git config

이 문서에서는 git config 명령에 대해 자세히 살펴볼 것입니다. 저장소 설정 페이지에서 git config 사용법에 대해 간략히 설명했습니다. git config 명령은 전역 또는 로컬 프로젝트 수준에서 Git 구성 값을 설정하는 데 사용되는 편리한 기능입니다. 이러한 구성 수준은 .gitconfig 텍스트 파일에 해당합니다. git config를 실행하면 설정 텍스트 파일이 수정됩니다. 이메일, 사용자 이름 및 에디터와 같은 일반적인 구성 설정을 다루겠습니다. 우리는 자주 사용하는 Git 작업에 대한 바로 가기를 만들 수있는 Git 별칭에 대해 설명합니다. git config와 다양한 Git 구성 설정에 익숙해지면 강력하고 사용자 정의 된 Git 워크 플로우를 만들 수 있습니다.

사용법

git config의 가장 기본적인 사용 예는 구성 이름을 사용하여 그 이름에 설정된 값을 표시하는 것입니다. 구성 이름은 계층 구조에 따라 '섹션'과 '키'로 구성된 점으로 구분 된 문자열입니다. 예 : user.email

```
git config user.email
```

이 예에서 전자 메일은 사용자 구성 블록의 하위 속성입니다. Git이 로컬에서 작성한 커밋과 연결할 전자 메일 주소를 반환합니다.

git config levels and files

git config 사용법을 더 자세히 설명하기 전에 잠시 시간을내어 구성 단계를 살펴 보겠습니다. git config 명령은 작동 할 구성 수준을 지정하는 인수를 허용 할 수 있습니다. 다음 구성 단계를 사용할 수 있습니다.

- **`--local`**

기본적으로 git config는 설정 옵션이 전달되지 않으면 로컬 레벨에 기록합니다. 로컬 레벨 설정은 git config가 호출 된 컨텍스트 저장소에 적용됩니다. 로컬 설정 값은 repo의 .git 디렉토리에있는 파일에 저장됩니다 : .git / config

- **`--global`**

전역 레벨 구성은 사용자별로 다르기 때문에 운영 체제 사용자에게 적용됩니다. 전역 구성 값은 사용자의 홈 디렉토리에있는 파일에 저장됩니다. ~ /.gitconfig는 유닉스 시스템에서는 C : ₩ Users ₩ <username> ₩. gitconfig in windows

- **`--system`**

시스템 레벨 구성은 전체 시스템에 적용됩니다. 여기에는 운영 체제의 모든 사용자와 모든 repos가 포함됩니다. 시스템 레벨 설정 파일은 시스템 루트 경로 밖의 gitconfig 파일에 있습니다. 유닉스 시스템에서는 $ (prefix) / etc / gitconfig. Windows에서이 파일은 Windows XP에서는 C : ₩ Documents and Settings ₩ All Users ₩ Application Data ₩ Git ₩ config, Windows Vista에서는 C : ₩ ProgramData ₩ Git ₩ config에 있습니다.

따라서 구성 수준의 우선 순위는 로컬, 글로벌, 시스템입니다. 이것은 구성 값을 찾을 때 힘내는 로컬 레벨에서 시작하여 시스템 레벨까지 버블 링된다는 것을 의미합니다.

변수 설정

git config에 대해 이미 알고있는 것에 대해 살펴보면서 값을 작성하는 예제를 살펴 보겠습니다.

```
git config --global user.email "developer@curvc.com"
```

이 예에서는 값 developer@curvc.com을 구성 이름 user.email에 씁니다. 이 값이 현재 운영 체제 사용자에 대해 설정되도록 --global 플래그를 사용합니다.

git config editor - core.editor

많은 Git 명령은 추가 입력을 요구하는 텍스트 편집기를 시작합니다. git config의 가장 일반적인 사용 사례 중 하나는 힘쓴 편집자가 사용해야하는 구성입니다. 다음은 인기있는 편집기와 일치하는 git config 명령의 표입니다.

Editor	config command
Atom	`~ git config --global core.editor "atom --wait"~`
emacs	`~ git config --global core.editor "emacs"~`
nano	`~ git config --global core.editor "nano -w"~`
vim	`~ git config --global core.editor "vim"~`
Sublime Text (Mac)	`~ git config --global core.editor "subl -n -w"~`
Sublime Text (Win, 32-bit install)	`~ git config --global core.editor "'c:/program files (x86)/sublime text 3/sublimetext.exe' -w"~`
Sublime Text (Win, 64-bit install)	`~ git config --global core.editor "'c:/program files/sublime text 3/sublimetext.exe' -w"~`
Textmate	`~ git config --global core.editor "mate -w"~`

Merge tools

병합이 충돌하는 경우 Git은 "병합 도구"를 실행합니다. 기본적으로 Git은 일반적인 유닉스 diff 프로그램의 내부 구현을 사용한다. 내부 Git diff는 최소한의 병합 충돌 뷰어입니다. 대신 사용할 수있는 외부 타사 병합 충돌 해결 방법이 많이 있습니다. 다양한 병합 도구와 구성에 대한 개요는 Git과의 충돌을 해결하기위한 팁과 도구에 대한 가이드를 참조하십시오.

```
git config --global merge.tool kdiff3
```

Colored outputs

Git은 신속하게 힘내 출력을 읽는 데 도움이 색깔 터미널 출력을 지원합니다. Git 출력을 맞춤 설정하여 맞춤 설정된 색상 테마를 사용할 수 있습니다. git config 명령은이 색상 값을 설정하는 데 사용됩니다.

`color.ui`

이것은 Git 색상의 마스터 변수입니다. false로 설정하면 Git의 컬러 터미널 출력이 모두 비활성화됩니다.

```
git config --global color.ui false
```

기본적으로 color.ui는 auto로 설정되어 즉각적인 터미널 출력 스트림에 색상을 적용합니다. 자동 설정은 출력 스트림이 파일로 리디렉션되거나 다른 프로세스로 파이프되는 경우 색상 코드 출력을 생략합니다.

color.ui 값을 always로 설정하면 출력 스트림을 파일이나 파이프로 리디렉션 할 때 색상 코드 출력도 적용됩니다. 수신 파이프가 색으로 구분 된 입력을 기대하지 않을 수 있으므로 의도하지 않은 문제가 발생할 수 있습니다.

Git color values

color.ui 외에도 다른 많은 세분화 된 색상 설정이 있습니다. color.ui와 마찬가지로이 색상 설정은 모두 false, auto 또는 always로 설정할 수 있습니다. 이 색상 설정에는 특정 색상 값 세트가있을 수도 있습니다. 지원되는 색상 값의 예는 다음과 같습니다.

- normal
- black
- red
- green
- yellow
- blue
- magenta
- cyan
- white

색상은 # ff0000과 같은 16 진수 색상 코드 또는 터미널에서 지원하는 경우 ANSI 256 색상 값으로 지정할 수도 있습니다.

Git color configuration settings

1. `color.branch`

    - Git 분기 명령의 출력 색상을 구성합니다.

2. `color.branch.<slot>`

이 값은 Git 브랜치 출력에도 적용됩니다. <slot>은 다음 중 하나입니다.

- current : 현재 브랜치
- local : 로컬 브랜치
- remote : 원격 지점 참조 ref/remotes
- upstream : 업스트림 tracking branch
- plain : 다른 ref

3. `color.diff`

    - git diff, git log 및 git show 출력에 색상 적용

4. `color.diff.<slot>`

    - color.diff 아래에 <slot> 값을 설정하면 패치의 어느 부분에 특정 색상을 사용할지 알려줍니다.
        - context : diff의 문맥 텍스트. 힘내 맥락은 변화를 강조하는 diff 또는 패치에 표시된 텍스트 내용의 줄입니다.
        - plain : 문맥의 동의어
        - meta : diff의 메타 정보에 색상을 적용합니다.
        - frag : 색을 "hunk header"또는 "hunk header의 기능"에 적용합니다.
        - old : diff에서 제거 된 선에 색상을 적용합니다.
        - new : diff의 추가 된 라인을 색칠합니다.
        - 커밋 (commit) : diff에서 헤더 색상을 커밋합니다.
        - 공백 : diff에있는 공백 오류의 색상을 설정합니다.

5. `color.decorate.<slot>`

    - git log --decorate 출력의 색상을 사용자 정의하십시오. 지원되는 <slot> 값은 branch, remoteBranch, tag, stash 또는 HEAD입니다. 로컬 브랜치, 원격 추적 분기, 태그, 숨겨진 변경 사항 및 HEAD에 각각 적용 가능합니다.

6. `color.grep`

    - git grep의 출력에 색상을 적용합니다.

7. `color.grep.<slot>`

    - git grep에도 적용됩니다. <slot> 변수는 grep 출력의 어느 부분에 색상을 적용 할지를 지정합니다.
        - 컨텍스트 : 컨텍스트 라인에서 일치하지 않는 텍스트
        - filename : 파일 이름 접두사
        - 함수 : 함수 이름 행
        - linenumber : 줄 번호 접두사
        - 일치 : 일치하는 텍스트
        - matchContext : 컨텍스트 라인에서 일치하는 텍스트
        - matchSelected : 선택한 줄의 일치하는 텍스트
        - selected : 선택한 줄에서 일치하지 않는 텍스트
        - 구분 기호 : 한 줄에있는 필드 사이의 구분 기호 (:, - 및 =) 및 헌크 (-) 사이의 구분 기호

8. color.interactive

    - 이 변수는 대화 형 프롬프트 및 표시에 색상을 적용합니다. 예는 git add --interactive 및 git clean입니다. --interactive

9. color.interactive.<slot>

    - 더 구체적인 "대화 형 출력"을 목표로 지정할 수있는 <slot> 변수가 있습니다. 사용 가능한 <slot> 값은 prompt, header, help, error입니다. 각각은 해당 대화식 출력에서 작동합니다.

10. `color.pager`

    - 호출기가 사용 중일 때 유색 출력을 활성화 또는 비활성화합니다.

11. `color.showBranch`

    - git show branch 명령의 컬러 출력을 활성화 또는 비활성화합니다.

12. color.status

    - Git 상태에 대한 색상 출력을 활성화 또는 비활성화하는 부울 값입니다.

13. color.status.<slot>

    - 지정된 자식 상태 요소에 대한 맞춤 색상을 지정하는 데 사용됩니다. <slot>은 다음 값을 지원합니다.
        - header

        - 상태 영역의 헤더 텍스트를 대상으로합니다.
    - added or updated
        - 추가되었지만 커밋되지 않은 두 대상 파일
    - changed
        - 수정되었지만 git 인덱스에 추가되지 않은 파일을 대상으로합니다.
    - untracked
        - Git에 의해 추적되지 않는 파일을 대상으로합니다.
    - branch
        - 현재 분기에 색상을 적용합니다.
    - nobranch
        - "분기 없음"경고가 표시된 색상
    - unmerged
        - 변경 사항이 병합되지 않은 색상 파일

Aliases

운영 체제 명령 줄에서 별칭 개념을 잘 알고있을 수 있습니다. 그렇지 않은 경우, 더 긴 명령이나 결합 된 명령으로 확장 할 명령을 정의하는 사용자 지정 바로 가기입니다. 별칭을 사용하면 자주 사용하는 명령을 입력하는 데 드는 시간과 에너지 비용을 절약 할 수 있습니다. 힘내 자체 별칭 시스템을 제공합니다. Git 별칭의 일반적인 사용 예는 커밋 명령을 줄이는 것입니다. 힘내 별칭은 힘내 설정 파일에 저장되어있다. 즉, git config 명령을 사용하여 별칭을 구성 할 수 있습니다.

```
git config --global alias.ci commit
```

이 예는 git commit 명령에 대한 ci 별명을 작성합니다. 그런 다음 git ci를 실행하여 git commit을 호출 할 수 있습니다. 별칭은 다른 별칭을 참조하여 강력한 콤보를 만들 수도 있습니다.

```
git config --global alias.amend ci --amend
```

이 예제는 ci 별명을 --amend flag를 사용하는 새 별명으로 작성하는 별명 수정을 작성합니다.

Formatting & whitespace

Git은 git diff를 사용할 때 공백 문제를 강조 표시하도록 구성 할 수있는 여러 "공백"기능을 가지고 있습니다. 공백 문제는 구성된 색상 color.diff.whitespace를 사용하여 강조 표시됩니다.

기본적으로 사용 가능한 기능은 다음과 같습니다.

- blank-at-eol :하이라이트 라인 엔딩에서 고아 공백
- space-before-tab :줄을 들여 쓰기 할 때 탭 문자 앞에 나타나는 공백 문자를 강조 표시합니다.
- blank-at-eof :파일 끝에 삽입 된 빈 줄을 강조합니다.

다음 기능은 기본적으로 사용하지 않도록 설정되어 있습니다.

- indent-with-non-tab: 탭 대신 공백으로 들여 쓰는 줄을 강조 표시합니다.
- tab-in-indent :초기 탭 들여 쓰기를 오류로 강조 표시합니다.
- tailing-space :blank-at-eol과 blank-at-eof 모두에 대한 줄임말입니다.
- cr-at-eol :줄 끝에서 캐리지 리턴을 강조 표시합니다.
- tabwidth = <n> :탭이 차지하는 문자 위치의 수를 정의합니다. 기본값은 8입니다. 허용되는 값은 1-63입니다.