INDUSTRIAL AI RESEARCH CENTER

SW 개발 업무 프로세스

2021.09.02

목 차

자가 테스트 (자아비판?)

조엘 (Joel) 테스트 (2005)

1. 소스코드 관리시스템을 사용하고 있습니까?
2. 한번에 빌드를 만들어낼 수 있습니까?
3. 일일 빌드를 하고 있습니까?
4. 버그 추적 시스템을 운영 하고 있습니까?
5. 코드를 새로 작성하기 전에 버그를 수정합니까?
6. 일정을 업데이트하고 있습니까?
7. 문서화를 하고 있습니까?
8. 조용한 환경에서 일하고 있습니까?
9. 경제적 범위 내에서 최고 성능의 도구를 사용하고 있습니까?
10. 테스트 팀을 구성하고 있습니까?
11. 프로그래머 채용 면접 때 코딩 테스트를 합니까?
12. 무작위의 사용편의성 테스트를 하고 있습니까?

개발 시 자주 사용하는 용어

- Alpha, Beta, RC, GA, FCS release
- Code freeze
- RTM (Release to Manufacturing)
- Baseline
- Tagging, Labeling
- Build
- Full Build, Partial Build, Incremental Build
- Daily Build
- Full Release, Partial Release, Patch
- Master, Master CD
- Broken Tree
- Branch, merge(2way merge, 3way merge)
- Go-live meeting
- Regression Test, Smoke Test
- Bug Resolved vs. Bug Closed
- Show-stopper
- Hot-fix
- SRS (Software Requirement Specification)
- SDS (Software Design Specification)

용어 설명

Alpha, Beta, RC, GA, FCS release

개발방법론 (100+개)

- Waterfall
- DoD
- Iterative
- XP Programming
- 구조공학
- 정보공학
- 객체지향
- CBD

용어 설명

Alpha Version

A very early version of a software product that should contain all of the features that are planned for the final version. But, product can be unstable and may contain show-stopper bugs.

Beta Version

Beta version software must include all features, but may also include known issues and bugs of a less serious variety.
Should not contain show-stopper bugs.
Open beta vs Closed beta

용어 설명

RC (Release Candidate)

- A version with potential to be a final product, ready to release.
- No major bugs.
- Apple: Golden master.

RTM (Release to Manufacturing)

- RTM is used to indicate that the software has met a defined quality level and is ready for mass distribution either by electronic means or by physical media.
- “Going gold“라고도 한다.
- RC 버전 중에서 테스트를 통과한 버전이 RTM 대상이다.
- Gold Master (예: CD)를 만든다.

FCS(First Customer Shipment), GA (General Availability)

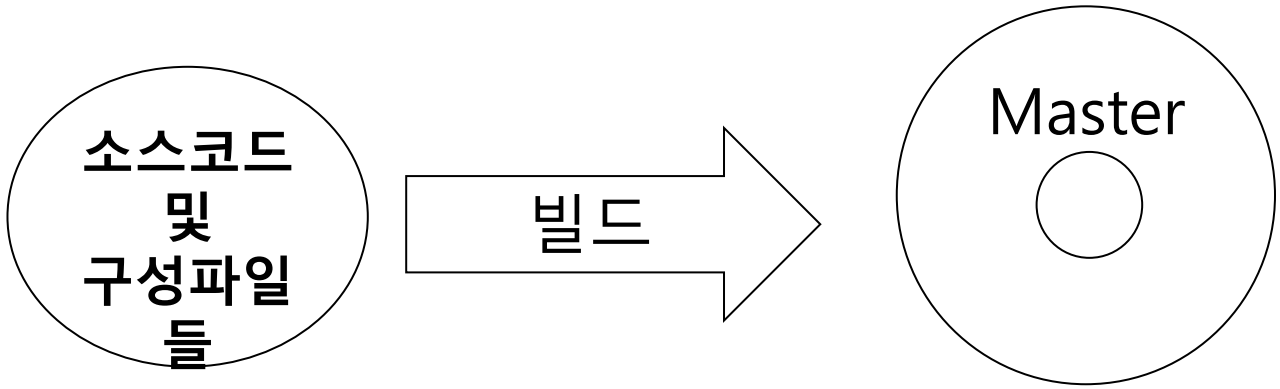
- Version to be released to the general market either via the web or physical media.
- All necessary commercialization activities have been completed.
- Gold Master 를 복사하여 배포한다

용어 설명

Build, Release, Master

Master

- Release 되는 설치 될 수 있는 상태의 파일 모음
- Installable Package



용어 설명

Build, Release, Master

Build

- 빌드의 종류는 ?
  - Official Build vs. Engineering Build
  - Full Build, Partial Build, Incremental Build
- 빌드의 결과물은 ?
  - Master 파일
  - 조합으로 인한 수 많은 패키지
    - H/W별, OS별, 언어별, 고객별,
- 빌드는 누가 할까?
  - 개발자, 빌드/릴리즈 부서
- 빌드는 어디서 할까?
  - 왜 공식 빌드는 개발자 기계에서 하면 안되나?
  - Dirty Environment

Build, Release, Master

Daily Build

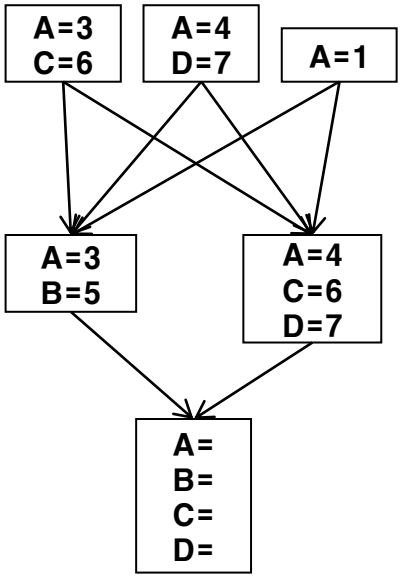
- Nightly build 라고도 함
- 목적
  - Broken Tree 방지
  - Continuous Integration(지속적 통합)
  - 인터페이스 변경 자동 감지
- 개발의 어느 시점부터 실행하나?
  - 상세설계 후부터 (가능한 지?)
- 소스 코드는 어디에서 가져오는가?

Release

- 종류 : Full Release, Partial Release, Patch

Branch/Merge

- Branch의 목적
  - Merge를 하는 경우(Feature Branch) 와 하지 않는 경우 (Product Branch)
  - 병행 개발을 가능하게 한다.
  - 전체 개발시간은 늘지만 출시시점은 빠르다.
    - Merge의 overhead
- Branch의 종류
  - Full Branch
  - Partial Branch – 공통 부분은 공유
- Merge의 종류
  - 2way merge
  - 3way merge – 글로벌회사의 필수 조건?
- Merge는 얼마나 자주하는 것이 좋은가?
  - Big Bang vs. Phased Approach
  - Water Fall vs. Agile



용어 설명

테스트종류

- **Regression Test (회귀 테스트 : 이전 버전, 이전 버그와 비교)**
- **Smoke Test (품질 검증의 가치)**

- Unit Test
- Integration Test
- System Test
- Acceptance Test
- Positive Test, Negative Test
- Black box Test, White box Test, Gray box Test
- Stress Test
- Boundary Test
- Field Test

개발자의 7계명

1. Broken Tree는 절대 만들지 않는다.
   (내가 만든 Build 실패는 회사의 막대한 손해를 끼친다)

2. 내가 올린 소스코드는 내가 탑승하는 비행기의 Software 라 생각하라.

3. 내가 올릴 소스코드에 대해 모든 검증(Unit test, Peer Review등) 이후 Commit 하라.
   (내가 올린 소스는 올리는 순간 모든 사람의 것이 된다)

4. Merge와 Conflict는 개발자인 내인생의 또 다른 반려자다 피하려고 하지 말라.

5. 버전관리 시스템의 자동 Merge Tool을 믿어라.
   그래도 정 의심이 가면 자동 Merge후 검토하면 된다.

6. Issue(Feature, Bug)는 반드시 등록 하고 나서 일하라.

7. 시스템이나 다른 개발자는 보조 수단일 뿐.
   시스템이나 동료 검토를 믿고 내가 할 일을 하지 않는 잘못을 범하지 말라.
   시스템이 나의 연봉을 책임져 주지 않는다.

이 모든 계명을 지키면 개발자의 인생이 즐거워 진다.

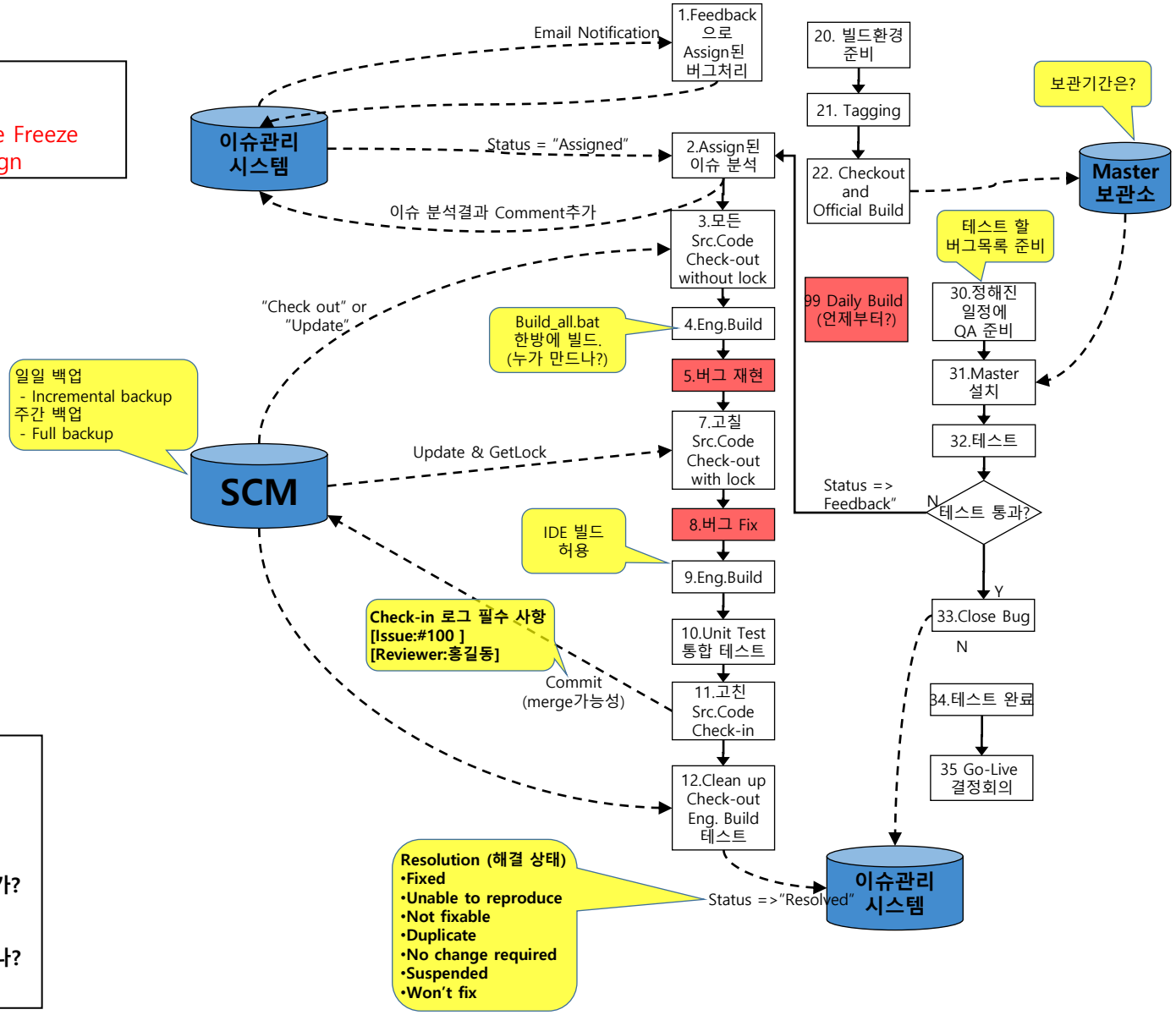
Peer Review (or Pair Programming, or Code Review?)

- Peer Review의 중요성
  - 테스트 보다 5배나 많은 Bug를 찾아낸다
  - 테스트와 다른 중요한 점은?
  - 그래도 안 하는 이유는?
    - 시간이 없다
    - 개발 프로세스상 환경이 준비되어 있지 않다
- 방법 – Desk Check, Review, Inspection,...
  - 무엇을
  - 언제
  - 누가
  - 어떻게
  - 왜
  - 어디서

일반적인(고전적인?) 개발 업무 프로세스

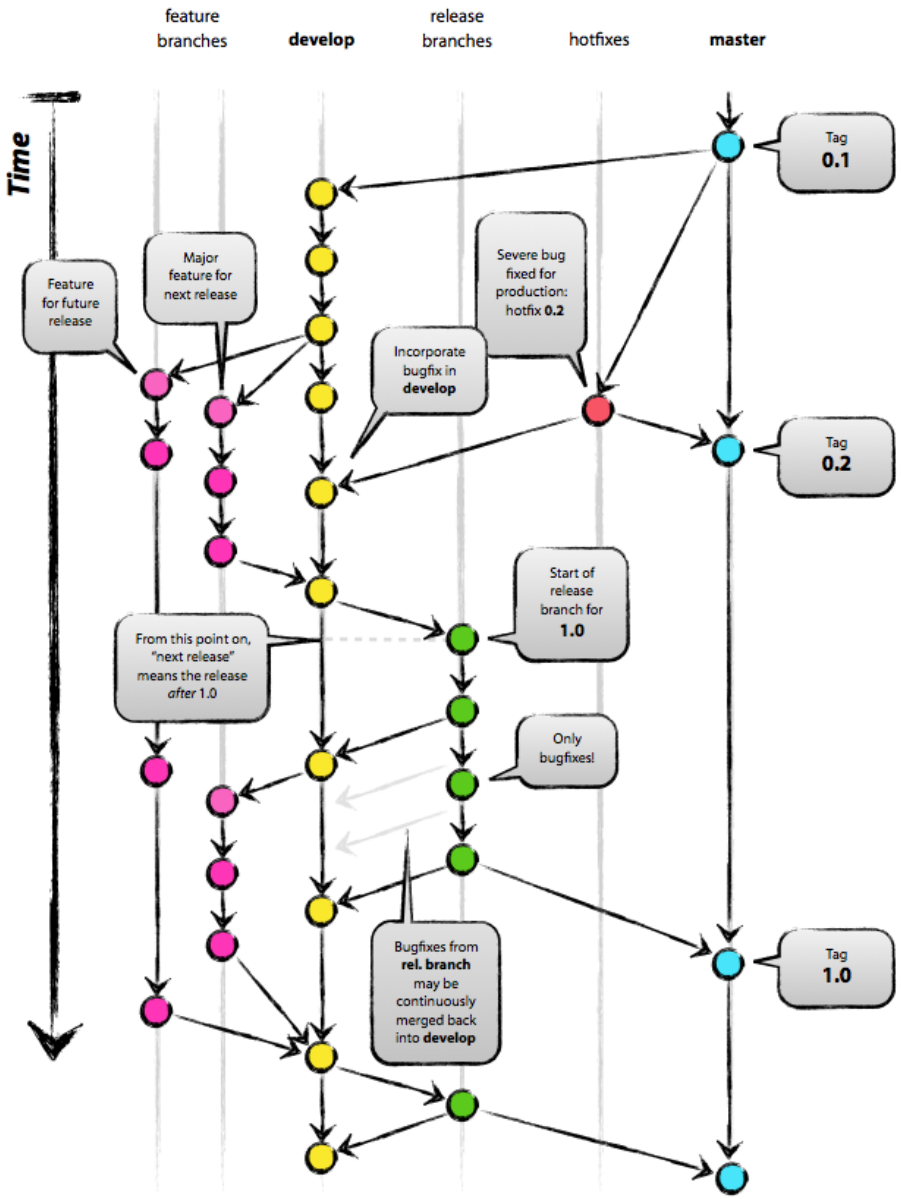
INDUSTRIAL AI RESEARCH CENTER

Daily 이슈 회의
-오늘 : 11월 1일 오전 9시
-12/23일 오후 6시에 RC2.1 Code Freeze
- Issue 123번은 홍길동에게 Assign



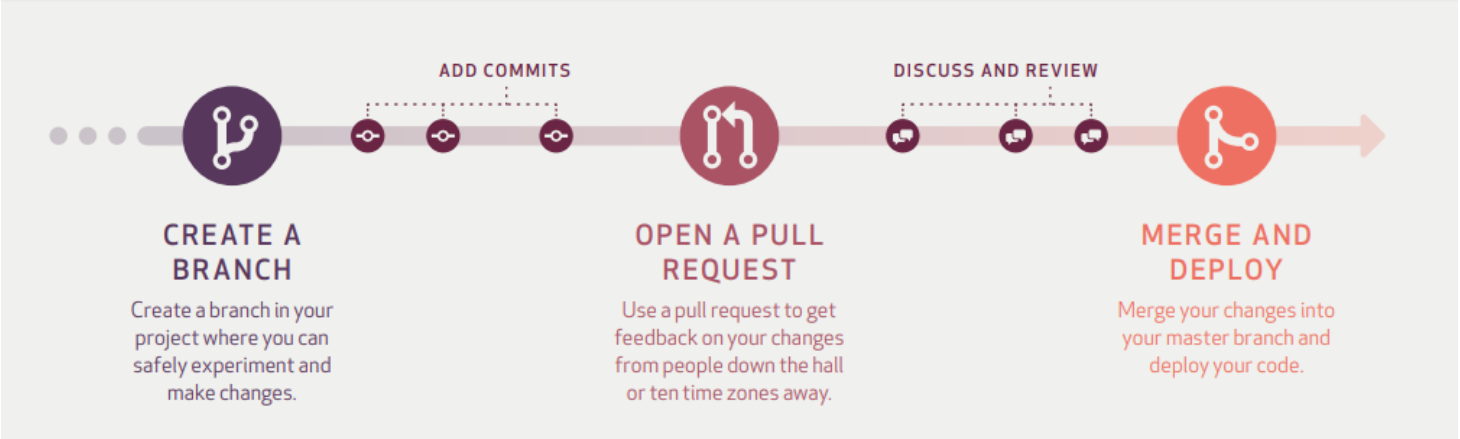
- 결재/승인 과정은?
-기존고객의 Hot Fix은 어떻게 처리?
- Peer Review는 언제?
- Go-Live 결정회의
- Lock의 목적은?
- 공식빌드는 왜 빌드기계에서만 하는가?
- Hot deployment
- Merge는 얼마나 자주하는가?
- Domain Knowledge은 어디서 배우나?
- Technical Skill은 어디서 배우나?

Git Flow



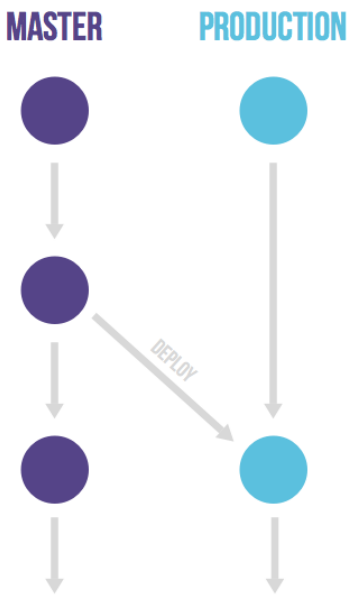
- 전통적인 개발 프로세스에 적합
- Release 시점 및 버전 관리가 명확해야 함.
- Branch or Tag 관리의 어려움.
- Issue Tracking System 필요

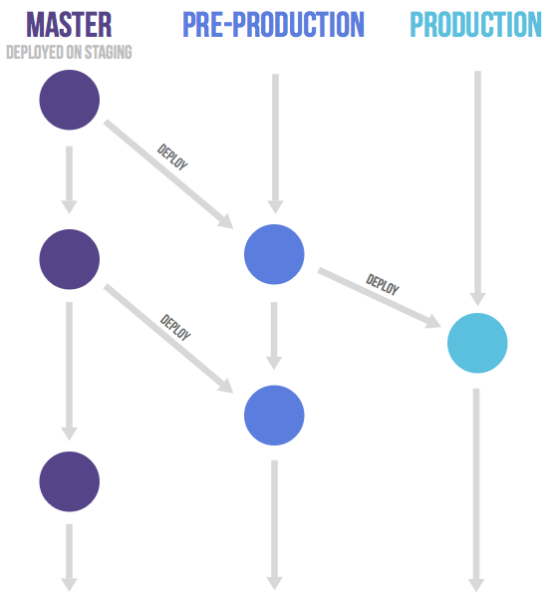
Github Flow

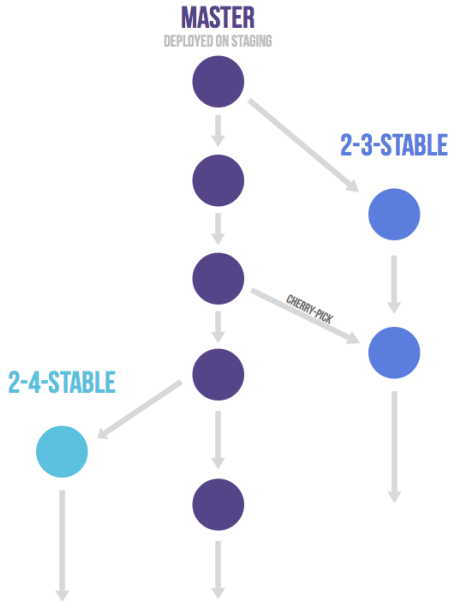


- 개별 개발자에 의한 오픈 소스 프로젝트 적합
- Pull Request로 인한 Code Review가 가능
- Source 관리자 필요?

Gitlab Flow







- Master Branch에 개발 집중. 자연스러운 개발 방향.
- 개별 Branch에 의한 작업 상황 무시

중요 체크 사항

- 개별 Branch에 작업하지 않는다면 작업 전 Pull하여 최신 버전을 유지한다.
  - ◆ 일반적인 경우 하루 단위로 Commit 한다.
- Commit하기 전 Check for Modification하여 본인이 수정한 코드를 다시 Review한다.
  - ◆ 프로젝트 파일(csproj), 리소스 파일도 확인
  - ◆ 프로젝트 파일을 확인
    - 불필요한 파일 추가되지 않도록 한다.
    - 프로젝트에 신규 생성한 파일이 Add되었는지 확인한다. (Broken Tree 방지)
- Commit Message는 자신이 작업한 내용을 한줄 요약하여 적는다.
  - ◆ 다음 줄 부터는 자세히 적어도 무방
- Push 전 다시 Pull하여 최신 버전을 만든 후 Build 확인 및 수정 사항을 다시 확인 후 Push한다.
  - ◆ Conflict 발생시 잘 모르겠다면 소스 확인하여 개발자와 협의하여 해결한다.

추후 검토 사항

- 각 개발 프로젝트에 맞는 업무 프로세스는?
- Issue Tracking System(Redmine, JIRA, Mantis)의 활성화
- 형상 관리 프로그램 도입 검토
  - ◆ 개별 Branch 개발 사항 체크 용이