2024 10 11 수업정리

◆ 어제 시험 친 내용 문제 풀이

**스크립트 주기적으로 실행하는 방법
파이썬 파일 여러 번 실행
  ))shell script만들어서 계속 돌림
  ))리눅스의 cron tab사용하여 자동화 시킴

**효율적으로 짜는법(먼저효율적이 뭐지?,최대한 문제가 뭔지)
가독성 좋게, 저장 장소 용량 줄이기, 메모리 덜 소모, 짧고,실행 시켰을 때 자원을 더

**코드는 표절 해도 상관없음.클론코드 많이 해보기(좋은 코드를 많이 봐야 경험치가 쌓임, 리팩도링 작업도 거쳐보기)

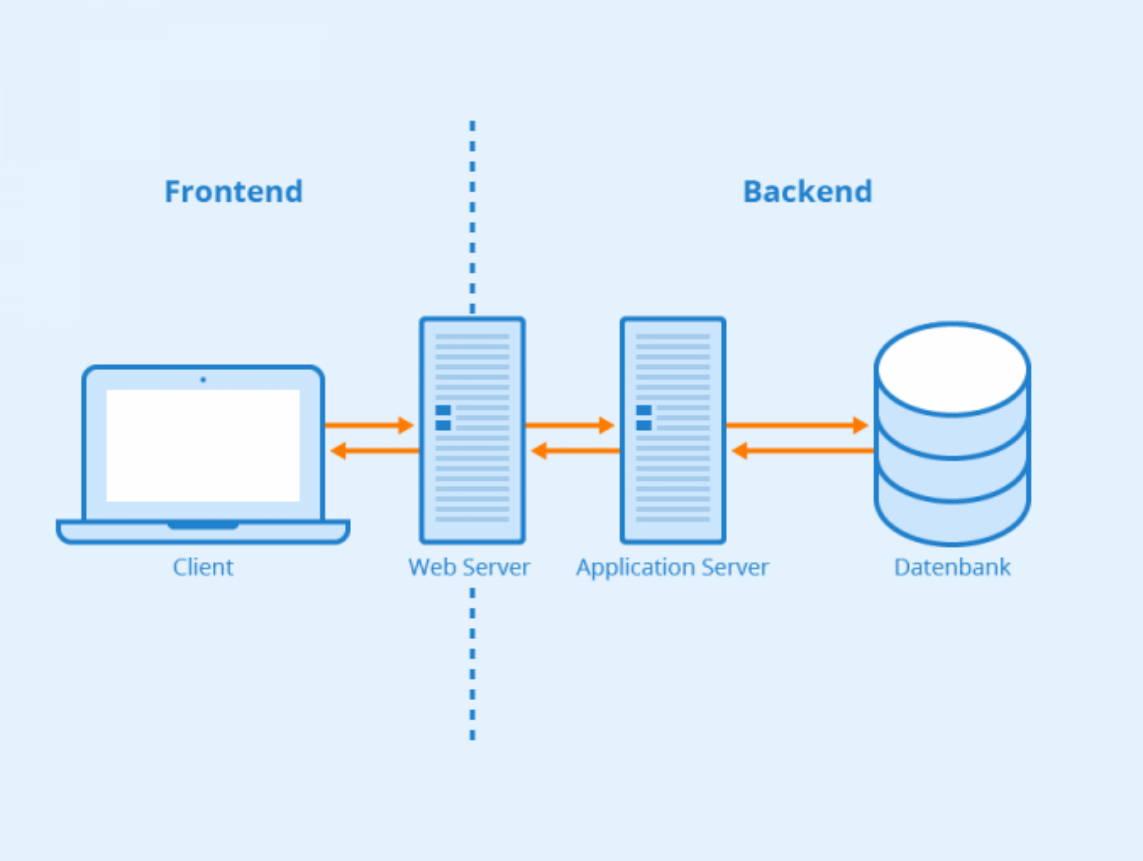
***★★for roof효율적 관리(이중 for문 써야 될때는 써야됨)★★**
** -if 한개만 쓰고 and and and**

** -if, if, if3개 각각 쓰기**
**   —> 상황에 따라 다름**

◆ 협업과 git hub

서버 연결 - 저장소는 백엔드랑 연결되어야한다.
벡엔드-개발환경(서버)위에서 코드를 짬
데브옵스(DevOps)-개발환경(서버) 서버관리자
(protocol):연결해주는 약속

PL,PM 은 협업시 각 역할 분담을 조정하고 관리하는 역할.



)) VCS이란?

-Version Control System의 약자
-지금 배율 git은 vcs의 종류(svn도씀), **git hub는 (참고로 hub는 모임.)
-git으로 협업하다 남이 짠 코드 건드리면 안됨.(merge가 안됨,충돌이일어남)
-git은 바뀐부분만 저장함.((추가된 부분과 제거된 부분이 뜸)

Q) git 관련 질문

-git은 항상 변경점(같이 변경해버리면 충돌함 둘이서 동시에 같은 부분 수정할 때 , 남이 완성한 코드 내가 망쳐서 저장해버리면 ㅠㅠ , **병합해서 complict뜨면 꼭 확인!!**)

-a,b,c 동시에 업로드하면 충돌이 일어날까? -git으로 하면 거의 안일어남(업로드 시간이 간발의 차로 다르기 때문에)
 - wheel(파이썬라이브러리 확장자) pip
-예를들어 (b코드를 짜려할때 a코드를 알아야 할 경우) PM이 그 흐름을 알아서 조정해준다.
-branch도 PM이 관리함, merge도 개인이 함부로 못함.
-merge굉장히 중요함.(다 날려버릴 수도 있으니까)
-branch이름 본인이 정하면 됨.

)) git 설치 및 사용

 리눅스 서버에서 git 사용할 때 비주얼 코드 들어가서 리눅스 서버 아이디 및 비번 입력 후

 호스트 연결 된 상태에서 git 명령어를 입력하여 git사용!!

** 깃으로 협업하다가 코드 저장시 충돌이 일어나는 경우 및 해결 방안

Git으로 협업할 때 코드 충돌이 발생하는 상황은 **두 명 이상의 개발자가 동일한 파일의 동일한 부분을 수정했을 때**입니다. Git은 이러한 상황을 자동으로 병합하지 못하고, 충돌이 발생했다고 알려줍니다. 충돌을 해결하지 않으면 코드를 병합할 수 없기 때문에, 수동으로 수정 후 병합 작업을 진행해야 합니다.

충돌 발생 과정:

1. **브랜치 분기**:
    - 팀에서 A 개발자는 `main` 브랜치에서 `feature-A`라는 새 브랜치를 생성해 새로운 기능을 개발하고 있습니다.
    - 동시에 B 개발자도 `main` 브랜치에서 `feature-B`라는 브랜치를 생성해 다른 작업을 하고 있습니다.
2. **각자 작업**:
    - A 개발자가 `feature-A` 브랜치에서 `file1.txt` 파일의 10번째 줄을 수정합니다.
    - B 개발자도 `feature-B` 브랜치에서 **동일한** `file1.txt` 파일의 10번째 줄을 수정합니다.
3. **브랜치 병합 시도**:
    - 먼저 A 개발자는 작업을 끝내고 `feature-A` 브랜치를 `main` 브랜치로 **Merge**합니다. 이때, Git은 문제없이 병합을 처리합니다.
    - 이제 B 개발자가 `feature-B` 브랜치를 `main` 브랜치에 병합하려고 합니다. 하지만 이 시점에서 B 개발자가 수정한 **file1.txt**의 동일한 부분이 이미 `main` 브랜치에서 수정되었기 때문에, Git은 **충돌(conflict)**을 감지하고 자동으로 병합할 수 없다고 알려줍니다.

충돌 해결 과정:

충돌이 발생하면 Git은 충돌이 난 파일을 표시해주며, 충돌 난 부분을 개발자가 수동으로 해결해야 합니다.

1. **충돌 파일 확인**:
    - 충돌이 발생한 파일을 열면, Git이 문제를 해결할 수 있도록 충돌 부분을 다음과 같이 표시해 줍니다:

```
bash
코드 복사
<<<<<<< HEAD
A 개발자가 작성한 코드
=======
B 개발자가 작성한 코드
>>>>>>> feature-B
```

여기서 `HEAD`는 `main` 브랜치의 최신 커밋을 나타내고, `feature-B`는 병합하려고 했던 B 개발자의 변경 사항을 나타냅니다.

2. **수동으로 충돌 해결**:
   - 개발자는 두 버전의 코드를 보고, 어떤 코드가 최종적으로 사용될지 선택해야 합니다. 두 코드를 합칠 수도 있고, 하나를 삭제할 수도 있습니다.
   - 예시:

A와 B가 같은 부분을 다르게 수정했으므로, 최종 코드에서 어떤 부분을 살릴지 선택합니다. 둘 다 합쳐질 수도 있고, 하나만 남길 수도 있습니다.

```
bash
코드 복사
// A의 코드와 B의 코드를 합치는 경우:
A 개발자가 작성한 코드
B 개발자가 작성한 코드
```

3. **충돌 해결 후 커밋**:
   - 충돌을 해결한 후 변경된 파일을 저장하고, 다시 **add**하고 **commit**합니다.

```
bash
코드 복사
git add file1.txt
git commit -m "Resolved merge conflict in file1.txt"
```

4. **병합 완료**:

- 이제 충돌이 해결되었으므로, `feature-B` 브랜치를 `main` 브랜치에 병합할 수 있습니다.
- 병합이 성공적으로 완료됩니다.

충돌을 방지하는 방법:

- **자주 병합**: 각 개발자가 작업하는 동안 너무 오래 기다리지 않고, 자주 병합하여 충돌을 방지할 수 있습니다.
- **작은 단위로 작업**: 작은 단위로 작업을 수행하고 자주 커밋하면, 충돌이 발생할 가능성이 줄어듭니다.
- **코드 리뷰 및 협업**: 팀 내에서 어떤 파일을 누가 수정하고 있는지 미리 파악하고, 필요 시 협업해 수정 범위를 조정하는 것도 좋습니다.

Git 충돌은 협업 과정에서 자주 발생할 수 있지만, 충돌이 발생한 부분을 수동으로 해결하면 큰 문제없이 프로젝트를 진행할 수 있습니다.