2020

버전 관리 모범 사례

팀의 협업을 향상시키기 위한 쉬운 5가지 방법

- ✓ 브랜치 전략 수립하기
- ✓ 작은 단위로 자주 커밋하기
- ✓ 상세한 커밋 메세지 작성
- ✓ 브랜치 상에서 작업하기
- ✓ 정기적인 코드 리뷰 수행

GitLab

프로덕트에 집중하세요.
인포그랩의이 돕습니다.

https://insight.infograb.net

목차

소개 01

파트 1: 브랜치 전략 수립하기 02
- 개요
- 협업에 미치는 효과

파트 2: 작은 단위로 자주 커밋하기 04
- 개요
- 협업에 미치는 효과

파트 3: 상세한 커밋 메세지 작성 05
- 개요
- 협업에 미치는 효과

파트 4: 브랜치 상에서 작업하기 06
- 개요
- 협업에 미치는 효과

파트5: 정기적인 코드 리뷰 수행 07
- 개요
- 협업에 미치는 효과

GitLab을 통한 버전 관리와 협업 09

소개

산업의 급격한 변화와 새로운 기능에 대한 고객의 요구 증가로 인해 팀은 서로 단절되고 고립된 업무 환경에서 처하게 될 수 있습니다. 애플리케이션 개발에는 반복되는 빠른 작업들이 요구되며, 비즈니스에 가치를 제공하기 위해서는 유기적인 협업이 필요합니다. 팀은 버전 관리를 통해 효율적으로 협업하고 정보의 고립(silo)을 해소할 수 있습니다.

버전 관리는 소프트웨어 프로젝트의 모든 변경 사항을 관리하며, 소스 파일, 디자인, 디지털 자산 및 메타데이터에 대한 변경 이력을 효과적으로 관리합니다. 이를 통해, 신속한 협업과 의견 공유가 가능하며 즉각적이고 실행 가능한 변화를 가져다 줍니다.

버전 관리 및 협업 도구를 채택한 팀들은 작업을 매끄럽게 조정하고, 변경 사항을 검토하며, 배포를 쉽게 관리함으로써 중요한 문제 해결과 가치에 전달에 집중할 수 있습니다.

버전 관리 및 협업은 단순한 변경 이력 조회를 넘어 다음과 같이 사용될 수 있습니다:

- 개발 팀이 분산 및 비동기 환경에서 작업할 수 있도록 지원
- 코드 및 산출물의 변경 및 버전 관리
- 코드 리뷰 및 기타 도구의 활성화
- 제안된 변경 사항에 대한 승인 내용 추적
- 병합(merge) 충돌 및 관련된 통합 문제 해결

특히 급변하는 환경에서는 버전 관리를 도입하는 것은 어려운 작업이 될 수 있습니다. 본 eBook은, 개발팀이 Git을 사용하여 반복되는 새로운 기능을 개발을 효율적으로 수행하고 비즈니스 가치 전달에 도움을 줄 수 있도록 협업 강화를 위한 5가지 모범 사례를 소개합니다.

파트 1

브랜치 전략 수립하기

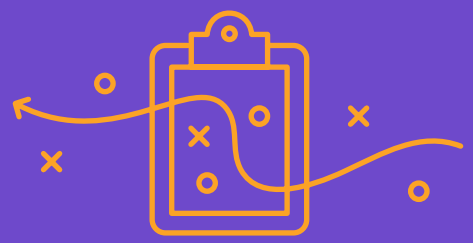
개요

다양한 전문분야와 지식을 가진 팀원들이 함께 일할 때 워크플로우에 대한 접근방식이 서로 상충될 수 있습니다. 이러한 혼란을 미연에 방지하기 위해 리더는 단일한 브랜치 전략을 세우고 이를 모두에게 전파해야 합니다.

브랜치 워크플로우의 결정은 팀 규모, 경험 수준, 확장에 대한 요구도 및 업무적인 제한 사항을 비롯한 여러 가지 요소에 따라 달라질 수 있습니다.

개발팀은 이미 정해진 워크플로우를 따라갈 수도 있지만, 팀에 필요한 요구사항에 맞춰 워크플로우를 만들 수도 있습니다. 어떤 전략을 선택하든 워크플로우를 팀에 전파하고, 필요한 경우 교육을 제공하는 것이 중요합니다.

일반적인 워크플로우는
다음을 포함합니다



중앙 집중식 워크플로우

중앙 집중식 워크플로우에는 하나의 저장소와 하나의 마스터 브랜치가 있습니다. 팀은 개발 시 다른 브랜치를 사용하지 않으므로, 변경 사항을 덮어쓸 위험이 높습니다.

기능(Feature) 브랜치

기능 브랜치는 새로운 기능을 추가할 때마다 새로운 브랜치를 만드는 것입니다. 기능에 대해 작업자들은 모두 해당 `feature` 브랜치에 커밋합니다.

GitFlow

GitFlow는 기능 브랜치의 극단적인 형태입니다. GitFlow를 사용한 개발에는 마스터(`master`)와 별도의 개발(`develop`) 브랜치가 존재하며, 기능, 릴리즈(release) 및 핫픽스(hotfix) 브랜치가 지원됩니다. `develop` 브랜치에서 개발이 수행되고 `release` 브랜치로 이동 후 `master` 브랜치로 병합됩니다.

태스크-브랜치 개발

GitLab Flow는 이러한 개발 유형의 한 예이며, 기능 중심(feature-driven) 개발 및 `feature` 브랜치를 이슈 추적과 결합한 형태입니다. GitLab Flow는 별도의 전용 브랜치를 사용하여 스테이징(staging), 운영 테스트(pre-production) 및 운영(production)과 같은 여러 환경을 유지 관리합니다. 이를 통해, 커밋되는 변경사항이 모든 환경에 걸쳐 테스트되도록 보장합니다.

협업에미치는효과

모두가 동일한 워크플로우 상에서 조화를 이룰 때 코드를 덮어 쓰거나 master를 망칠 일이 줄어듭니다.

또한 모든 작업자가 개발 및 배포 프로세스에 익숙하기 때문에 팀 구성원이 서로의 작업에 쉽게 기여할 수 있습니다. 명확하고 간결한 브랜치 전략은 새로운 코드를 병합하고 프로젝트를 발전시키는 선순환으로 이어집니다. 이러한 환경은 팀 구성원이 회의를 주선하고 마감일과 작업량을 관리하는 데 도움이 됩니다.

다음은 인기 있는 워크플로우가 어떻게 협업에 영향을 미치는지 자세히 살펴볼 것입니다.

중앙 집중식 워크플로우

이러한 유형의 워크플로우에는 협업이 없지만, 두 명의 개발자가 동일한 코드를 동시에 중복해서 작업하지 않도록 좋은 의사소통이 가능한 소규모 팀(개발자 5명 미만)에게 효과적입니다.

기능(Feature) 브랜치와 GitFlow

기능 브랜치는 더 많은 코드 리뷰, 푸시 규칙, 코드 승인자 및 더 광범위한 테스트가 필요하므로 다양한 팀을 연결해 줍니다.

태스크-브랜치 개발

태스크 브랜치 개발은 매우 빠른 속도로 진행되며, 팀 구성원 들로 하여금 요구 사항을 task 브랜치를 통해 전달되는 작은 기능의 덩어리로 분해하도록 합니다. 이러한 유형의 워크플로우에는 코드 스니펫, 코드 리뷰 및 단위 테스트와 같은 협업 사례를 포함합니다. 또한 테스트가 실패할 경우 팀원이 협력하여 무엇이 잘못되었는지 확인할 수 있습니다.

파트 2

작은 단위로 자주 커밋하기

개요

완성도를 우선시하는 나머지, 프로젝트가 거의 완성됐을 때만 대규모로 커밋하는 습관에 빠질 수 있습니다.

그러나 간단한 기능들의 검증과 작은 단계들을 건너뛰게 되면, 결국 잘못된 기능 개발이나 엉뚱한 방향성으로 시간을 소비 할 위험이 있습니다.

소프트웨어 개발에서 비즈니스 가치를 제공하고 고객 요구를 충족시키기 위한 가장 효과적인 방법은, 동작 가능한 테스트와 코드가 있을 때마다 커밋하는 것입니다.

프로젝트를 작은 단계로 단순화시키고, 잦은 커밋을 통해 큰 목표를 달성하기 위한 방법을 찾아야 합니다.

협업에 미치는 효과

자주 커밋하는 문화가 갖춰지면, 모두가 코드 저장소에 대한 가시성을 갖게 되므로 다른 팀원들이 무엇을 하고 있는지 쉽게 알 수 있습니다.

기능 브랜치 또는 병합 요청에서 작업물을 공유하게 되면 다른 팀원의 중복 작업을 막을 수 있으므로, 아직 검토 준비가 되지 않은 경우에도 기능 브랜치로 자주 푸시해야 합니다. 완료되기 전에 작업을 공유하면 토론 및 피드백이 활성화되어 코드 리뷰 이전에도 품질을 개선할 수 있습니다.

작업을 여러 개의 커밋으로 나누면, 개발자와 다른 팀 (예 : QA 및 보안)이 향후에 코드를 검토할 때 유용한 정보로 활용될 수 있습니다. 기능이 어떻게 개발되었는지 커밋 별로 명확하게 식별이 가능하며, 관련 없는 변경 이력은 그대로 두고 원하는 특정 시점으로 롤백하거나 특정 코드의 변경 사항만 선택적으로 되돌리는 것이 용이해집니다.

파트 3

상세한 커밋 메세지 작성

개요

커밋 메시지는 커밋 내용뿐만 아니라 개발자의 의도를 반영해야 합니다. 커밋으로 인한 변경 사항을 쉽게 확인할 수 있는 만큼, 커밋 메시지를 통해 이러한 변경이 발생한 이유를 설명할 수 있어야 합니다.

팀 전체의 일관성을 유지하고 혼동과 잘못된 의사소통을 줄이기 위해, 커밋 메시지 내용의 규칙을 정하는 것이 중요합니다.

좋은 커밋 메시지의 예는 다음과 같습니다.
"사용자 화면의 중복 코드를 줄이기 위해 템플릿을 결합함."

커밋 메시지에 담긴 *"변경"*, *"개선"*, *"수정"* 및 *"재구성"* 과 같은 단어는 유용한 정보가 되지 않습니다.

예를 들어, *"XML 생성을 개선함"* 대신 *"XML 생성에서 특수 문자를 올바르게 이스케이프 처리함"* 이라고 하는 것이 더 좋습니다.

협업에 미치는 효과

상세한 커밋 메시지는 투명성을 높이고 진행 상황에 대한 가시성을 제공하여 팀원, 고객 및 미래의 기여자들이 개발 프로세스를 이해할 수 있도록 도와줍니다.

코드 리뷰를 수행할 때 커밋 메시지는 팀 구성원이 반복된 절차를 따르고, 릴리즈, 협의 또는 요구 사항의 변경 이후에 어떤 변화가 이루어졌는지 확인하는데 도움이 됩니다.

또한, 자세한 커밋 메시지는 QA와 보안팀이 코드를 검사할 때 문제 영역을 식별하고 특정 변경 사항을 되돌릴 수 있도록 도와줍니다. 뿐만 아니라, 개발자가 자세한 커밋 메시지를 작성하면 다른 팀원의 중복 작업을 막아주고, 작업 지연을 최소화하여 프로젝트의 진척률을 보다 안정적으로 이끌어나갈 수 있게 됩니다.

파트 4

브랜치를 사용하여 개발하기

개요

브랜치에서 개발하는 것은 특정 브랜치 (보통 master)에서 현 시점의 상태를 복제하여 작업하는 것과 같습니다.

브랜치를 사용하면 메인 코드베이스에 영향을 주지 않고 코드를 변경하는 것이 가능합니다. 변경 이력은 브랜치 내에서 관리 됩니다. 코드가 준비되면 코드를 master 브랜치에 병합할 수 있습니다.

브랜치에서 코딩하면 개발에 대한 좀 더 체계적인 방식으로 접근할 수 있으며, 개발중인 초안을 안정적인 테스트를 거친 master 코드와 분리하여 별도로 관리할 수 있습니다.

협업에 미치는 효과

브랜치를 사용한 개발은, 구성원들이 복잡한 문제를 해결하기 위한 혁신적인 해결책과 실험을 시도해 볼 수 있게 합니다.

개발팀은 master 브랜치에 영향을 줄 것이라는 불안감에서 벗어나 창의적인 도전이 가능합니다. 뿐만 아니라, master 브랜치에 병합하기 전에 솔루션에 대한 정상 작동을 검증하기 위해 협업할 수 있습니다. 운영, QA 및 보안팀은 배포 전에 코드를 검토하고, 모든 구성원이 릴리즈 전에 아이디어를 제시하고 잠재적인 문제에 대해 논의할 수 있는 기회를 보장해 줍니다.

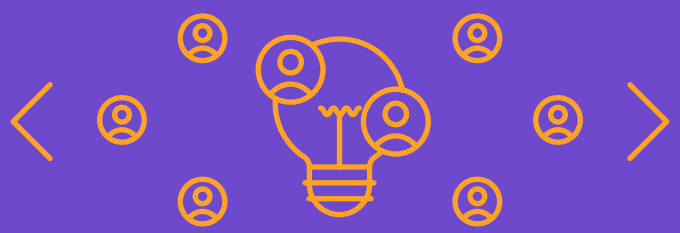
파트 5

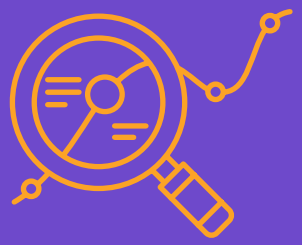
정기적인 코드 리뷰 수행

개요

정기적인 코드 리뷰 문화를 구축하면 지속적인 개선을 보장하고 불안정한 코드를 방지해 줍니다. 모든 팀원은 누구나 다른 사람의 코드를 검토하고 의견을 제시할 수 있습니다.

검토할 코드가 생기면 바로 프로젝트에 대해 잘 아는 동료, 팀원 또는 업무영역 전문가에게 코드 리뷰를 요청해야 합니다.





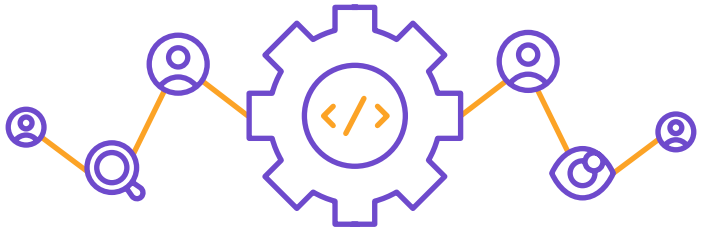
코드 리뷰를 수행할 때 팀원들은 다음을 유의해야 합니다:

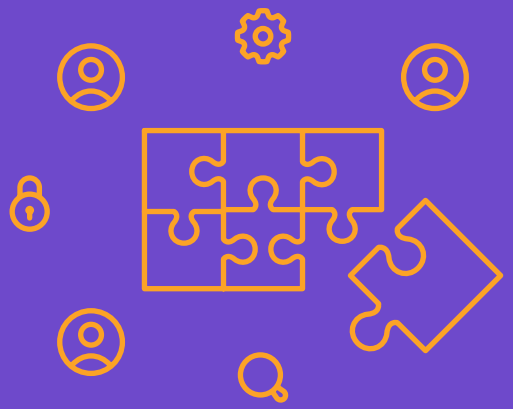
- 어떠한 변경이 필요한지 **설명해** 주어야 합니다 (예: 버그 수정, 사용자 경험 개선, 기존 코드 리팩토링).
- 사소하고 필수가 아닌 개선 사항이 보이는 경우, 코드 작성자에게 이 제안이 선택적이며, 즉시 혹은 다음 반복 주기에서 해결될 수 있음을 알리기 위해 "Not blocking:"이라는 **접두어**로 시작하는 주석을 달아 줍니다.
- 대안을 **제공**하되, 개발자가 이미 이를 고려했다고 가정해야 합니다 (예: "이 부분에서는 사용자 정의 유효성 검사를 사용하는 것이 어떨까요?")
- 반복 횟수를 줄이려면 신중히 고려해야 합니다.
- 문제를 해결하는 과정에서도 코드를 단순화시킬 수 있는 방법을 찾아야 합니다.

협업에 미치는 효과

코드 리뷰는 문제 해결과 구현에 또 다른 의견을 제시해주며, 버그, 로직 문제 혹은 잠재적인 코너 케이스를 찾을 수 있는 또 다른 눈이 되어줍니다.

코드 리뷰는 어려운 이슈들을 해쳐나가며 릴리즈를 향해 달려갈 때 발생할 수 있는 문제들을 완화시켜 줍니다. 문제 해결방안을 검토하고 의견을 제시해 줌으로써 팀원들의 협력을 통한 리뷰가 가능합니다.





코드 리뷰를 통해 협업 함으로써, 팀원들은 다양한 코딩 방법, 워크플로우 노하우 및 새로운 문제 해결 방식을 배울 수 있습니다.
이는 혁신과 효율성을 향상시키고 지식의 고립(silo)을 해소해 줍니다.

GitLab을 통한 버전 관리와 협업

GitLab은 더 좋은 소프트웨어를 더욱 빠르게 제공하기 위한 통합된 버전 관리 및 협업 (VC&C) 솔루션입니다.

모든 개발 이력과 메타데이터를 수집하는 Git 기반 웹 저장소인 GitLab은, 명확한 코드 리뷰, 산출물 버전 관리, 피드백루프 (feedback loop) 및 강력한 브랜치 패턴을 통해 능률적인 소프트웨어 개발 환경을 제공합니다. GitLab은 규정을 준수 하며 보다 빠르고 효율적으로 개발이 가능하도록 도와줍니다.

GitLab 버전 관리 및 협업은 조직에게 향상된 **가시성**과, **협업**, 그리고 **속도**를 제공합니다.

유기적인 협업은 좋은 제품을 더 빠르게 만들 수 있는 핵심입니다. GitLab의 버전 관리 솔루션은 팀이 비즈니스 목표를 달성하고 빠른 구현이 가능하도록 도와줍니다.

가시성

GitLab Insights와 Compliance Dashboard는 정보를 집계하고 대시보드를 구성할 수 있습니다. 이를 통해 병목 현상을 찾아내고 해결책을 적용하며 지속적인 개선을 촉진할 수 있습니다.

협업

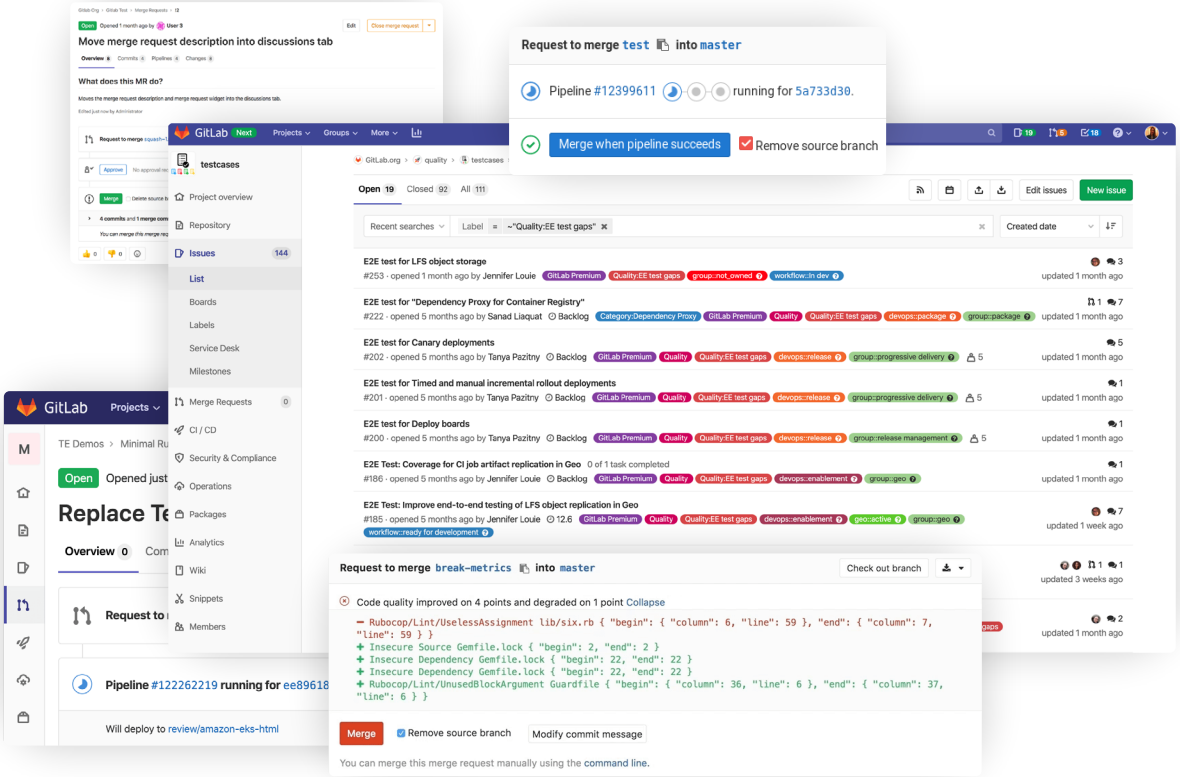
GitLab의 Merge Requests와 Code Quality는 코드 및 프로젝트 자산의 변경과 관련된 모든 측면을 관리할 수 있는 완벽한 화면을 제공하며, 검토를 통해 피드백을 제공하고 코드를 개선하기 위한 즉각적인 조치를 취할 수 있습니다.

속도

GitLab Merge Requests는 자동화된 코드 병합 승인과 문제 해결의 촉진을 통해 지루한 작업을 자동화함으로써, 개발자가 기능 구현에 집중할 수 있도록 합니다.

GitLab의 버전 관리는 독보적인 소스 코드 관리를 통해, 개발팀의 공유, 협업과 생산성 극대화에 도움을 줍니다. GitLab은 모든 프로젝트의 중요한 파일과 자산을 하나로 모아, 단일 애플리케이션 상에서 전체 소프트웨어 개발 라이프 사이클에 걸친 손쉬운 소스 코드 관리를 제공합니다.

GitLab 무료 체험하기



GitLab 무료 체험하기

GitLab의 모든 기능 — 30일간 무료
GitLab은 단순한 버전 관리나 협업 툴 그 이상입니다. 전체 소프트웨어 라이프 사이클 관리 및 DevOps를 위한 단일 애플리케이션 입니다.

무료체험 시작하기

About GitLab

GitLab은 제품, 개발, QA, 보안 및 운영 팀이 동일한 프로젝트에서 동시에 작업할 수 있도록 DevOps 라이프 사이클의 모든 단계를 위한 단일 애플리케이션으로 개발된 DevOps 플랫폼입니다.

GitLab은 DevOps 라이프 사이클에서, 단일한 데이터 저장소, 사용자 인터페이스 및 권한 모델을 제공합니다. 보다 효율적인 협업과 집중을 통해 개발 주기를 대폭 단축시켜 줍니다.

오픈 소스를 기반으로 하는 GitLab은 수천 명의 개발자와 수백만 사용자의 커뮤니티 기여를 통해 새로운DevOps 혁신을 지속적으로 제공합니다.

티켓마스터, 재규어 랜드 로버, NASDAQ, 디쉬 네트워크, 컴캐스트를 포함하여 신생 기업에서 글로벌 기업에 이르는 **10만 개 이상의 조직이** 빠른 속도로 뛰어난 소프트웨어를 제공하기 위해 GitLab에 의지하고 있습니다.

GitLab은 세계 최대의 원격 기업으로 65개국 이상에 1,250명 이상의 팀원이 있습니다.

GitLab.com에서 더 알아보기

GitLab 한국 공식 파트너 InfoGrab 입니다. / https://insight.infograb.net

담당자 : 신철호 dexter@infograb.net