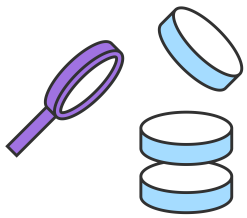
git tag



이 문서는 Git의 태그 지정과 git tag 명령에 대해 설명합니다. 태그는 Git 히스토리의 특정 지점을 가리키는 참조입니다. 태그 지정은 일반적으로 마크 업 된 버전 릴리스 (예 : v1.0.1)에 사용되는 기록의 한 지점을 캡처하는 데 사용됩니다. 태그는 변경되지 않는 분기와 같습니다. 브랜치와는 달리 태그는 생성 된 후에 더 이상 커밋 된 이력이 없습니다. 브랜치에 대한 자세한 정보는 git 브랜치 페이지를 참조하십시오. 이 문서에서는 다양한 종류의 태그, 태그 생성 방법, 모든 태그 나열 방법, 태그 삭제 방법, 태그 공유 방법 등에 대해 다룹니다.

태그 만들기

다음 명령어를 이용해 태그를 만들 수 있습니다.

```
git tag <tagname>
```

<tagname>을 태그가 생성 될 때의 레포 상태에 대한 의미 적 식별자로 바꿉니다. 일반적인 패턴은 git tag v1.4와 같은 버전 번호를 사용하는 것입니다. Git은 두 가지 유형의 태그, 주석이 달린 태그 및 간단한 태그를 지원합니다. 이전 예제에서는 간단한 태그를 만들었습니다. 경량 태그와 주석 태그는 저장된 메타 데이터 양이 다릅니다. 주석이 달린 태그는 공개로, 경량 태그는 비공개로 간주하는 것이 가장 좋습니다. 주석 태그는 태그 이름, 전자 메일 및 날짜와 같은 추가 메타 데이터를 저장합니다. 이것은 공개 자료의 중요한 데이터입니다. 경량 태그는 본질적으로 커밋에 '북마크'이며, 커밋에 대한 포인터와 포인터이며, 관련 커밋에 대한 빠른 링크를 만드는 데 유용합니다.

Annotated Tags

주석이 달린 태그는 Git 데이터베이스에 완전한 객체로 저장됩니다. 다시 말하면, 그들은 tagger 이름, 이메일 및 날짜와 같은 추가 메타 데이터를 저장합니다. 커밋 및 커밋 메시지와 유사 주석이 달린 태그에는 태그 지정 메시지가 있습니다. 또한 보안을 위해 주석이 달린 태그는 GNU Privacy Guard (GPG)를 사용하여 서명하고 확인할 수 있습니다. git 태그 지정에 권장되는 권장 사항은 모든 관련 메타 데이터를 가질 수 있도록 가벼운 태그보다 주석 태그를 선호하는 것입니다.

```
git tag -a v1.4 -m "my version 1.4"
```

이 명령을 실행하면 이전 호출과 유사하지만이 명령 버전에는 -m 옵션과 메시지가 전달됩니다. 이는 git commit -m과 비슷한 편리한 메소드로, 즉시 새 태그를 작성하고 로컬 텍스트 편집기를 열어 -m 옵션과 함께 전달 된 메시지를 저장하지 않아도됩니다.

Lightweight Tags

```
git tag v1.4-lw
```

이 명령을 실행하면 v1.4-lw로 식별되는 경량의 태그가 작성됩니다. 경량 태그는 -a, -s 또는 -m 옵션이 없으면 작성됩니다. 경량 태그는 새 태그 체크섬을 만들어 프로젝트의 repo .git / 디렉토리에 저장합니다.

태그 목록 보기

다음 명령을 통해 저장소에 저장된 태그 목록을 볼 수 있습니다.

```
git tag
```

명령의 결과는 다음과 같습니다.

```
v0.10.0
 v0.10.0-rc1
 v0.11.0
 v0.11.0-rc1
 v0.11.1
 v0.11.2
 v0.12.0
 v0.12.0-rc1
 v0.12.1
 v0.12.2
 v0.13.0
 v0.13.0-rc1
 v0.13.0-rc2
```

태그 목록을 구체화하기 위해 와일드 카드 표현식과 함께 -l 옵션을 전달할 수 있습니다.

```
$ git tag -l *-rc*
 v0.10.0-rc1
 v0.11.0-rc1
 v0.12.0-rc1
 v0.13.0-rc1
 v0.13.0-rc2
 v0.14.0-rc1
 v0.9.0-rc1
 v15.0.0-rc.1
 v15.0.0-rc.2
 v15.4.0-rc.3
```

이 이전 예제는 -l 옵션과 -rc 접두어로 표시된 모든 태그 목록을 반환하는 -rc의 와일드 카드 표현식을 사용합니다.이 태그는 전통적으로 릴리스 후보를 식별하는 데 사용됩니다.

과거 커밋에 태그 생성하기

이전 태깅 예제는 암시 적 커밋에 대한 작업을 보여주었습니다. 기본적으로 git 태그는 HEAD가 참조하는 커밋에 태그를 생성합니다. 또는 git 태그를 특정 커밋에 대한 참조로 전달할 수 있습니다. HEAD를 기본값으로하는 대신 커밋 된 커밋에 태그를 붙입니다. 이전 커밋 목록을 수집하려면 git log 명령을 실행하십시오.

```
$ git log --pretty=oneline
 15027957951b64cf874c3557a0f3547bd83b3ff6 Merge branch 'feature'
 a6b4c97498bd301d84096da251c98a07c7723e65 add update method for thing
 0d52aaab4479697da7686c15f77a3d64d9165190 one more thing
 6d52a271eda8725415634dd79daabbc4d9b6008e Merge branch 'experiment'
```

git log를 실행하면 커밋 목록이 출력됩니다. 이 예제에서 우리는 가장 새로운 커밋 Merge branch 'feature'를 선택합니다. Git에 전달하기 위해 SHA 해시 커밋을 참조해야합니다.

```
git tag -a v1.2 15027957951b64cf874c3557a0f3547bd83b3ff6
```

위의 git 태그 호출을 실행하면 이전의 git 로그 예제에서 선택한 커밋에 대해 v1.2로 식별 된 새로운 주석 된 커밋이 만들어집니다.

태그 다시 하기 / 생성된 태그 교체하기

기존 태그와 동일한 식별자를 가진 태그를 만들려고 시도하면 Git은 다음과 같은 오류를 발생시킵니다.

```
fatal: tag 'v0.4' already exists
```

또한 이전 커밋에 기존 태그 식별자를 사용하여 태그를 지정하려고하면 Git에서 동일한 오류가 발생합니다.

기존 태그를 갱신해야하는 경우 -f FORCE 옵션을 사용해야합니다.

```
git tag -a -f v1.4 15027957951b64cf874c3557a0f3547bd83b3ff6
```

위의 명령을 실행하면 15027957951b64cf874c3557a0f3547bd83b3ff6 커밋을 v1.4 태그 식별자에 매핑합니다. v1.4 태그의 기존 내용을 덮어 씁니다.

태그 공유하기: Pushing Tags to Remote

태그를 공유하는 것은 분기를 푸시하는 것과 유사합니다. 기본적으로 git push는 태그를 푸시하지 않습니다. 태그는 git push에 명시 적으로 전달되어야합니다.

```
$ git push origin v1.4
 Counting objects: 14, done.
 Delta compression using up to 8 threads.
 Compressing objects: 100% (12/12), done.
 Writing objects: 100% (14/14), 2.05 KiB | 0 bytes/s, done.
 Total 14 (delta 3), reused 0 (delta 0)
 To git@bitbucket.com:atlasbro/gittagdocs.git
 * [new tag] v1.4 -> v1.4
```

여러 개의 태그를 동시에 푸시하려면 --tags 옵션을 git push 명령에 전달하십시오. 다른 사용자가 Repo를 복제하거나 가져 오면 새 태그가 수신됩니다.

태그 체크 아웃하기

git checkout 명령을 사용하여 태그에서 repo의 상태를 볼 수 있습니다.

```
git checkout v1.4
```

위의 명령은 v1.4 태그를 체크 아웃합니다. 이렇게하면 repo가 분리 된 HEAD 상태가됩니다. 이는 변경 한 내용이 태그를 업데이트하지 않음을 의미합니다. 그들은 새로운 분리 된 커밋을 만들 것입니다. 이 새로운 분리 된 커밋은 모든 분기의 일부가 아니며 SHA 해시 커밋에 의해서만 직접 접근 할 수 있습니다. 그러므로 분리 된 HEAD 상태에서 변경을 수행 할 때마다 새 분기를 작성하는 것이 가장 좋습니다.

태그 삭제하기

태그를 삭제하는 것은 간단합니다. git 태그에 -d 옵션과 태그 식별자를 전달하면 식별 된 태그가 삭제됩니다.

```
$ git tag
 v1
 v2
 v3
 $ git tag -d v1
 $ git tag
 v2
 v3
```

이 예제에서는 git 태그가 실행되어 v1, v2, v3을 나타내는 태그 목록을 표시 한 다음 v1 태그를 삭제하는 git 태그 -d v1이 실행됩니다.