

|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **맑은 유림조** | **구성원** | **김영훈 김유림 지경민 이수경 석주연** |
| **주제** | **JIRA** | | |
| **주요 내용** | | | |
| **지라**(JIRA)는 [아틀라시안](https://ko.wikipedia.org/wiki/%EC%95%84%ED%8B%80%EB%9D%BC%EC%8B%9C%EC%95%88)이 개발하고 2002년에 발표한 [사유](https://ko.wikipedia.org/wiki/%EC%82%AC%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4) [이슈 추적](https://ko.wikipedia.org/wiki/%EC%9D%B4%EC%8A%88_%EC%B6%94%EC%A0%81_%EC%8B%9C%EC%8A%A4%ED%85%9C) 제품이다. [버그 추적](https://ko.wikipedia.org/wiki/%EB%B2%84%EA%B7%B8_%EC%B6%94%EC%A0%81_%EC%8B%9C%EC%8A%A4%ED%85%9C), 이슈 추적, [프로젝트 관리](https://ko.wikipedia.org/wiki/%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8_%EA%B4%80%EB%A6%AC) 기능을 제공하는 소프트웨어이며 유료 서비스다.  설치 링크 : <https://ko.atlassian.com/software/jira>  설명서 : <https://www.lesstif.com/pages/viewpage.action?pageId=12943700>    프로그래밍 언어 : 자바  운영 체제 플랫폼 : 독립  종류 : 버그 추적 시스템, 프로젝트 관리 소프트웨어  라이선스 사유: 공식적인 비영리 단체, 오픈 소스 프로젝트, 비정부 기관, 학술 및 종교 단체에는 이용 제한 없음  웹사이트 : atlassian.com/software/jira  지라(JIRA)라는 이름은 고지라라는 일본식 이름에서 따왔다.  2002년부터 지금까지 개발되고 있다.  영어, 프랑스어, 독일어, 일본어, 스페인어를 포함한 여러 번역을 포함하고 있다.  지라는 경쟁 제품인 버그질라로부터의 마이그레이션을 도와주는 도구를 포함하고 있다.  프로젝트의 3대 특성(일시적, 고유함, 점진적 구체화)을 유연하고 검증되어있는 방법론과  잘 구현된 프로세스를 도입하기 위하는 이들이 JIRA를 사용한다.  `협업 전문도구 Atlassian의 Issue 및 프로젝트 관리 시스템  `깔끔한 UI, 좋은 기술과 기능 지원제공  `자사 제품과 통합지원  `장기적으로 보았을 때 비용적 측면의 이익  `Open Source 제품에 비해 기술력과 시간, 비용적으로 절약이 가능  구성요소  이슈 관리 시스템  -오류, 버그 및 '새로운 기능', 작업요청, 사소한 질문이나 의견등 제품에 관해 회사에서 대화의 대상이 되는 거의 모든 것을 이슈라고 한다.  -S/W 공학의 정수로 형상 관리와 더불어 프로젝트의 핵심인프라  -일반적으로 Web 기반이라 별도의 설치없이 손쉽게 사용 가능  사용법(주요필드)  Issue 등록시 주요 필드  Summary: 대략적인 이슈 내용을 짐작할 수 있도록 요약해서 기술  - 이슈 공유합니다(X)  - 요청사항입니다(X)  - IE 11 사용시 일일 정산 및 통계 기능 미작동(O)  Type: 이슈의 유형(Bug, 개선, Task 등) 지정  Description: 상세한 이슈 내용 기록  비개발 직군의 참여를 높이기 위해 등록시 필수 항목은 최소화하고 추가항목은 화면 아래에 배치  Project  JIRA 프로젝트는 이슈들의 집합  - 소프트웨어 개발 프로젝트  - 마케팅 캠페인  - 웹사이트 개선 요청 시스템  Project는 Sub-Project를 가질 수 없음(redmine과의 차이점)  보통 하나의 제품은 하나의 프로젝트에 매핑  Component  이슈의 논리적인 집합으로 Sub-project 개념으로 사용 가능  - 문서 작업  - 백엔드 구성  - 고객 교육등  모든 Issue는 하나 이상의 컴포넌트를 가질 수 있음  기본 담당자 할당  -프로젝트와 Component마다 담당자를 지정해두면 이슈 등록시 자동 할당되므로 이슈 등록시 편리  Version  SW의 기능이나 개선점 등의 논리적인 묶음  V1.0, V2.0-alpha 등  제품과 프로젝트의 과거 주요 변경 내역과 앞으로 나아갈 방향을 손쉽게 만들고 공유가능  개별 Issue는 버전 관련 2가지 필드를 설정할 수 있음  - Affect Ver.버그등의 경우 발생하는 버전  - Fix Ver. 해당 이슈가 해결된 버전  이슈의 상태(Status)  이슈가 life cycle(work flow)내에 어떤 상태인지를 의미  일반적으로 이슈는 생성 후 처리되고 종료되지만 프로젝트의 유형에 따라 다른 work flow를 가질 수 있음  종료된 이슈라도 다시 열릴 수 있음(좀비버그)  Reopen이 될 수 있다는 것은 실제 프로젝트의 현실을 반영하며 이에 따른 유연한 대응이 가능  이슈처리상태(Resolution)  이슈가 어떻게 종료되었는지 구체적인 사유를 표시하는 항목  일반적으로 Fixed나 Closed로 해결된 이슈인 것을 표시  현실적으로 등록된 모든 이슈를 종료할 수는 없고 프로젝트의 우선 순위에 따라 선별하여 처리  자체적으로 처리할 수 없는 이슈(OS버그, 브라우저 버그 등)일 경우 Won't fix등으로 변경  작업흐름(Workflow)  이슈 관리 시스템의 가장 중요한 기능  이슈 lifecycle 이 완료될 때까지의 각 상태(Status)로의 전이(transition)관리 및 추적 가능  대부분의 ITS는 Workflow 생성/관리 기능 내장  프로젝트의 특성에 맞게 임의로 설정 가능  새로운 Workflow가 필요할 경우  관리자 권한으로 새로운 workflow 생성  또는 Market place에 올라온 workflow 검색 & 설치    이슈연결(Link)  PMP의 PDM보다 더 실용적으로 이슈간 관계 및 우선순위를 관리하는 방법  https://www.lesstif.com/pages/viewpage.action?pageld=51282196  이슈 검색  살아있는 프로젝트 진행 정보를 다양한 조건으로 검색 가능  강력한 검색을 위한 전용 언어 JQL 제공  개별 담당자는 보고를 위한 자료를 만들 필요없이 개별 Issue만 최신 정보로 갱신하면 되며 아래와 같은 요청 최소화  ex)  -금주 주간업무 작성해 주세요  -주요 업무 현황 보고 바랍니다.  이슈 삭제(Delete)  중복등의 이유로 이슈를 삭제하는 것  동일한 버그라도 사용자 관점에서는 다른 증상으로 나타날 수 있으므로 삭제보다는 ISSUE Link 사용하며 관계를 중복적으로 설정  이슈를 삭제할 경우 이슈 등록자는 무시당했다는 느낌을 가질 수 있음  Issue 처리 내역을 댓글로 하지 않고 본문 수정  해당 이슈를 처리한 이력과 변경사항은 댓글로 기술하면 한 눈에 이력이 표시됨  위키처럼 변경 이력을 이슈인 Description(본문) 부분만 수정하여 최종본만 남기는 경우 처리 이력을 보기가 어려움  History 탭을 클릭해서 변경 이력을 확인해야 하나 가독성이 떨어짐  지라는 해당 이슈를 처리한 이력도 중요하기 때문에 최근 버전이 필요한 위키처럼 사용하면 안 됨.  PHP에서 JIRA와 연동 필요시  https://github.com/lesstif/php-jira-rest-client 추천 | | | |



|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **맑은 유림조** | **구성원** | **김영훈 김유림 지경민 이수경 석주연** |
| **주제** | **GIT** | | |
| **주요 내용** | | | |
| **깃(Git /ɡɪt)**    [분산 버전 관리 시스템](https://ko.wikipedia.org/wiki/%EB%B6%84%EC%82%B0_%EB%B2%84%EC%A0%84_%EA%B4%80%EB%A6%AC_%EC%8B%9C%EC%8A%A4%ED%85%9C)  [컴퓨터 파일](https://ko.wikipedia.org/wiki/%EC%BB%B4%ED%93%A8%ED%84%B0_%ED%8C%8C%EC%9D%BC)의 변경사항을 추적하고 여러 명의 사용자들 간의 작업을 조율이 쉬움  빠른 수행 속도에 중점. 데이터 무결성, 분산, 비선형 워크플로를 지원  출처:[ttps://ko.wikipedia.org/wiki/%EA%B9%83\_(%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)#cite\_note-5](https://ko.wikipedia.org/wiki/%EA%B9%83_(%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)#cite_note-5)  **설치(링크 참조)**  -[OS X용 git 다운로드](http://git-scm.com/download/mac)  -[Windows용 git 다운로드](http://msysgit.github.io/)  -[Linux용 git 다운로드](http://book.git-scm.com/2_installing_git.html)  **기본 용어**  이 튜토리얼에서 반복적으로 사용하려고 하는 몇 개의 용어가 있다. 나도 배우기 시작하기 전에는 들어본 적이 없는 것들이다. 여기 중요한 것들이 있다:  커맨트 라인(Command Line): 깃 명령어를 입력할 때 사용하는 컴퓨터 프로그램. 맥에선 터미널이라고 한다. PC에선 기본적인 프로그램이 아니어서 처음엔 깃을 다운로드해야 한다(다음 섹션에서 다룰 것이다). 두 경우 모두 마우스를 사용하는 것이 아닌 프롬프트로 알려진 텍스트 기반 명령어를 입력한다.  저장소(Repository): 프로젝트가 거주(live)할 수 있는 디렉토리나 저장 공간. 깃허브 사용자는 종종 “repo”로 줄여서 사용한다. 당신의 컴퓨터 안의 로컬 폴더가 될 수도 있고, 깃허브나 다른 온라인 호스트의 저장 공간이 될 수도 있다. 저장소 안에 코드 화일, 텍스트 화일, 이미지 화일을 저장하고, 이름붙일 수 있다.  버전관리(Version Control): 기본적으로, 깃이 서비스되도록 고안된 목적. MS 워드 작업할 때, 저장하면 이전 화일 위에 겹쳐쓰거나 여러 버전으로 나누어 저장한다. 깃을 사용하면 그럴 필요가 없다. 프로젝트 히스토리의 모든 시점의 “스냅샷”을 유지하므로, 결코 잃어버리거나 겹쳐쓰지 않을 수 있다.  커밋(Commit): 깃에게 파워를 주는 명령이다. 커밋하면, 그 시점의 당신의 저장소의 “스냅샷”을 찍어, 프로젝트를 이전의 어떠한 상태로든 재평가하거나 복원할 수 있는 체크포인트를 가질 수 있다.  브랜치(Branch): 여러 명이 하나의 프로젝트에서 깃 없이 작업하는 것이 얼마나 혼란스러울 것인가? 일반적으로, 작업자들은 메인 프로젝트의 브랜치를 따와서(branch off), 자신이 변경하고 싶은 자신만의 버전을 만든다. 작업을 끝낸 후, 프로젝트의 메인 디렉토리인 “master”에 브랜치를 다시 “Merge”한다.  **주요 명령어**  깃은 리눅스와 같은 큰 프로젝트를 염두에 두고 디자인되었기 때문에, 깃 명령어는 아주 많다. 그러나, 깃의 기본을 사용할 때에는 몇 개의 명령어만 알면된다. 모두 “git”이란 단어로 시작된다.  git init: 깃 저장소를 초기화한다. 저장소나 디렉토리 안에서 이 명령을 실행하기 전까지는 그냥 일반 폴더이다. 이것을 입력한 후에야 추가적인 깃 명령어들을 줄 수 있다.  git config: “configure”의 준말, 처음에 깃을 설정할 때 가장 유용하다.  git help: 명령어를 잊어버렸다? 커맨드 라인에 이걸 타이핑하면 21개의 가장 많이 사용하는 깃 명령어들이 나타난다. 좀 더 자세하게 “git help init”이나 다른 용어를 타이핑하여 특정 깃 명령어를 사용하고 설정하는 법을 이해할 수도 있다.  git status: 저장소 상태를 체크. 어떤 화일이 저장소 안에 있는지, 커밋이 필요한 변경사항이 있는지, 현재 저장소의 어떤 브랜치에서 작업하고 있는지 등을 볼 수 있다.  git add: 이 명령이 저장소에 새 화일들을 추가하진 않는다. 대신, 깃이 새 화일들을 지켜보게 한다. 화일을 추가하면, 깃의 저장소 “스냅샷”에 포함된다.  git commit: 깃의 가장 중요한 명령어. 어떤 변경사항이라도 만든 후, 저장소의 “스냅샷”을 찍기 위해 이것을 입력한다. 보통 “git commit -m “Message hear.” 형식으로 사용한다. -m은 명령어의 그 다음 부분을 메시지로 읽어야 한다는 것을 말한다.  git branch: 여러 협업자와 작업하고 자신만의 변경을 원한다? 이 명령어는 새로운 브랜치를 만들고, 자신만의 변경사항과 화일 추가 등의 커밋 타임라인을 만든다. 당신의 제목이 명령어 다음에 온다. 새 브랜치를 “cats”로 부르고 싶으면, git branch cats를 타이핑한다.  git checkout: 글자 그대로, 현재 위치하고 있지 않은 저장소를 “체크아웃”할 수 있다. 이것은 체크하길 원하는 저장소로 옮겨가게 해주는 탐색 명령이다. master 브랜치를 들여다 보고 싶으면, git checkout master를 사용할 수 있고, git checkout cats로 또 다른 브랜치를 들여다 볼 수 있다.  git merge: 브랜치에서 작업을 끝내고, 모든 협업자가 볼 수 있는 master 브랜치로 병합할 수 있다. git merge cats는 “cats” 브랜치에서 만든 모든 변경사항을 master로 추가한다.  git push: 로컬 컴퓨터에서 작업하고 당신의 커밋을 깃허브에서 온라인으로도 볼 수 있기를 원한다면, 이 명령어로 깃허브에 변경사항을 “push”한다.  git pull: 로컬 컴퓨터에서 작업할 때, 작업하고 있는 저장소의 최신 버전을 원하면, 이 명령어로 깃허브로부터 변경사항을 다운로드한다(“pull”).  **처음으로 깃/깃허브 설정하기**  먼저, GitHub.com에 가입한다. 다른 소셜 네트워크에 가입하는 것처럼 간단하다.  로컬 컴퓨터에서 작업하려면 깃을 설치해야 한다. 필요에 따라 윈도우, 맥, 리눅스 용 깃을 설치하라.  이제 커맨드 라인으로 넘어갈 시점이다. 윈도우에선 방금 설치한 Git Bash 앱으로, OS X에선 터미널로 시작한다. 깃에 자신을 소개할 차례이다. 다음 코드를 타이핑한다:  git config --global user.name "Your Name Here"  물론, “Your Name Here”의 인용부호 안에 자신의 이름을 넣어야 한다.  다음엔, 당신의 이메일을 말해준다. 조금 전에 GitHub.com을 가입할 때 사용한 이메일이어야 한다. 다음과 같이 한다:  git config --global user.email "your\_email@youremail.com"  이것이 로컬 컴퓨터에서 깃을 사용할 때 필요한 모든 것이다. 원한다면, 깃과 소통할 때마다 GitHub.com 계정에 로드인하는 것을 요청하지 않도록 깃을 설정할 수 있다. 이것과 관련된 풀 튜토리얼은 깃허브에 있다.  참고 사이트: <https://nolboo.kim/blog/2013/10/06/github-for-beginner/> | | | |



|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **맑은 유림조** | **구성원** | **김영훈 김유림 지경민 이수경 석주연** |
| **주제** | **Azure(Cloud)** | | |
| **주요 내용** | | | |
| 1. 개요 :  Microsoft의 애플리케이션 개발, 테스트, 배포를 통합하여 관리할 수 있는  클라우드 컴퓨팅 플랫폼 및 인프라 스트럭처 서비스    2. 기능 :    2.1 컴퓨팅  2.2 네트워킹  2.3 스토리지  2.4 데이터베이스  2.5 웹 및 모바일  2.6 사물 인터넷, 빅 데이터, AI  2.1 컴퓨팅 :  Azure에서 호스팅되는 windows, linux, sql server 등의 가상 머신을 제공    2.2 네트워킹 :  VPN(가상 사설망에)에 VM을 연결, DDoS 보호, 트래픽 분산, 부하 분산  2.3 스토리지 :  스키마와 관계없이 비정형 데이터를 호스팅하는 Azure Table 스토리지  비디오나 비트맵 같은 대용량 개체를 위한 스토리지 서비스,  파일 서버처럼 액세스하고 관리할 수 있음,    2.4 DB :  NoSQL 옵션을 지원하는 Azure Cosmos DB, SQL DB,  SQL Server MariaDB등을 바로 사용가능  2.5 웹 및 모바일 :  .NET, Node.js, PHP, Python, Java 등으로 개발된 웹 앱을 지원  iOS, Android, Windows, macOS에서 돌아가는 앱을 개발할 수 있다.  오프라인 데이터 동기화를 통해 네트워크 문제가 있을 때에도 문제없이 앱이 작동할 수 있다.  자체 데이터센터와 연결해 전세계에서 자체 데이터 센터의 데이터에 쉽게 접근할 수 있다.  2.6 사물 인터넷, 빅 데이터, AI  대규모 IoT에 연결 및 모니터링  Hadoop 클러스터를 사용하여 대량의 데이터 처리  기계 학습 모델의 개발, 교육, 테스트, 배포, 관리를 할 수 있는 환경 제공  참고 사이트:  <https://docs.microsoft.com/ko-kr/learn/modules/welcome-to-azure/3-tour-of-azure-services> | | | |



|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **맑은 유림조** | **구성원** | **김영훈 김유림 지경민 이수경 석주연** |
| **주제** | **Jenkins(CI/CD)** | | |
| **주요 내용** | | | |
| * Jenkins :   소프트웨어를 작성, 테스트 및 배포하거나 배포하는 것과 관련된 모든 작업을 자동화하는 데 사용할 수있는 독립적인 오픈 소스 자동화 서버  소프트웨어 개발 시 지속적 통합(continuous integration) 서비스를 제공하는 툴  다수의 개발자들이 하나의 프로그램을 개발할 때 버전 충돌을 방지하기 위해 각자 작업한 내용을 공유 영역에 있는 저장소에 빈번히 업로드함으로써 지속적 통합이 가능  Blue Ocean을 서프 프로젝트로 구축하면서 CD(continuous deployment)툴로도 사용     * Jenkins Pipeline :   Jenkins에 지속 적인 전송 파이프라인을 구현하고 통합하는 것을 지원하는 플러그인 모음   * Jenkins 공식 사이트 : <https://jenkins.io/> → 무료로 설치가능 * Jenkins 사용 이유 : * 프로젝트의 빌드가 정상적으로 되고 있는지 체크할 때. * 자동으로 유닛 테스트(Unit Test)와 통합 테스트(Integration Test)의 정기적인 실행과 결과에 문제가 있을 경우에 이메일을 통해서 결과확인. * 코드의 질을 조절. * 젠킨스가 주는 이점   개발중인 프로젝트에서 커밋은 매우 빈번히 일어나기 때문에 커밋 횟수만큼 빌드를 실행하는 것이 아니라 작업이 큐잉되어 자신이 실행될 차례를 기다리게 된다  코드의 변경과 함께 이뤄지는 이 같은 자동화된 빌드와 테스트 작업들은 다음과 같은 이점들을 가져다 준다.   * 프로젝트 표준 컴파일 환경에서의 컴파일 오류 검출 * 자동화 테스트 수행 * 정적 코드 분석에 의한 코딩 규약 준수여부 체크 * 프로파일링 툴을 이용한 소스 변경에 따른 성능 변화 감시 * 결합 테스트 환경에 대한 배포작업   이 외에도 플러그인을 온라인으로 간단히 인스톨 할 수 있는 기능을 제공   * Jenkins가 제공하는 기능   •웹 인터페이스를 통한 간편한 설정  •강력하고 편리한 Reporting 기능  •지속적인 자동화 빌드  •지속적인 자동화 테스트  •커버리지 검사  •코드 품질 검사  •다양한 인증기반과 결합한 인증 및 권한 관리 기능  •Groovy script를 이용한 고수준의 Job Scheduling 기능  •커맨드라인 인터페이스 제공  •자동화된 배포 관리  •분산빌드 기능  •윈도우 커맨드 스케쥴링 실행기능  이 외에도 간단하게 플러그인 설치를 통해 추가/확장 가능   * Jenkins를 효율적으로 사용하기 위해서는..   빌드 자동화 : 빌드 자동화 스크립트를 만들어 놓음으로써 Jenkins에게 빌드를 명령하여 지속적으로 빌드할 수 있도록 한다.  테스트 자동화 : 단위 테스트, UI 테스트 등 테스트를 자동화해놓음으로써 개인이 미처 발견하지 못한 오류를 검출하고 개인에게 알림이 갈 수 있도록 한다.  코드 표준 준수 여부 검사 : 정적 코드 분석을 통하여 코드 품질을 높일 수 있도록 한다. Build Pipeline 구성 : 2개 이상의 모듈로 구성되는 프로젝트의 경우 각 모듈의 참조관계에 따라서 순차적으로 빌드할 수 있도록 한다.   * Jenkins 설치 참고 블로그 :   https://roxxy.tistory.com/entry/Jenkins-%EC%A0%A0%ED%82%A8%EC%8A%A4-%EC%84%A4%EC%B9%98-%EB%B0%8F-%EC%84%A4%EC%A0%95 | | | |