

|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **Give Me More** | **구성원** | **김동욱, 유지원, 최홍용** |
| **주제** | **JIRA** | | |
| **주요 내용** | | | |
| **지라(JIRA)란?**  - 아틀라시안이 개발한 사유 이슈 추적 제품. 버그 추적, 이슈 추적, 프로젝트 관리 기능을 제공하는 소프트웨어. ‘고지라’ 라는 일본식 이름에서 따옴.  **아틀라시안(Atlassian)**  - 호주의 시드니에 기반을 두고 있으며 소프트웨어 개발자를 목표로 하는 비즈니스 엔터프라이즈 소프트웨어를 만드는 소프트웨어 기업.  **이슈 추적 시스템(Issue tracking system, ITS)**  - 단체의 필요에 의해 이슈 목록을 관리하고 유지 보수하는 컴퓨터 소프트웨어의 하나. 트러블 티켓 시스템(Trouble Ticket System)라고도 함.  - 프로젝트에서 예상되거나 또는 이미 발생한 이슈를 관리하는 것.  - 관리 대상 이슈로는 업무(작업), 문제점, 개선 사항 등이 있음.  - 이슈 관리 방법: 목록으로 정리, 우선 순위 부여, 담당자 지정, 진행 과정을 추적.  - 이슈 추적 시스템의 티켓(ticket)은 특정 문제, 상태, 기타 관련 데이터에 대해 실행 중인 보고임. 헬프데스크나 콜센터 환경에서 만들어지며 유일한 참조 번호를 가지고 있는데, 이를 케이스(case), 콜 로그(call log)번호라 부름. 직원이나 사용자가 사용자의 이슈나 요청의 상태를 빠르게 위치시키고, 추가하고 주고받을 수 있게 함.  **버그 추적 시스템(Bug Tracking System)**  - 소프트웨어 개발 프로젝트에서 알려진 소프트웨어 버그를 추적하는 응용 소프트웨어.  - 이슈 추적 시스템의 일종.  - 오픈 소스 소프트웨어 프로젝트가 사용하는 것과 같은 버그 추적 시스템을 통해 최종 사용자가 버그 보고서를 직접 입력할 수 있음. 다른 시스템은 소프트웨어 개발을 수행하는 기업이나 단체 내에서 내부적으로만 사용.  - 일반적으로 버그 추적 시스템은 다른 소프트웨어 프로젝트 관리 응용프로그램에 통합.  - 버그 추적 시스템은 소프트웨어 개발 인프라의 필수적인 요소.  **프로젝트 관리(Project Management)**  - 프로젝트의 성공적인 완성을 목표로 움직이는 활동을 말함.  - 프로젝트를 구성하는 각각의 활동계획입안, 일정표 작성 및 진척 관리를 포함. | | | |



|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **Give Me More** | **구성원** | **김동욱, 유지원, 최홍용** |
| **주제** | **GIT** | | |
| **주요 내용** | | | |
| **GIT이란?**  소스코드를 효과적으로 관리하기 위해 개발된 분산형 버전 관리 시스템.  **원격 저장소(Remote Repository)**  - 파일이 원격 저장소 전용 서버에서 관리되며 여러 사람이 함께 공유하기 위한 저장소.  **로컬 저장소(Local Repository)**  - 내 PC에 파일이 저장되는 개인 전용 저장소.  **브랜치(Branch)**  - 독립적으로 어떤 작업을 진행하기 위한 개념. 필요에 의해 만들어지는 각각의 브랜치는 다른 브랜치의 영향을 받지 않기 때문에, 여러 작업을 동시에 진행할 수 있음.  **기본 명령어**  - git –version : 현재 git의 버전을 확인 할 수 있음.  - git init: git 저장소를 생성.  - git status: 커밋되지 않은 변경사항을 조회.  - git add 파일명: git add는 로컬디렉토리에서 작업한 파일을 스테이징 영역에 올려 놓음.  - git commit –m “커밋메세지”: git add를 통해서 스테이징에 올라간 파일들을 커밋함.  - git diff: 작업트리와 스테이징 영역에 대한 차이점을 보여줌.  - git mv 파일명 새파일명: 기존에 존재하던 파일명을 새파일명로 변경.  - git checkout –파일명: 아직 커밋 전인 파일에 대해 변경내용을 취소하고 이전 커밋을 되돌림.  **원격저장소 관련 명령어**  - git clone 원격저장소주소: 원격저장소 주소에 있는 내용을 복제하여 현재 폴더에 저장.  - git pull: 원격저장소의 변경사항을 가져와서 로컬저장소에 있는 내용과 합침  - git push: 커밋한 파일들을 원격서버에 올림.  - git remote add 이름 저장소주소: 새로운 원격 저장소를 추가.  - git remote: 추가한 원격저장소 리스트를 보여줌.  - git remote rm 이름: 해당 이름의 원격저장소를 제거. | | | |



|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **Give Me More** | **구성원** | **김동욱, 유지원, 최홍용** |
| **주제** | **Azure(Cloud)** | | |
| **주요 내용** | | | |
| **Azure(애저)란?**  마이크로소프트가 관리하는 글로벌 데이터센터와 네트워크를 활용하여 응용 프로그램을 빠르게 개발, 배포 및 관리할 수 있게 하는 서비스 집합  **Azure로 어떤 것을 할 수 있을까?**   * 가상머신에서 기존 어플리케이션을 실행하는 것부터 지능형 봇 및 합성 현실과 같은 새 소프트웨어 패러다임을 탐색하는 것까지 모든 것을 수행할 수 있는 100개 이상의 서비스를 제공한다. * Azure에서 실행되는 가상 머신으로 기존 어플리케이션을 이동하여 클라우드 탐색을 한다. 가상 머신으로 기존 앱을 마이그레이션하는 것 등을 통해서. * 단순 마이그레이션을 넘어, 시/청각 및 음성을 통해 사용자와 자연스럽게 대화할 수 있는 AI및 기계 학습 서비스를 제공한다. 또한 대량의 데이터를 수용할 수 있도록 동적으로 확장되는 스토리지 솔루션도 제공한다.   **Azure에서 소개하는 여러 기능들**    **Azure 시작** : <https://docs.microsoft.com/ko-kr/azure/>  **Azure 소개페이지** : <https://docs.microsoft.com/ko-kr/learn/modules/welcome-to-azure/> (배울 수 있다.)  **Cloud 서비스 비교**   * 현존하는 클라우드 서비스는 AWS, Azure, GCP 등이다. 이 3사를 비교해본다.      |  |  |  | | --- | --- | --- | | vendor | 강 | 약 | | AWS | 마켓 포지션 큼  광범위하고 전문적인 제품  대규모 조직 지원  많은 교육 프로그램 | 쓰기 어렵다  비용 관리 어렵다  옵션이 너무 많다 | | MS Azure | 두번째로 큰 마켓셰어  MS도구 및 SW와의 통합  광범위한 기능 세트  하이브리드 클라우드  오픈소스 지원 | 문서 관련 문제  불완전한 management tooling | | GCP | 클라우드 기반 비즈니스를 위해 설계됨  오픈소스 지원  큰 할인율과 유동적인 계약금  DevOps expertise | IaaS시장에 늦은 진출  적은 기능과 서비스  역사적으로 기업중심이 아님 | | | | |



|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **Give Me More** | **구성원** | **김동욱, 유지원, 최홍용** |
| **주제** | **Jenkins(CI/CD)** | | |
| **주요 내용** | | | |
| **CI/CD란?**  - 애플리케이션 개발 단계를 자동화하여 애플리케이션을 보다 짧은 주기로 고객에게 제공하는 방법.  - CI/CD의 기본 개념은 지속적인 통합, 지속적인 서비스 제고, 배포.  - Jenkins와 같은 CI툴이 등장하기 전에는 일정시간마다 빌드를 실행하는 방식이 일반적. 특히 개발자들이 당일 작성한 소스들의 커밋이 모두 끝난 심야 시간대에 이러한 빌드가 타이머에 의해 집중적으로 진행되었음. Jenkins는 Git와 같은 버전관리 툴과 연동하여 소스의 커밋을 감지하면 자동적으로 자동화 테스트가 포함된 빌드가 작동되도록 설정 할 수 있음. 개발 도중의 커밋은 빈번하게 일어나기 때문에 커밋 횟수만큼 빌드를 실행하는 것이 아니라 큐에 들어가 자신이 실행될 차례를 기다림. 코드의 변경과 함께 이뤄지는 자동화 빌드와 테스트 작업들은 몇 가지 이점이 있음.  - **이점**   1. 프로젝트 표준 컴파일 환경에서의 컴파일 오류 검출. 2. 자동화 테스트 수행. 3. 정적 코드 분석에 의한 코딩 규약 준수여부 체크. 4. 프로파일링 툴을 이용한 소스 변경에 따른 성능변화 감시. 5. 결합 테스트 환경에 대한 배포작업.   - 이 외에도 Jenkins는 1400가지가 넘는 플러그인을 온라인으로 간단히 인스톨 할 수 있는 기능을 제공하고 있으며 파이썬과 같은 스크립트를 이용해 손쉽게 자신에게 필요한 기능을 추가 할 수 있음. | | | |