형상 관리를 위한 Git & Github

Chapter 1

Git을 이용한 포트폴리오 형상관리

Chapter 1

Git을 이용한 포트폴리오 형상관리

Semver를 이용한 버전 관리

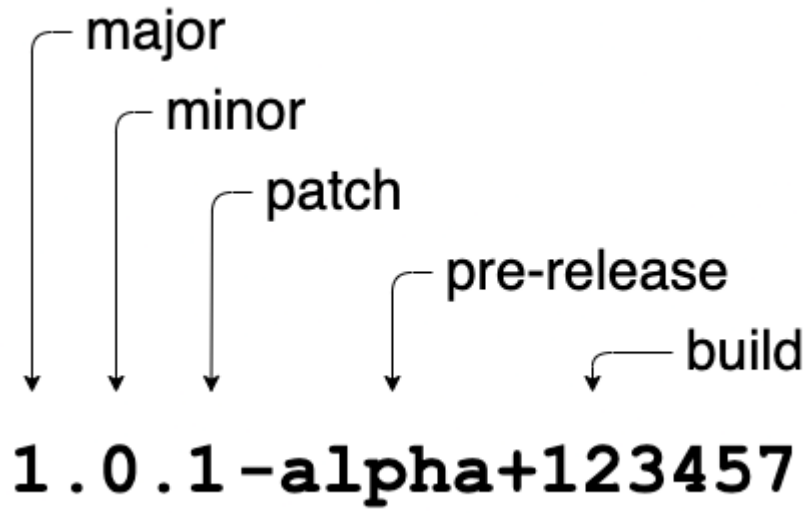
Semver

- **Versioning**

특정 상태에 유일한 버전 이름 혹은 버전 번호를 결정하는 과정

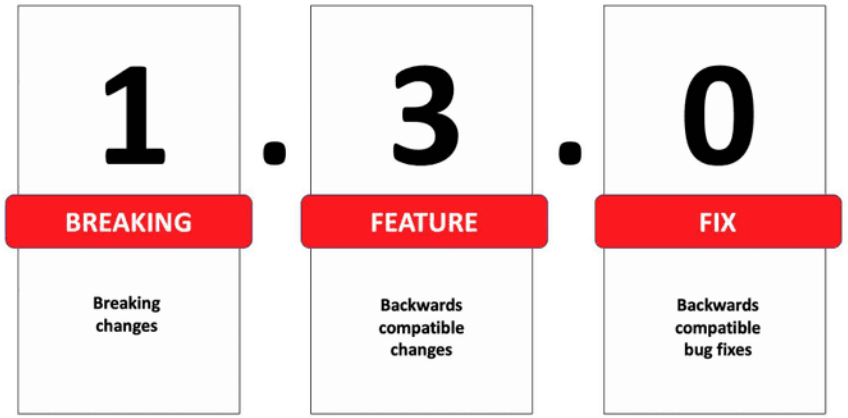
- **Semantic Versioning (Semver)**

Versioning에 대한 규칙과 요구사항이 담긴 명세서



기본 규약

- Major 버전 : 하위 버전과 호환성을 보장하지 않는 API 변경이 발생
- Minor 버전 : 하위 버전과 호환되며 API 및 기능이 추가 됨
- Patch 버전 : 하위 버전과 호환되며 버그가 수정된 것



소프트웨어 생명 주기

•Alpha
- 조직 내부적으로 이뤄지는 버전 테스트 시작 단계
- 더 이상 기능이 추가되지 않을 때 alpha 종료

•Beta
- public 한 사용자의 테스트로 이뤄지는 버전
- 기능 개발은 완료됐지만 어떤 문제가 있을지 모르는 단계

•RC(Release Candidate)
- 최종 제품이 될 가능성이 있는 베타 버전
- 릴리스 후보는 프로덕션 배포용이 아니라 테스트용
- 부하 테스트 등 모든 테스트 종류의 최종 단계가 이뤄짐

우선 순위

- 1.0.0-alpha
- 1.0.0-beta
- 1.0.0-alpha.1
- 1.0.0-beta.2
- 1.0.0
- 1.0.0-rc.1
- 1.2.1

우선 순위

- 1.0.0-alpha
- 1.0.0-beta
- 1.0.0-alpha.1
- 1.0.0-beta.2
- 1.0.0
- 1.0.0-rc.1
- 1.2.1

- 1.0.0-alpha < 1.0.0-alpha.1 < 1.0.0-beta < 1.0.0-beta.2 < 1.0.0-rc.1 < 1.0.0 < 1.2.1

호환성

- **Tilde (~)**

  ~1.2.3 : >= 1.2.3 < 1.3.0

  ~1.2 : >= 1.2.0 < 1.3.0 : 1.2.x

  ~1 : >= 1.0.0 < 2.0.0 : 1.x

- **Caret (^)**

  ^1.2.3 : >= 1.2.3 < 2.0.0

  ^1.2 : >= 1.2.0 < 2.0.0

  ^1 : >= 1.0.0 < 2.0.0

  ^0.1.2 : >= 0.1.2 < 0.2.0 (1.0.0 미만일 경우 Tilde와 같이 동작)