1. 기본 과정
   1. 리포짓 생성 – git init
   2. 파일 생성 및 수정 – working tree
   3. git add – staging area
   4. git commit – 버전 생성 repositroy
   5. git push – 원격 저장소에 저장
      1. 실패한 경우 git pull 로 원격저장소와 로컬저장소 동기화
2. 버전 만들기
   1. 리포짓 만들기
      1. git init 🡪 local 리포짓 생성
      2. 깃허브 리포짓 생성 -> remote 리포짓 생성
   2. 저장소 종류
      1. working Tree: 실제로 코딩하는 장소, 버전과 무관
         1. untraked
      2. Staging Area : 버전으로 만들려고하는 파일들
         1. add 된 파일들
         2. tracked 되는 파일들로 이파일들만 git에 적용됨
      3. Repository : .git 파일 or 버전이 저장되는곳
         1. commit 으로 버전이 되어history에 올라간 것들
   3. 버전간의 차이점 비교
      1. git diff : 현재 워킹트리와 마지막 버전과의 차이점 출력
      2. git reset –hard : 워킹트리 내용을 마지막 버전으로 돌아가게함
   4. 버전 이동 – checkout(HEAD를 제어)
      1. git checkout [log hash] 로 HEAD을 특정 버전으로 옮길수 있다.
         1. 지우지 않고 HEAD만 이동
      2. got checkout [file name] : 변경된 파일을 최신버전의 파일로 이동
   5. 버전 삭제 – git reset(HEAD가 속한 브랜치를 제어)
      1. git reset –hard [log hash] : 해당 버전으로 reset하겠다.
         1. 해당 버전앞의 버전들을 삭제
      2. git reset HEAD [filename]: add된 파일을 add에서 빼버림
   6. 버전 되돌리기 – git revert -> push된 버전을 삭제하고 싶을 때 사용
      1. git revert [log hash] : 자동으로 그버전이 revert 됬다고 커밋하고 해당 버전의 수정값을 삭제함
         1. a->b->c / git revert c / a->b->c->c revert~ 의 log을 가짐
         2. revert a 처럼 최신버전이 아닌값을 하면 충돌이 일어날 확률이 높아서 거의 사용안함
         3. revert c -> revert b -> revert a가 최선 -> 그냥 reset 쓰겠다 싶은데 이는 이미 push 된 버전에 대해서 쓰면 remote랑 연동이 편하기 때문
3. branch 관리
   1. 기본 용어
      1. HEAD : 현재 가리키고 있는 곳
      2. master : 기본적으로 생성되는 branch
   2. branch 명령어
      1. git branch : 브랜치 목록 출력
      2. git branch topic : HEAD가 가리키는 버전의 복사본인 topic의 이름을 가진 branch 생성
      3. git branch topic [hash]: hash의 버전의 복사본인 topic 브랜치 생성
      4. git branch -d [branch name]: 해당 브랜치 삭제
   3. checkout 명령어 – HEAD을 움직이는 명렁어로 이해
      1. git checkout [branch name] : HEAD가 해당 브랜치를 가리키게 함
      2. git checkout -b [branch name]: 브랜치 생성과 동시에 그브랜치로 이동
   4. merge 명령어
      1. git merge [branch name] : 현재 HEAD가 가리키는 버전에 명시된 브랜치를 병합함
         1. 충돌이 없으면 바로 커밋창 나옴, 커밋 내용 지정후 저장
         2. 충돌 과정
            1. merge 후 충돌 발생
            2. 충돌한 발생한 파일에 충돌된 부분 명시되어 있음, 확인후 수정
            3. 수정한 파일을 add 하고 commit을 통해서 merge했다는 내용을 추가.
      2. 3 way merge : 병합할려는 브랜치들의 공통조상을 가져와서 merge을 한다.
         1. 더더욱 효율적인 merge 자동화
      3. merge tool 로 p4merge등등 사용
         1. 사용법은 필요하면 구글링
   5. rebase 명령어 : merge와 결과는 같지만 과정의 차이로 버전모습이 일자로됨
      1. 주의점은 원격저장소의 history는 변해서는 안된다!!!!
      2. git rebase [br name] : 해당 브랜치를 조상으로 병합을 한다.
         1. 1—2—3—4(origin/master)
         2. \*5—6—7(HEAD->master) 일때 git rebase origin/master
         3. -> 1—2—3—4(origin/master)—5’—6’—7’(HEAD->master)의 결과가 나옴
         4. 1-2-3-4의 원격저장소 기록(history)은 건들지 않았기 때문에 문제없이 push가 가능하다.
4. remote 저장소(원격 저장소)
   1. 종류
      1. github : 매우 대중적, 부분 유료
      2. gitlab : 무료
   2. 원격 저장소와 연결
      1. 이미 존재하는 저장소를 원격저장소에 연결하는 경우
         1. git remote add origin [addr] : origin이라는 별명의 remote 생성
            1. 원격 저장소를 여러 개 쓰는경우가 있으니 이름 붙이는거 중요
         2. git push –set-upstream origin master
            1. 기본 psush의 원격저장소를 지정
      2. 원격저장소를 복제해서 지역저장소를 만드는 경우
         1. git clone [addr]
            1. 원격저장소의 내용을 복제해서 local 리포짓 생성해줌(git init없이)
         2. git clone [addr] [folder name]
            1. 원격저장소의 내장된 파일이름이 아닌 특정 파일이름으로 리포짓 만들고 싶을경우
   3. push 명령어
      1. git push origin master : 현재 브랜치를 origin에 입력된 원격저장소의 master 브랜치로 push
      2. 위의 내용을 ser upstream 옵션줘서 git push 처럼 사용
   4. git fetch 명령어
      1. 원격저장소에 수정된 내용이 있을경우 그내용을 가져온다.
         1. git log등으로 수정내용을 확인
         2. 최신의 버전(ex. origin/master)과 merge후에 push가능
      2. 바뀐 내용을 확인후에 merge하는 과정이므로 git pull보다 세심한 merge 가능
   5. pull 명령어 == git fetch + git merge FETCH\_HEAD(remote의 최신 버전)
      1. 원격저장소에 git push가 거절당하면 원격저장소의 내용이 바낀경우 일수 있다.
         1. git pull을 통해 현재 버전과 원격저장소를 merge을 한다.
         2. 충돌이 발생한 경우에는 수정후에 add-commit 과정으로 merge완료
         3. 이제 git push
5. 커밋 내역
   1. git status : 파일 상태 확인
   2. git log : 커밋내역 확인
   3. git reset –hard [hash] : 해당 커밋으로 돌아감, 이후 내용 삭제 수정후
   4. git commit –amend : 최신 커밋 내용수정 ->:wq
   5. git rebase -i 명령어 : 필요할때 구글링
      1. git rebase -i HEAD~3 // 최근 3개 버전 수정가능한 창 뜸
      2. git rebase -i [hash] : 해당해쉬 제외 위의 버전들 나옴
      3. git rebase -i 로 들어와서 pick부분을 수정한후에 나갈 때 :wq! 로 나가야 자동적으로 변경이되니 주의해야함
6. 소개글 READ.md작성 : <https://github.com/leejongcheal/remote_ex/new/master?readme=1>
   1. 프로젝트에 대한 설명을 쓰는 공간
   2. #내용 ##내용 ###내용 : 크기 큰순
   3. ‘’’c ~~~~ ‘’’ 으로 c언어 포맷 나오게함
   4. 링크 [클릭내용](addr 주소) -> 클릭내용 // 클릭시 addr로 이동
   5. 목록은 밑처럼 \*이용
      1. \*목록1
      2. \*하위내용
   6. 인용구문: >인용구문 //으로 하면된다.
   7. 테이블
      1. |으로 구분해서 쓰면 알아서 테이블화
   8. 강조
      1. \*\*치킨\*\* : 굵은 글자
      2. ~~두드러기~~ : 점근선으로 가리는 글자