GIT REPOS 實務管理
以 AZURE REPOS 為例

Presented By
Duran Hsieh
Site Reliability Engineer

Microsoft MVP Most Valuable Professional

STUDY4.TW 為學習而生

GDG Taichung

ABOUT ME

Duran Hsieh (謝政廷)

專長於開發測試、效能調教、DevOps 相關技術，過去曾擔任技術顧問，提供國內外企業技術諮詢與數位轉型。目前同時為 Study4TW 社群核心成員、微軟最有價值專家、 Google Developer Group Taichung 共同創辦人，曾積極參與技術社群與經營技術部落格，曾獲得三次 IT 邦幫忙鐵人賽佳作與撰寫 <動手學 GitHub 現代人不能不知道的協同合作平台>書籍。

- Duran 技術冶煉廠 : https://dog0416.blogspot.com/
- Duran 速寫筆記: https://note.duranhsieh.com/

GIT 實務管理

- Git ignore
- Repository Management
- Branch
- Branch Strategy
- Branch Permission

GIT IGNORE

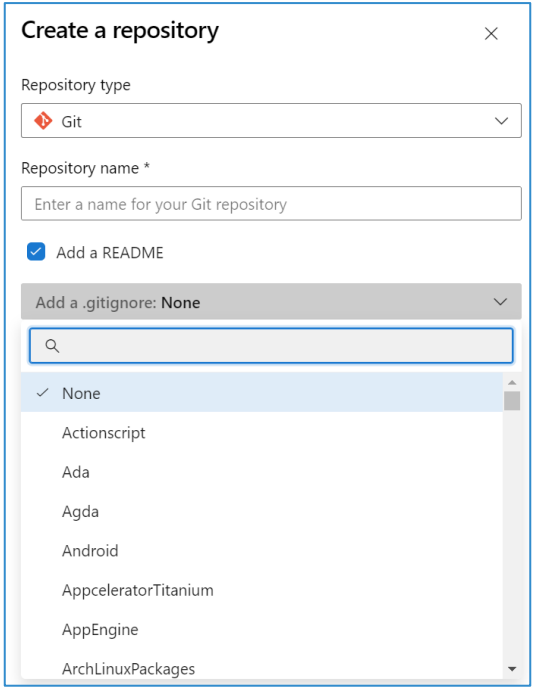
- 避免工作和存儲庫中的文件混亂示
  - Binary 檔案
- 頁面上忽略的文件既不提交也不推送
- 在 Repo 內預設分支中共享 .gitignore，以便您和您的團隊可以隨時更新

```
# ignore a single file
mycode.class

# ignore an entire directory
/mydebugdir/

# ignore a file type
*.json

# add an exception (using !) to the preceding rule to track a specific file
!package.json
```

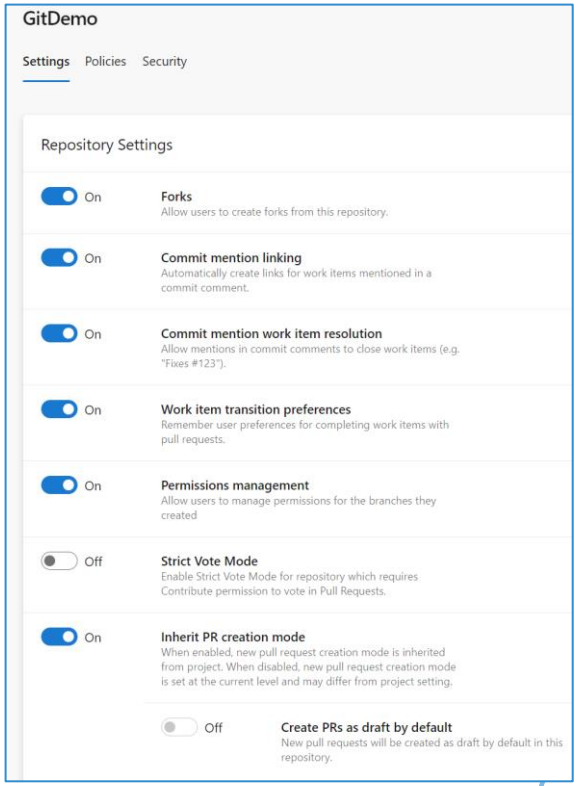


盡可能不要提交BINARY檔案
尤其是大容量類型

Git 可以追蹤文字檔案內容，但不適用於 Binary
容易造成切換速度慢、不容易完整刪除問題

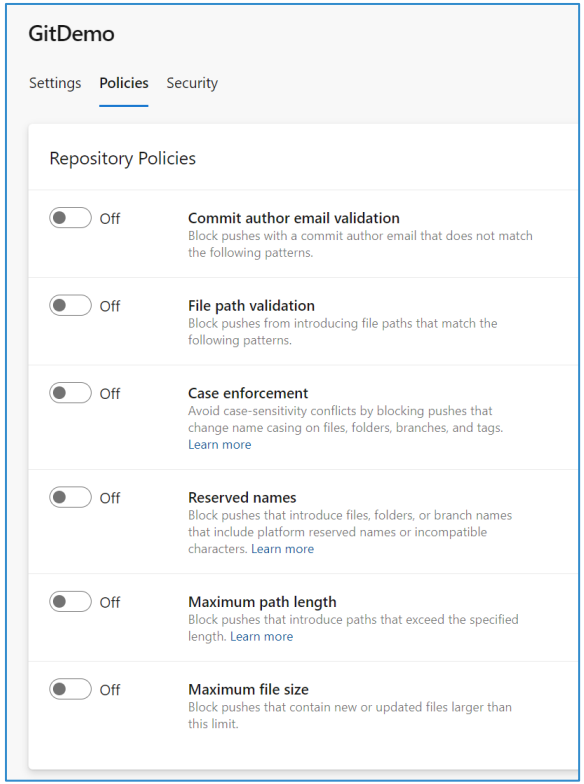
REPOSITORY MANAGEMENT

- **Repository 設定**
  - 對於整個 Repo 進行設定
  - (實用) 工作項目與 Git Commit 連結
    - 在工作項目內快速的了解歷程與程式修改紀錄
    - 從 Git 的歷史紀錄中，了解 Commit 與哪個工作項目有關
    - 更新版本時，管理人員能清楚了解那些工作項目 (Bug 修正、新功能) 已經完成



REPOSITORY MANAGEMENT

- **Repository 管理策略**
  - 可以依據團隊需求，進行相關限制
    - 作者信箱
    - 檔案路徑驗證
    - 大小寫驗證
    - 保留名稱
    - 最長路徑長度
    - 最長檔案限制

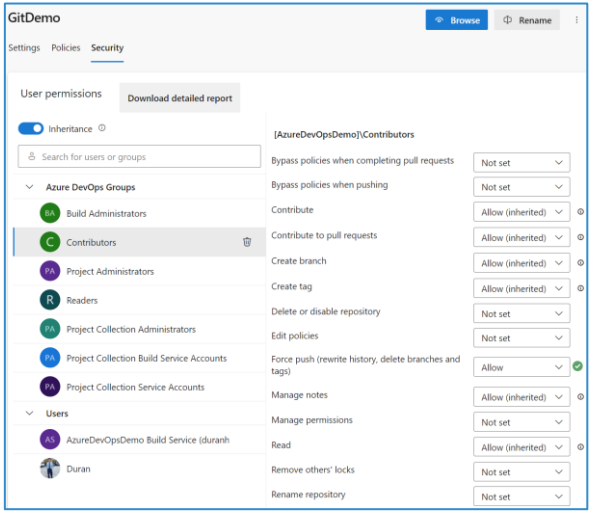


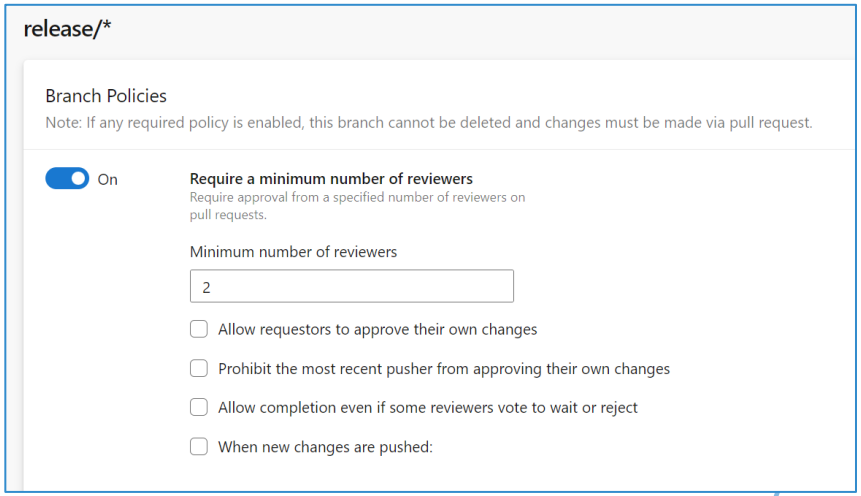
設定禁止刪除特定 BRANCH
應該為最優先事項

與刪除資料庫到跑路有異曲同工之妙

REPOSITORY MANAGEMENT

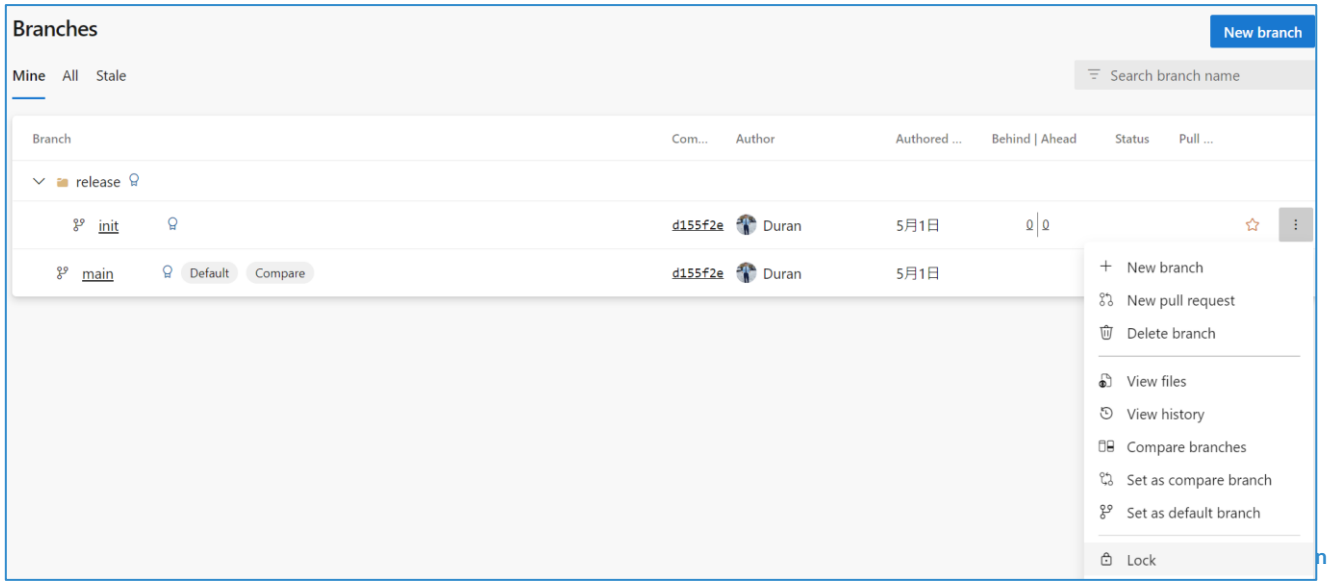
- 如何設定禁止刪除 Branch
  - 設定 Branch Policy (特定 branch)
  - 禁止 Force Push
  - 恢復刪除 (Demo)





REPOSITORY MANAGEMENT

- Lock branch
  - 防止衝突 (特定情境)
  - 為什麼在 Git 很少 Lock branch (說明)

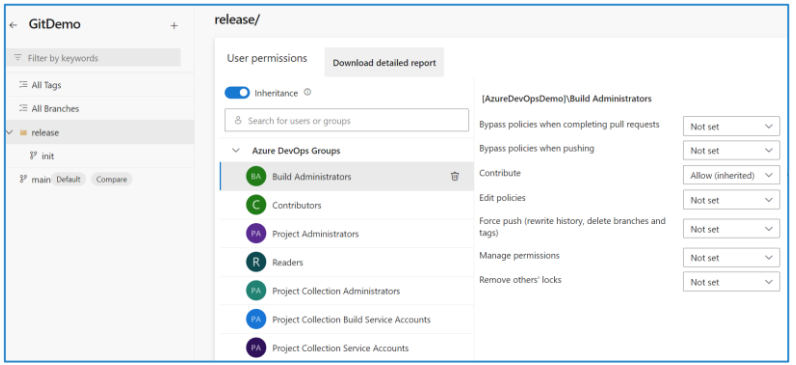


BRANCH

- **對所有功能和錯誤修復使用分支**
  - 因為 Git 開分支成本非常低廉
  - 按照慣例命名您的功能分支
  - 使用 feature flag 管理長時間運作的分支 (連結)
  - 可以透過分支策略進行資料夾管理 (Demo)
    - 限制只有管理者可以建立 Release 分支

```
function createToggleRouter(featureConfig){
  return {
    setFeature(featureName,isEnabled){
      featureConfig[featureName] = isEnabled;
    },
    featureIsEnabled(featureName){
      return featureConfig[featureName];
    }
  };
}
```

users/username/description
users/username/workitem
bugfix/description
feature/feature-name
feature/feature-area/feature-name
hotfix/description



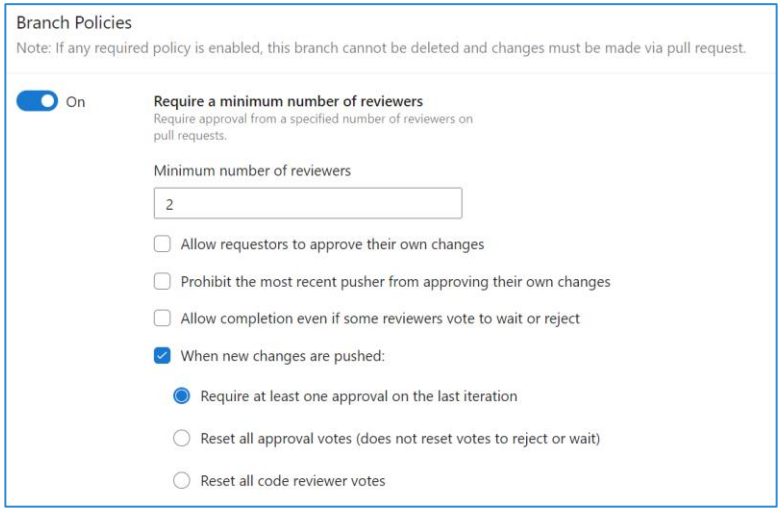
BRANCHING STRATEGY

- **Git 工作流程的重要組成部分，使您能夠：**
  - 將正在進行的工作與主要分支中已完成的工作分隔開來
  - 在合併前進行建置(Build)
  - 限制特定成員可以為特定分支進行提交
  - 每次程式碼更改自動加入合適的審閱者
  - 所需的審閱者一起進行最佳實踐

Policy	Default	Description
Require a minimum number of reviewers	Off	Require approval from a specified number of reviewers on pull requests.
Check for linked work items	Off	Encourage traceability by checking for linked work items on pull requests.
Check for comment resolution	Off	Check to see that all comments have been resolved on pull requests.
Limit merge types	Off	Control branch history by limiting the available types of merge when pull requests are completed.
Add Build Validation policies	Off	Add one or more policies to validate code by pre-merging and building pull request changes. Can also enable or disable policies.
Add Status Check policies	Off	Add one or more policies to require other services to post successful status to complete pull requests. Can also enable or disable policies.
Automatically included reviewers	Off	Add one or more policies to designate code reviewers to automatically include when pull requests change certain areas of code. Can also enable or disable policies.

BRANCHING STRATEGY

- 使用 **Pull request** 將 Feature branch 合併到主要分支
  - 提高品質最佳方法
  - 避免沒有 PR 合併回主要分支
  - 研究顯示最佳審核者為 2 人
  - 避免大量 PR 指派相同的審核者 (團隊共享)



BRANCHING STRATEGY

- **使用 Branch policy 管理 Pull Request，以維持高品質程式碼**
  - 必須要 Pull Request
    - 防止直接推送到主分支，並確保對提議的更改進行討論
  - 建立 Pull Require 時自動加入 Reviewer。
    - 添加的團隊成員審查代碼並對 Pull Request 中的更改發表評論
  - 需要成功構建才能完成 Pull Request
    - 合併到主要分支的程式碼應該乾淨且成功 Build。

BRANCHING STRATEGY

- **管理發佈 (Manage Release)**
  - 確保修改出現在功能分支與主要分支 (有機會做 cherry-pick)
  - 另一種方式：始終在主要分支修改
    - Release Tag 與 Release branch 則一即可
    - 開發人員經常忘記上 Tag

在某些情況下，您需要繞過策略要求
即使不滿足分支策略

對使用者或組群組授予所需的權限是必要的

QUESTION & ANSWERS

THANK YOU FOR WATCHING

若您對 SRE 相關工作有興趣
歡迎找我討論