

Git 가이드

이 문서는 Git 가이드를 공유하기 위해 작성되었다.

- 1. Git 시작하기
- 2. Git 협업하기
- 3. Git Branching
- 4. Git on the Server
- Git tips
- Migration

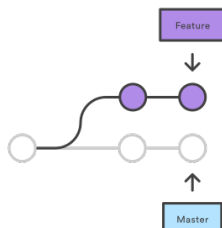
Git Overview

Git은 가장 널리 사용되는 버전 관리 시스템으로 사실상 버전 관리의 표준임

Performance - 성능에 최적화된 committing, branching, merging 동작 - Git 저장소는 멀티 엔코딩, 압축, 디렉토리 내용과 바이트 메타데이터 오브젝트로 구성됨 - 분산 관리로 상당한 성능 개선 이점 제공	Security - 파일 내용, 파일과 디렉토리 관계, 버전, 태그 커밋 등 모든 오브젝트는 SHA1 암호 해시 보안 알고리즘으로 보호 - Git의 보안 방식은 실수 또는 악의적인 변경으로부터 코드와 변경 기록 보호 - 타 버전 관리 시스템에는 없는 사후 변경에 대한 보호 가능 제공
Flexibility - Nonlinear 개발 업무 흐름 지원, 소규모부터 대규모 프로젝트 적용 가능 - 기존 시스템과 및 프로토콜과 호환성 제공 - 분기 및 태그 지정 지원	Distributed, 오픈소스 - 분산 저장소 운영 지원으로 소스코드 형상 관리 유연성 제공 - 오픈소스로 비용 없이 사용 가능 - 사실상의 표준 버전 관리 시스템

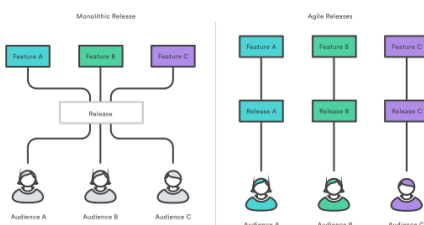
Git Overview

- 브랜치 지원
 - 코드 변경에 대한 독립적인 환경 제공
 - 빠르고 쉬운 브랜치 머지 제공
 - 모든 변경 작업은 브랜치를 생성해서 작업 → 언제나 배포 가능한 master 브랜치 유지
 - 애자일 백로그와 동일한 작업 세분화 유도



Git Overview

- 단일 저장소에 다양한 배포 일정 운영 가능
 - Feature C: UI 부분만 수정 (2주 후 배포)
 - Feature B: 현재 고객에게 영향을 주는 작은 기능 추가 (2개월 후 배포)
 - Feature A: 중요 기능 추가 (6개월 후 배포)



1. Git 시작하기

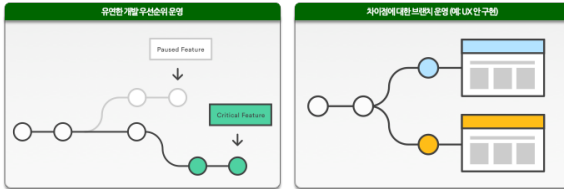
- Git 저장소
 - git init
 - git clone
 - git config
- 변경 저장하기
 - git add
 - git commit
 - git diff
 - git stash
 - .gitignore
- 저장소 점검하기
 - git status
 - git log
 - git tag
 - git blame
- 변경 취소하기
 - git 실행 취소
 - git clean
 - git revert
 - git reset
 - git rm
- Rewriting history
 - git commit --amend
 - git rebase
 - git reflog

2. Git 협업하기

- 동기화하기
 - git remote
 - git fetch
 - git pull
 - git push
- 브랜치 사용하기
 - git branch
 - git checkout
 - git merge
 - 병합 충돌 해결하기 (Merge conflicts)
 - 병합 전략 (Merge strategies)
- Pull request 만들기

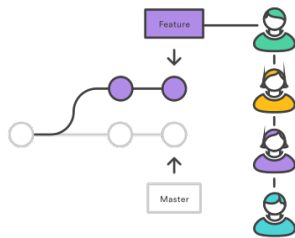
Git Overview

- 브랜치 운영의 이점들



Git Overview

- Pull Request
 - 다른 개발자에게 다른 브랜치에 여러분의 브랜치를 머지 요청하는 방법
 - 프로젝트 리더가 변경 추적 용이
 - 작업에 대해 개발자들간의 논의 유도
 - 기술적으로 어려운 문제에 대해 동료의 도움을 구하는 방법 제공
 - 주니어는 pull request를 이용해 코드 리뷰를 요청하여 실수 예방



Git Overview

- 분산 개발 지원
 - 로컬 이력 관리로 Git를 빠르게 유지 가능
 - 네트워크 연결되지 않아도 커밋, 비교등의 기능 수행 가능
 - 쉬운 개발 팀 스케일링
 - 개발자별 기능 브랜치 운영 가능해 빠른 개발 환경 제공

