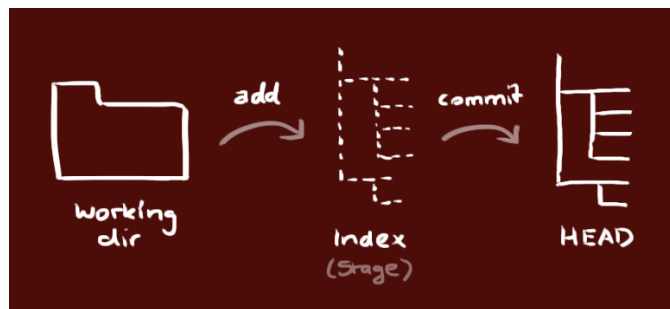
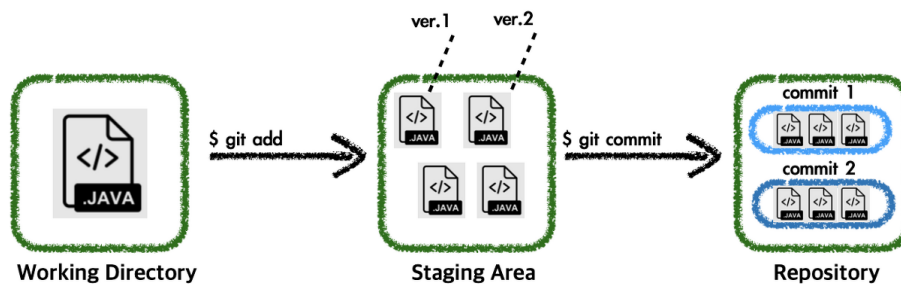


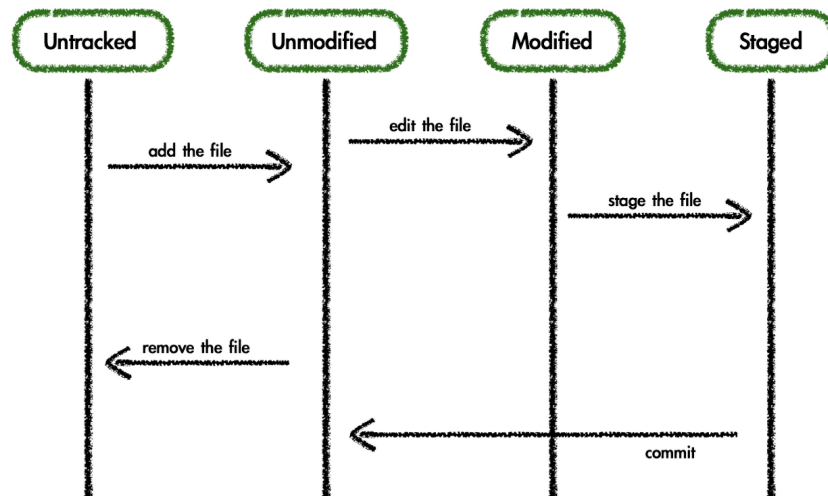
# GIT 실습 2 (Add, Commit)

## Staging Area



- Working Directory : 작업 디렉토리, 내가 작업하고 있는 프로젝트의 디렉토리
- Staging Area(Index) : 준비영역, commit을 하기 위해 add 명령어로 추가한 파일들이 모여있는 가상 공간.  
commit 내역을 검토하고 일부 파일만 선택해서 commit하거나 충돌 해결
- Repository : 최종 확정본, commit한 내용들이 저장되어 있는 저장소

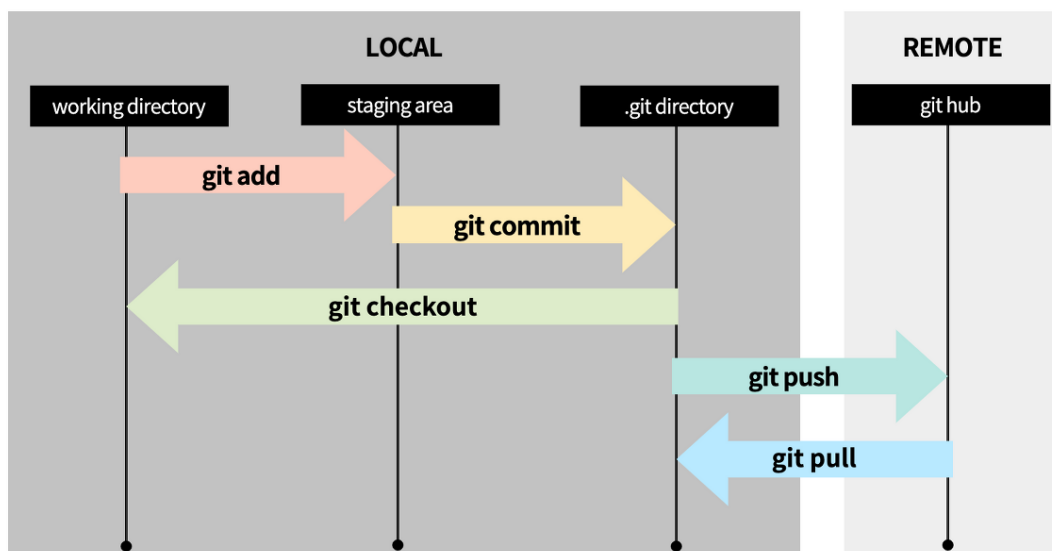
## File Status LifeCycle



**File** 관점에서는 다시 **4가지 단계**로 나뉜다.

- Untracked : Working Directory에 있는 파일이지만 **Git으로 버전관리**를 하지 않는 상태
- Unmodified : 신규로 파일이 추가되었을 때, new file 상태와 같다. ( add 상태 )
- Modified : 파일이 추가된 이후 해당 파일이 수정되었을 때의 상태
- Staged : **Staging Area**에 반영된 상태

## About git workflow



<https://m-ur-phy.tistory.com/entry/Git-깃-기초-Mac-맥-1>

## 1. add

파일 하나 담기

```
git add tigers.yaml
```

- `git status` 로 확인

모든 파일 담기(VSCode 파일 아이콘 확인)

```
git add .
```

- `git status` 로 확인

작업한 내용들을 각각 다른 버전으로 추가하는 상황도 있음

일반적으로는 모두 담기

## 2. commit

아래 명령어로 **commit**

```
git commit
```

- Vi 입력 모드로 진입

작업	Vi 명령어	상세
입력 시작	<b>i</b>	명령어 입력 모드에서 텍스트 입력 모드로 전환
입력 종료	<b>ESC</b>	텍스트 입력 모드에서 명령어 입력 모드로 전환
저장 없이 종료	<b>:q</b>	
저장 없이 강제 종료	<b>:q!</b>	입력한 것이 있을 때 사용
저장하고 종료	<b>:wq</b>	입력한 것이 있을 때 사용

위로 스크롤	k	git log 등에서 내역이 길 때 사용
아래로 스크롤	j	git log 등에서 내역이 길 때 사용

- **FIRST COMMIT** 입력한 뒤 저장하고 종료(:wq)
- 저장을 완료해야 commit이 완료됨(VSCode 파일 아이콘 확인)
- Sourcetree로 History 확인(main 브랜치)

커밋 메시지까지 함께 작성하기

(이후 작업은 이렇게 사용, 실무에서는 VI로 들어가는 일은 많이 없음)

```
git commit -m "FIRST COMMIT"
```

아래 명령어와 소스트리(F5)로 확인

```
git log
```

- 커밋마다 고유 ID확인, Sourcetree에서도 확인

### 3. 추가 수정 사항 commit

- **lions.yaml** 파일 삭제
- **tigers.yaml** 의 manager를 **Donald** 로 변경(VSCode 아이콘 변경)
- **leopards.yaml** 파일 추가

```
team: Leopards
```

```
manager: Luke
```

```
members:
```

```
- Linda
```

- William
- David

- `git status` 로 확인
  - 이전 작업 꼭 저장!
  - 파일의 **추가, 변경, 삭제** 모두 내역으로 저장할 대상
- `git diff` 로 확인
  - 변경사항을 구체적인 내용으로 확인
  - leopards.yaml은 아직 추적되지 않음
  - q로 종료

## add, commit

```
git add .
```

- `git status` 로 확인

```
git commit -m "Replace Lions with Leopards"
```

- `git log`, 소스트리로 확인
- 2개의 커밋 확인

### TIP `add` 와 `commit` 한꺼번에

```
git commit -am "(메시지)"
```

- 새로 추가된(untracked) 파일이 없을 때 한정

---

## 4. 실습

다음의 세 커밋들을 추가하세요.

## 첫 번째 추가 커밋

- Tigers의 `members`에 `George` 추가
- 커밋 메시지: `Add George to Tigers`

소스트리에서 History로 확인

F5로 화면 갱신

## 두 번째 추가 커밋

- `cheetas.yaml` 추가

```
team: Cheetas

manager: Laura

members:
- Ryan
- Anna
- Justin
```

- 커밋 메시지: `Add team Cheetas`

//커밋은 잘되었나요? -am명령어 쓰진 않으셨나요?

## 세 번째 추가 커밋

- `cheetas.yaml` 삭제
- Leopards의 `manager`를 `Nora`로 수정
- `panthers.yaml` 추가

```
team: Panthers
```

```
manager: Sebastian
```

```
members:
```

- Violet
- Stella
- Anthony

- VS Code → 왼쪽 Source Control에서 tigers.yaml 더블클릭해서 변경사항 확인 커밋 메시지:

```
Replace Cheetas with Panthers
```

## 소스트리 확인 결과

The screenshot shows the VS Code Source Control interface. On the left is a sidebar with icons for '워크스페이스' (Workspace), '파일 상태' (File State), '히스토리' (History), '검색' (Search), '브랜치' (Branch), and 'main'. The '히스토리' (History) view is selected. The main area displays a commit history for the 'main' branch. The top bar shows '모든 브랜치' (All Branches), '원격 브랜치 표시' (Show Remote Branches), and '원형 순' (Original Order). Below this, there are two tabs: '그래프' (Graph) and '설명' (Description). The '설명' (Description) tab is active, showing a list of commits. The top commit is 'main Replace Cheetas with Panthers'. Below it are four other commits: 'Add team Cheetas', 'Add George to Tigers', 'Replace Lions with Leopards', and 'FIRST COMMIT'.

모든 브랜치	원격 브랜치 표시	원형 순
그래프	설명	
<b>main Replace Cheetas with Panthers</b>		
Add team Cheetas		
Add George to Tigers		
Replace Lions with Leopards		
FIRST COMMIT		