git init

이 페이지에서는 git init 명령을 자세히 살펴 봅니다. 이 페이지의 끝 부분에서 git init의 핵심 기능과 확장 기능 세트에 대해 알게 될 것입니다. 이 탐험은 다음을 포함한다 :

- git init 옵션과 사용법
- .git 디렉토리 개요
- 사용자 정의 git init 디렉토리 환경 값
- git init 과 git clone
- git init bare 저장소
- git init 템플릿

git init 명령은 새로운 Git 저장소를 만듭니다. 그것은 버전없는 기존 프로젝트를 Git 저장소로 변환하거나 새로운 빈 저장소를 초기화하는 데 사용할 수 있습니다. 다른 Git 명령은 초기화 된 저장소 외부에서 사용할 수 없으므로 일반적으로 새 프로젝트에서 실행할 첫 번째 명령입니다.

git init을 실행하면 현재 작업 디렉토리에 .git 하위 디렉토리가 생성됩니다.이 디렉토리에는 새 저장소에 필요한 모든 Git 메타 데이터가 들어 있습니다. 이 메타 데이터에는 개체, 참조 및 템플릿 파일의 하위 디렉터리가 포함됩니다. 현재 체크 아웃 된 커밋을 가리키는 HEAD 파일도 생성됩니다.

프로젝트의 루트 디렉토리에서 .git 디렉토리를 제외하고 기존 프로젝트는 변경되지 않습니다 (SVN과 달리 모든 하위 디렉토리에는 .git 서브 디렉토리가 필요하지 않습니다).

기본적으로 git init은 Git 구성을 .git 하위 디렉토리 경로로 초기화합니다. 다른 곳에서 살기를 원하면 하위 디렉토리 경로를 수정하고 사용자 정의 할 수 있습니다. $ GIT_DIR 환경 변수를 커스텀 경로로 설정하면 git init은 거기에있는 Git 설정 파일을 초기화 할 것이다. 또한 동일한 결과에 대해 --separate-git-dir 인수를 전달할 수 있습니다. 별도의 .git 하위 디렉토리를 사용하는 일반적인 경우는 .git 폴더를 다른 위치에 유지하면서 홈 디렉토리에 시스템 구성 "dotfiles"(.bashrc, .vimrc 등)을 유지하는 것입니다.

사용법

SVN과 비교하여 git init 명령은 새로운 버전 제어 프로젝트를 생성하는 매우 쉬운 방법입니다. Git에서는 저장소를 만들고, 파일을 가져오고, 작업 복사본을 체크 아웃 할 필요가 없다. 또한, 힘내는 기존의 서버 또는 관리자 권한이 필요하지 않습니다. 당신이해야 할 일은 프로젝트 서브 디렉토리에 cd하고 git init을 실행하는 것입니다. 여러분은 Git 저장소를 완벽하게 사용할 수 있습니다.

```
git init
```

현재 디렉토리를 Git 저장소로 변환하십시오. 이렇게하면 .git 서브 디렉토리가 현재 디렉토리에 추가되고 프로젝트의 개정판을 기록 할 수있게됩니다.

```
git init <directory>
```

지정된 디렉토리에 빈 Git 저장소를 만듭니다. 이 명령을 실행하면 .git 서브 디렉토리 만 포함하는 새 서브 디렉토리가 작성됩니다.

이미 프로젝트 디렉토리에서 git init을 실행했고 .git 서브 디렉토리가 있다면, 동일한 프로젝트 디렉토리에서 git init을 다시 안전하게 실행할 수 있습니다. 기존 .git 구성을 재정의하지 않습니다.

git init vs. git clone

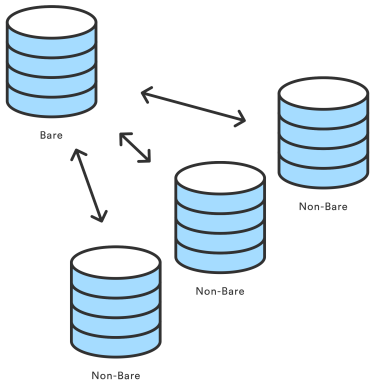
빠른 참고 사항 : git init 및 git clone은 쉽게 혼동 될 수 있습니다. 높은 수준에서, 그들은 둘 다 "새로운 자식 저장소를 초기화하는 데 사용할 수 있습니다." 그러나 git clone은 git init에 의존합니다. git clone은 기존 저장소의 복사본을 만드는 데 사용됩니다. 내부적으로, git clone은 먼저 git init을 호출하여 새로운 저장소를 만듭니다. 그런 다음 기존 저장소의 데이터를 복사하고 새로운 작업 파일 세트를 체크 아웃합니다. git clone 페이지에 대해 자세히 알아보십시오.

Bare repositories --- git init --bare

```
git init --bare <directory>
```

빈 저장소를 초기화하고 작업 디렉토리를 생략하십시오. 공유 저장소는 항상 --bare 플래그를 사용하여 만들어야합니다 (아래 토론 참조). 일반적으로, --bare 플래그로 초기화 된 저장소는 .git로 끝납니다. 예를 들어, my-project라는 저장소의 베어 버전은 my-project.git라는 디렉토리에 저장되어야합니다.

--bare 플래그는 작업 디렉토리가없는 저장소를 작성하므로 파일을 편집하고 해당 저장소의 변경 사항을 커밋할 수 없습니다. git push와 git pull을위한 맨 처음 저장소를 만들지 만 직접적으로 커밋하지 마십시오. 중앙 저장소는 베어지 저장소로 분기를 푸는 것이 변경 내용을 덮어 쓸 가능성이 있기 때문에 항상 베어 리지 저장소로 만들어야합니다. 개발 환경과는 반대로 저장소를 저장소 기능으로 표시하는 방법으로 생각하십시오. 즉, 사실상 모든 Git 워크 플로우에서 중앙 저장소는 노출되지 않았으며 개발자 로컬 저장소는 노출되지 않았습니다.



git init --bare의 가장 일반적인 사용 사례는 원격 중앙 저장소를 만드는 것입니다 :

```
ssh <user>@<host> cd path/above/repo git init --bare my-project.git
```

먼저, 중앙 저장소를 포함 할 서버로 SSH합니다. 그런 다음 프로젝트를 저장하려는 위치로 이동합니다. 마지막으로 --bare 플래그를 사용하여 중앙 저장소 저장소를 만듭니다. 개발자는 my-project.git을 복제하여 개발 컴퓨터에서 로컬 복사본을 만듭니다.

git init templates

```
git init <directory> --template=<template_directory>
```

새 Git 저장소를 초기화하고 <template_directory>의 파일을 저장소에 복사합니다.

템플릿을 사용하면 미리 정의 된 .git 하위 디렉토리로 새 저장소를 초기화 할 수 있습니다. 새 저장소의 .git 서브 디렉토리에 복사 할 기본 디렉토리와 파일을 갖도록 템플리트를 구성 할 수 있습니다. 디폴트의 Git 템플릿은 일반적으로`/ usr / share / git-core / templates` 디렉토리에 있지만 여러분 머신의 다른 경로 일 수 있습니다.

기본 템플리트는 좋은 참조이며 템플리트 기능을 사용하는 방법의 예입니다. 기본 템플릿에 표시되는 템플릿의 강력한 기능은 Git Hook 구성입니다. 미리 정의 된 Git 훅을 사용하여 템플릿을 만들 수 있으며, 일반적인 훅을 사용할 준비가 된 새로운 git 리포지토리를 초기화 할 수 있습니다. Git Hook에 대한 자세한 내용은 Git Hook 페이지를 참조하십시오.

구성

git init <directory>의 모든 설정은 <directory> 인수를 취합니다. <directory>를 제공하면 명령이 그 안에 실행됩니다. 이 디렉토리가 존재하지 않으면 생성됩니다. 이미 논의 된 옵션과 설정 외에도 Git init에는 몇 가지 다른 명령 행 옵션이 있습니다. 전체 목록은 다음과 같습니다.

```
-Q, --QUIET :" ",     .    .

--BARE :bare  .  "bare repository" .

--TEMPLATE=<TEMPLATEDIRECTORY> :   .  "Git Init Templates" .

--SEPARATE-GIT-DIR=<GIT DIR> :<git dir>      .   .git    . .git            . --
separate-git-dir     .
```

- .git 폴더를 다른 위치에 유지하면서 홈 디렉토리에 시스템 구성 "dotfiles"(.bashrc, .vimrc 등)을 유지하려면
- Git 히스토리는 디스크 크기가 매우 커 졌으므로 별도의 대용량 드라이브로 옮겨야합니다.
- Git 프로젝트를`www : root`와 같이 공개적으로 접근 가능한 디렉토리에두고 싶습니다.

기존 저장소에서 git init --separate-git-dir을 호출하면 .git 디렉토리가 지정된 <git 디렉토리> 경로로 이동합니다.

--SHARED[=(FALSE|TRUE|UMASK|GROUP|ALL|WORLD|EVERYBODY|0XXX)] :새 저장소에 대한 액세스 권한을 설정하십시오. Unix 레벨 권한을 사용하는 사용자와 그룹이 저장소로 푸시 / 풀을 허용하는 사용자와 그룹을 지정합니다.

Examples

기존 코드베이스를위한 새로운 git 저장소 만들기

```
cd /path/to/code \
git init \
git add . \
git commit
```

Bare 저장소 생성하기

```
git init --bare /path/to/repo.git
```

템플릿을 이용해 저장소 생성하기

```
mkdir -p /path/to/template \ echo "Hello World" >> /absolute/path/to
/template/README \ git init /new/repo/path --template=/absolute/path/to
/template \ cd /new/repo/path \ cat /new/repo/path/README
```