

Agile 및 Scrum 방법론의 이해



Agile 개발 방식의 철학

변화에 빠르게 대응하고, 고객 중심의 개발 강조

소통과 협력을 통한 팀 중심의 접근법

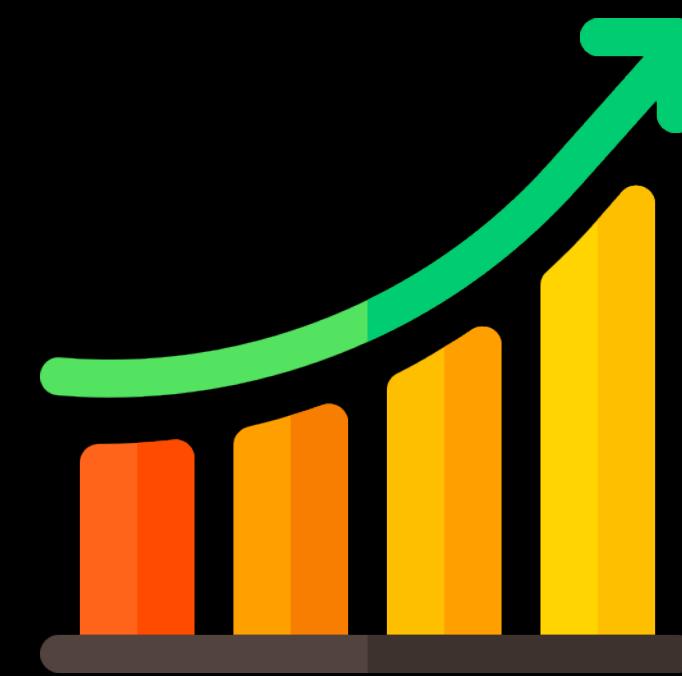
Agile의 3가지 핵심 개념



적응성



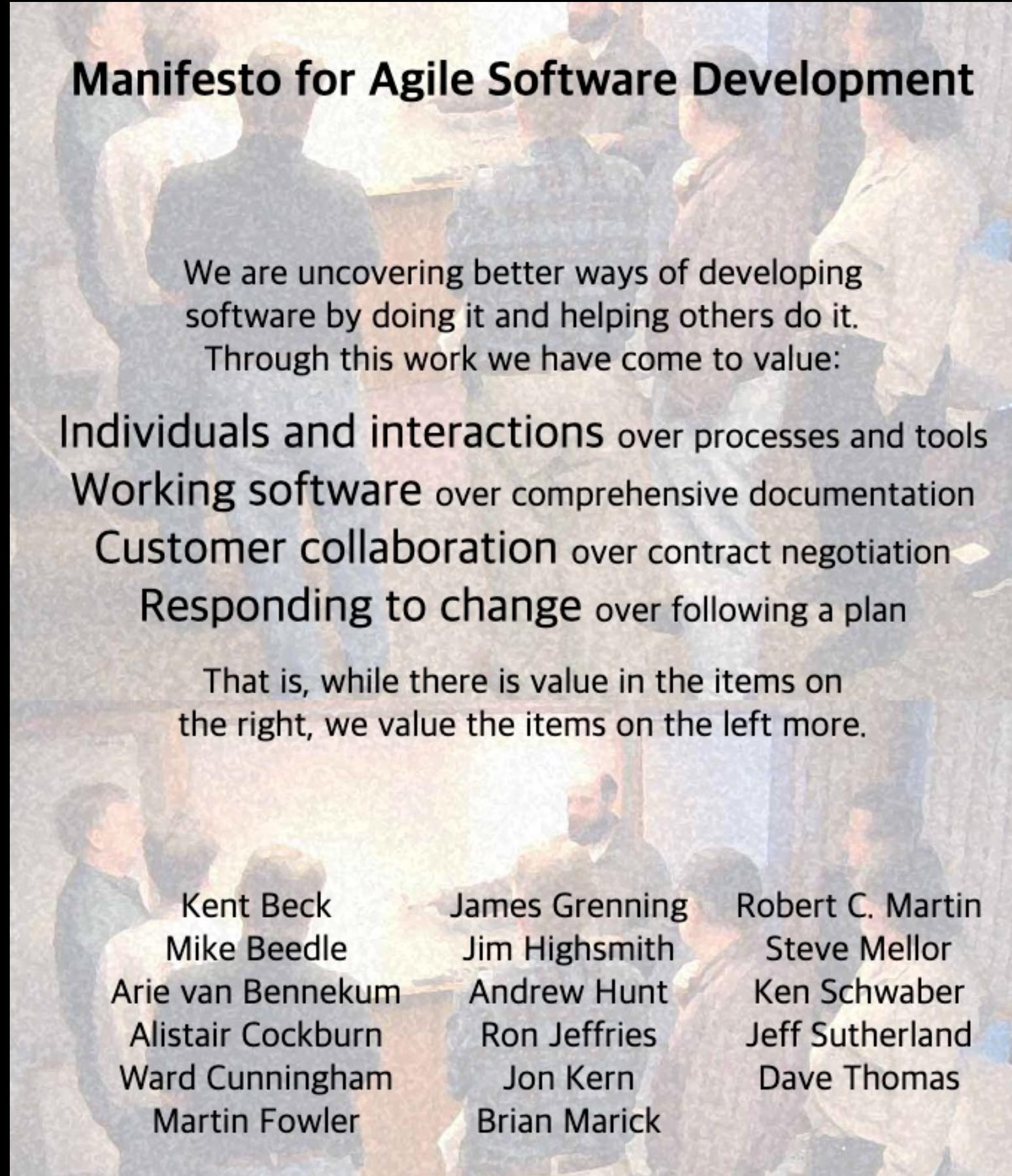
협력



지속적 개선



Agile의 역사적 배경



2001년 미국 유타주 스노우버드에서 17명의 개발자들이 **Agile Manifesto**를 작성

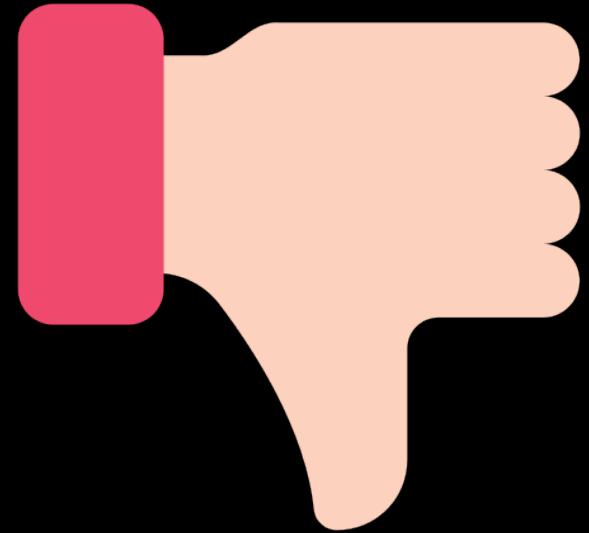
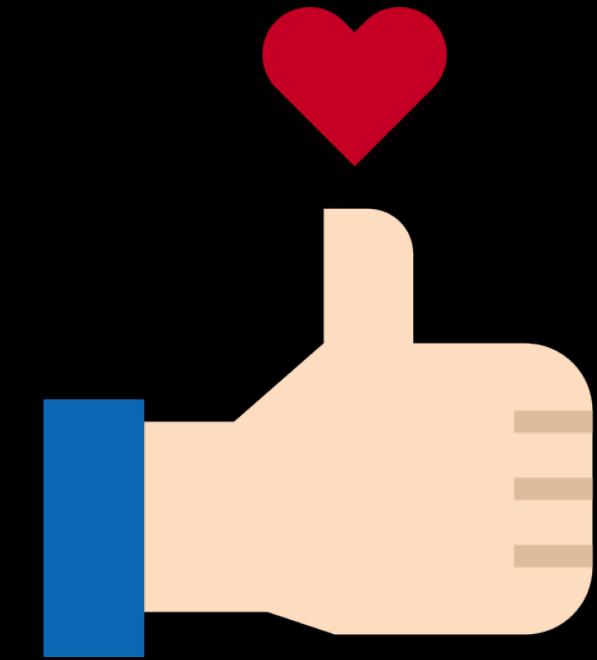
기존 개발 방식(**Waterfall**)의 비효율성을 극복하기 위한 대안으로 등장

[Waterfall 방식의 한계]

- 긴 개발 주기
- 변화에 대한 적응 부족
- 고객 요구사항과의 불일치

Agile Manifesto의 4가지 가치

Agile Manifesto



- 개인과 상호작용
- 동작하는 소프트웨어
- 고객과의 협력
- 변화에 대한 대응
- 프로세스와 도구
- 포괄적인 문서화
- 계약 협상
- 계획을 따르는 것



Agile Manifesto의 실제 적용

1. 개인과 상호작용



팀 내 정기 미팅

2. 동작하는 소프트웨어



빠른 고객 피드백 반영



Agile Manifesto의 실제 적용

3. 고객과의 협력



고객 요구사항 적극 반영

4. 변화 대응



최신 요구사항 반영



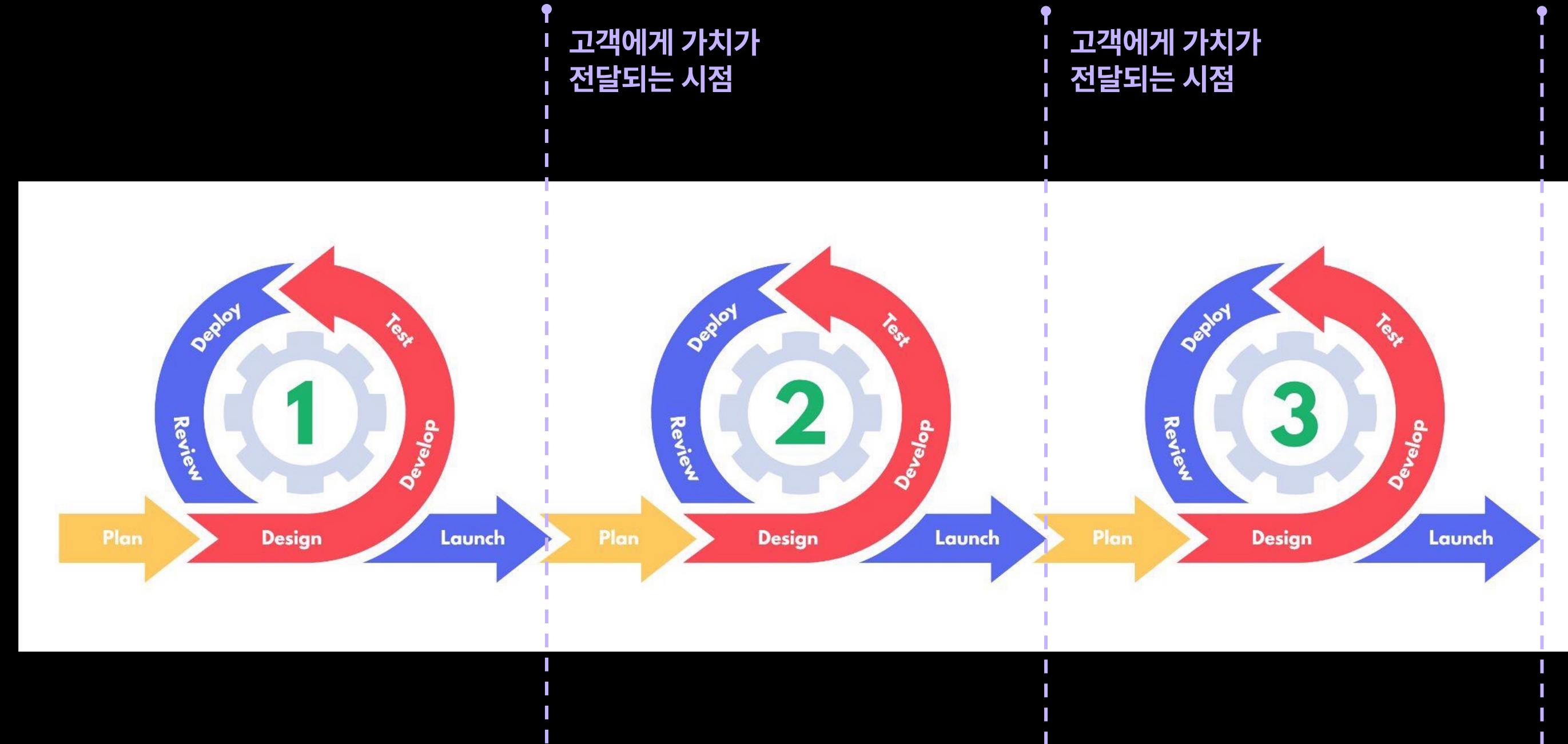
Agile 12가지 원칙

Agile의 12가지 실질적 행동 지침

초기부터 지속적으로 고객 만족 	12 Agile Principles @OlgaHeismann 요구사항 변경 수용	짧은 배포 간격 함께 일하기
동기 부여된 팀원들로 프로젝트팀 만들기 	얼굴 보고 대화하기	동작되는 소프트웨어로 진도 측정 지속 가능한 개발 속도 유지
좋은 기술, 설계에 관심 	단순성	자기 조직화 팀 정기적으로 효율성 제고



Agile 원칙의 세부내용



- **가치 전달 우선:** 사용자가 가치 있게 느끼는 소프트웨어를 지속적으로 제공
- **변화 수용:** 초기 계획에서 벗어나더라도 고객 요구를 반영
- **짧은 개발 주기:** 2~4주 단위로 작업을 반복하여 기능을 점진적으로 추가

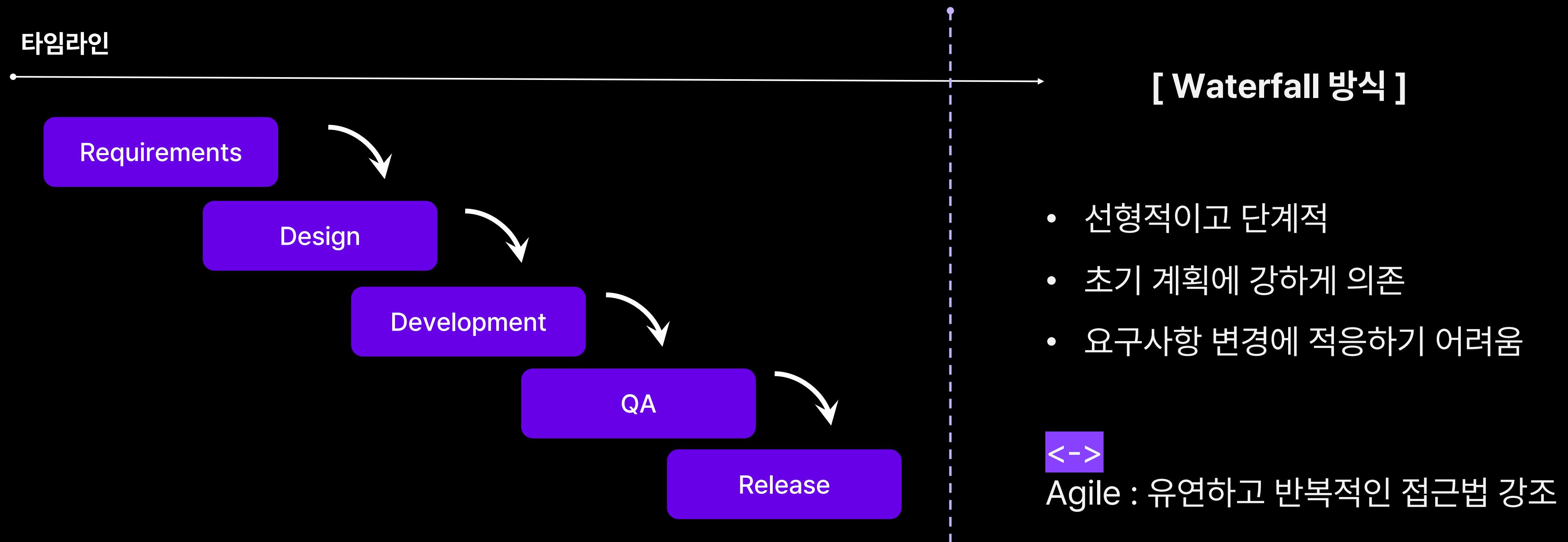


Agile 원칙의 세부내용



- **팀 중심 협력:** 개발자와 비즈니스 팀의 협력을 강화
- **성과 측정:** 완료된 소프트웨어를 성과의 주요 척도로 평가
- **지속 가능한 개발:** 개발자와 팀이 일정한 페이스를 유지하도록 관리

☰ Waterfall 방식과 Agile의 차이



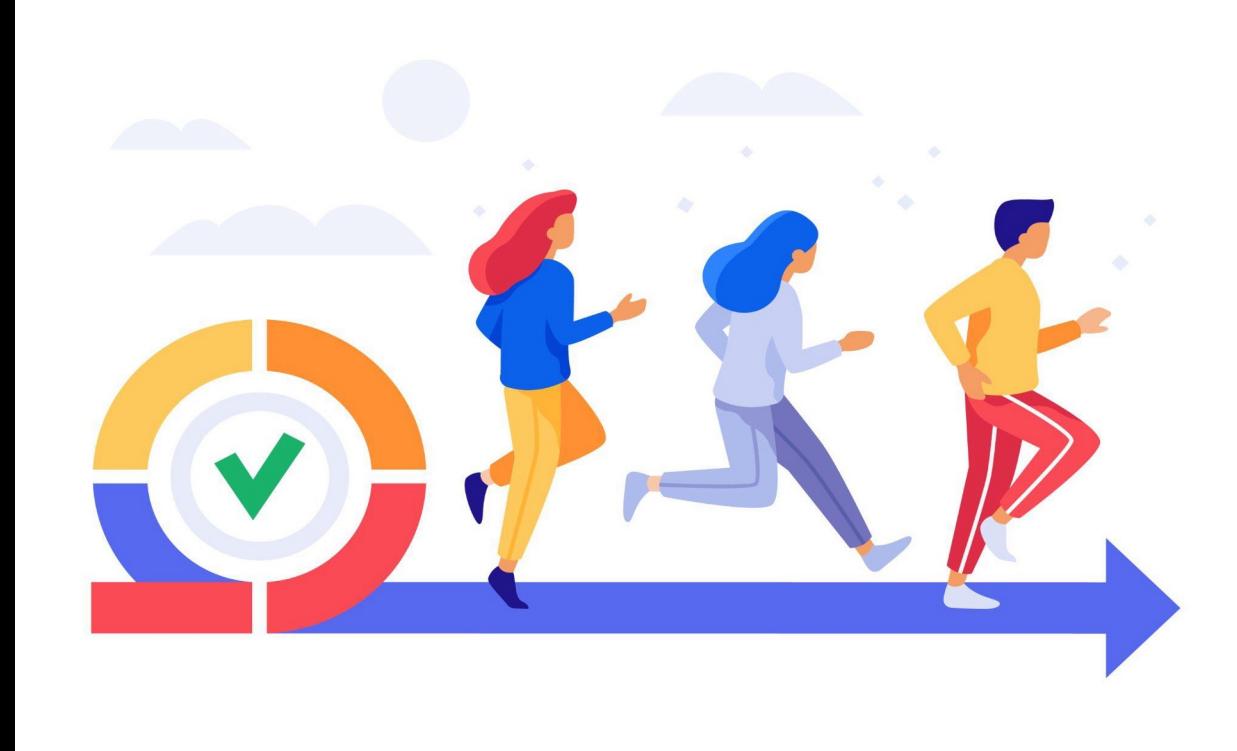


Waterfall 방식과 Agile의 차이

특징	Waterfall	Agile
개발 단계 진행 방식	순차적 단계별 진행	반복적이고 점진적인 개발
변경 수용성	초기 계획 이후 변경이 제한적	지속적 요구사항 변경과 고객 피드백 수용
고객 참여	개발 초기 단계에서만 제한적 참여	개발 전 과정에서 지속적 참여와 피드백
문제 식별 시점	후반부(테스트 단계)에서 주로 발견	초기 단계에서 문제 식별과 즉각적 해결
계획과 문서화	상세한 초기 계획과 문서 중심	최소한의 문서화와 실질적 작업에 초점
프로젝트 종료 시점	모든 개발 단계 완료 후 결과물 제공	주기적으로 실행 가능한 결과물 제공

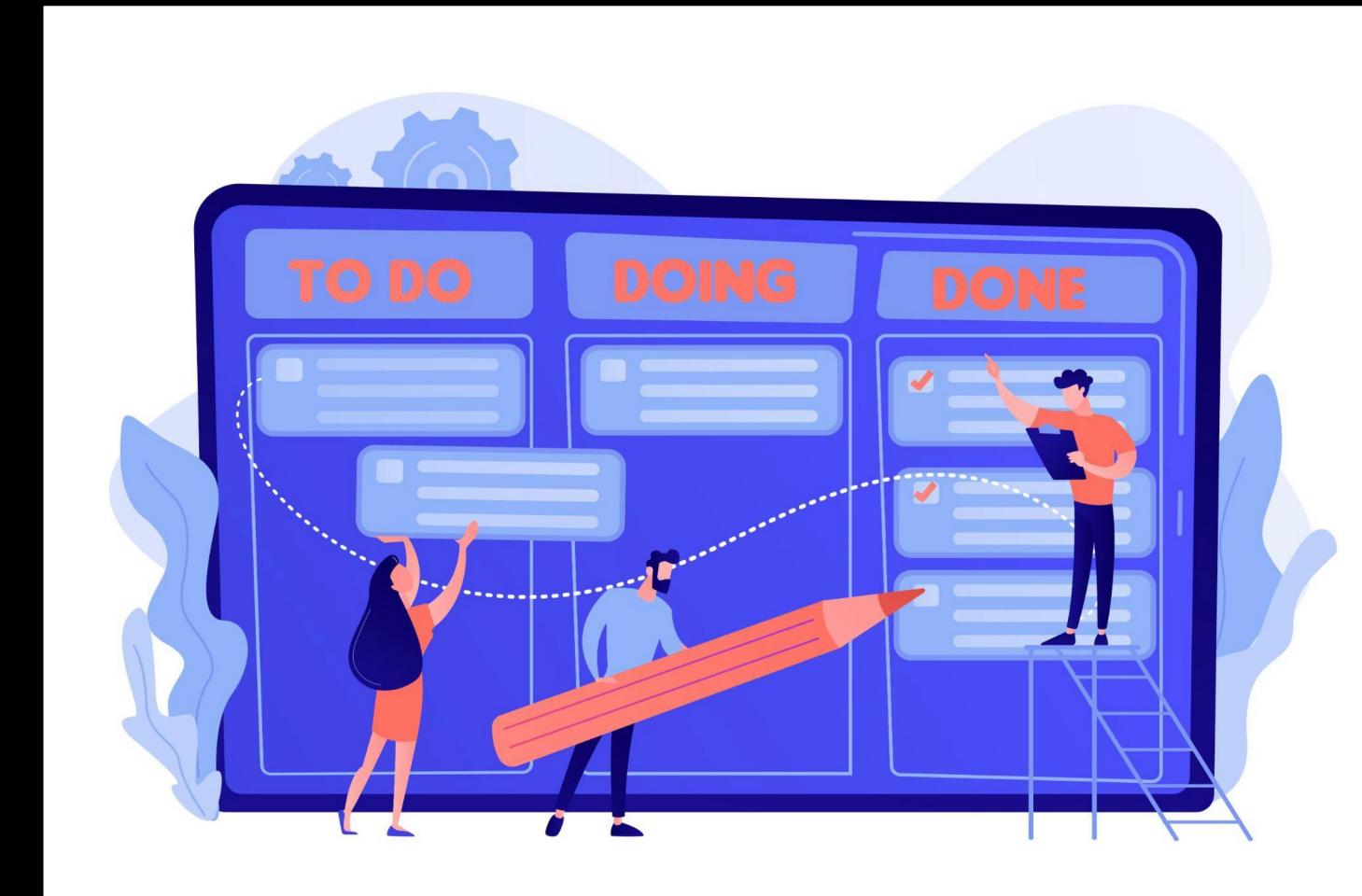


Agile의 다양한 프레임워크



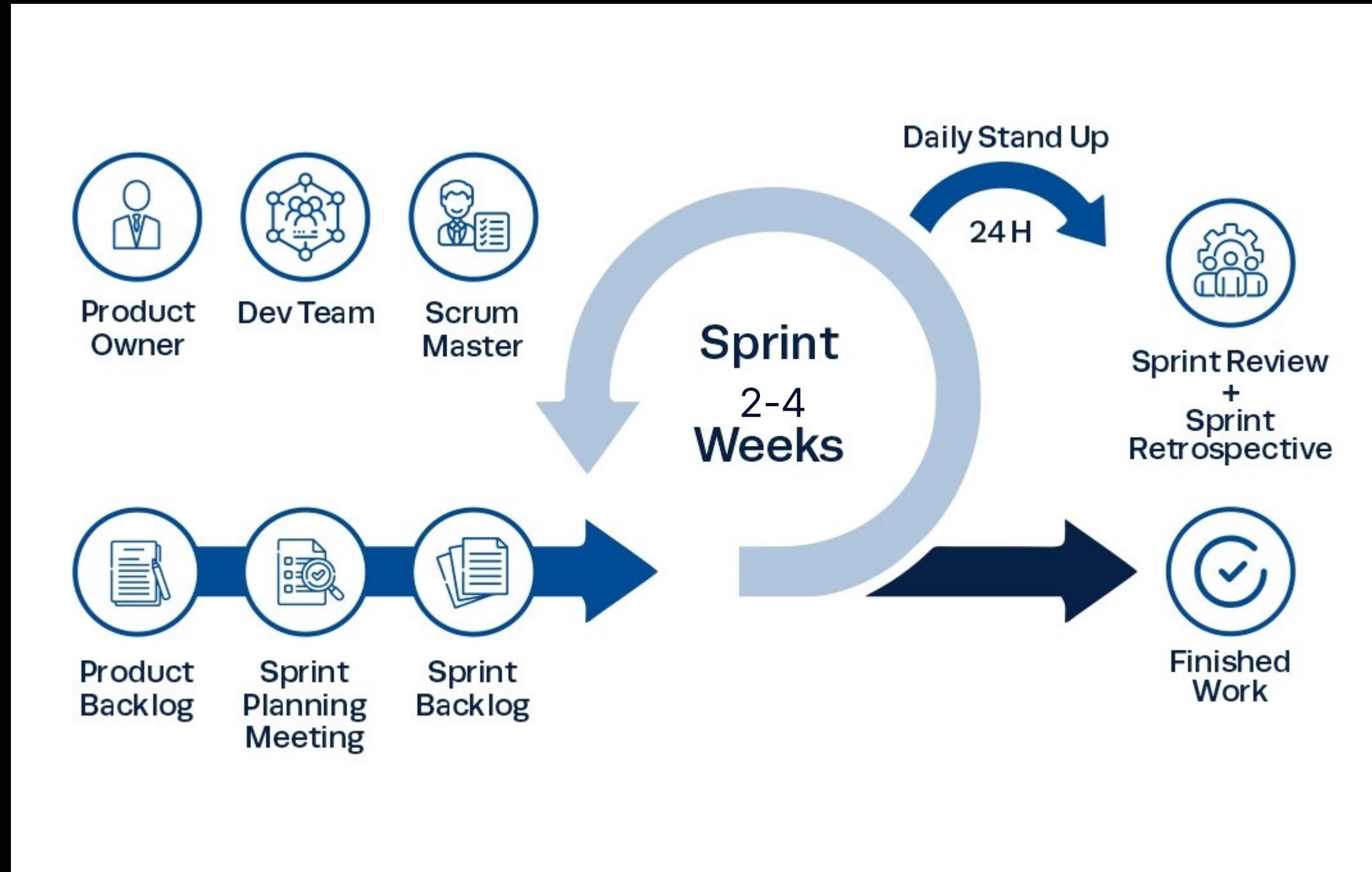
Scrum: 반복적이고 점진적인 Sprint 기반 관리

Kanban: 작업 흐름과 가시성을 중점으로 관리





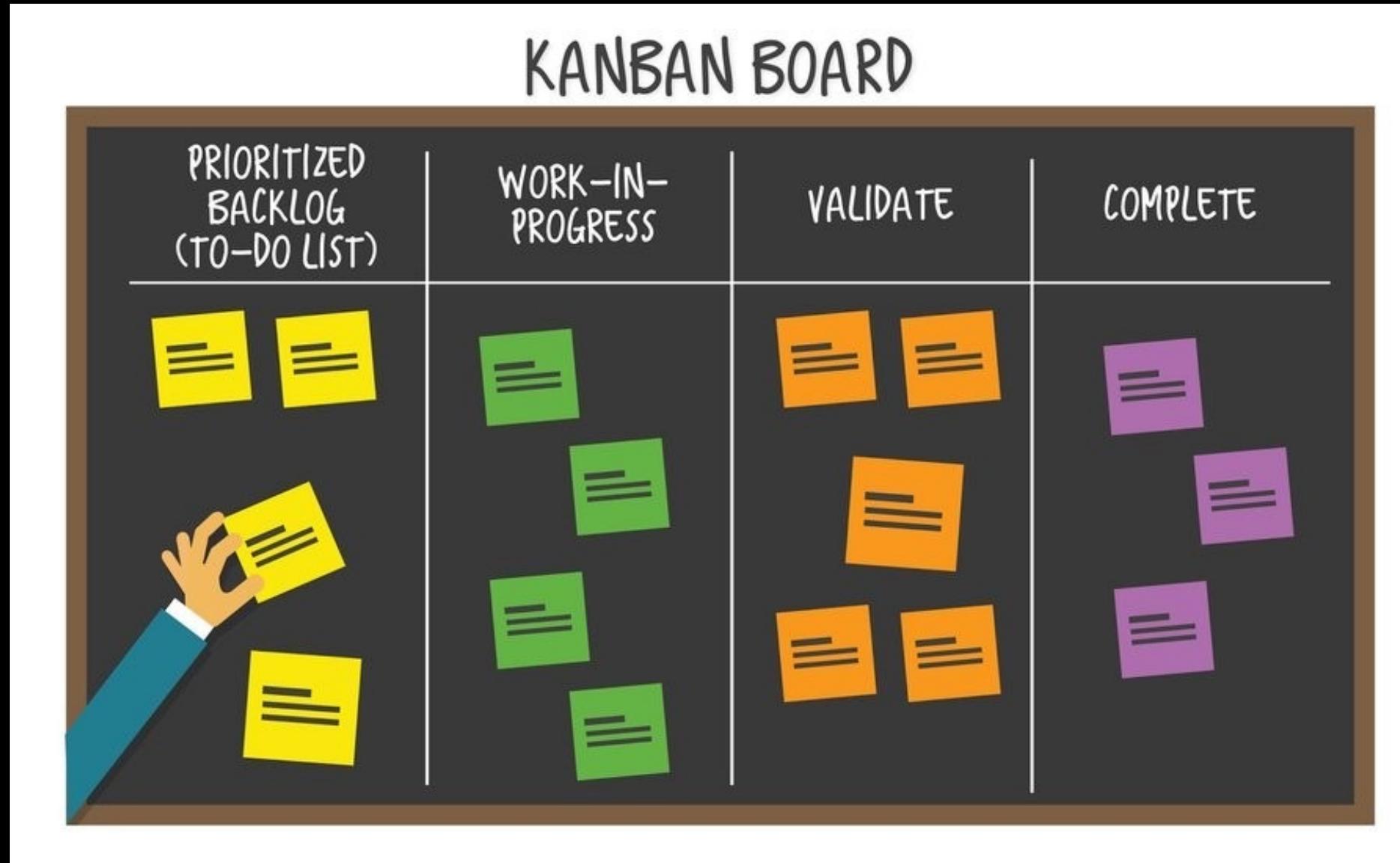
Scrum의 개념과 특징



- 반복적이고 점진적인 Sprint(2~4주 단위) 관리
- Product Owner, Scrum Master, 개발 팀으로 역할 분담
- Daily Scrum, Sprint Review 등 정기적 이벤트 활용
- Scrum은 빠른 피드백과 적응성을 강조하며, 소규모 팀에 적합



Kanban의 개념과 특징

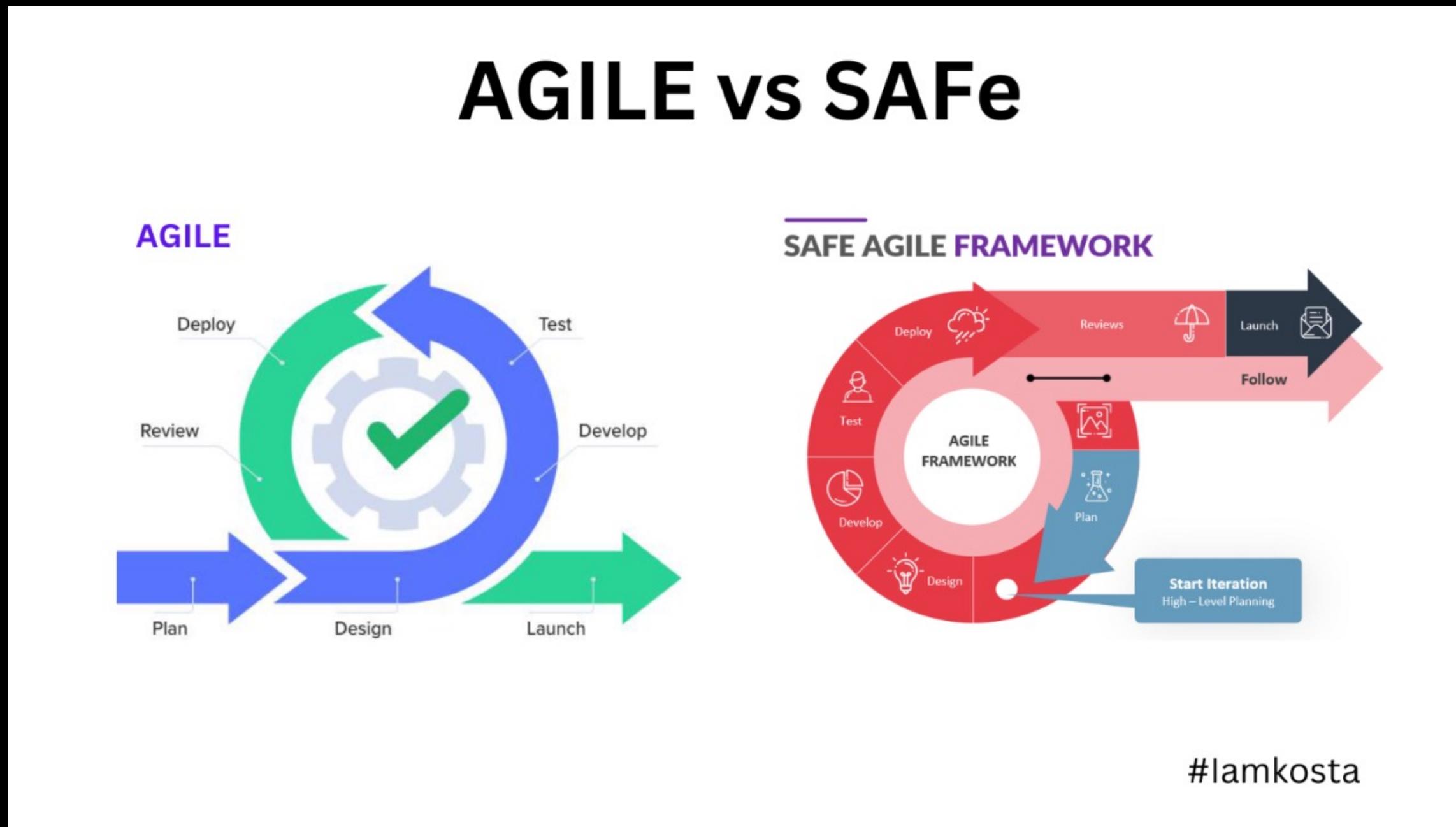


- 작업 흐름(Workflow) 시각화: Kanban 보드 사용
- WIP(Work In Progress) 제한: 작업 병목 현상 방지
- 작업의 지속적 흐름을 유지하여 낭비 최소화
- Kanban은 작업 진행 상황을 명확히 하며, 지속적인 개선에 중점



SAFe의 개념과 특징

AGILE vs SAFe



- Scaled Agile Framework
 - 대규모 조직에서 Agile 도입을 지원하는 프레임워크
 - 프로그램, 포트폴리오 관리 계층 추가
 - 여러 팀 간의 협업과 동기화를 지원
 - SAFe는 Agile을 확장하여 대규모 환경에서도 일관성을 유지



고객 중심 개발의 개념



고객 중심 개발

고객의 요구와 가치를 최우선으로 두는 개발 방식

- 정기적인 고객 리뷰(Sprint Review) 활용
- 빠른 배포를 통해 사용자 경험을 즉시 반영



고객 중심 개발의 중요성



- 제품 만족도 증가
- 시장 요구 변화에 빠르게 대응
- 불필요한 기능 개발 방지
- 고객 피드백을 통해 요구사항의 우선순위 조정



Agile 팀의 자율성



- 팀의 창의성과 동기 부여 향상
- 문제 해결 속도 증가
- 자율성은 팀의 높은 책임감을 전제로 함

팀이 작업 방식을 스스로 결정



Agile 팀의 책임 문화



책임 문화

각 팀원이 자신의 작업 결과에 책임을 지는 문화

- 팀의 목표 달성을 위한 협업
- 개인과 팀 모두의 책임 공유
- 책임 문화는 자율성을 강화하는 핵심 요소



자율성과 책임의 상호작용

자율성



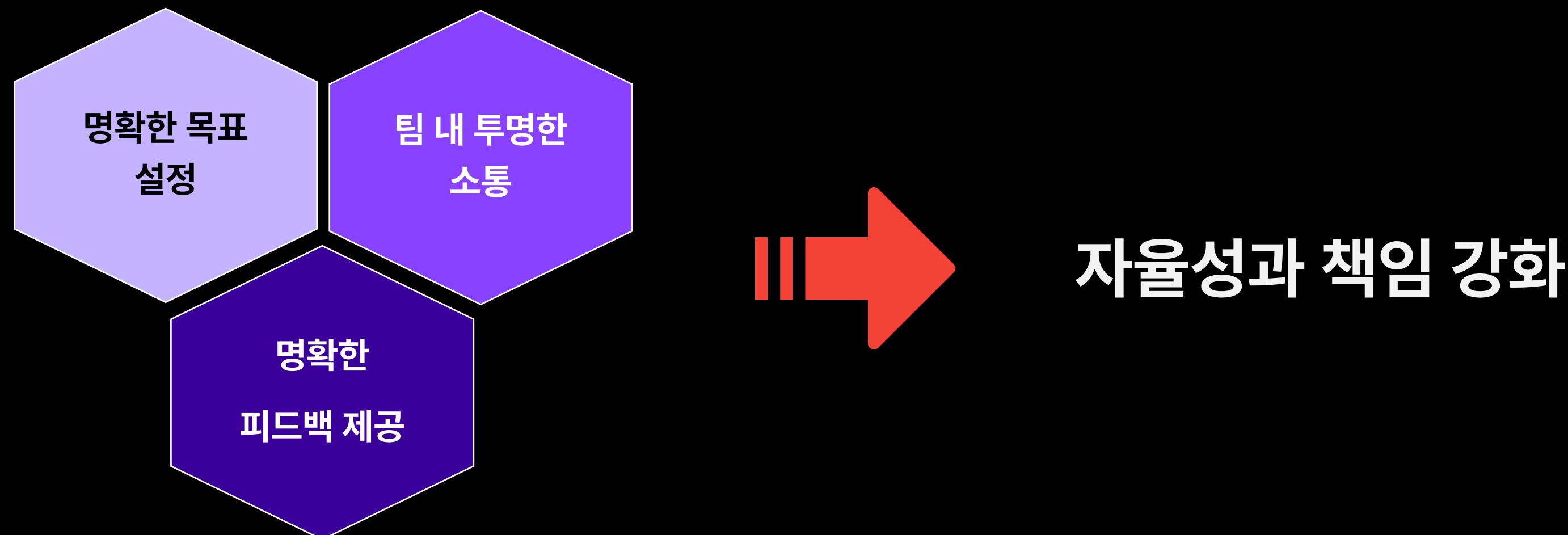
책임

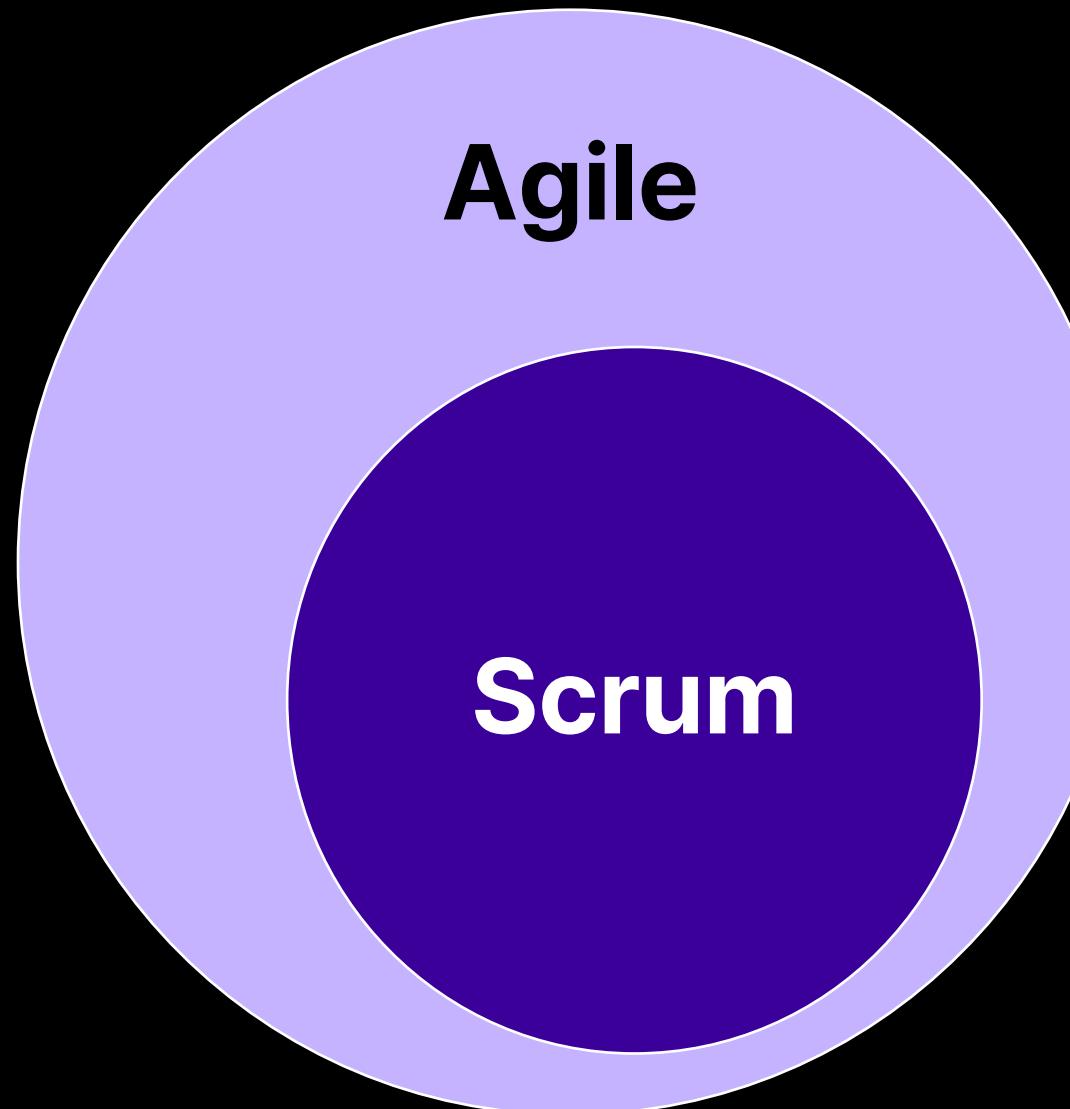


Agile 팀의 성공적인 운영을 위한 요소



자율성과 책임 극대화 방법





- Agile 프레임워크 중 가장 널리 사용되는 방식
- 반복적이고 점진적인 개발(Sprint)
- 명확한 프로세스와 작업 구조 제공
- 작은 팀과 빠른 개발 주기에 최적화



Scrum과 Kanban, SAFe의 차이점

특징	Scrum	Kanban	SAFe
목표	고정된 Sprint 내 작업 완료	작업 흐름(Workflow) 최적화	대규모 조직의 협업 지원
팀 구조	역할 기반(PO, SM, 개발팀)	유연한 팀 구성	여러 팀과 포트폴리오 레벨의 협업
주요 특징	반복적, Sprint 중심	지속적 흐름 관리	규모 확장에 중점, Agile Practice 확장
적합 대상	소규모 팀	모든 규모의 팀	대규모 조직 및 프로젝트
작업 계획	고정된 기간 (Sprint, 일반적으로 2주)	작업이 준비되는 즉시 처리	릴리스 계획과 대규모 협업
가시화	Sprint Backlog와 Burndown Chart	Kanban 보드	여러 레벨의 가시화 (팀, 프로그램 등)



Scrum의 3가지 역할 개요

각 역할은 독립적이지만 **상호 보완적**으로 작동



Product Owner

고객과 비즈니스 요구를 반영



Scrum Master

팀의 협업 조정 및 장애물 제거



개발 팀

제품 설계 및 개발의 실행 주체



Product Owner의 역할과 책임



Product Owner

- 제품 백로그 관리(Product Backlog)
- 고객 요구사항과 비즈니스 목표를 팀에 전달
- 우선순위 설정과 제품 비전 공유
- 고객의 목소리를 대표하며, 팀과의 소통이 핵심



Product Owner의 성공 사례



좋은 PO가 되려면?

- 명확한 우선순위 설정을 통해 개발 시간 단축
- 고객 피드백을 빠르게 반영하여 제품 가치 극대화
- 개발 팀과의 원활한 소통으로 요구사항 전달 시간 단축



Scrum Master의 역할과 책임



Scrum Master

- 팀의 장애물을 식별하고 제거
- Scrum 이벤트(Daily Scrum, Sprint Review 등) 관리
- Scrum 프로세스와 원칙이 올바르게 실행되도록 지원
- 팀의 성과를 극대화하기 위한 촉진자 역할



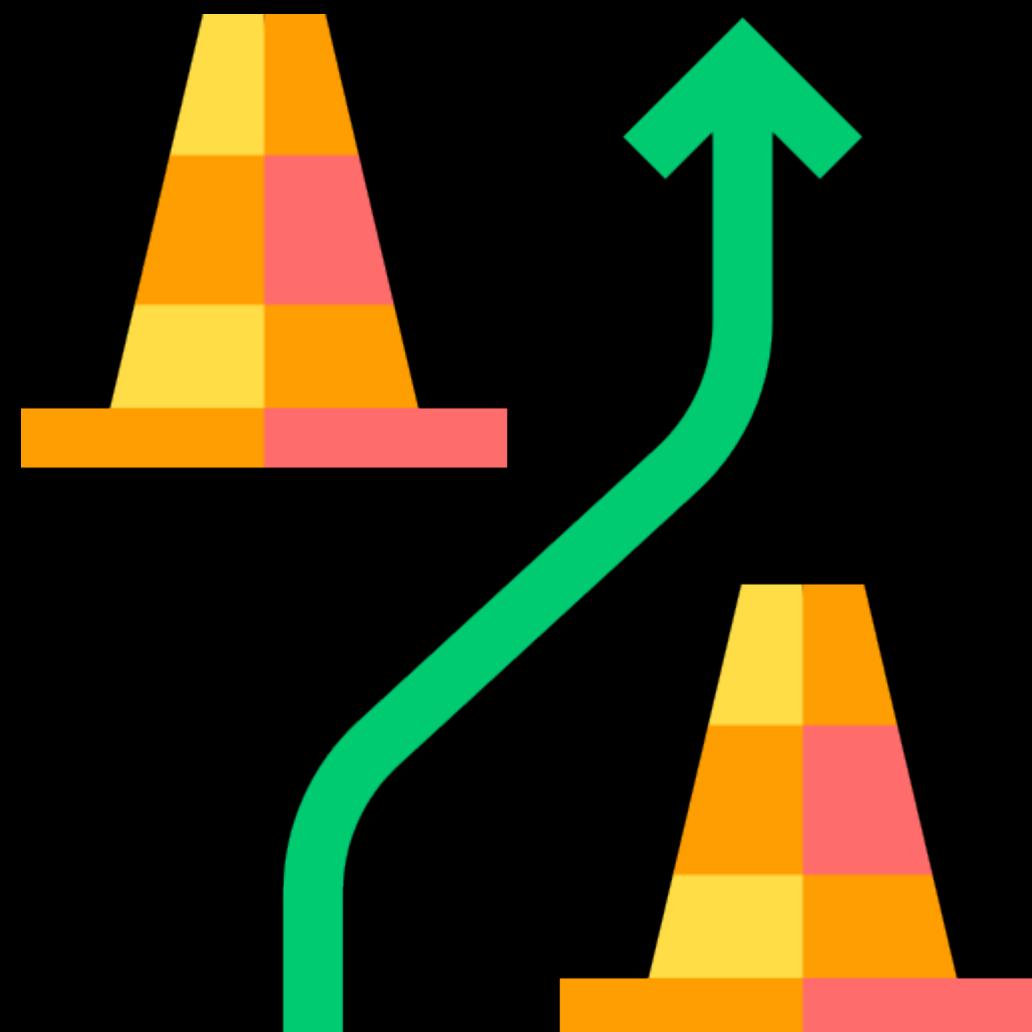
Scrum Master의 팀 조정 기술



- 갈등 관리: 팀 내 의사소통 문제 해결
- 생산성 향상: 팀의 작업 흐름 최적화
- 동기 부여: 팀원들이 목표에 집중하도록 유도



Scrum Master의 장애물 제거 성공 사례

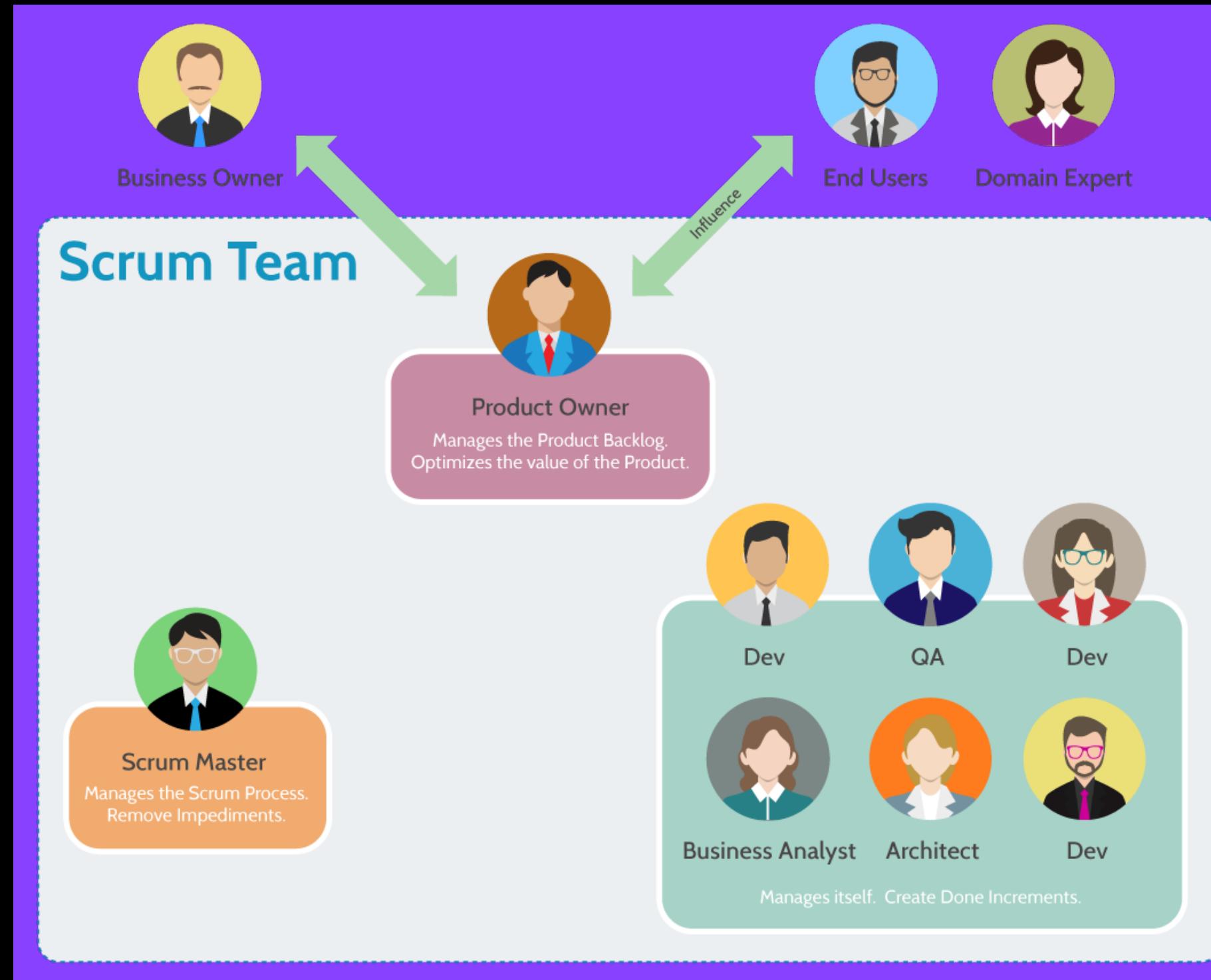


좋은 Scrum Master가 되려면?

- 팀 간 커뮤니케이션 문제 해결로 작업 지연 방지
- 외부 간섭으로 인한 작업 중단을 최소화
- 개발 환경 문제(예: 테스트 서버 부족) 해결



Scrum의 팀 기반 구조



- 자율적이고 교차 기능적인 팀 구성
- Product Owner, Scrum Master, 개발 팀 간의 명확한 역할 분담
- 의사결정 속도와 협업 효율성을 향상



Scrum 성공을 위한 핵심 요건



팀 구성원 모두의 노력

지속적인 피드백과 개선

Good Product Owner

명확한 제품 목표

팀 내 신뢰와 협업 문화

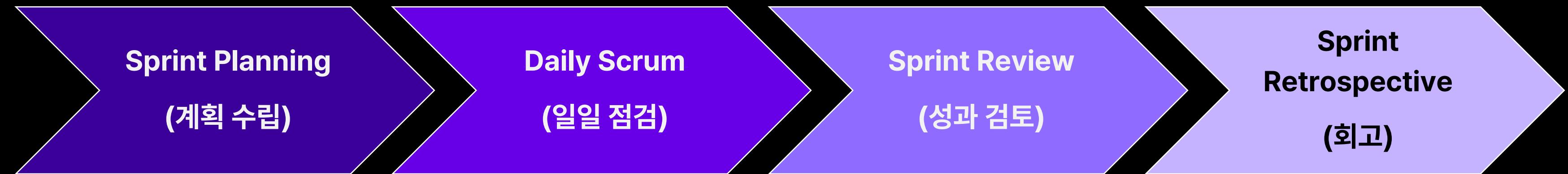
Good Scrum Master



Sprint란 무엇인가?

Sprint

: Scrum에서 정의된 일정(2~4주) 동안 특정 목표를 달성하기 위한 개발 주기



Sprint Planning
(계획 수립)

Daily Scrum
(일일 점검)

Sprint Review
(성과 검토)

Sprint
Retrospective
(회고)

Sprint 기간 동안 매일 점검

스프린트 목표가 해당 기간 동안
얼마나 달성됐는지 검토

회고에서 나온 피드백으로
새로운 스프린트 시작

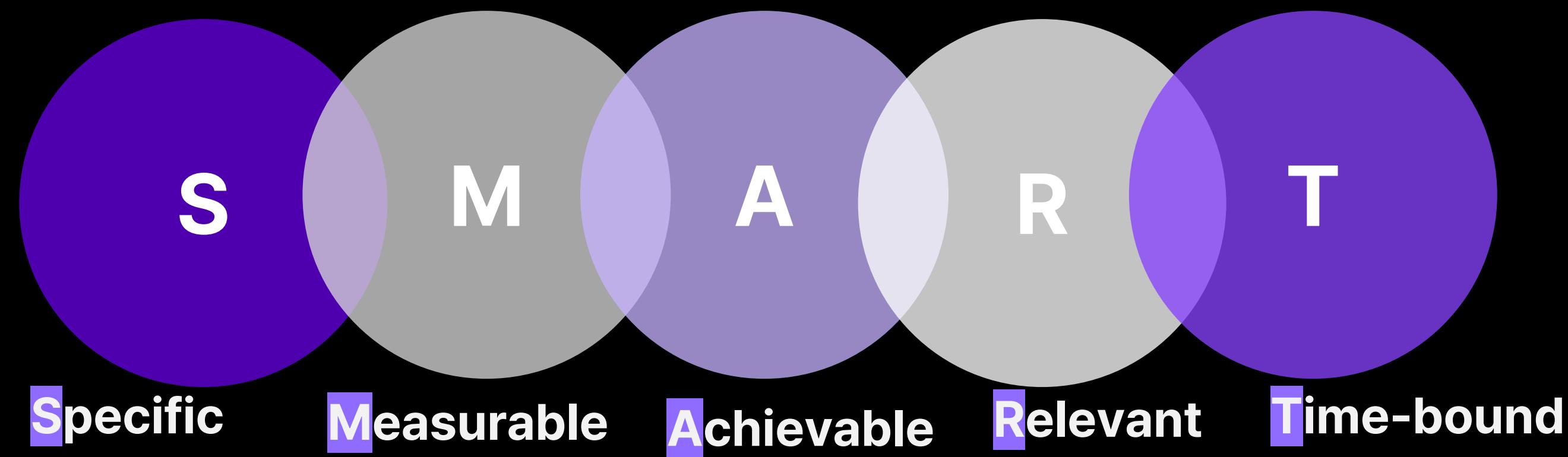


Sprint 목표 설정 방법

Sprint 목표

Sprint 기간 동안 달성해야 할 핵심 목표
팀의 방향성을 제공하고, 초점을 맞추도록 도움

SMART 원칙 활용





성공적인 Sprint 실행을 위한 요건



명확한 목표



장애물 제거



효율적인 커뮤니케이션

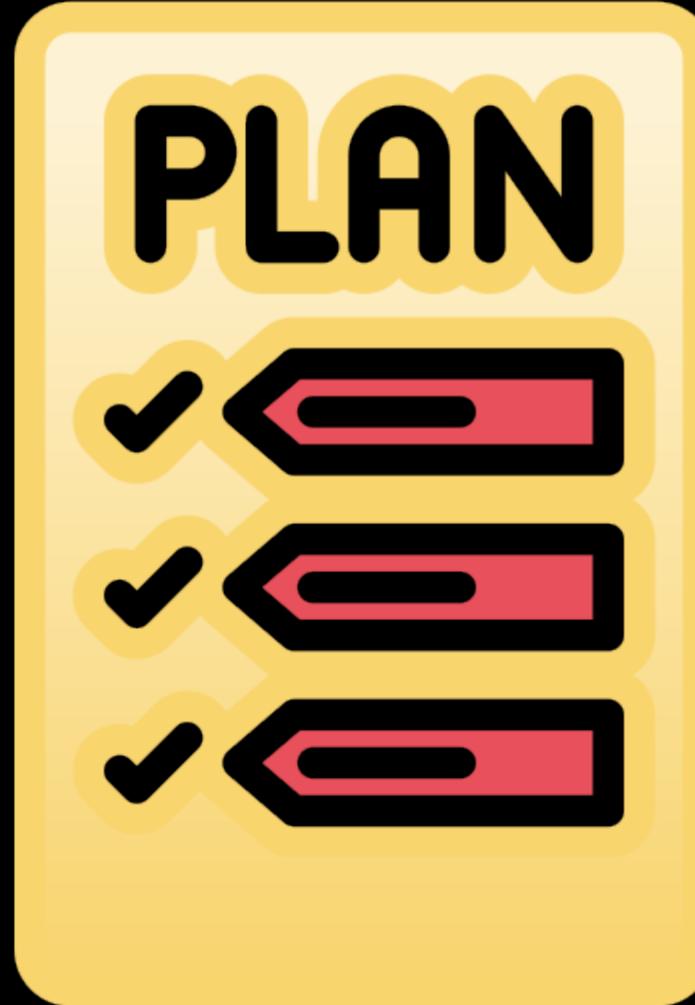


우선순위 관리

모든 정의된 작업이 완료되었는지 확인 후 종료



Sprint Planning이란?



Sprint Planning

Sprint 시작 전에 목표와 작업 계획을 수립하는 이벤트

- Sprint 목표 정의
- Product Backlog에서 작업 항목을 선정
- 팀의 작업 가능성(velocity) 평가

Backlog를 효과적으로 관리하는 방법

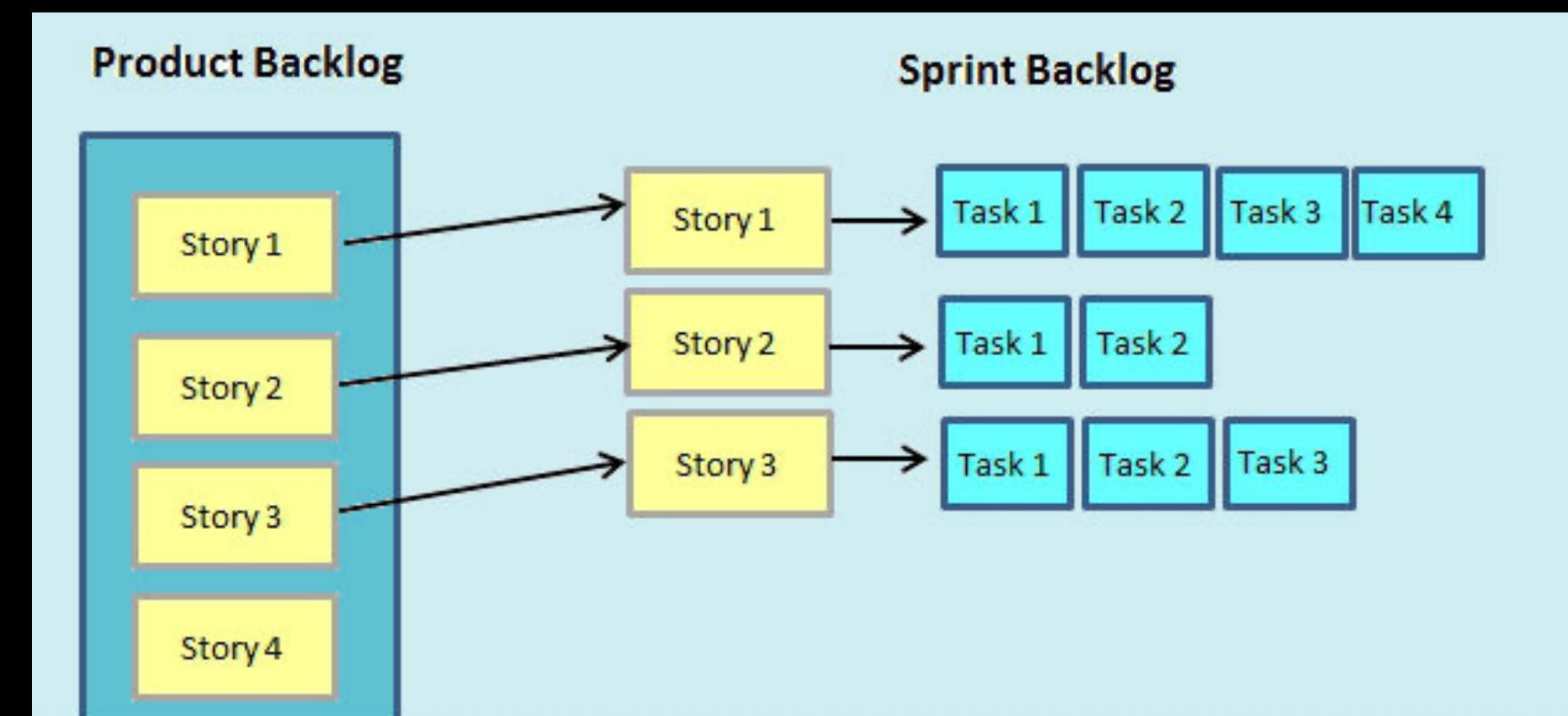
Backlog 관리

Product Backlog에서 Sprint에 포함할 작업 항목을 선택 및 정리

Product Backlog - 제품 개발을 위해 해야 할 모든 작업의 리스트

Sprint Backlog - 특정 Sprint 동안 수행할 작업의 리스트

"Sprint Backlog는 Product Backlog의 부분집합"



Backlog 우선순위를 설정하는 전략

MoSCoW 분석(Must, Should, Could, Won't)



- **고객 가치:** 사용자가 가장 필요로 하는 기능을 우선 개발
- **비즈니스 요구:** 회사의 전략적 목표와 일치
- **기술적 위험:** 복잡한 항목은 초기 Sprint에 포함
- **작업 규모:** 팀이 Sprint 내에 완료 가능한 항목 선택

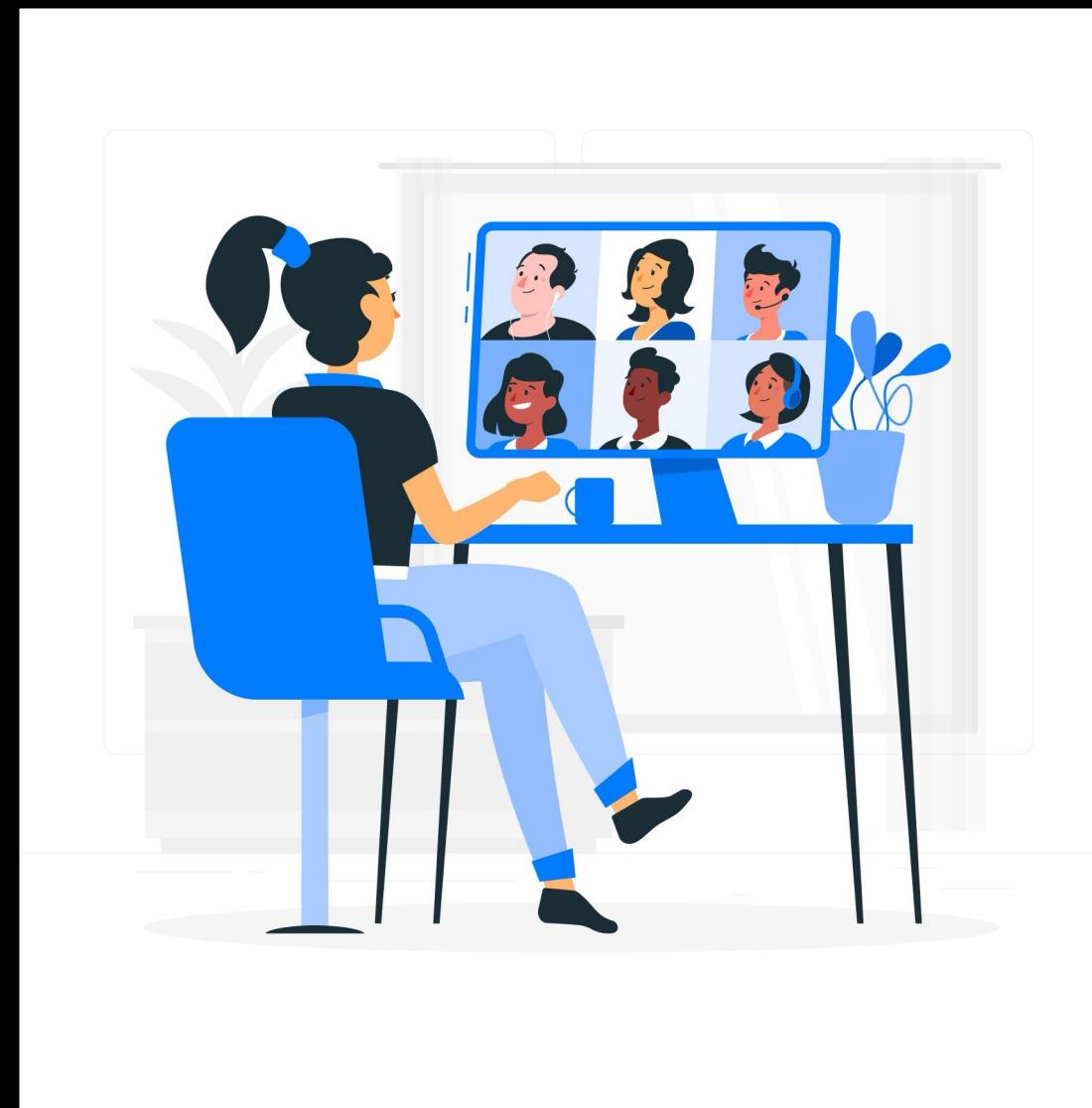


Daily Scrum이란?

Daily Scrum

- 매일 15분간 진행되는 팀 점검 회의
- 팀원들이 작업 진행 상황을 공유하고, 장애물을 식별
- 팀의 협업과 목표 집중을 유지하는 데 필수적

“어제 무엇을 했는가?”
“오늘 무엇을 할 것인가?”
“작업을 방해하는 장애물은 무엇인가?”





Daily Scrum을 효과적으로 운영하는 방법

A screenshot of a Slack 'Thread' titled 'Standuply'. It shows a message from 'Standuply APP' with a blue speech bubble icon, dated 'Today at 12:56 PM', which reads: 'Check this out! Here are the results on Standup report from 2017-08-10 😎'. Below it, 'Gleb Dvoryatkin' (with a Google icon) replies 17 minutes ago: 'What did you do yesterday? Working on market analysis', 'What do you plan on doing today? Customer development activities', and 'Okay, any obstacles? No'. Then 'Artem Borodin' (with a person icon) replies 16 minutes ago: 'What did you do yesterday? bug fixing: started to do back-end for ST-1235', 'What do you plan on doing today? back-end for ST-1235, maybe fixes for stat plugin', and 'Okay, any obstacles? I think no, I should finish according to schedule'. At the bottom is a 'Reply...' input field with a smiley face emoji and a 'Send' button.

슬랙 스탠드업 미팅 예시

성과 극대화 Tip

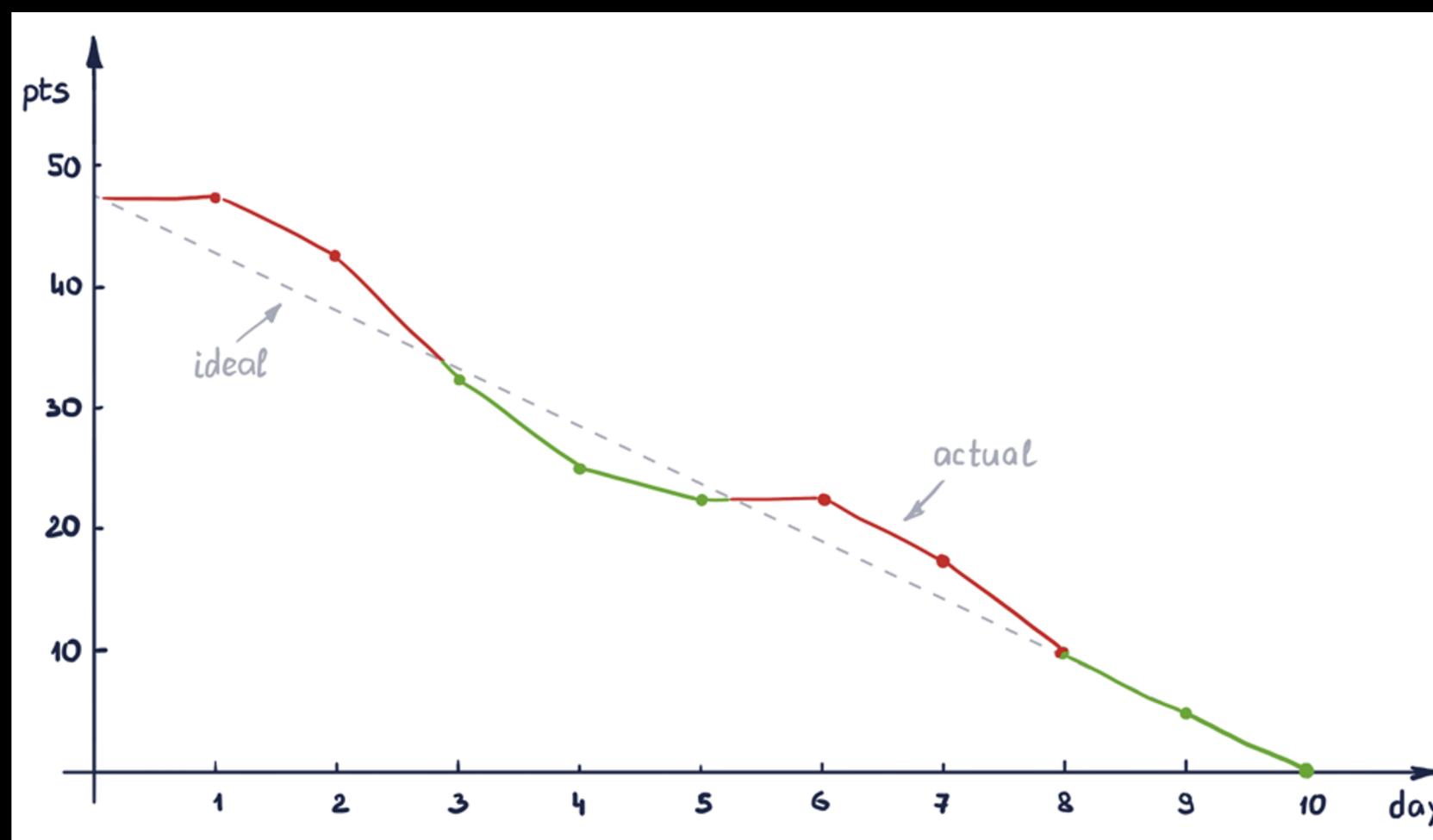
- 회의를 시간 내(15분) 완료
- 각 팀원이 간결하고 명확하게 작업 상황 공유
- Scrum Master가 장애물 제거 지원
- 회의는 작업 진척 상황에 초점
- Daily Scrum은 문제 해결보다는 공유와 협업의 장으로 활용



Sprint Review란?

Sprint Review

Sprint 종료 후, 팀이 작업한 결과물을 이해관계자와 공유하는 이벤트
제품 인크리먼트를 검토하고 피드백 수집

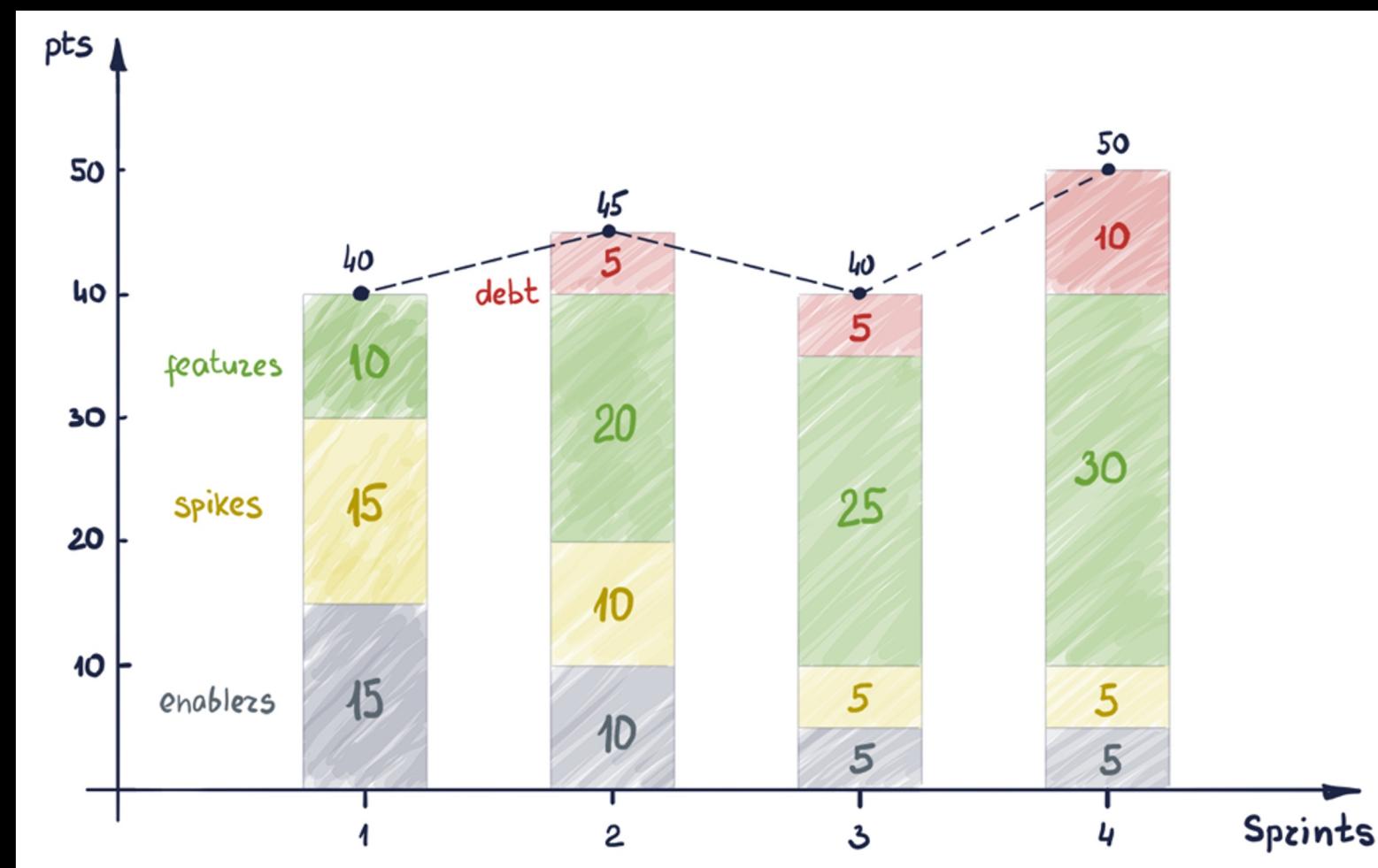


Burn-down Chart

- 스프린트 기간 대비 남아있는 작업규모 추이를 보여주는 그래프
- 백로그가 완료될 때마다 그래프가 내려가며, 스프린트에 백로그가 추가되는 등 작업량 변경 발생을 모니터링 및 감지하는데 활용



Sprint Review란?

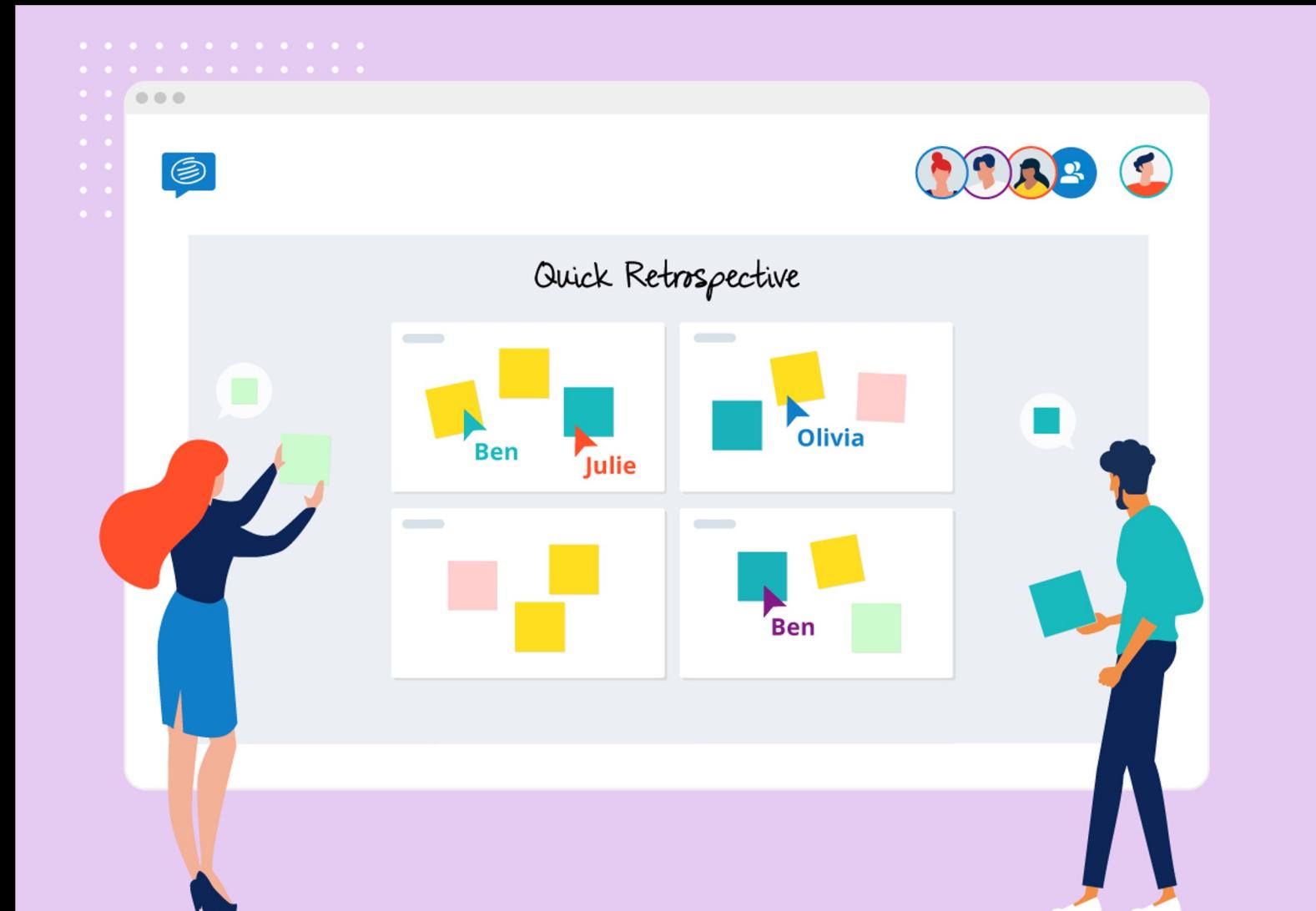


Velocity

- 매 스프린트마다 실제 완료한 총 스토리포인트의 합계 / 스프린트 횟수
- 매 스프린트마다 계획 대비 실제 완료한 결과를 정량적으로 살펴봄으로써, 다음 스프린트에 팀이 어느 정도 scope의 일을 계획하고 완료할 수 있는지 예측할 수 있음
- 작업한 기능을 시연
- 고객과 이해관계자의 피드백 수렴
- 다음 Sprint를 위한 아이디어 논의



Sprint Retrospective란?



주요 목적

- 팀의 생산성 향상
- 협업 방식 개선
- 반복적인 문제 해결

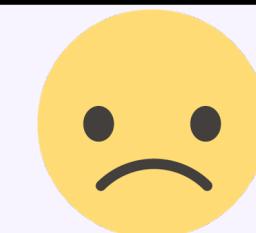
Sprint Retrospective

Sprint 종료 후 팀의 작업 프로세스와 협업 방식을 되돌아보는 회의
개선 가능한 부분을 식별하고, 다음 Sprint에 반영

Sprint Retrospective의 핵심 활동

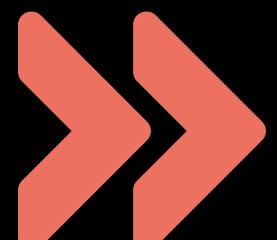


- 첫 Sprint인데도 잘 끝냄 +4
- 첫 스프린트 무사히 목표한 바 달성한 것 +1
- Daily Stand-up 시작 +3
- Stand up meeting +1
- 같은 공간에서 개발할 수 있어서 만족 +3
- QA 달려주심! Brundown +3
- 각 파트별로 협업이 잘 되었다 +4
- Scrum Reboot +1
- 제로 베이스에서 시작, 시도해볼 수 있는 것이 많았다 +1
- Sprint Review 첫 시도 +1



- 생각보다 산출물이 적어서 아쉬움 +2
- 다음 스프린트 목표는 좀 더 도전적으로?! +1
- 첫 시도인 만큼 익숙하지 않은 것이 많음. 아직도. 데모 날에는 배포 안하는 걸로 (가능하면) +1
- Daily Stand-up 같이 함께 집중해요~+4
- 커뮤니케이션 셀프 반성... 더 잘할게요! +1
- Scrum, Agile 공감대 형성 어려웠음

“무엇이 잘 되었는가?”
“개선할 점은 무엇인가?”
“다음 Sprint에서 적용할 개선 방안은?”



팀이 성공적으로 수행한 작업 식별
문제점이나 비효율적인 요소 도출
실행 가능한 행동 계획 도출



Sprint Retrospective의 효과와 중요성

- 팀의 문제 해결 능력 향상
- 협업과 소통 문화 강화
- 반복적인 문제 방지
- 팀원의 만족도와 참여도 증가





Product Backlog란?

Product Backlog

프로젝트의 모든 작업 항목을 정리한 목록

Product Owner가 관리하며, Sprint Planning의 작업 항목



구성 요소

- 기능 요구사항(User Story)
- 버그 수정
- 기술적 작업(예: 리팩토링)
- 연구 및 탐구 작업(예: 프로토타입 개발)

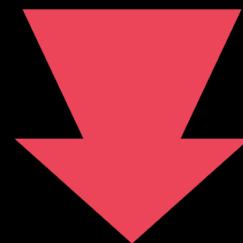


Product Backlog 관리가 중요한 이유



관리 실패

- 불명확하거나 우선순위가 없는 작업 항목
- 팀이 작업 목표나 방향성을 이해하지 못함
- 고객 요구사항 변화를 반영하지 못함
- 작업 항목이 과도하거나 누락되어 혼란 발생

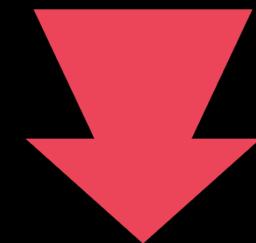


프로젝트 지연, 팀의 생산성 저하, 고객 불만족



관리 성공

- 고객 및 비즈니스 목표에 따라 명확히 설정된 우선순위
- 팀이 명확한 작업 계획과 목표를 이해하고 수행
- Product Backlog를 지속적으로 업데이트하고 반영
- 적절한 작업량과 지속적인 흐름 유지



프로젝트 목표 달성, 팀의 효율적 작업, 고객 만족



Product Backlog의 우선순위 설정 방법

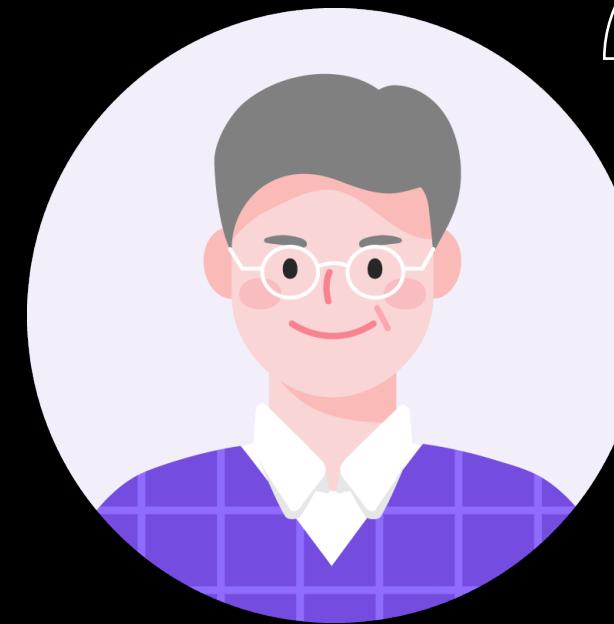
MosCoW 분석

우선순위 분류	작업 항목 예시
Must Have	- 핵심 기능 개발 (예: 사용자 로그인, 결제 시스템 구현) - 주요 버그 수정 (예: 사용자 데이터 유실 문제 해결)
Should Have	- 사용성 개선 (예: 대시보드 레이아웃 최적화) - 추가 보고서 생성 기능
Could Have	- 소셜 미디어 공유 버튼 추가 - UI 테마 옵션 제공
Won't Have	- 다국어 지원 (현 Sprint에서 제외, 추후 우선순위 재검토 예정) - 비주얼 애니메이션 추가

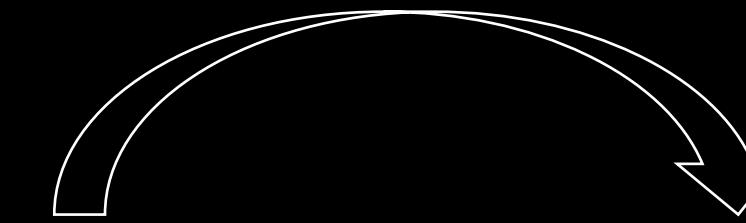
고객 가치가 높은 작업 항목
비즈니스 목표와의 일치
기술적 위험도가 높은 작업
Sprint 기간 내 완료 가능한 작업

우선순위 ↑

Product Backlog 관리 모범 사례



Product Owner



- 정기적으로 Backlog를 검토하고 정리
- 작업 항목을 작은 단위로 나누어 관리
- 각 항목에 명확한 완료 기준(Definition of Done) 설정
- 고객 피드백을 기반으로 Backlog를 업데이트

☰ Product Backlog 관리 모범 사례

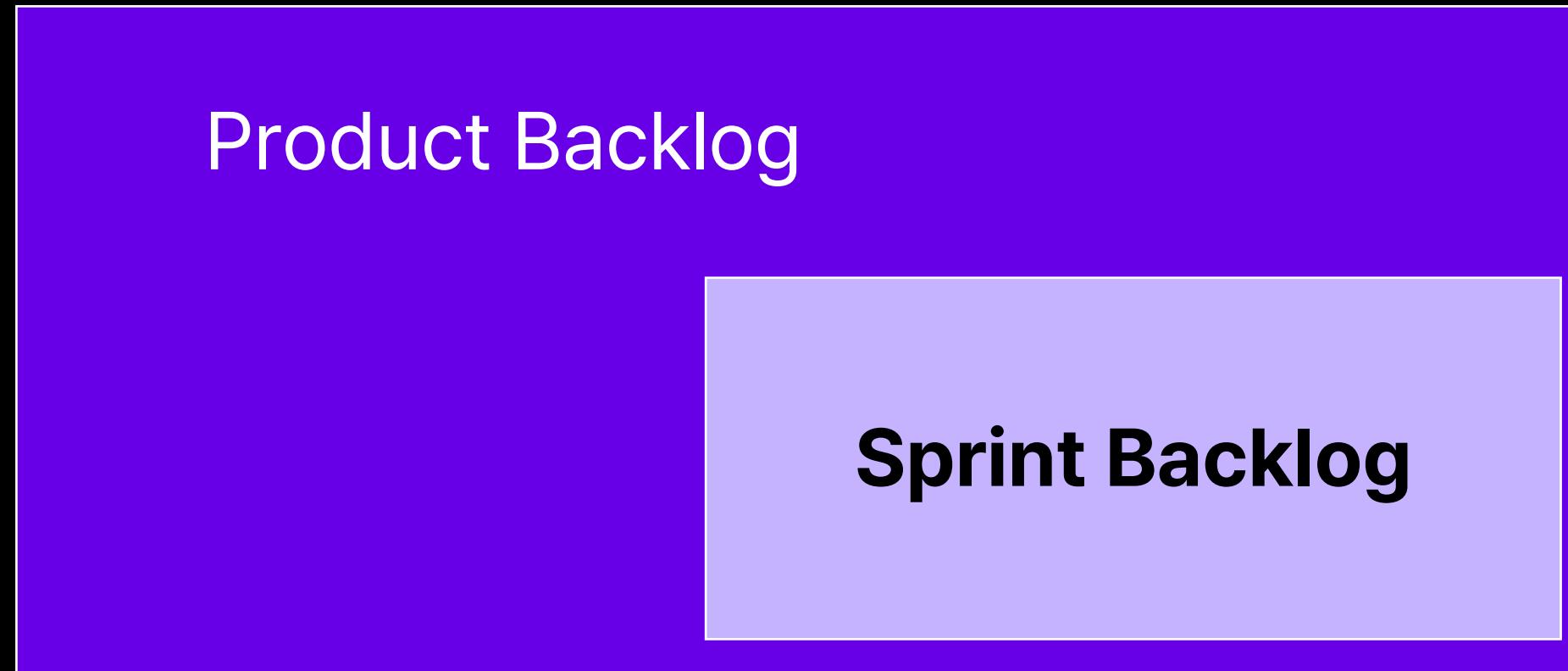
제품 백로그 예시: 온라인 쇼핑몰 개발 프로젝트

우선순위	백로그 항목	설명	스토리 포인트	상태
Must	사용자 회원가입 및 로그인 기능	신규 사용자 등록 및 기존 사용자 로그인 가능.	8	준비 완료
Must	상품 검색 기능	키워드를 기반으로 상품 검색 및 필터링 제공.	13	진행 중
Should	장바구니 기능	사용자가 선택한 상품을 장바구니에 담고 확인 가능.	8	준비 완료
Should	주문 및 결제 기능	결제 게이트웨이를 통한 신용카드 및 PayPal 결제 지원.	20	대기 중
Could	추천 상품 표시 기능	사용자의 관심사에 따라 개인화된 추천 상품 표시.	5	대기 중
Could	리뷰 및 평점 기능	상품에 대한 리뷰 작성 및 별점 평가 가능.	8	대기 중
Won't	다국어 지원	영어 외 다른 언어 지원(현재 Sprint에서는 제외).	13	제외

- 이해관계자와 협업하여 우선순위 조정
- 기술적 부채를 Backlog에 포함하고 추적
- 장기적인 목표와 단기적인 작업 항목 간의 균형 유지
- Sprint 종료 시 Backlog를 검토하고 업데이트



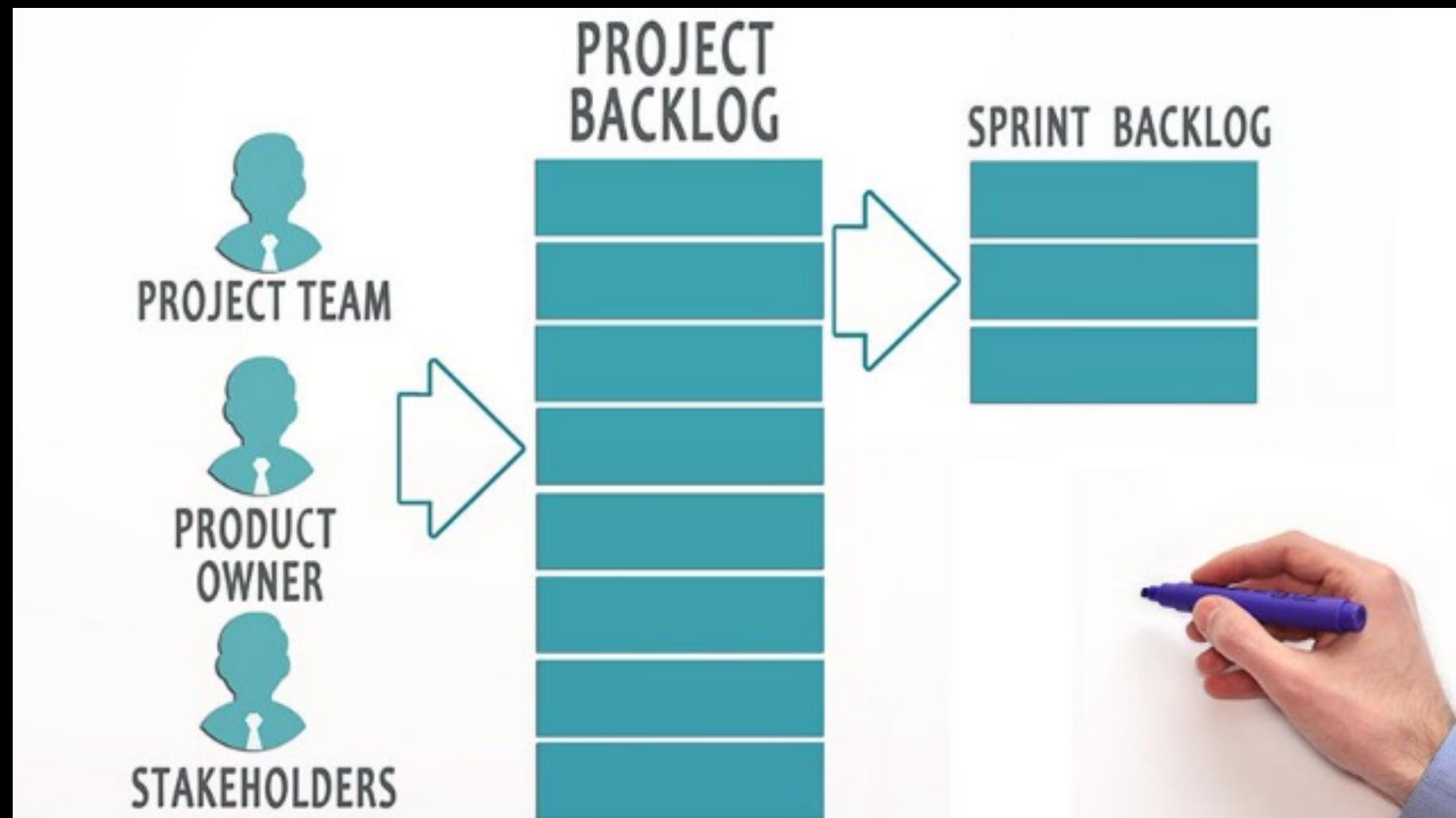
Sprint Backlog의 정의와 역할



Sprint Backlog

- Sprint 기간 동안 개발 팀이 완료할 작업 항목의 목록
- Product Backlog에서 선택된 항목으로 구성

Sprint Backlog 관리의 모범 사례



- Sprint 시작 시 팀 전체가 작업 항목을 명확히 이해
- 작업 항목의 진척도를 지속적으로 업데이트
- 작업 완료 기준(Definition of Done)을 충족하는지 확인
- 중간 목표 설정으로 작업 진행 상황을 점검
- 팀의 작업 상태를 한눈에 보여주는 도구로 활용



퀴즈 타임

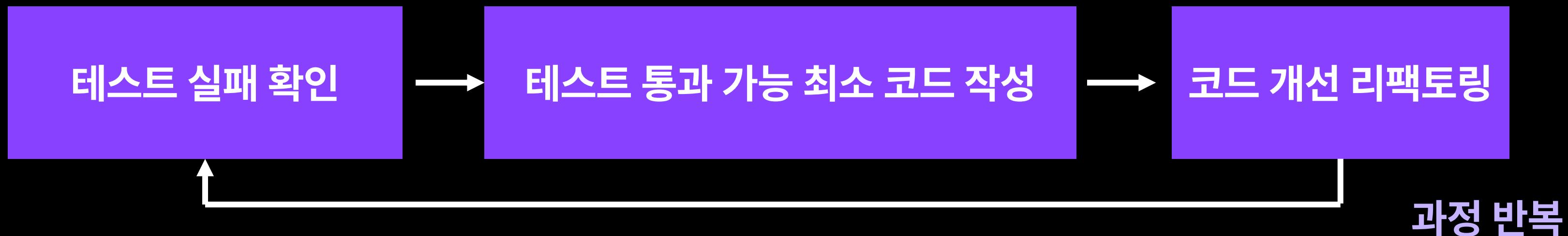


QUIZ 1번부터 8번을 풀어봐요!



TDD(Test Driven Development)

- ✓ 테스트 코드를 먼저 작성하고, 그 테스트를 통과하는 실제 코드를 작성하는 개발 방법론
- ✓ TDD의 철학 : "테스트를 실패하게 하고, 테스트를 통과하며, 코드를 개선한다."





TDD의 진정한 목적

요구사항 중심 개발



TDD의
목적

리팩토링 용이성



개발 효율성 향상





TDD의 3가지 핵심 원칙



테스트 우선: 코드 작성 전에 테스트를 작성



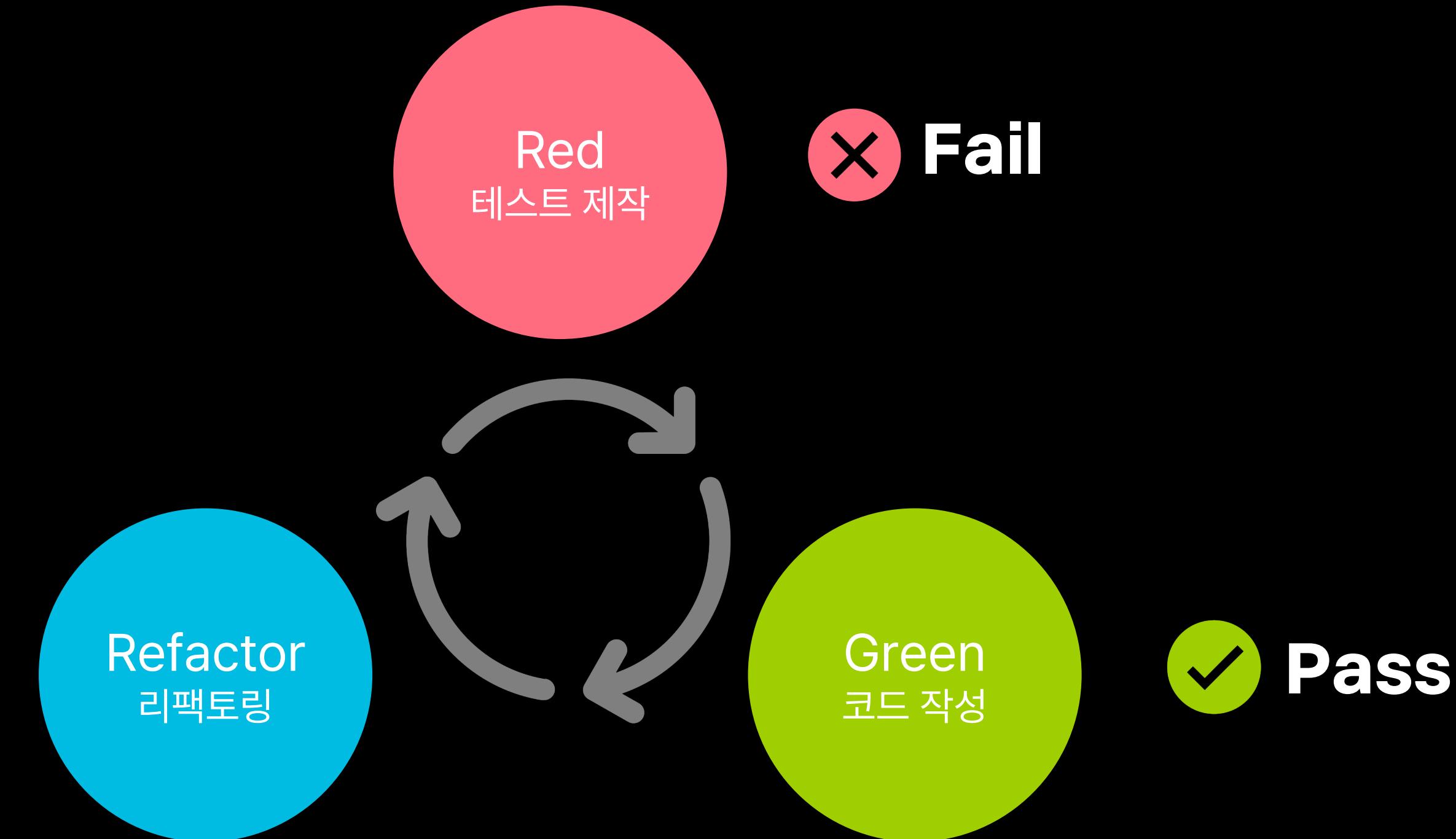
작은 단계로 나누기: 한 번에 하나의 기능만 테스트



리팩토링: 테스트를 통과한 후, 코드를 정리



TDD의 기본 절차





TDD에서 테스트 작성 예시

계산기 프로그램에서 덧셈 기능 테스트



테스트 케이스 작성

" $1 + 2 = 3$ "

코드 작성

테스트를 통과하기 위한 최소한의 덧셈 코드 작성

테스트 실행

테스트를 실행해 통과 여부 확인

실패

성공



코드 작성과 리팩토링의 중요성





BDD의 정의



BDD(Behavior-Driven Development)

TDD에서 발전된 방식
사용자 행동(Behavior)에 초점을 맞춘 테스트 주도 개발

- 테스트와 요구사항의 연결
- 비즈니스 요구를 코드로 전환
- 팀 간 소통 강화



Gherkin 언어의 개념



Gherkin 언어

BDD에서 사용되는 문법으로, 요구사항을 시나리오 형식으로 표현
사람이 읽기 쉬운 자연어로 작성

기본 문법

Given: 초기 상태 정의

When: 동작 정의

Then: 기대 결과 정의

예시

Given 사용자가 로그인 화면에 있을 때

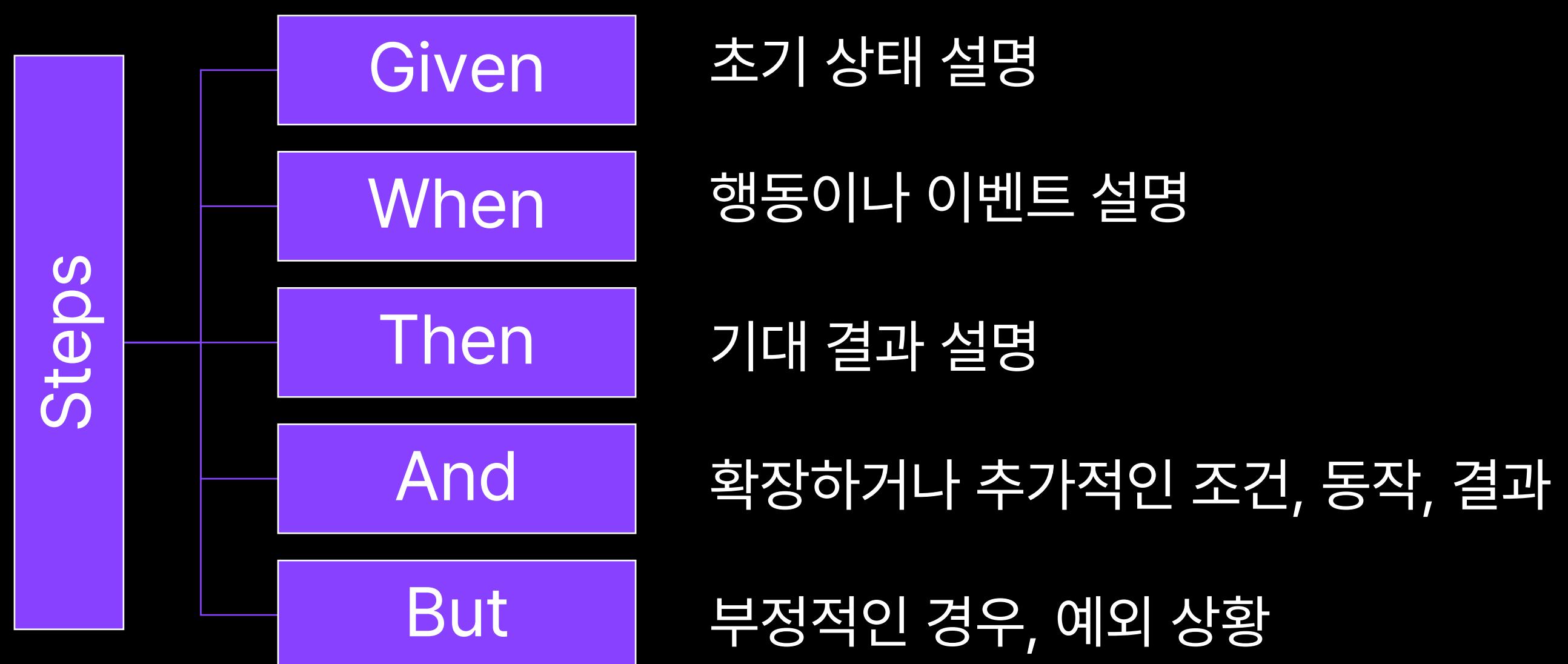
When 로그인 버튼을 클릭하면

Then 대시보드 화면이 표시된다

Gherkin 언어를 활용한 시나리오 작성

Feature : 기능의 큰 주제 또는 목표

Scenario : 특정 상황에서 시스템 동작 정의



[시나리오 예시]

Feature: 로그인 기능

사용자는 자신의 계정으로 로그인하여 대시보드에 접근할 수 있어야 한다.

Scenario: 유효한 계정으로 로그인 성공

Given 사용자가 로그인 페이지에 있다

And 사용자에게 유효한 계정이 있다

When 사용자가 올바른 아이디와 비밀번호를 입력한다

Then 사용자는 대시보드로 이동한다

And 환영 메시지를 볼 수 있다

Scenario: 잘못된 비밀번호로 로그인 실패

Given 사용자가 로그인 페이지에 있다

And 사용자에게 유효한 계정이 있다

Gherkin 언어를 활용한 시나리오 작성

[시나리오 예시]

Feature: 로그인 기능

사용자는 자신의 계정으로 로그인하여 대시보드에 접근할 수 있어야 한다.

Scenario: 유효한 계정으로 로그인 성공

Given 사용자가 로그인 페이지에 있다

And 사용자에게 유효한 계정이 있다

When 사용자가 올바른 아이디와 비밀번호를 입력한다

Then 사용자는 대시보드로 이동한다

And 환영 메시지를 볼 수 있다

Scenario: 잘못된 비밀번호로 로그인 실패

Given 사용자가 로그인 페이지에 있다

And 사용자에게 유효한 계정이 있다

When 사용자가 올바르지 않은 비밀번호를 입력한다

Then "잘못된 비밀번호입니다"라는 에러 메시지가 표시된다

But 사용자는 비밀번호 입력 필드를 수정할 수 있다

Scenario: 계정이 없는 사용자의 로그인 시도

Given 사용자가 로그인 페이지에 있다

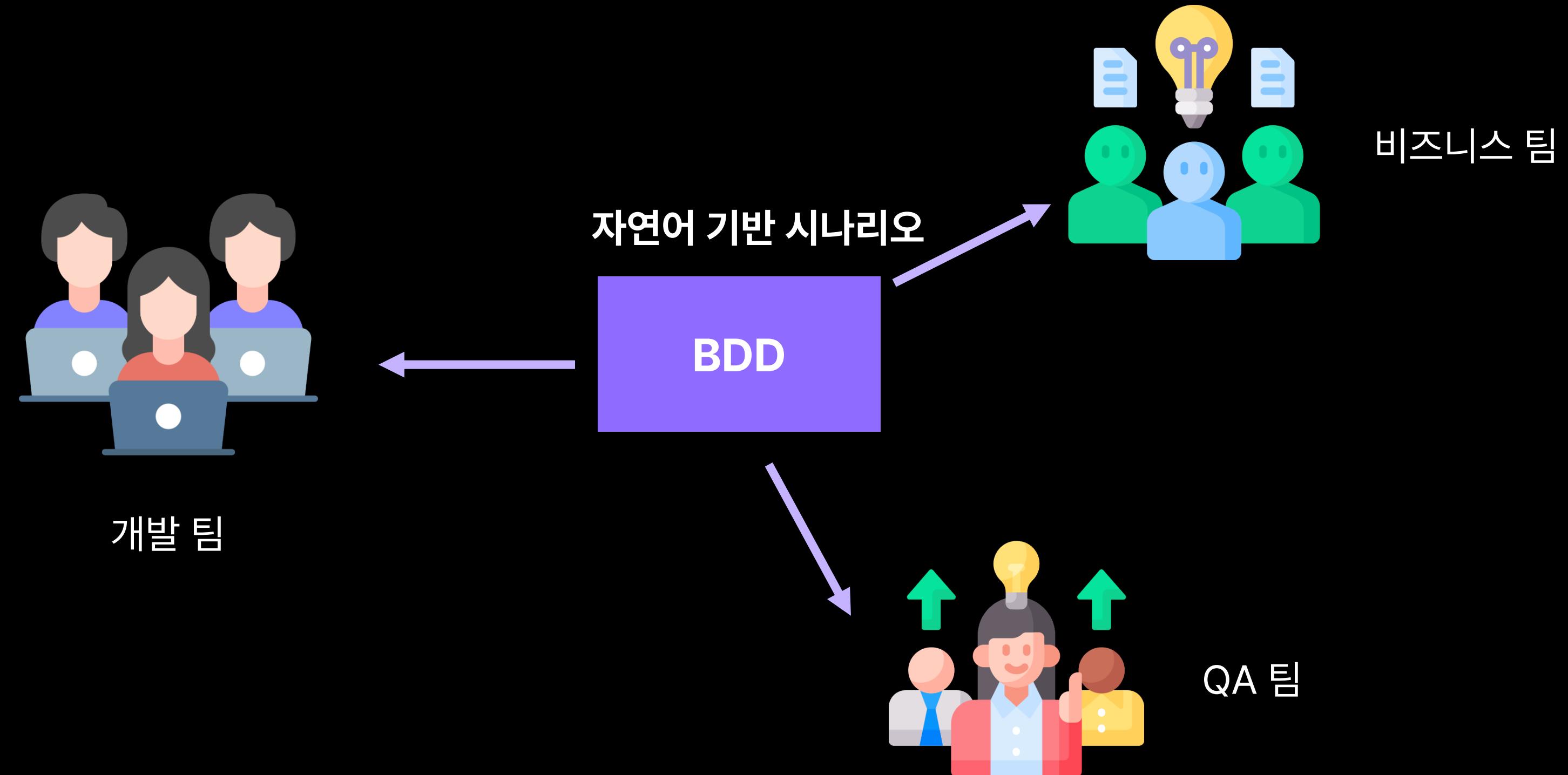
When 사용자가 존재하지 않는 아이디를 입력한다

And 임의의 비밀번호를 입력한다

Then "계정이 존재하지 않습니다"라는 에러 메시지가 표시된다

And 회원가입 버튼이 표시된다

BDD가 팀 간 소통을 강화하는 방법





TDD와 BDD의 주요 차이점

구분	TDD	BDD
초점	기술적 요구사항	사용자 행동 및 비즈니스 요구
목적	코드가 요구사항을 정확히 만족하도록 보장	비즈니스 요구사항과 사용자 행동의 일치 보장
출발점	테스트 케이스 작성	사용자 시나리오 작성 (Gherkin 언어 활용)
참여자	주로 개발 팀	개발자, 비즈니스 팀, QA 팀 간의 협업
사용 언어	프로그래밍 언어로 테스트 코드 작성	자연어 기반 (Given, When, Then)
강점	코드 품질 향상 및 리팩토링 용이	팀 간 협업 강화 및 명확한 요구사항 전달
적용 대상	기술적 구현 및 내부 로직	비즈니스 요구사항과 사용자 중심 기능



TDD와 BDD의 상호 보완 사례



사용자 중심 시나리오 작성

상세 테스트 케이스 작성 및 구현

TDD와 BDD는 상호 보완적으로 사용 가능

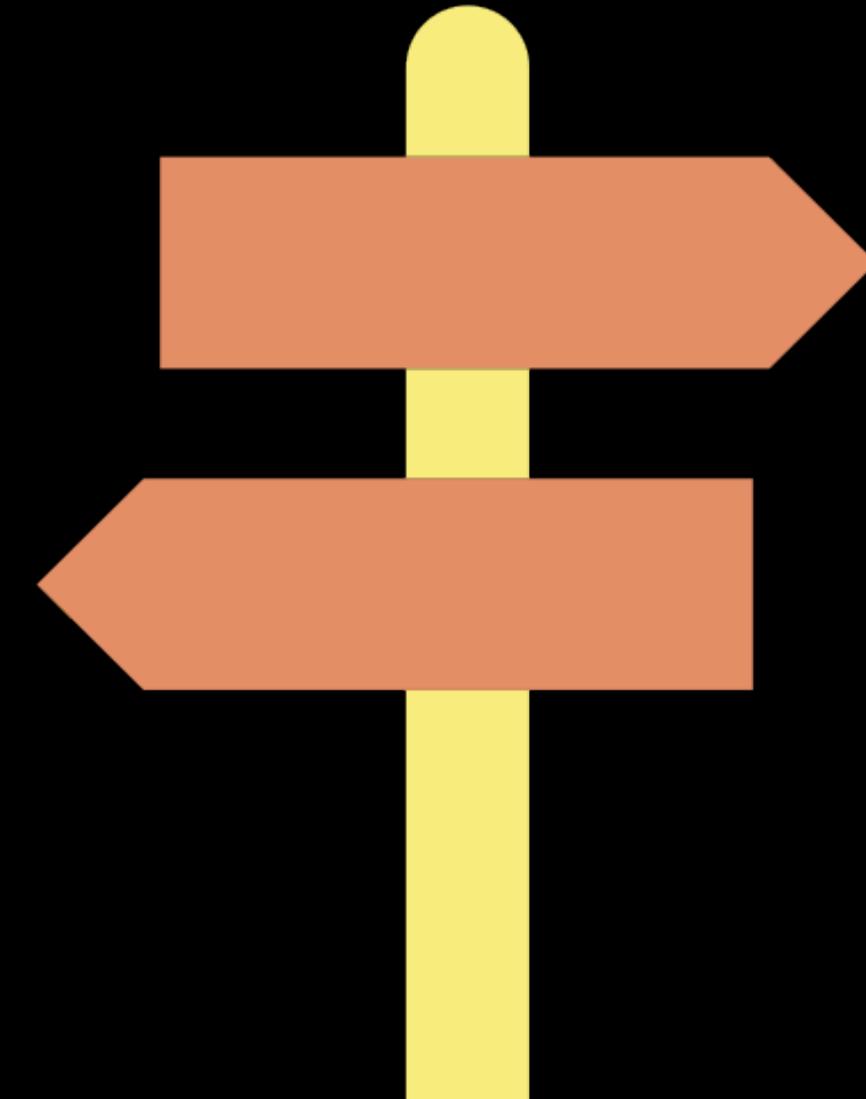


TDD와 BDD를 선택하는 기준

TDD냐? BDD냐?

BDD

TDD



- 비즈니스 요구사항이 명확하지 않고, 팀 간 협업이 필요한 경우
- 사용자 행동 중심의 개발이 중요한 경우

- 기술적 요구사항이 복잡하고 세부 테스트가 필요한 경우
- 개발 팀 내부에서 테스트 중심의 워크플로우를 유지할 때



TDD/BDD의 실질적 적용 사례

TDD 적용 사례



두 방식은 요구사항 정의와 기술 구현을 조화롭게 연결

은행 시스템

금융 거래 로직의 정확성을
검증하기 위해
세부 테스트 케이스 작성

IoT 장치

센서 데이터 처리의
정확성과 성능 테스트





BDD 적용 사례

두 방식은 요구사항 정의와 기술 구현을 조화롭게 연결

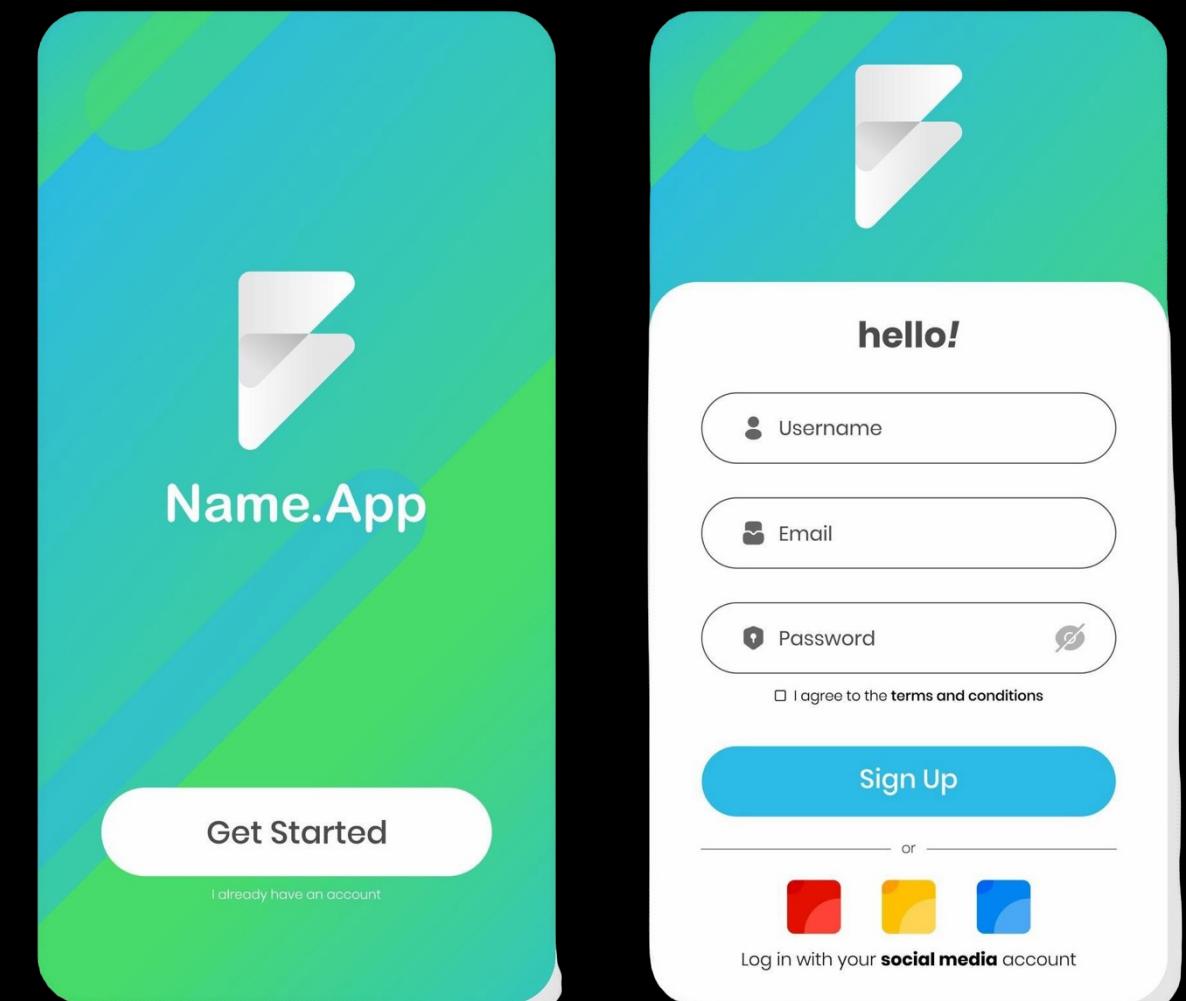


전자 상거래

사용자가 상품을 검색하고
장바구니에 추가하는 시나리오

모바일 앱

사용자가 로그인한 후 개인화된
콘텐츠를 확인하는 시나리오





TDD/BDD 적용 실패를 방지하는 방법론

TDD 실패 방지 방안	BDD 실패 방지 방안
테스트 케이스가 모호하지 않고 구체적으로 작성되어야 함	Gherkin 시나리오가 명확하고 간결해야 함
지나치게 많은 테스트 작성으로 인한 속도 저하를 방지해야 함	개발 팀과 비즈니스 팀 간의 협업 부족 방지
공통 요구사항을 명확히 정의 테스트 커버리지를 지속적으로 관리 팀 간 커뮤니케이션 강화	



퀴즈 타임



QUIZ 9번부터 18번을 풀어봐요!

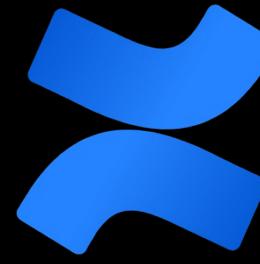
Jira와 Confluence란 무엇인가?



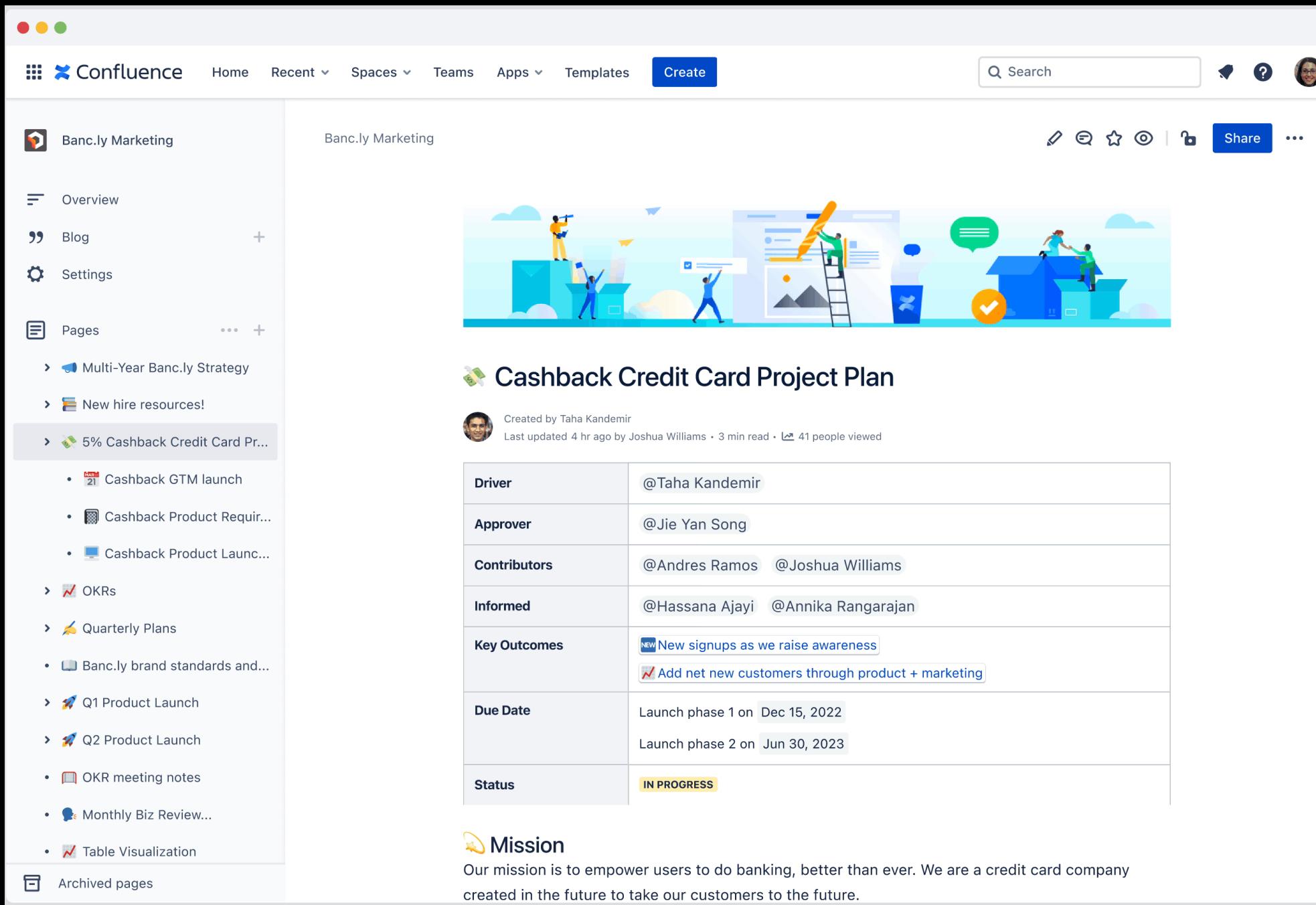
JIRA

A screenshot showing two project management boards side-by-side. On the left is a Jira board titled 'Beyond Gravity' with four columns: TO DO (12 issues), IN PROGRESS (4 issues), IN REVIEW (4 issues), and DONE (4 issues). The DONE column contains items like 'Multi-dst search UI web' and 'Quick booking for accomodations -'. On the right is a Confluence board titled 'Board' for the 'Teams in Space' project, also with four columns: TO DO (5 issues), IN PROGRESS (5 issues), IN REVIEW (2 issues), and DONE (8 issues). The IN PROGRESS column has one item highlighted with a red border, which reads: 'Requesting available flights is now taking > 5 seconds'. Both boards show issue keys (e.g., NUC-205, TIS-37) and small user icons.

- 애자일 프로젝트 관리 도구
- 스프린트, 백로그 관리, 작업 추적, 버그 보고를 위한 기능 제공
- Scrum 및 Kanban 보드 활용



Confluence



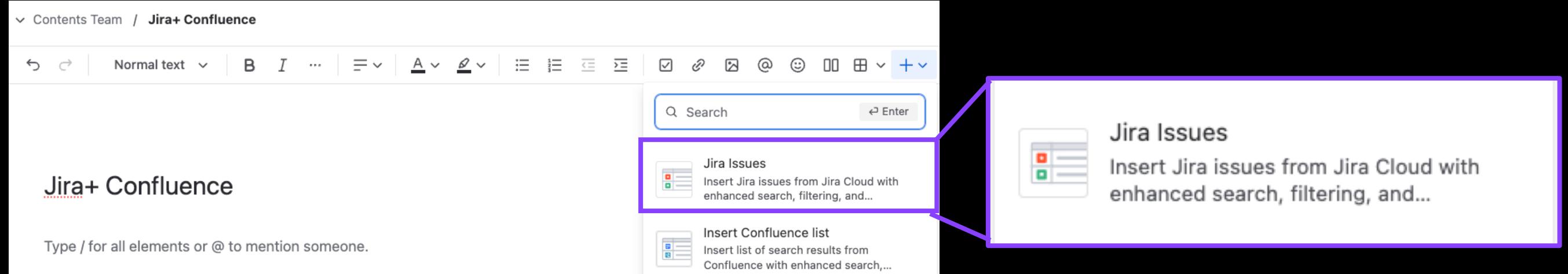
The screenshot shows the Confluence interface. On the left, there's a sidebar with navigation links like 'Overview', 'Blog', 'Settings', 'Pages', and 'Archived pages'. The main content area displays a page titled 'Cashback Credit Card Project Plan'. The page features a decorative header image with people working on a building, a brief description, and a table of contents. Below the table of contents is a table with columns for Driver, Approver, Contributors, Informed, Key Outcomes, Due Date, and Status. The 'Status' column shows 'IN PROGRESS'. At the bottom, there's a section for the mission statement.

Driver	@Taha Kandemir
Approver	@Jie Yan Song
Contributors	@Andres Ramos @Joshua Williams
Informed	@Hassana Ajayi @Annika Rangarajan
Key Outcomes	New signups as we raise awareness Add net new customers through product + marketing
Due Date	Launch phase 1 on Dec 15, 2022 Launch phase 2 on Jun 30, 2023
Status	IN PROGRESS

Mission
Our mission is to empower users to do banking, better than ever. We are a credit card company created in the future to take our customers to the future.

- 팀 문서 관리 및 협업 도구
 - 요구사항, 회의 노트, 프로젝트 계획
- 문서화를 지원**
- Jira와 Confluence는 연계하여 팀 간 소통과 작업 효율성을 강화

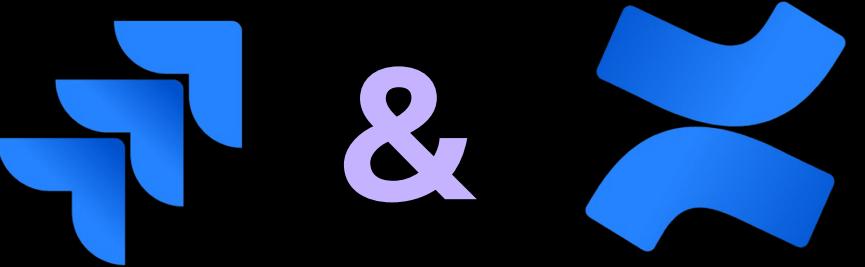
Jira와 Confluence의 통합 활용



This screenshot shows the 'Insert Jira issues' modal. At the top, there's a search bar and filters for 'Project', 'Type', 'Status', and 'Assignee'. The main area is a table with columns: Type, Key, Summary, Assignee, Priority, Status, and Updated. The table lists several Jira tickets, including LXP-6382, AIDT-3322, AIDT-3321, LXP-6381, and AIDT-3320. The bottom right of the modal has 'Cancel' and 'Insert issues' buttons.

Type	Key	Summary	Assignee	Priority	Status	Updated
BUG	QUALITY-107		-	P1(High)	TO DO	Jan 21, 2025, 18:13
BUG	QUALITY-106		-	P1(High)	TO DO	Jan 21, 2025, 18:13
BUG	QUALITY-105		-	P2(Medium)	TO DO	Jan 21, 2025, 18:13
BUG	LXP-6382		-	P4(Low)	TO DO	Jan 21, 2025, 17:59
BUG	AIDT-3322		!	P1(High)	TO DO	Jan 21, 2025, 17:59
BUG	AIDT-3321		!	P1(High)	TO DO	Jan 21, 2025, 17:59
BUG	LXP-6381		...	P1(High)	TO DO	Jan 21, 2025, 17:58
BUG	AIDT-3320		!	P2(Medium)	TO DO	Jan 21, 2025, 17:58

32,259 issues



- Jira 작업 상태를 Confluence 문서에 자동으로 연결
- Confluence에서 Jira 티켓 생성 및 상태 확인
- 요구사항과 작업 상태를 하나의 플랫폼에서 확인하여 팀 간 협업 강화



Jira와 Confluence 활용의 모범 사례

Jira : Sprint 상태 관리

Jira Software

Your work Projects Filters Dashboards People Apps Create

Search

Nucleus Software project

Roadmap

Backlog

Board

Code

On-call

Project pages

Add item

Project settings

Projects / Nucleus

Backlog

Epic Label Version

Epics

- > Forms
- > Feedback
- > Accounts
- > Billing
- > AWS spike

+ Create epic

Sprint 3 5 issues

3 2 0 Complete sprint

NUC-335 Affelite links integration - frontend BILLING

NUC-342 Fast trip search ACCOUNTS

NUC-341 Quick payment FEEDBACK

NUC-340 Account settings defaults ACCOUNTS

NUC-339 Billing system integration - frontend ACCOUNTS

NUC-337 Multi-dest search UI mobileweb ACCOUNTS

NUC-360 Onboard workout options (OWO) ACCOUNTS

NUC-344 Optimize experience for mobile web BILLING

NUC-338 Multi-dest search UI web ACCOUNTS

NUC-354 Shopping cart purchasing error - quick fix required.

NUC-343 Fluid booking on tablets FEEDBACK

NUC-346 Adapt web app no new payments provider

NUC-336 Quick booking for accomodations - web

+ Create issue

(추가 가능)

- 팀 규칙, 작업 프로세스를 Confluence에 문서화
 - 자동화 기능을 통해 Jira 티켓 상태를 Confluence에 업데이트

Confluence : Sprint 리뷰 작성



What did we do well?

- Big improvement on fixing broken builds as soon as they break.
- Code reviews were quite thorough.
- The analytics hooks are all implemented – woot!
- Pairing closely with designers cut out a ton of back n' forth.

What should we have done better?

- Lots of broken builds this sprint.
- Had a couple instances of people starting work on a story without assigning it to themselves, and then someone else started working on it too (oops!).

What should we start doing?

- 🌟 Use Vagrant for our dev environments so they don't get out of sync with staging and production

What should we stop doing?

- Implementing more analytics hooks
- 🚫 Flagging the entire team as reviewers for each pull request. 2 or 3 (including

What should we keep doing?

- ✅ Assign the issue to yourself before starting work on it – every. time.
- Keep rockin' the code reviews.

Jira와 Confluence 활용의 모범 사례

1. QA 팀이 Confluence를 활용해 테스트 계획 문서화

J_<VU/OU>_<version>_04_06			
Project Name	JVU751_2K18	Designed By	Alana Fernando
Test Scenario Name	Login verification	Designed Date	27 Aug 2018
Test Scenario ID	J_VU_7.5.1_04_06	Last Modified By	
Test Priority	Medium	Last Modified Date	
Description	this test case verifies Login ability , Login restrictions applied.	Executed By	Alana Fernando
Final Results	PENDING PASS FAIL	Execution Date	
Pre-Condition	1. Follow instructions 2. Test user should have to have system access	Supplementary	System URL
TC_Step No	Test Case Step	Expected Results	Actual Results comment Results
			As Expected Not As Expected
1	Navigate to system log in page	Page should load successfully	<input type="checkbox"/> <input type="checkbox"/>
2	Log in to system using test-user credentials	User should be able to log in successfully	<input type="checkbox"/> <input type="checkbox"/>
Post Condition	Update results		
Verified BY			

2. Jira로 결함 상태를 관리

The screenshot shows the TestCycles app interface in Jira. The top navigation bar includes 'Project' (with a robot icon), 'Test Cases', 'Test Cycles' (which is the active tab), 'Test Plans', and 'Reports'. Below the tabs are buttons for '+ New Folder', 'Search...', 'Group by', 'Filter', 'Edit', 'Run', 'Clone', and 'Delete'. A prominent blue button '+ New Test Cycle' is located on the right. The main content area displays a list of 'All test cycles (114)'. Under 'Pre-release tests (15)', there are three categories: 'Regression tests (26)', 'Backend releases (4)', and 'Frontend releases (1)'. The 'Regression tests (26)' section lists five items: 'P C11 Regression 1.0 Android' (100% DONE), 'P C12 Regression 1.0 iOS' (100% DONE), 'P C13 Regression 2.0 Android' (100% DONE), 'P C14 Regression 2.0 iOS' (58% IN PROGRESS), and 'P C15 Regression 2.1 Android' (54% IN PROGRESS). A message at the top right says '1 test cycle is selected. Clear selection'. On the bottom right, a sidebar titled 'Execution status' provides summary statistics: Not Executed: 11, In Progress: 3, Pass: 14, Fail: 3.

Jira와 Confluence 활용의 모범 사례

3. Confluence에서 결과 공유

NewSystem - GoLive Test Plan



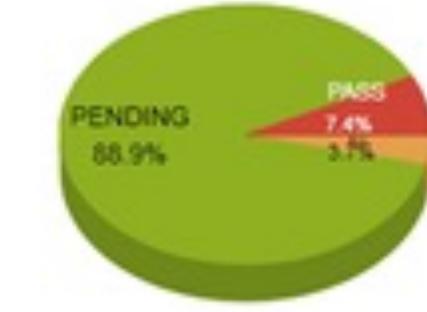
Common Instructions :

- 1 Test Environment
- 2 Please use test user credentials to log in instead of your AD account credentials. (test user credentials are provided in this page)
- 3 If any issue raised, contact Heshan Manamperi.
- 4 Note down any issue in comment or actual results area provided for each test step.
- 5 Update results in results section provided for each test step.

Regression of functionality and functional verification Summary Report

Title	Test Scenario Name	Test Scenario ID	Test Priority	Final Results
J_<VUIOU>_<version>_04_01	Email template verification	J_VU_7.5.1_04_01	Medium	FAIL
J_<VUIOU>_<version>_01_03	Transition verification	J_VU_7.5.1_01_03	Medium	PENDING
J_<VUIOU>_<version>_01_02	Log in verification	J_VU_7.5.1_01_02	Medium	PASS
J_<VUIOU>_<version>_01_01	Report verification	J_VU_7.5.1_01_01	Medium	PASS
J_<VUIOU>_<version>_04_02	Search function verification	J_VU_7.5.1_04_02	Medium	PENDING
J_<VUIOU>_<version>_04_03	Comments verification	J_VU_7.5.1_04_03	Medium	PENDING

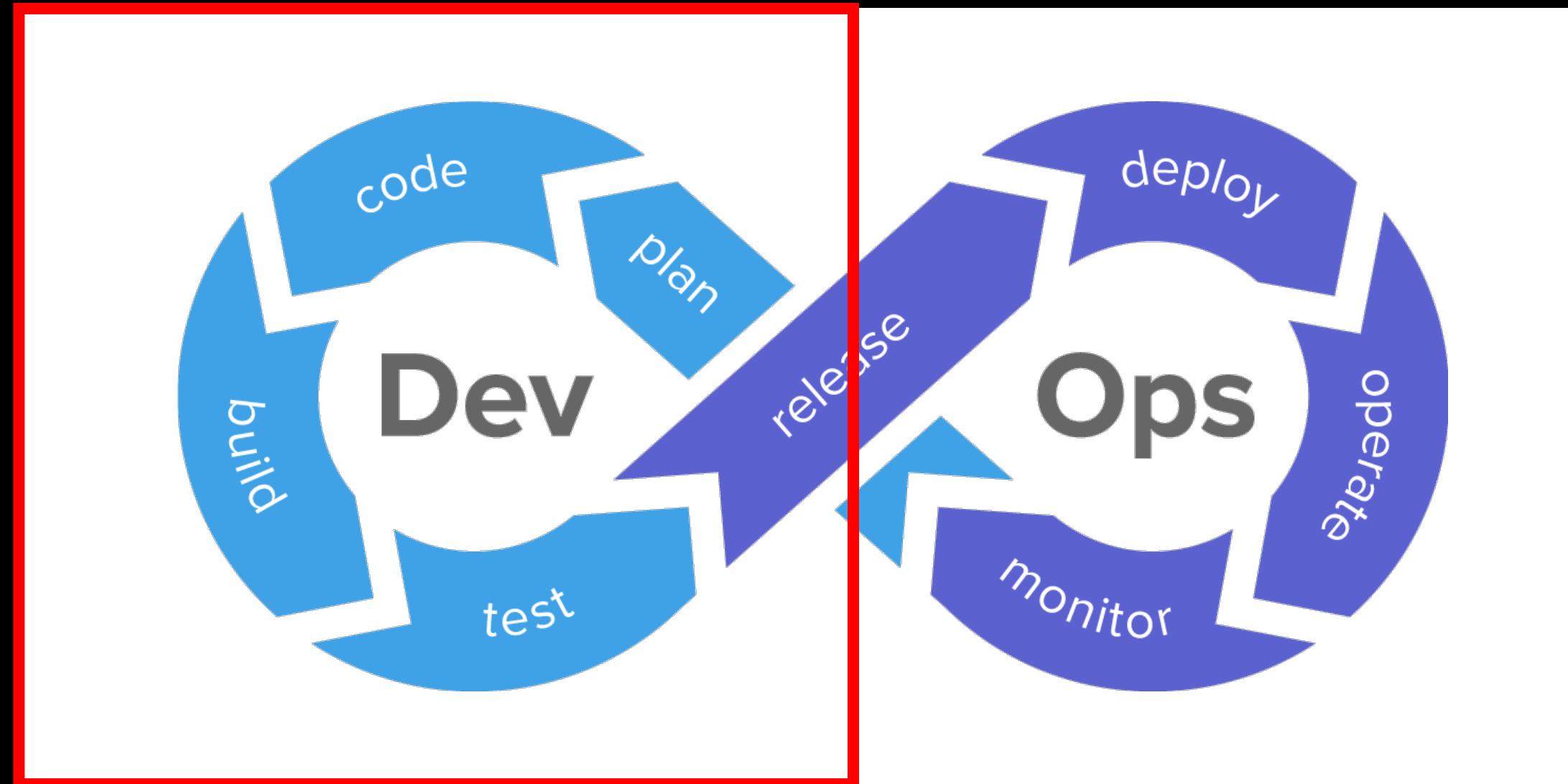
Test Results



Labels column	Final Results
Values column	Test Scenario ID
Type	Pie
Height	200
Width	200



CI/CD란 무엇인가?



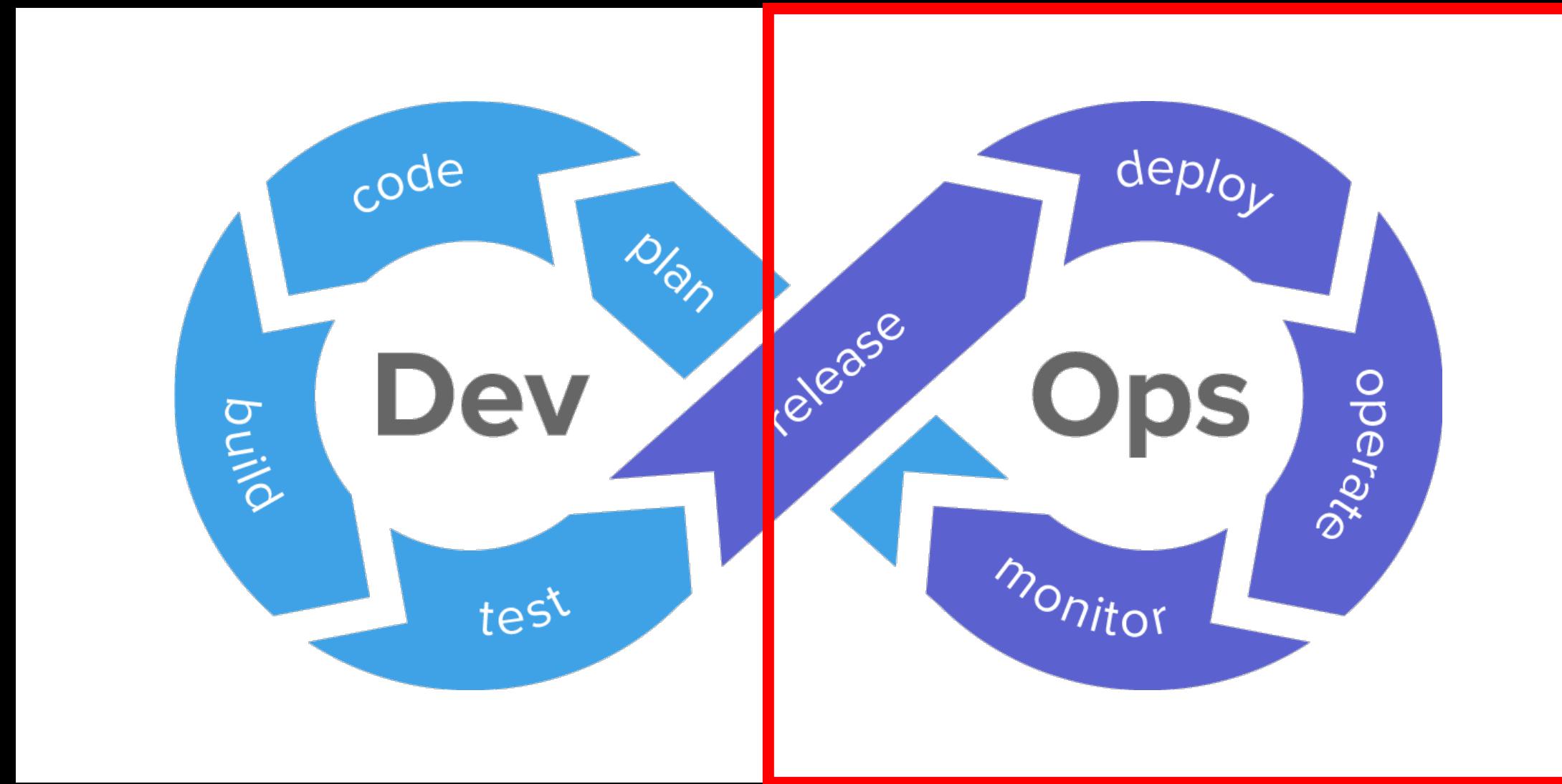
CI

(Continuous Integration)

- 코드를 자주 병합하여 결함을 조기에 발견
- 빌드와 테스트 자동화를 통해 개발 주기 단축



CI/CD란 무엇인가?



CD

(Continuous Delivery/Deployment)

- CI 이후의 단계로, 소프트웨어를 안정적으로 배포
- Continuous Delivery는 승인 후 배포,
Continuous Deployment는 자동 배포
- CI/CD는 Agile 개발의 핵심 요소로, 빠르고
신뢰성 있는 배포를 가능하게 함



CI/CD와 Agile 개발의 연계



CI/CD 도구와 Agile의 공통 목표

- 짧은 개발 주기
- 지속적인 개선과 배포

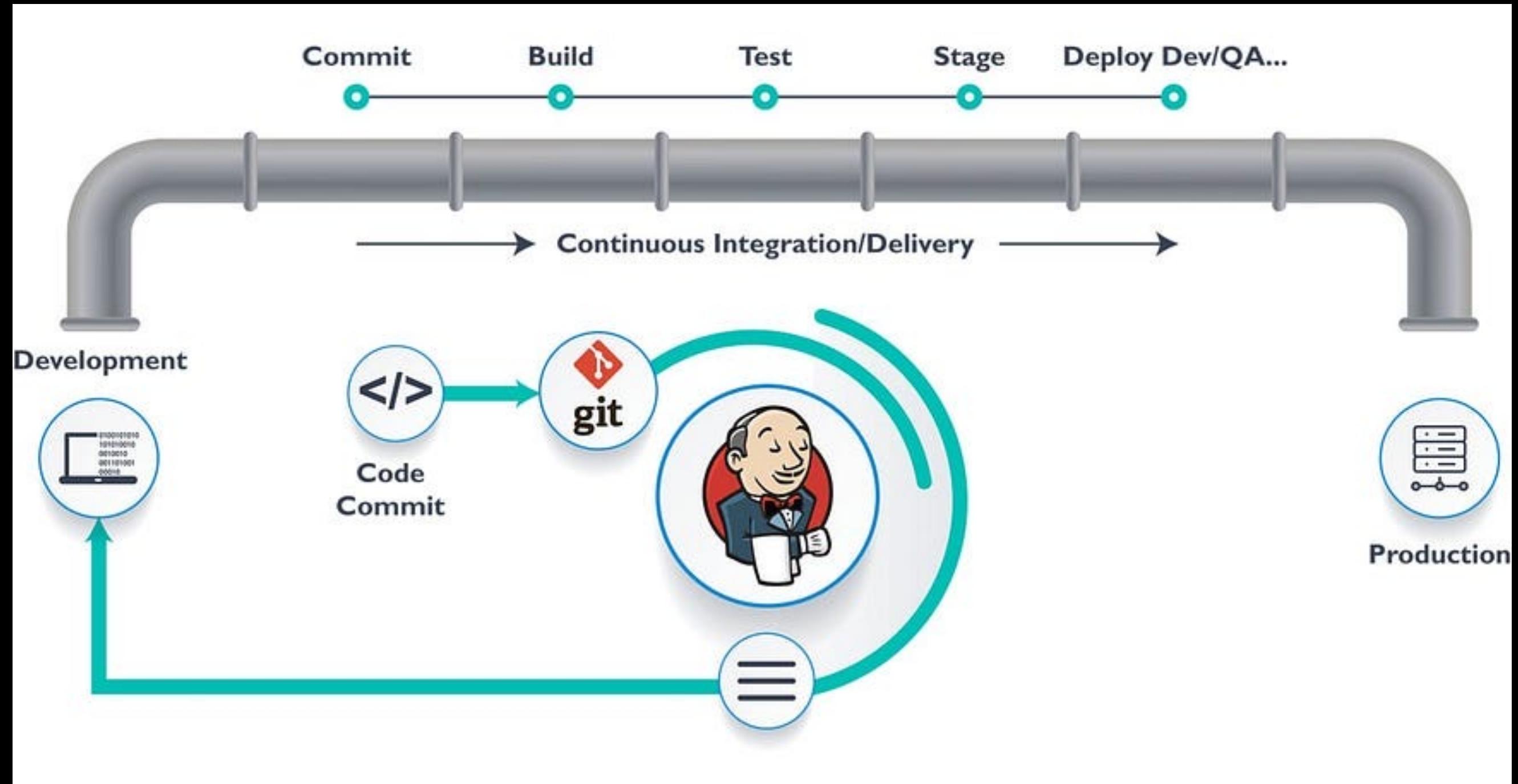
Agile 개발에 CI/CD가 주는 이점

- 팀이 변경 사항을 빠르게 확인하고 수정
- 제품 품질과 배포 속도 향상



CI/CD와 Agile 개발의 연계

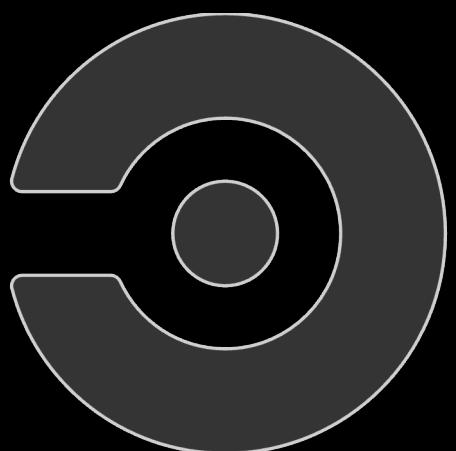
Jenkins Pipeline



Jenkins



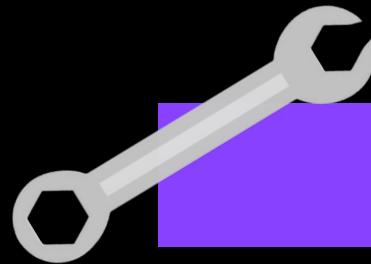
Gitlab CI/CD



Circle CI



Agile QA 팀의 도구 개요

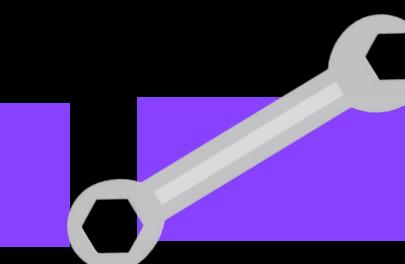


TestRail

The screenshot shows the TestRail interface for a specific test case. At the top, there's a navigation bar with links like 'Return to Dashboard', 'Working On', 'Help & Feedback', and social media icons. Below the navigation is a 'Datahub' section with tabs for 'Overview', 'Todo', 'Milestones', 'Test Runs & Results' (which is selected), 'Test Suites & Cases', and 'Reports'. The main content area displays a test case titled 'T218763 Verify CSV import with attached test data file'. It includes a pie chart showing 170 Passed (58%), 2 Blocked (1%), 6 Retest (2%), and 22 Failed (7%). A table provides details: Type (Other), Priority (3 - Test If Time), Estimate (5 minutes), and References (DH-123). The 'Preconditions' section states: 'No page has been opened and all user settings have been reset.' Below this is a table for 'Steps':

Page	Option	Value	Error Range
Page 1	Page Margin	2cm	0.5cm
Page 2	Zoom Percent	120%	0

The 'Prerequisites' section lists 19 items with various status indicators (Passed, Blocked, Failed, etc.). The 'Results & Comments' section shows a comment from Dennis G. dated 2015/11/28 at 04:25 PM.



Zephyr

The screenshot shows the Zephyr interface for a 'Cycle Summary' report. The title is 'Sprint 2'. It includes a search bar and a sidebar with navigation links: 'Create New Test Cycle', 'Activity', 'Progress', and 'Defects'. The main area displays a table of test cases:

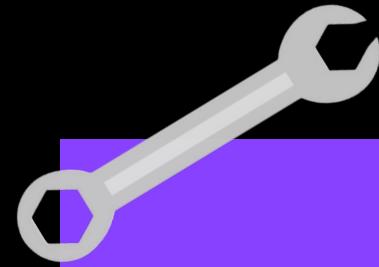
ID	Status	Summary	Defect	Component	Label	Total Exec...	Actions	
SMART-67	FAIL	Wishlist items persist across sessions	110	SMART-66	shopping cart	unit, browser		
SMART-68	BLOCKED	Payment data data is obfuscated	-	payment processing	unit	-		
SMART-69	PASS	Currency converted from USD based on location	-	payment processing	integration	-		
SMART-70	FAIL	Payment with valid credit card	-	-	unit/integration	-		
SMART-71	INPROGRESS	Maintain data integrity of shopping cart across sessions	110	SMART-58	shopping cart	-		
SMART-72	FAIL	Production options - only options available in inventory are selectable	210	SMART-60	shopping cart	end-to-end		
SMART-73	INPROGRESS	SMART-61	-	-	-	-		

테스트 케이스 관리 및 결과 분석

Jira와 통합된 테스트 관리



Agile QA 팀의 도구 개요



QTest

The screenshot shows the QTest Manager interface. At the top, there's a navigation bar with 'qTest Project', 'Test Plan', 'Requirements', 'Test Design', and 'Test Execution'. Below it is a summary bar with counts: 387 Test Runs, 243 Executed, 144 Unexecuted. A progress bar indicates the status of these runs. The main area is titled 'Execution Summary' and lists test items by release: LSP Sprint 1, LSP Sprint 2, LSP Sprint 3, and a Test Cycle: Automated Tests. Each item has a breakdown of test runs, executed, failed, incomplete, blocked, ready for next tester, and unexecuted counts. Below this is a section for 'UNEXECUTED' items, showing 32 (27%) items with a breakdown of 33 (green), 45 (red), 6 (yellow), 2 (purple), and 32 (grey). Other sections include 29 (94%) items (11 green, 29 grey), 51 (100%) items (51 grey), 0 (0%) items (65 green, 52 red, 12 yellow), and 32 (57%) items (7 green, 16 red, 1 yellow, 32 grey). A large 'EXPORT' button is at the bottom.

The screenshot shows the QTest Pulse interface. At the top, there's a navigation bar with 'qTest Project' and 'My Rules'. Below it is a chart titled 'Rule Performance' showing 'Action Executions' over time. A modal dialog is open for creating a 'New Rule'. It has fields for 'Name' (NewRule), 'Tag' (Active checked), 'Triggers' (qTestDefectSubmitted selected), 'Rule' (When: qTestDefectSubmitted), 'Actions' (CreateDefectInAzureDevops and SyncDefectFromAzureDevops selected), and a 'Drag Action' button. The background shows a list of triggers and actions, including Azure DevOps Work Item triggers and actions like CreateDefectInAzureDevops and SyncRequirementFromAzureDevops.

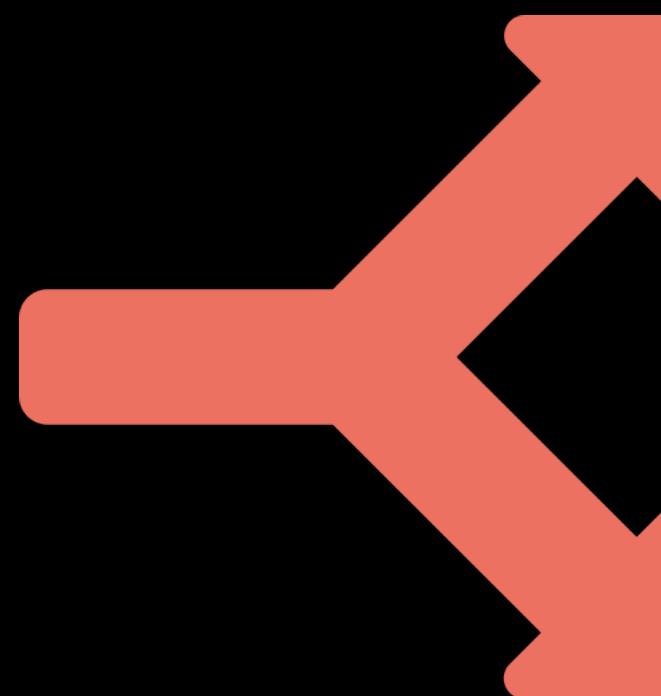
대규모 프로젝트에 적합한 테스트 관리



TestRail과 Zephyr 비교

	TestRail	Zephyr
소개	독립형 테스트 관리 도구	Jira와 통합되어 Agile 프로젝트와 연계
특징	상세한 보고 기능과 사용자 정의 가능	테스트 결과를 Jira 티켓과 직접 연결 가능

[선택 기준]



Jira 기반 프로젝트 = TestRail

독립적인 테스트 관리와 상세 분석 필요 = Zephyr



Slack과 Zoom을 활용한 원격 협업

빠른 소통 및 효율적 의사결정을 위한 원격 협업 툴



Slack

실시간 채팅과 파일 공유로 팀 간 소통 강화
통합 기능(GitHub, Jira 등)으로 작업 상태 공유



Zoom

화상 회의와 화면 공유로 팀 협업 촉진
Sprint Planning, 회고 등을 원격으로 진행 가능



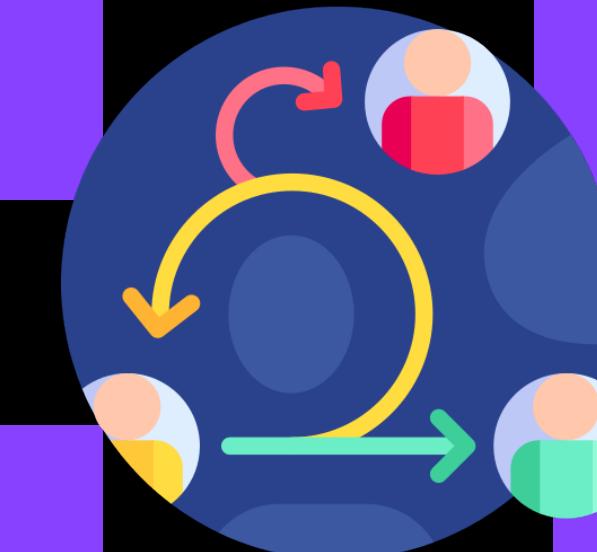
원격 Agile 환경 구축의 핵심

팀 간 투명한 의사소통

명확한 역할 분담과
목표 설정

업무 상태를 공유할 수 있는
협업 도구 활용

원격 환경에서도 Agile
원칙을 유지하기 위한 노력



Slack/Zoom: 실시간 소통과 회의
Jira: 작업 추적
Confluence: 문서화와 지식 공유



Agile QA 도구 활용의 실제 사례

Agile 도구 활용을 통해 QA와 개발 팀 간 협업 효율성 극대화

테스트 결과 논의를 위한 Zoom 회의



Slack에서 테스트 상태를 실시간 공유

QA2L APP 7:43 PM ⚡

Failure | Account Create | Sun May 21 2017 21:44:01 GMT-0500 (CDT)

Home Page - Click on "Individuals"

Success: 15 of 15 checks

- ✓ c1 equals www
- ✓ v1 equals D=c1
- ✓ v2 matches [0-9]* [Show more...](#)

Select "Start Free Trial"

Success: 5 of 5 checks

- ✓ events equals event1
- ✓ pev2 equals Start Free Trial Click
- ✓ v2 matches [0-9]* [Show more...](#)

Account Completed

Failure: 9 of 10 checks

- ✗ events equals event1 **expected event2**
- ✓ c1 equals account complete
- ✓ v1 equals D=c1 [Show more...](#)

[Edit Task](#) [QA2L Task Manager](#)



퀴즈 타임



QUIZ 19번부터 23번을 풀어봐요!