

For Developers

FADU와 함께 진행하는 산학주제

목차

- 산학 기관(FADU)의 주제
- 제작 의도
- 시스템 구성도
- 만들고자 하는 기술과 해당 기술의 필요성
- 최종 목표
- 전체적인 개발 단계
- 전체적인 프로그램 흐름(사용법)
- 일정
- 멘토링, 프로젝트 관리

산학 기관(FADU)의 주제

Jenkins Plug-in 개발 프로젝트

FADU 펌웨어 및 내부 툴 개발 단계에서 소프트웨어 품질 관리를 위한 Plugin 개발

A팀

- Code coverage status reporter
- Build status reporter
- Unit test status reporter

B팀

- Code styling status reporter
- Static analysis status reporter

팀간 협업

- Dashboard for status

산학 기관(FADU)의 주제

Jenkins Plug-in 개발 프로젝트

FADU 펌웨어 및 내부 툴 개발 단계에서 소프트웨어 품질 관리를 위한 Plugin 개발

A팀

- Code coverage status reporter
- Build status reporter
- Unit test status reporter



12 TEAM

B팀

- Code styling status reporter
- Static analysis status reporter

팀간 협업

- Dashboard for status

산학 기관(FADU)의 주제

Jenkins Plug-in 개발 프로젝트

FADU 펌웨어 및 내부 툴 개발 단계에서 소프트웨어 품질 관리를 위한 Plugin 개발

A팀

- Code coverage status reporter
- Build status reporter
- Unit test status reporter



12 TEAM

B팀

- **Code styling status reporter**
- **Static analysis status reporter**



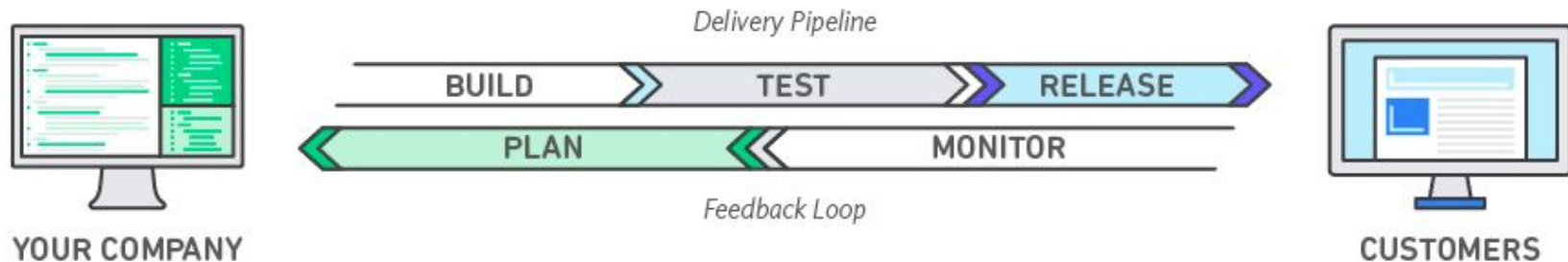
19 TEAM

팀간 협업

- Dashboard for status

제작 의도

DevOps의 등장으로 개발, 테스트 주기가 빨라져 **CI/CD**가 자동화될 필요 존재



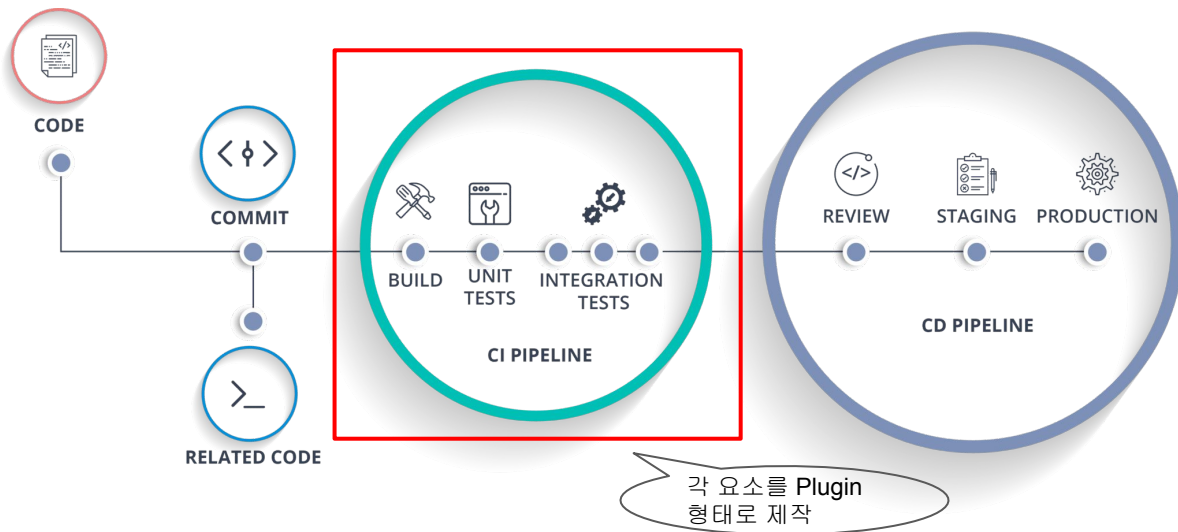
DevOps: 제품을 보다 빠르게 혁신하고 개선, 이로 인해 조직은 고객을 더 잘 지원하고 시장에서 좀 더 효과적으로 경쟁 가능하게 해주는 이점이 있다.

제작 의도

CI/CD를 각 조직에 적절하게 **자동화**해줄 필요

-> CI/CD 지원 Jenkins의 **플러그인** 형태로 자동화를 도와주도록 제작 예정

-> 이로 인해 개발자는 **개발**에만 집중할 수 있는 환경이 구성됨



제작 의도

아; 다른 사람 코드 알아보기 너무 힘들어..
왜 막 쓰는거야 도대체!

-> **Code style** 검사

팀마다 규정된 스타일을 따라 코드가
작성되었는지 확인하는 검사



제작 의도

간단한 프로그램인데 코드가 거의 1000줄 가까이 되네요;; 정말 이게 다 필요한 코드예요?

-> **Static analysis** 검사

조건문에서 수행되지 않을 코드가 작성되었는지 등 불필요한 코드가 있는지 확인하는 검사



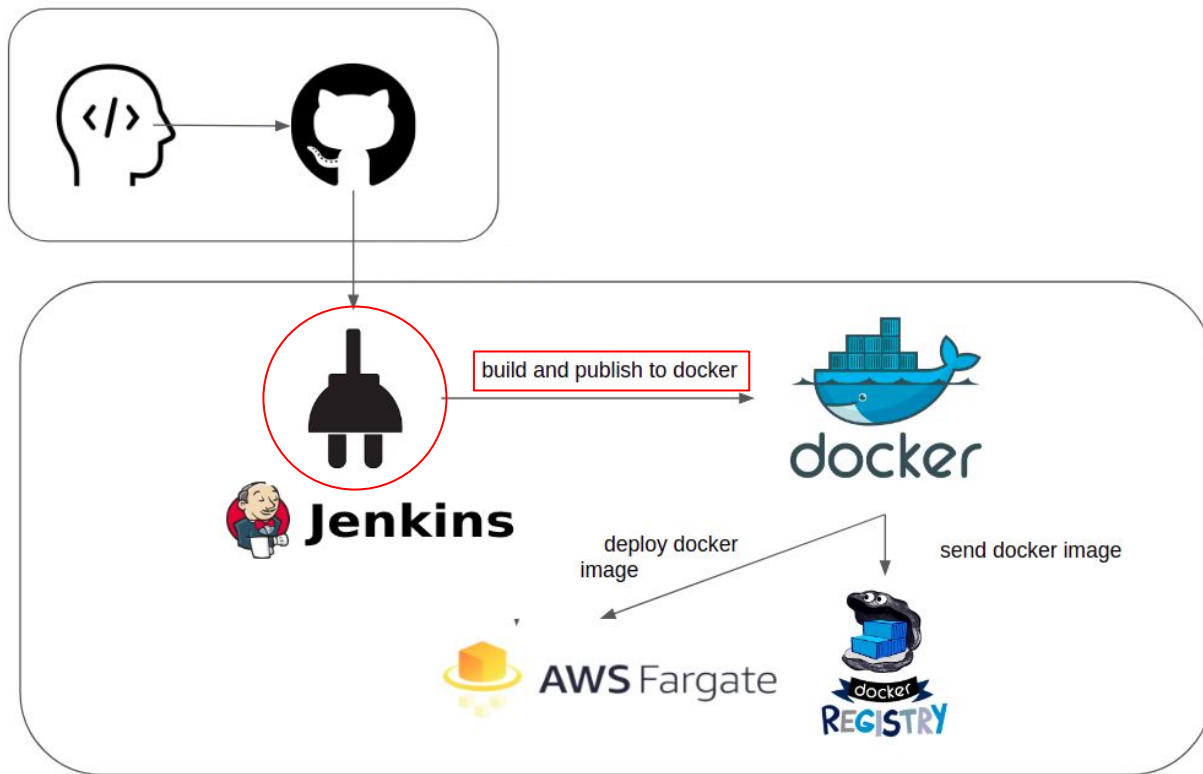
제작 의도

검사만 한다고 끝나는 게 아니야 피드백을
해야지!

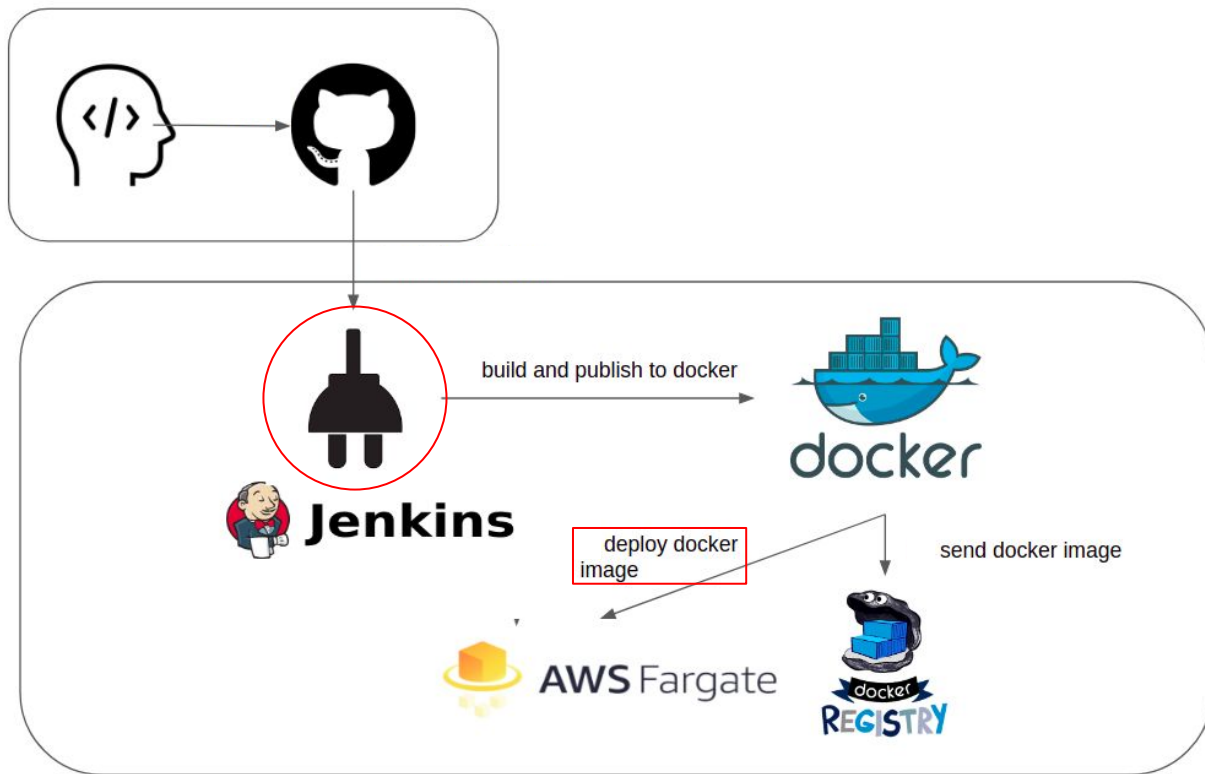


코드 상에서 어느 부분이 스타일을 위반했고,
어느 부분이 불필요하게 작성되었는지 보여주는
report와 **대시보드** 형태로 보여주도록 피드백
제공

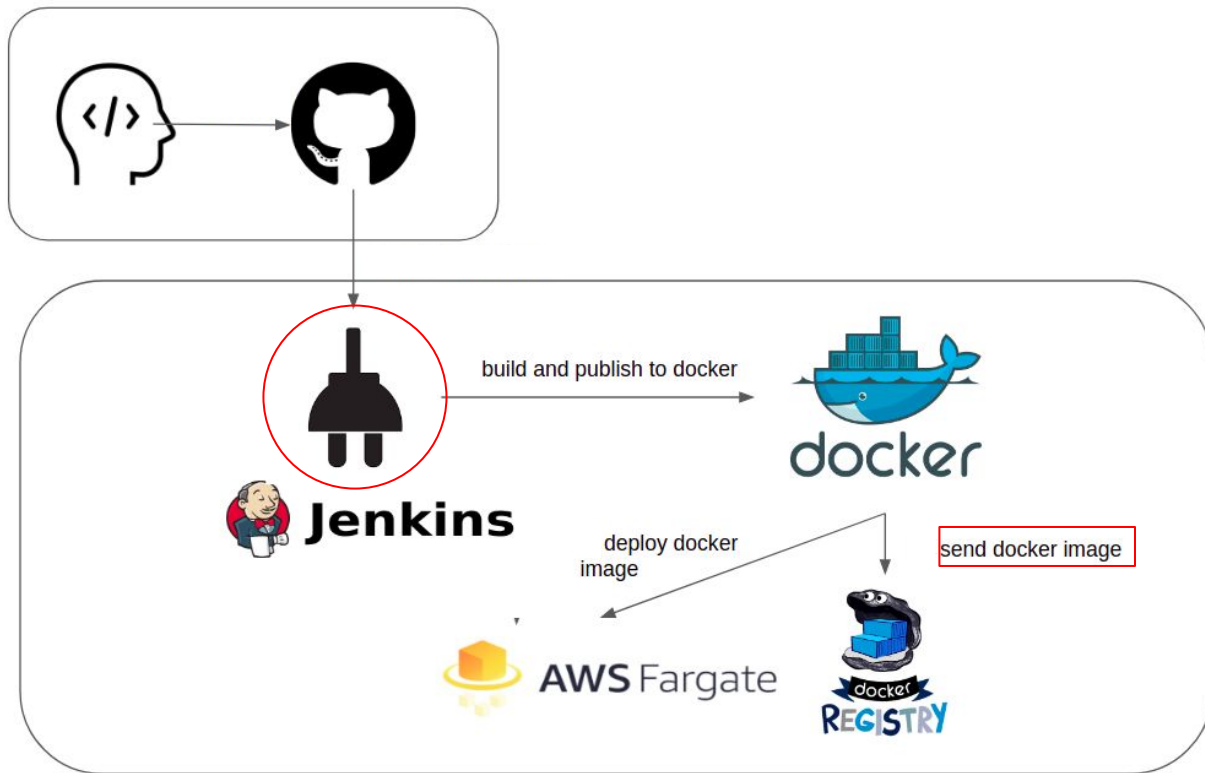
시스템 구성도



시스템 구성도



시스템 구성도

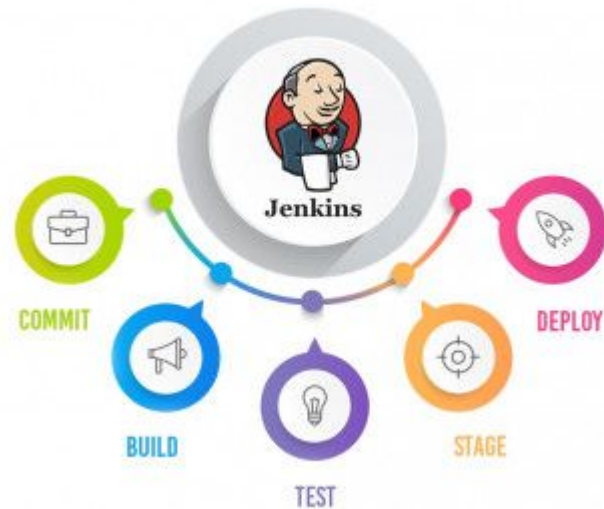


Jenkins

지속적이고 안정적으로 통합 서비스를 제공하는 CI 툴

선택 이유

- 가장 많이 사용되는 CI/CD 툴
- 실제 산학 협력 담당 회사에서 사용중인 툴



만들려는 기술 - Code styling status reporter plugin

Build 자동화 도구인 **Jenkins**에서 이용 가능한 플러그인 제작

1. Code styling status reporter plugin

- 기업의 요청에 따른 코드 스타일 검사 툴 **Clang-format** 사용
- 사전에 정의해놓은 스타일을 따르지 않는 부분을 코드상에서 찾아서 알려줌

해당 기술이 필요한 이유 - Code styling status reporter plugin

- 각 개발자가 사용하는 스타일이 다르면 다른 개발자가 개발한 코드를 볼 때 가독성이 좋지 않음
- 가독성이 좋지 않다는 말은 개발 외에 다른 부분에 또다른 노력을 해야할 필요가 있다는 의미
- 사용하는 환경에 따라 다른 스타일은 에러를 유발할 수도 있음

=> 코드 구조의 일관성을 유지하여 가독성을 높이고 협업 및 유지보수의 효율을 향상시키는 것에 도움을 주는 것이 목표

만들려는 기술 - Static analysis Status reporter plugin

Build 자동화 도구인 Jenkins에서 이용 가능한 플러그인 제작

2. Static analysis Status reporter plugin

- 기업의 요청에 따른 IKOS 툴을 사용하여 코드 분석
- Unreachable, UnexpectedOperend, Division By Zero, UnknownMemoryAccess, BufferOverflow..등 의 에러를 찾아줌

해당 기술이 필요한 이유 - Static analysis Status reporter plugin

- 런타임 오류가 없는 것을 감지하고 증명하기 위한 확장 가능한 분석을 구현
- 사전에 코드가 필요하지 않거나 에러가 나는 코드를 알게 됨으로 미리 제거 할 수 있음

=> 따라서 코드의 오류를 사전에 방지하고 필요 없는 코드를 제거함으로 가독성을 높이고 코드를 읽는 것을 편리하게 한다.

최종 목표

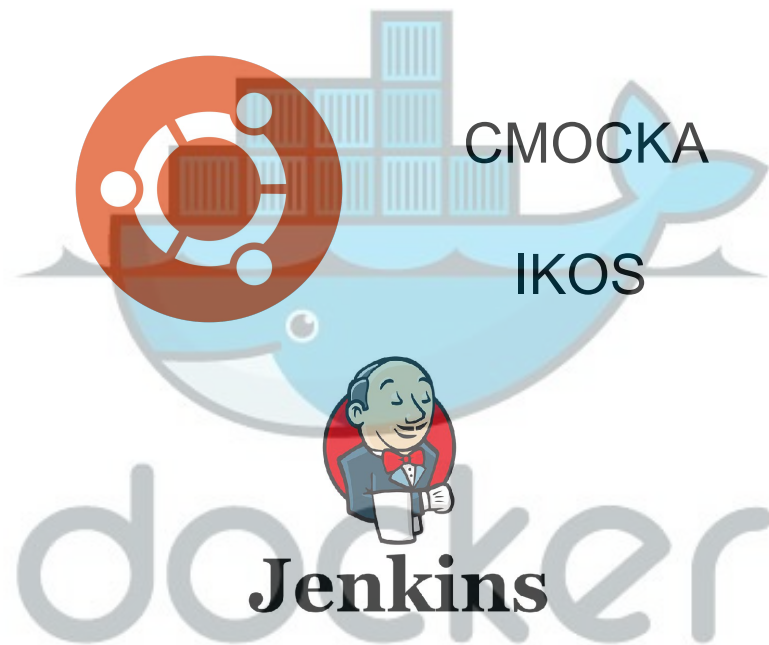
- 젠킨스에서 간편하게 설치해 문제가 될 수 있는 부분을 미연에 방지
- 가독성을 높이고 유지보수가 원활하게 이루어질 수 있는 프로젝트를 할 수 있도록 돕는 것이 목표



전체적인 개발 단계

1. 도커에 개발 환경 작성

-> 동일한 환경에서 개발함으로써
버전 차이 등의 문제를 없애기 위함



전체적인 개발 단계

2. AWS fargate와 도커 연동

AWS ECS가 아닌 새로운 Fargate를 사용?

-> 관리할 클러스터가 없음!

-> EC2를 직접 올리지 않음!



AWS Fargate

전체적인 개발 단계

3. 예제 코드를 *Static Analysis Tool / Clang-format* 툴을 사용해 해당 결과를 텍스트 파일로 저장

- IKOS 실행 결과

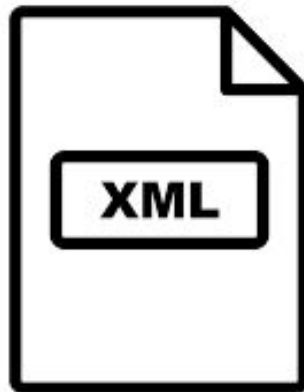
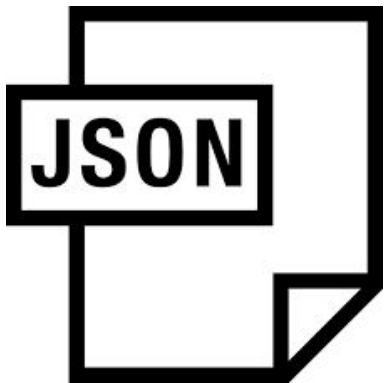
```
loop.c: In function 'main':  
loop.c:9:13: error: division by zero  
    printf(i%0);  
            ^
```

- Clang-format 실행 결과

```
./test.c:14:17: warning: code should be clang-formatted [-Wclang-format-violations]  
class EmptyClass  
      ^  
./test.c:15:2: warning: code should be clang-formatted [-Wclang-format-violations]  
{  
^
```

전체적인 개발 단계

4. 텍스트 파일을 JSON/XML로 변환하는 *translator* 작성



전체적인 개발 단계

5. 변환한 파일을 실제 사용자가 볼 정보로 구성되도록 *parser* 작성

```
{
  "1": [
    {
      "file": "loop.c",
      "code": "a[i] = i;",
      "function": "In function 'main'",
      "column": "8",
      "row": "10",
      "check": "      ^",
      "status": "error",
      "info": "buffer overflow, accessing index 10 of global variable 'a' of 10 elements"
    }
  ]
}
```



1th

loop.c In function 'main' 8:10 error buffer overflow, accessing index 10 of global variable 'a' of 10 elements

```
a[i] = i;
      ^
```


전체적인 개발 단계

6. *translator*와 파서 *plugin*으로 빌드

Failed

DocumentProviderImpl.cpp.[658/5] [bugprone-inaccurate-erase]

Error Message

warning: this call will remove at most one item even when multiple items should be removed

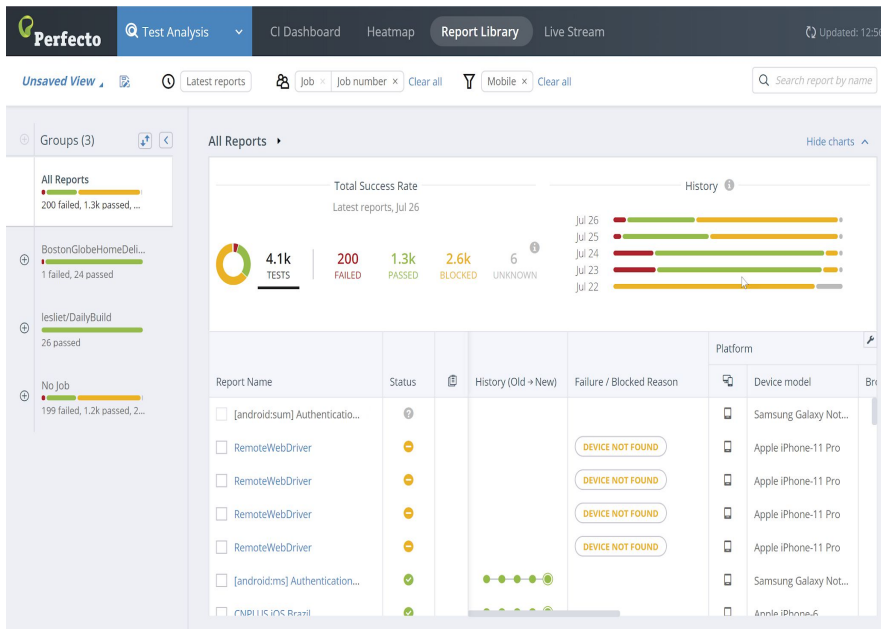
Stacktrace

```
vector.erase(std::remove_if(vector.begin(), vector.end(), [](int x) { return x > 2; }));
```

^

전체적인 개발 단계

7. 대시보드 작성



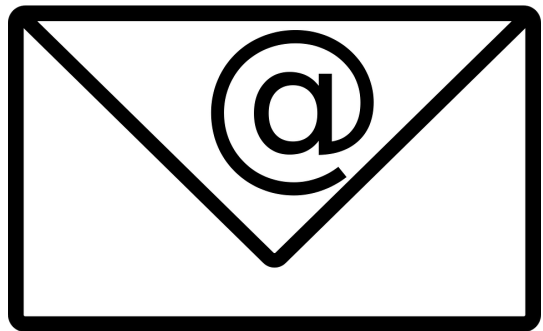
- Back to Project
- Status
- Changes
- Console Output
- Edit Build Information
- Delete Build
- Greeting**
- Previous Build

전체적인 개발 단계

8. *사용자에게 알람*



slack



전체적인 프로그램 흐름(사용자)

1. Jenkins와 git 연동
2. git commit&push가 일어나면 webhook으로 젠킨스에 자동으로 빌드 설정
3. 개발하고 있는 코드 수정 및 github에 commit & push
4. Jenkins에서 자동 빌드
5. Jenkins에서 코드상 잘못된 부분 dashboard에서 알려줌
6. 동시에 slack / mail 등으로 해당 결과를 받아볼 수 있음



Jenkins



GitHub

현재까지 진행 상황

2월 1주차	2월 2주차	2월 3주차	2월 4주차	3월 1주차	3월 2주차	3월 3주차	3월 4주차
Jenkins, Maven 스터디	Jenkins, Maven 스터디			Docker study			
	Jenkins tutorial	Jenkins tutorial		Dockerfile 작성	Dockerfile 작성		
		플러그인 별 툴 사용법 익히기	플러그인 별 툴 사용법 익히기		AWS 스터디	AWS 스터디	
						AWS tutorial	AWS Tutorial
						Dockerfile 관리	AWS makefile Study

일정 계획

4월 1주차	4월 2주차	4월 3주차	4월 4주차	5월 1주차	5월 2주차	5월 3주차
parser에 정보 추가	parser에 정보 추가					
AWS Makefile 작성	Side panel 작성	Side panel 작성				
	AWS Makefile 작성	Side panel에 parser 연결	Side panel에 parser 연결			
			대시보드 작성	대시보드 작성		
				내부테스트 진행	내부테스트 진행	
					테스트 결과 보완	테스트 결과 보완

파두 고원석 박사님의 멘토링

- Study자료 및 개인이 사용할 툴 제시
- Dockerfile 직접 사용 후 조언
- 개인별 질문
- Github 사용법 및 Commit 하는 방법
- 미팅에 참여 후 피드백
- IKOS/Clang-Format이 지원하는 Json/XML 형식을 사용하지 않고 customized된 translator 작성할 것을 조언
- 젠킨스에서 직접 만든 플러그인을 빠르고 간단하게 테스트 할 수 있는 방법 제시

멘토님의 멘토링

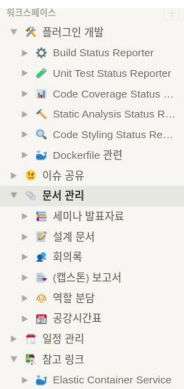
- 일정 관리 툴 소개와 일정 관리법 안내
- 팀 프로젝트에 걸맞은 분업 방법 안내
- 현업에서 젠킨스 사용 경험으로 젠킨스의 필요성을 안내
- 개발 진행 과정 기록의 필요성 안내

교수님과의 멘토링

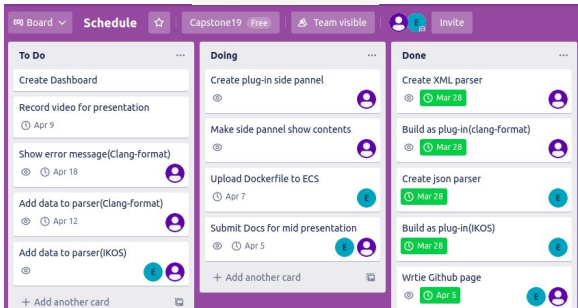
- 전체적인 프로젝트의 구조와 개발 단계 안내
- AWS에 대한 필요성 안내
- AWS Fargate의 활용도에 대한 안내
- 회의를 통한 피드백 제시

개발 관리

- Notion -> 전체 팀의 문서, 이슈 관리




- Trello -> 개발 일정 관리



- slack -> Study 관리



 **kimmj** 오전 1:30
sudo ls -i:50000
sudo kill PID

 **eun ji** 오후 3:15
<https://www.44bits.io/ko/post/building-docker-image-basic-commit-diff-and-dockerfile>
44BITS
도커 이미지 빌드와 Dockerfile 기초
도커(Docker)에서 이미지 빌드는 컨테이너와 함께 가장 중요한 개념입니다. 이미지는 컨테이너의 기반이 되는 파일들의 집합입니다. 사용자는 리눅스 배포판 이미지를 기반으로 자유롭게 프로비저닝을 하고 커스텀 이미지를 만들 수 있습니다. 이 글에서는 도커 이미지의 기본적인 빌드 원리와 Dockerfile의 사용 방법에 대해서 소개합니다. (43kB)

개발 관리

- 매주 금요일 A, B 팀 통합 회의
- 2주에 한번씩 기업, 교수님 미팅
- 주기적으로 멘토님과 미팅
- 팀원과 매주 2번 이상 회의 및 진행 상황 공유

앞으로 멘토링을 받으며 진행할 내용

- 사용자가 사용할 때, 편의성을 높이기 위한 구조는 어떻게 구성해야 하는가?
- 가독성을 높이기 위한 대시보드는 어떠한 형태인가?
- 사용자가 편하게 사용할 수 있는 환경은 무엇인가?
- 개발하며 발생하는 이슈를 공유하며 해결책 논의

감사합니다