Git ????

?? rev-parse

```
# Getting the top-level directory
git rev-parse --show-toplevel

# Find your way home
git rev-parse --show-cdup

## Current location
# 判斷是否在專案目錄 <git-repo>/.git 底下
git rev-parse --is-inside-git-dir
# 判斷是否在專案目錄 <git-repo> 底下 (不包含 .git 目錄下)
git rev-parse --is-inside-work-tree
```

??????

- How to Fix Merge Conflicts in Git

Git credential cache

This command caches credentials for use by future Git programs. The stored credentials are kept in memory of the cache-daemon process (instead of being written to a file) and are forgotten after a configurable timeout.

```
git config credential.helper cache
git config credential.helper 'cache --timeout=3600'
```

Shallow Clone (???)

???? shallow clone:

- You don’t need the full commit history
- You want faster setup for CI/CD pipelines
- You’re working with large repositories

```
# Shallow clone
# Limit the clone to 100 commits before the current repository HEAD.
git clone --depth=100 <Repository_URL>

# Shallow Cloning Only a Single Branch
git clone --depth 100 <repository_URL> --single-branch --branch=<branch-name>

# Clone commits since a specific date
git clone --shallow-since="2024-01-01" <repository_URL>

# Skip large file downloads
git clone --filter=blob:none <Repository_URL>

# Unshallow a shallow clone
git fetch --unshallow
```

Common issues with shallow clones

- git merge-base
- git rebase
- git bisect

Fix: Run `git fetch --unshallow` or avoid shallow clone in those contexts.

Efficient Fetching Strategies for Shallow Clones

```
# Fetch with Limited Depth
git fetch origin --depth=5

# Fetch by Date
git fetch origin --shallow-since="2024-01-01"

# Fetch by Excluding Old Commits
git fetch origin --shallow-exclude=a1b2c3d

# Fetch Specific Branches/Tags
git fetch origin main --depth=10
git fetch origin refs/tags/v2.1.0 --depth=1
```

Revision #7
Created 2023-03-27 09:12:11 CST by A-Lang (Admin)

Updated 2026-01-12 16:31:13 CST by A-Lang (Admin)