**📌 1. 협업 준비**

**✅ (1) GitHub 원격 저장소 준비**

* 팀장이 GitHub에 springboot-project 저장소 생성
* 팀원들을 Collaborator(공동 편집자)로 초대
* 원격 URL: https://github.com/팀이름/springboot-project.git

**📌 2. IntelliJ로 Clone 받기**

1. IntelliJ 실행 ➜ Get from VCS 클릭
2. URL에 GitHub 저장소 주소 입력 ➜ Clone
3. 로컬에 저장 경로 선택 ➜ 프로젝트 열기

**📌 3.**main**브랜치 보호 설정 (GitHub)**

* GitHub 저장소 ➜ Settings ➜ Branches ➜ Branch protection rules
* main 브랜치에:
  + **직접 Push 금지**
  + **Pull Request로만 Merge**
  + 최소 리뷰어 1명 지정

이렇게 하면 실수로 바로 main에 올리는 걸 방지할 수 있어요.

**📌 4. 각자 작업용 브랜치 생성하기**

**예시: 팀원이**feature-login**브랜치에서 로그인 기능 개발**

bash

복사편집

*# 최신 main 가져오기* git checkout main git pull origin main *# 새로운 브랜치 생성 & 이동* git checkout -b feature-login *# 브랜치 작업 (코드 작성)* *# 변경사항 확인* git status *# 변경파일 add* git add . *# 커밋 작성* git commit -m "feat: 로그인 기능 구현" *# 원격에 새 브랜치 Push* git push origin feature-login

**📌 5. Pull Request(PR)로 병합 요청하기**

1. GitHub 저장소 페이지 이동
2. Compare & pull request 버튼 클릭
3. Base Branch ➜ main, Compare ➜ feature-login
4. PR 제목 & 설명 작성 (어떤 기능인지)
5. 리뷰어 지정 ➜ 리뷰 요청
6. 리뷰 승인되면 Merge Pull Request 클릭
7. 병합 완료 후 브랜치 삭제

**📌 6. 다른 사람의 변경사항 가져오기**

팀원 A가 main에 PR Merge를 완료하면  
다른 팀원 B는 반드시 main을 최신화하고 **자기 브랜치에 반영**해야 합니다.

bash

복사편집

*# main 최신화* git checkout main git pull origin main *# 내 작업 브랜치로 이동* git checkout feature-cart *# main의 최신 내용 반영 (rebase 또는 merge)* git merge main *# 또는* git rebase main

**Tip:** rebase는 커밋 기록이 깔끔해지고 merge는 기록이 그대로 남음.

**📌 7. 충돌(Conflict) 발생 시**

* IntelliJ에서는 충돌 파일에 <<<<< HEAD 표시가 생김.
* 충돌 난 부분을 직접 수정 ➜ add ➜ commit.
* 다시 push.

**📌 8. 브랜치 삭제하기**

PR Merge 후 깃허브에서 Delete Branch 클릭해서 불필요한 브랜치 제거.  
로컬에서도 삭제:

bash

복사편집

git branch -d feature-login *# 로컬 삭제* git push origin --delete feature-login *# 원격 삭제*

**✅ 요약: 3~4인 협업 핵심 흐름**

1️⃣ main에서 branch 분기 → feature-기능명  
2️⃣ 각자 로컬에서 작업하고 커밋 ➜ 원격 push  
3️⃣ GitHub에서 PR 올려 코드 리뷰 & Merge  
4️⃣ Merge되면 main 최신화 ➜ 자기 branch에 반영  
5️⃣ 필요 없는 브랜치 정리

**🚀 IntelliJ에서 Git 쉽게 쓰는 팁**

* Git 탭에서 Commit/Push/Checkout 버튼 사용
* 충돌 발생 시 IntelliJ Merge Tool 사용 가능
* Log 탭에서 커밋 히스토리 시각화 가능
* Pull 할 때 Rebase 체크박스 활용 (옵션)

**두 명이 각자**feature**브랜치를 따로 만들어 올렸을 때, 서로 최신화해서 충돌 없이**main**에 병합하려면?**

**✅ 상황 정리**

* main: 배포 기준 안정된 브랜치
* 당신: feature-fff1 브랜치에서 작업 ➜ 원격 push 완료
* 팀원: feature-fff4 브랜치에서 작업 ➜ 원격 push 완료
* 이제 각자 **main 최신 내용 반영하고**, PR(Pull Request)로 main에 머지할 계획

**✅ 협업 흐름**

1️⃣ 각자 자신의 feature 브랜치를 main과 최대한 동기화해야 함  
2️⃣ 충돌 최소화하려면 **순서 지켜서 Merge 또는 Rebase**  
3️⃣ 충돌 해결 후 main에 PR ➜ Merge ➜ 다음 작업자 pull ➜ 다시 동기화

**✅ 흐름 예시**

**🔹 1.**main**최신화**

bash

복사편집

*# main으로 이동* git checkout main *# 원격 main 최신 내용 가져오기* git pull origin main

**🔹 2. 내**feature**브랜치 최신화**

내 작업이 feature-fff1이라면:

bash

복사편집

*# 내 작업 브랜치로 이동* git checkout feature-fff1 *# main의 최신 내용 가져오기 (Merge 방식)* git merge main *# 또는 Rebase 방식* *# git rebase main*

🔑 merge는 커밋 히스토리를 그대로 남기고, rebase는 커밋을 깔끔하게 덮어씌움.  
팀 내 규칙에 따라 선택!

**🔹 3. 충돌 나면 반드시 로컬에서 해결 후**

* 충돌 파일 수정 ➜ git add 파일
* merge: git commit
* rebase: git rebase --continue

**🔹 4. 최신화된**feature**브랜치 다시 원격 push**

bash

복사편집

*# merge 방식이라면* git push origin feature-fff1 *# rebase 후라면 force push 필요* git push -f origin feature-fff1

**🔹 5. Pull Request(PR) ➜ main으로 병합 요청**

* GitHub에서 **base: main**, **compare: feature-fff1**
* 리뷰 받고 ➜ Merge Pull Request ➜ 완료 후 원격 feature 브랜치 삭제

**🔹 6. 다른 팀원도 동일하게 반복**

팀원이 feature-fff4 브랜치도:

bash

복사편집

*# main 최신 pull* git checkout main git pull origin main *# 내 작업 브랜치로 이동* git checkout feature-fff4 *# main 병합* git merge main *# 충돌 해결 ➜ push ➜ PR ➜ Merge Pull Request*

**✅ 충돌 최소화 팁**

✅ 팀원끼리 **누가 먼저 Merge할지 순서 합의**  
✅ 먼저 Merge된 사람 ➜ main 최신화 ➜ 나머지 사람 Pull ➜ 다시 Merge  
✅ 중복 충돌 방지!

**✅ ✔️ 요약: 협업 순서**

| **순서** | **해야 할 일** |
| --- | --- |
| 1️⃣ | main 최신 pull |
| 2️⃣ | 내 작업 브랜치 최신화 (merge/rebase) |
| 3️⃣ | 충돌 해결 |
| 4️⃣ | 최신화된 브랜치 push |
| 5️⃣ | PR ➜ 리뷰 ➜ Merge Pull Request |
| 6️⃣ | 나머지 팀원은 Merge된 최신 main 다시 pull 후 반복 |

**✅ 추천: Merge vs Rebase 언제?**

| **방법** | **특징** | **언제 사용** |
| --- | --- | --- |
| Merge | 커밋 기록 그대로 남음 | 기록 중요할 때 |
| Rebase | 깔끔한 선형 히스토리 | 팀원 간 충돌 적고 깔끔히 정리할 때 |
| Squash Merge | 기능 단위로 커밋 압축 | PR 병합 시 |