

So verwenden Sie mehrere Git-Konfigurationen auf einem Computer



Möglicherweise fällt es Ihnen schwer, viele Katzen zu verwalten, aber wenn es um Git-Profile geht, können Sie etwas tun.

Kommen wir gleich zur Lösung – Die Antwort liegt in der `.gitconfig` Datei. Dies ist der Ausgangspunkt für Git, um zu ermitteln, welche Konfigurationen verwendet werden müssen.

Die Idee besteht darin, die Repos auf Ihrem Computer in mehrere Verzeichnisse aufzuteilen, indem Sie die gewünschten Profile trennen und dann eine `.gitconfig` Datei pro Profil definieren.

Schritt 1 → separate Verzeichnisse für Repos erstellen

Organisieren Sie die Projekte, an denen Sie arbeiten, in separaten Ordnern nach den Profilen, mit denen Sie arbeiten möchten.

Nehmen wir zum Beispiel an, Sie arbeiten mit zwei Git-Profilen. Dies ist für die meisten von uns ein häufiger Anwendungsfall:

- WORK → für arbeitsbezogene Projekte
- PERSONAL → für Open Source und Nebenprojekte

Schritt 2 → Erstellen Sie eine globale Git-Konfiguration

Erstellen Sie die globale `.gitconfig` Datei in Ihrem Home-Verzeichnis, falls sie noch nicht vorhanden ist. Fügen Sie dann alle Profilverzeichnisse als Eintrag hinzu, wie im Beispiel unten.

Die Funktionsweise ist sehr intuitiv – wenn der Verzeichnispfad, in dem Sie das Git-Verzeichnis erstellt haben, mit einem der Pfade in Übereinstimmung `includeIF`, verwendet Git diese bestimmte Profilkonfigurationsdatei. Andernfalls wird die Standardkonfiguration verwendet.

```
$HOME/.gitconfig

[includeIf "gitdir:~/personal/"]
  path = ~/.gitconfig-personal
[includeIf "gitdir:~/work/"]
  path = ~/.gitconfig-work
```

Schritt 3 → individuelle Git-Konfigurationen für Profile erstellen

Falls Sie es noch nicht bemerkt haben: Wir haben gerade die Dateien `.gitconfig-personal` und `.gitconfig-work` in der globalen `.gitconfig` Datei erwähnt, sie aber noch nicht erstellt. Diese einzelnen Dateien können alle erforderlichen Anpassungen enthalten, vom Benutzernamen und der E-Mail-Adresse bis hin zu Commit-Hooks.

```
$HOME/.gitconfig-work

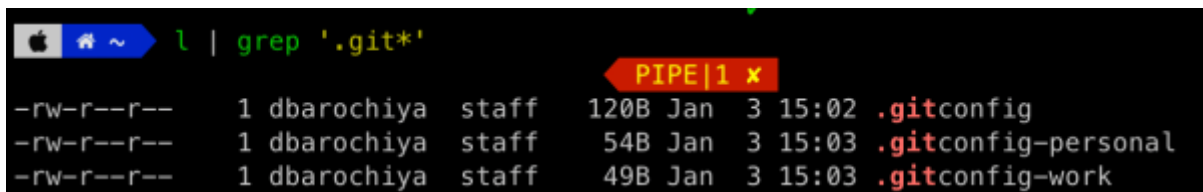
[user]
  name = work_user
  email = work_email
```

```
$HOME/.gitconfig-personal

[user]
  name = personal_user
  email = personal_email
```

Lassen Sie uns überprüfen

Wir sind bereit! Jetzt haben Sie drei Git-Dateien in Ihrem Home-Verzeichnis.



```
l | grep '.git*'
-rw-r--r--  1 dbarochiya  staff  120B Jan  3 15:02 .gitconfig
-rw-r--r--  1 dbarochiya  staff   54B Jan  3 15:03 .gitconfig-personal
-rw-r--r--  1 dbarochiya  staff   49B Jan  3 15:03 .gitconfig-work
```

Jetzt erstellen und initiieren wir ein neues Git-Repo im Arbeits- und Privatverzeichnis und überprüfen die Konfigurationen.

```
$ cd ~/work
```

```
$ mkdir work-test-repo
$ cd work-test-repo
$ git init
*Initialized empty Git repository in /Users/dbarochiya/work/work-
test-repo/.git/*
$ git config -l
credential.helper=osxkeychain
includeif.gitdir:~/personal/.path=~/.gitconfig-personal
includeif.gitdir:~/work/.path=~/.gitconfig-work
**user.name=working_me
user.email = work@work.com**
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true*
```

```
$ cd ~/personal
$ mkdir personal-test-repo
$ git init
*Initialized empty Git repository in /Users/dbarochiya/personal/.git/*
$ git config -l
credential.helper=osxkeychain
includeif.gitdir:~/personal/.path=~/.gitconfig-personal
**user.name=me_personal
user.email=personal@personal.com**
includeif.gitdir:~/work/.path=~/.gitconfig-work
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true*
```

Voilà – wie Sie sehen, sind E-Mail und Benutzername in beiden Fällen unterschiedlich. Abhängig vom Pfad des Git-Repositorys ist es in der Lage, die benutzerdefinierten .gitconfig Dateien zu verwenden.

From:
<https://blog.cooltux.net/> - TuxNet DokuWiki

Permanent link:
https://blog.cooltux.net/doku.php?id=it-wiki:git:mehrere_git_konfigurationen_ein_computer

Last update: 2024/04/16 04:13

