git clone

git clone 명령을 자세히 살펴 보겠습니다. git clone은 기존 저장소를 대상으로하고 대상 저장소의 복제본 또는 복사본을 만드는 데 사용되는 Git 명령 줄 유틸리티입니다. 이 페이지에서는 확장 구성 옵션과 git clone의 일반적인 사용 사례에 대해 설명합니다. 여기서 다루는 몇 가지 사항은 다음과 같습니다.

- 로컬 또는 원격 저장소 복제
- 원시 저장소 복제
- 얕은 옵션을 사용하여 저장소를 부분적으로 복제
- URL 구문 및 지원 프로토콜

이 페이지는보다 복잡한 복제 및 구성 시나리오를 탐색합니다.

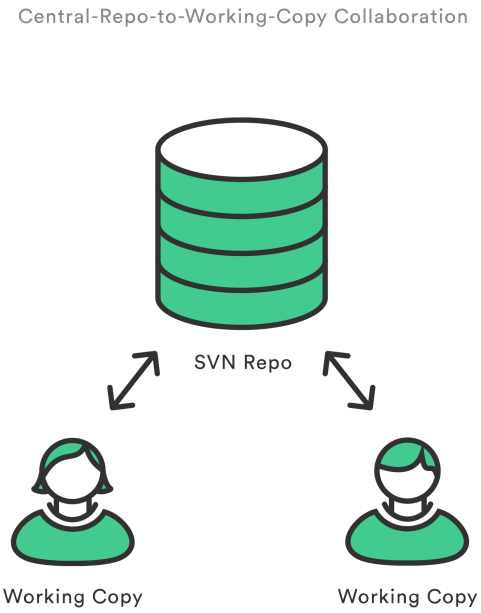
목적: 저장소 간 협업 (작업을 위한 저장소 복제)

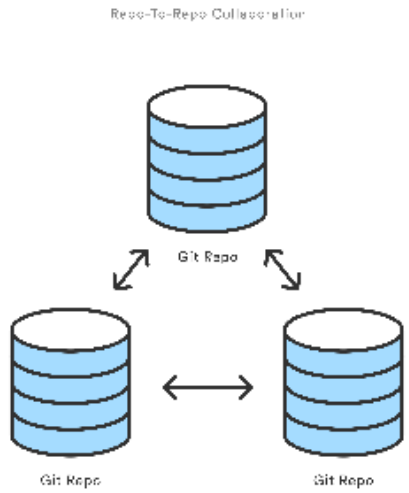
프로젝트가 중앙 저장소에 이미 설정된 경우 git clone 명령은 사용자가 개발 복사본을 얻는 가장 일반적인 방법입니다. git init과 마찬가지로, clone은 일반적으로 일회성 작업입니다. 개발자가 작업 복사본을 얻으면 모든 버전 제어 작업과 공동 작업이 로컬 저장소를 통해 관리됩니다.

저장소 간 협업

Git의 "working copy"에 대한 아이디어가 SVN 저장소에서 코드를 체크 아웃하여 얻은 작업 카피와 매우 다르다는 것을 이해하는 것이 중요합니다. SVN과는 달리, Git은 작업 카피와 중앙 저장소를 구별하지 않는다. 그것들은 모두 본격적인 Git 저장소이다.

이것은 Git과 SVN과 근본적으로 다른 협력을 만든다. SVN이 중앙 저장소와 작업 사본 사이의 관계에 의존하는 반면, Git의 공동 작업 모델은 저장소와 저장소 간의 상호 작용을 기반으로합니다. 작업 복사본을 SVN의 중앙 저장소로 검사하는 대신 한 저장소에서 다른 저장소로 커밋을 push 하거나 pull 합니다.





물론, Git repos라는 특별한 의미를주는 것을 막을 수있는 방법은 없습니다. 예를 들어 Git repo를 "중앙"저장소로 지정하면 Git을 사용하여 중앙 집중식 워크 플로를 복제 할 수 있습니다. 요컨대, 이것은 VCS 자체에 내장되어 있지 않고 규칙을 통해 수행됩니다.

사용법

git clone은 주로 기존 repo를 가리키고 다른 위치의 새 디렉토리에서 해당 repo의 복제본 또는 복사본을 만드는 데 주로 사용됩니다. 원래 저장소는 로컬 파일 시스템이나 원격 머신이 액세스 할 수있는 지원 프로토콜에 위치 할 수 있습니다. git clone 명령은 기존 Git 저장소를 복사합니다. 이것은 일종의 "작업 카피 (working copy)"가 본래의 저장소를 가지고 있다는 것을 제외하고는 SVN 체크 아웃과 비슷합니다. 그것은 자체 히스토리를 가지고 있고, 자체 파일을 관리하며, 원래 저장소와 완전히 격리 된 환경입니다.

편의상 복제는 원래 저장소로 다시 향하는 "origin"이라는 원격 연결을 자동으로 만듭니다. 따라서 중앙 저장소와 쉽게 상호 작용할 수 있습니다. 이 자동 연결은 refs / remotes / origin에서 원격 지점 HEAD에 Git 참조를 만들고 remote.origin.url 및 remote.origin.fetch 구성 변수를 초기화하여 설정됩니다.

git clone 사용 예제는 저장소 가이드 설정에서 찾을 수 있습니다. 아래 예제는 SSH 사용자 이름 john을 사용하여 example.com에 액세스 할 수있는 서버에 저장된 중앙 저장소의 로컬 복사본을 얻는 방법을 보여줍니다.

```
git clone ssh://john@example.com/path/to/my-project.git
cd my-project
# Start working on the project
```

첫 번째 명령은 로컬 컴퓨터의 my-project 폴더에있는 새 Git 저장소를 초기화하고 중앙 저장소의 내용으로 채웁니다. 그런 다음 프로젝트에 들어가서 파일 편집, 스냅 샷 작성 및 다른 저장소와의 상호 작용을 시작할 수 있습니다. 또한 .git 확장자는 복제 된 저장소에서 생략됩니다. 이것은 로컬 복사본의 베어 메어가 아닌 상태를 반영합니다.

폴더에 복제하기

```
git clone <repo> <directory>
```

<repo>에있는 저장소를 ~ <directory>라는 폴더에 복제하십시오! 로컬 컴퓨터에서.

복제 대상 테그 설정하기

```
git clone --branch <tag> <repo>
```

<repo>에있는 저장소를 복제하고 <tag>에 대한 참조 만 복제하십시오.

Shallow clone

```
git clone -depth=1 <repo>
```

<repo>에있는 저장소를 복제하고
옵션 깊이 = 1로 지정된 커밋의 이력 이 예에서는 <repo>의 복제가 만들어지고 가장 최근의 커밋 만 새 복제 된 Repo에 포함됩니다. 얕은 복제는 광범위한 커밋 내역이있는 repos로 작업 할 때 가장 유용합니다. 광범위한 커밋 내역으로 인해 디스크 공간 사용 제한 및 복제시 긴 대기 시간과 같은 확장 문제가 발생할 수 있습니다. 얕은 복제는 이러한 확장 문제를 완화하는 데 도움이 될 수 있습니다.

구성 옵션

git clone -branch

-branch 인수를 사용하면 원격 HEAD가 가리키는 분기 (대개 마스터 분기) 대신 복제 할 특정 분기를 지정할 수 있습니다. 또한 동일한 효과를 위해 분기 대신 태그를 전달할 수 있습니다.

```
git clone -branch new_feature git://remoterepository.git
```

위의 예제는 원격 Git 저장소에서 new_feature 분기 만 복제합니다. 이는 저장소의 HEAD ref 파일을 다운로드하고 필요한 ref를 추가로 가져와야하는 시간을 절약 할 수있는 순전히 유틸리티입니다.

git clone -mirror vs. git clone -bare

git clone --bare

git init --bare와 비슷하게 -bare 인수가 git clone에 전달되면 원격 저장소의 복사본은 생략 된 작업 디렉토리로 만들어집니다. 즉, 리포지토리는 푸시 및 푸시가 가능하지만 직접 편집 할 수없는 프로젝트의 기록으로 설정됩니다. 또한 repo에 대한 원격 브랜치는 -bare 저장소로 구성되지 않습니다. git init --bare와 마찬가지로 개발자가 직접 편집하지 않는 호스트 저장소를 만드는 데 사용됩니다.

git clone --mirror

--mirror 인수를 전달하면 암시 적으로 --bare 인수도 전달됩니다. 즉, --bare의 동작은 --mirror에 의해 상속됩니다. 편집 가능한 작업 파일이없는 맨 페이지가 생성되었습니다. 또한 --mirror는 원격 저장소의 모든 확장 된 참조를 복제하고 원격 분기 추적 구성을 유지 관리합니다. 그런 다음 미러에서 git remote update를 실행하면 origin repo의 모든 참조를 덮어 씁니다. 정확한 '미러링 된'기능을 제공합니다.

기타 구성 옵션

다른 git clone 옵션의 포괄적 인 목록은 Git 공식 문서를 참조하십시오. 이 문서에서는 다른 일반적인 옵션에 대해서도 다룰 것입니다.

git clone --template

```
git clone --template=<template_directory> <repo location>
```

<repo location>에서 repo를 복제하고 <template directory>의 템플릿을 새로 만든 로컬 분기에 적용합니다. Git 템플릿에 대한 철저한 언급은 git init 페이지에서 찾을 수 있습니다.

Git URLs

Git은 원격 저장소 위치를 Git 명령에 전달하는 데 사용되는 자체 URL 구문을 가지고 있습니다. git clone은 원격 저장소에서 가장 일반적으로 사용되기 때문에 여기에서 Git URL 구문을 검사합니다.

Git URL protocols

-SSH

SSH (Secure Shell)는 유비쿼터스 인증 네트워크 프로토콜로 대부분의 서버에서 기본적으로 공통적으로 구성됩니다. SSH는 인증 된 프로토콜이므로 연결하기 전에 호스팅 서버에 자격 증명을 설정해야합니다.

ssh : // [사용자 @] host.xz [: 포트] /path/to/repo.git/

- GIT

자식에게 유일한 프로토콜. 자식은 포트 (9418)에서 실행되는 데몬과 함께 제공됩니다. 프로토콜은 SSH와 유사하지만 인증이 없습니다.

git : //host.xz [: port] /path/to/repo.git/

- HTTP

하이퍼 텍스트 전송 프로토콜. 웹의 프로토콜로 인터넷을 통해 웹 페이지 HTML 데이터를 전송하는 데 가장 일반적으로 사용됩니다. Git은 HTTP http [s]를 통해 통신하도록 구성 될 수 있습니다 :

http[s]://host.xz [: port] /path/to/repo.git/