

> **Git 翻譯會跟著版本一直維護下去。**

> 入口變簡單了，治理工作還在。

謝謝大家。接下來，讓我們把 POT 更新、提交整理和檢查再做得省力一點。

Design By COSCUP 2025 Design Team CC-BY 4.0



留五分鐘，聊聊你遇到的流程

Q&A

Q&A

Git 翻譯、Weblate、維護流程，都可以問。

Design By COSCUP 2025 Design Team CC-BY 4.0