git diff

변경 비교하기: git diff

Diffing은 두 개의 입력 데이터 집합을 가져 와서 변경 내용을 출력하는 함수입니다. git diff는 실행시 Git 데이터 소스에서 diff 함수를 실행하는 다중 사용 Git 명령입니다. 이러한 데이터 소스는 커밋, 분기, 파일 등이 될 수 있습니다. 이 문서는 git diff와 diffing 작업 흐름 패턴의 일반적인 호출에 대해 설명합니다. git diff 명령은 git status 및 git log와 함께 사용되어 Git repo의 현재 상태를 분석합니다.

Reading diffs: outputs

Raw output format

다음 예제는 간단한 repo에서 실행됩니다. repo는 아래 명령으로 생성됩니다.

```
$ mkdir diff_test_repo
$ cd diff_test_repo
$ touch diff_test.txt
$ echo "this is a git diff test example" > diff_test.txt
$ git init .
Initialized empty Git repository in /Users/kev/code/test/.git/
$ git add diff_test.txt
$ git commit -am"add diff test file"
[master (root-commit) 6f77fc3] add diff test file
1 file changed, 1 insertion(+)
create mode 100644 diff_test.txt
```

이 시점에서 git diff를 실행하면 결과가 출력되지 않습니다. diff에 대한 변경 사항이 없으므로 이는 예상되는 동작입니다. repo가 생성되고 diff_test.txt 파일이 추가되면 diff 출력을 실험하기 위해 파일 내용을 변경할 수 있습니다.

```
$ echo "this is a diff example" > diff_test.txt
```

이 명령을 실행하면 diff_test.txt 파일의 내용이 변경됩니다. 일단 수정되면 diff를보고 출력을 분석 할 수 있습니다. 이제 git diff를 실행하면 다음과 같은 결과가 출력됩니다.

```
$ diff --git a/diff_test.txt b/diff_test.txt
index 6b0c6cf..b37e70a 100644
--- a/diff_test.txt
+++ b/diff_test.txt
@@ -1 +1 @@
-this is a git diff test example
+this is a diff example
```

이제 diff 출력에 대한 자세한 분석을 살펴 보겠습니다.

1. Comparison input

Git 가이드

1. Git 시작하기
   - Git 저장소
     - git init
     - git clone
     - git config
   - 변경 저장하기
     - git add
     - git commit
     - git diff
     - git stash
     - .gitignore
   - 저장소 점검하기
     - git status
     - git log
     - git tag
     - git blame
   - 변경 취소하기
     - git 실행 취소
     - git clean
     - git revert
     - git reset
     - git rm
   - Rewriting history
     - git commit --amend
     - git rebase
     - git reflog
2. Git 협업하기
   - 동기화하기
     - git remote
     - git fetch
     - git pull
     - git push
   - 브랜치 사용하기
     - git branch
     - git checkout
     - git merge
     - 병합 충돌 해결하기 (Merge conflicts)
     - 병합 전략 (Merge strategies)
   - Pull request 만들기

```
$ diff --git a/diff_test.txt b/diff_test.txt
```

이 행은 diff의 입력 소스를 표시합니다. a / diff_test.txt와 b / diff_test.txt가 diff에 전달되었음을 알 수 있습니다.

2. Meta data

```
index 6b0c6cf..b37e70a 100644
```

이 행은 내부 Git 메타 데이터를 표시합니다. 이 정보가 필요 없을 것입니다. 이 출력의 숫자는 Git 객체 버전 해시 식별자에 해당합니다.

3. Markers for changes

```
--- a/diff_test.txt
+++ b/diff_test.txt
```

이 행은 각 diff 입력 소스에 기호를 지정하는 범례입니다. 이 경우 / diff_test.txt의 변경 사항은 ---로 표시되고 b / diff_test.txt의 변경 사항은 +++ 기호로 표시됩니다.

4. Diff chunks

나머지 diff 출력은 diff 'chunk'목록입니다. diff는 변경 사항이있는 파일의 섹션 만 표시합니다. 현재 시나리오에서는 간단한 시나리오로 작업하면서 하나의 청크 만 있습니다. 덩어리는 독자적인 세분화 된 출력 의미론을 가지고 있습니다.

```
@@ -1 +1 @@
-this is a git diff test example
+this is a diff example
```

첫 번째 줄은 청크 헤더입니다. 각 청크는 @@ 기호 안에 포함 된 헤더 앞에 추가됩니다. 헤더의 내용은 파일에 대한 변경 사항 요약입니다. 우리의 단순화 된 예에서 우리는 -1 +1 의미 라인 하나가 변경되었습니다. 보다 현실적인 diff에서는 다음과 같은 헤더가 표시됩니다.

```
@@ -34,6 +34,8 @@
```

이 머리글 예제에서는 34 행부터 6 행을 추출했습니다. 또한 34 행부터 8 행을 추가했습니다.

diff chunk의 나머지 내용은 최근 변경 사항을 표시합니다. 변경된 각 줄 앞에는 + 또는 - 기호가 표시되어 변경 사항의 원본 버전을 나타냅니다. 앞에서 설명한 것처럼 -는 a / diff_test.txt의 변경 사항을 나타내며 +는 b / diff_test.txt의 변경 사항을 나타냅니다.

변경 사항 강조 표시

1. `git diff --color-words`

git diff는 --color-words보다 훨씬 세분화 된 변경 사항을 강조하기위한 특수 모드도 제공합니다. 이 모드는 추가되거나 제거 된 행을 공백으로 토큰 화 한 다음 그 행과 비교합니다.

```
$ git diff --color-words
$ diff --git a/diff_test.txt b/diff_test.txt
index 6b0c6cf..b37e70a 100644
--- a/diff_test.txt
+++ b/diff_test.txt
@@ -1 +1 @@
this is agit difftest example
```

이제 출력물에는 변경된 색으로 구분 된 단어 만 표시됩니다.

2. `git diff-highlight`

git 소스를 복제하면 contrib이라는 하위 디렉토리가 있습니다. 여기에는 여러 git 관련 도구가 포함되어 있으며 흥미로운 비트와 조각은 아직 핵심 노드로 승격되지 않았습니다. 이 중 하나는 diff-highlight라는 Perl 스크립트입니다. diff 강조 표시는 diff 출력의 일치하는 행을 짝 지워주고 변경된 하위 단어 조각을 강조 표시합니다.

```
$ git diff | /your/local/path/to/git-core/contrib/diff-highlight/diff-
highlight
diff --git a/diff_test.txt b/diff_test.txt
index 6b0c6cf..b37e70a 100644
--- a/diff_test.txt
+++ b/diff_test.txt
@@ -1 +1 @@
-this is a git diff test example
+this is a diff example
```

이제 우리는 가능한 가장 작은 변경 사항을 비교해 보았습니다.

바이너리 파일 비교하기

우리가 지금까지 보여준 텍스트 파일 유틸리티 외에도 git diff는 바이너리 파일에서 실행될 수 있습니다. 불행히도, 기본 출력은별로 도움이되지 않습니다.

```
$ git diff
Binary files a/script.pdf and b/script.pdf differ
```

Git에는 diff를 수행하기 전에 이진 파일의 내용을 텍스트로 변환하는 쉘 명령을 지정할 수있는 기능이 있습니다. 그래도 약간의 설정이 필요합니다. 먼저 특정 유형의 바이너리를 텍스트로 변환하는 방법을 설명하는 textconv 필터를 지정해야합니다. 우리는 PDF 파일을 사람이 읽을 수있는 HTML로 변환하기 위해 pdftohtml이라는 간단한 유틸리티 (homebrew에서 사용 가능)를 사용하고 있습니다. 이 설정은 .git / config 파일을 편집하여 단일 저장소에 대해 설정할 수도 있고 ~ /.gitconfig를 편집하여 전역으로 설정할 수도 있습니다

```
[diff "pdfconv"]
textconv=pdftohtml -stdout
```

그런 다음 하나 이상의 파일 패턴을 pdfconv 필터와 연결하기 만하면됩니다. 리포지토리의 루트에 .gitattributes 파일을 만들어이 작업을 수행 할 수 있습니다.

```
*.pdf diff=pdfconv
```

일단 설정되면, git diff는 먼저 구성된 변환기 스크립트를 통해 이진 파일을 실행하고 변환기 출력을 diff합니다. jips, jars 및 기타 아카이브 : pdf2html 대신 unzip -l (또는 유사)을 사용하면 추가되거나 제거 된 경로가 표시됩니다. 이미지 커밋 : exiv2는 이미지 차원 문서와 같은 메타 데이터 변경 사항을 표시하는 데 사용할 수 있습니다. 변환 도구는 .odf, .doc 및 기타 문서 형식을 일반 텍스트로 변환하기 위해 존재합니다. 핀치 (pinch)에서 문자열은 형식 변환기가없는 바이너리 파일에서 종종 작동합니다.

파일 비교하기: git diff file

git diff 명령은 명시 적 파일 경로 옵션을 전달할 수 있습니다. git diff에 파일 경로가 전달되면 diff 작업의 범위가 지정된 파일로 지정됩니다. 아래 예제는 이러한 사용법을 보여줍니다.

```
$ git diff HEAD ./path/to/file
```

이 예제는 호출 될 때 ./path/to/file로 범위가 지정되며 작업 디렉토리의 특정 변경 사항을 인덱스와 비교하여 아직 준비되지 않은 변경 사항을 보여줍니다. 기본적으로 git diff는 HEAD와의 비교를 실행합니다. 위의 예에서 HEAD를 생략하면 git diff ./path/to/file과 동일한 효과가 나타납니다.

```
$ git diff --cached ./path/to/file
```

git diff가 --cached 옵션과 함께 호출되면 diff는 단계적 변경 사항을 로컬 저장소와 비교합니다. --cached 옵션은 --staged와 동의어입니다.

모든 변경 비교하기

git diff를 파일 경로없이 사용하면 전체 저장소의 변경 사항을 비교할 수 있습니다. 위의 파일 별 예제는 ./path /to/file 인수없이 호출 할 수 있으며 로컬 repo의 모든 파일에서 동일한 출력 결과를 갖습니다.

마지막 커밋 이후 변경 보기

기본적으로 git diff는 마지막 커밋 이후 커밋되지 않은 변경 사항을 보여줍니다.

```
$ git diff
```

서로다른 커밋의 파일 비교하기

git diff는 diff에 커밋을 전달할 수 있습니다. 몇 가지 예제 ref는 HEAD, 태그 및 분기 이름입니다. Git의 모든 커밋에는 GIT LOG를 실행할 때 얻을 수있는 커밋 ID가 있습니다. git diff에이 커밋 ID를 전달할 수도 있습니다.

```
$ git log --prety=oneline
957fbc92b123030c389bf8b4b874522bdf2db72c add feature
ce489262a1ee34340440e55a0b99ea6918e19e7a rename some classes
6b539f280d8b0ec4874671bae9c6bed80b788006 refactor some code for feature
646e7863348a427e1ed9163a9a96fa759112f102 add some copy to body

$ git diff 957fbc92b123030c389bf8b4b874522bdf2db72c
ce489262a1ee34340440e55a0b99ea6918e19e7a
```

브랜치 비교하기

두 개의 브랜치 비교하기

브랜치는 다른 모든 ref 입력과 마찬가지로 git diff와 비교됩니다.

```
$ git diff branch1..other-feature-branch
```

이 예제는 도트 연산자를 소개합니다. 이 예에서 두 점은 diff 입력이 두 가지의 팁임을 나타냅니다. 점이 생략되고 분기 사이에 공백이 사용되는 경우에도 동일한 효과가 발생합니다. 또한 점 연산자 3 개가 있습니다.

```
$ git diff branch1...other-feature-branch
```

세개의 도트 연산자는 첫 번째 입력 매개 변수 branch1을 변경하여 diff를 시작합니다. branch1을 두 diff 입력, 즉 branch1과 other-feature-branch의 공유 조상 사이의 공유 공통 조상 커밋의 ref로 변경합니다. 마지막 매개 변수 입력 매개 변수는 other-feature-branch의 끝으로 변경되지 않습니다.

두 브랜치의 파일 비교하기

브랜치간에 특정 파일을 비교하려면 파일 경로를 git diff의 세 번째 인수로 전달하십시오

```
git diff master new_branch ./diff_test.txt
```