학습 정리

|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **5678** | **구성원** | **김호영, 박수현, 윤성철, 배민수** |
| **주제** | **JIRA** | | |
| **주요 내용** | | | |
| **JIRA**  지라(JIRA)는 아틀라시안이 개발한 사유 이슈 추적 제품으로 버그추적, 이슈추적, 프로젝트 관리 기능을 제공하여 애자일팀에 유용한 소프트웨어이다.  **Issue**  **정의**   * 오류, 버그 및 새로운 기능, 작업요청, 사소한 질문이나 의견 등 회사에 대상이 되는 거의 모든 것 * 이슈의 종류의 따라 개발자의 구현이 필요 없는 경우도 존재함   **종류**   * User Story   + 사용자의 요구 사항이나 개발의 대상이 되는 기능 * Task   + User Story의 하위 작업   + User Story를 위해 개발자가 실제로 작업해야하는 각각의 작업단위 * Bugs   + 제품이 설계대로 동작하지 않는 문제점 * Enhancenment Request   + 기능 개선 요청으로 기능 추가 작업   **Issue 단위 작업 절차**   1. PM이 요구 사항을 취합하여 User Story 를 작성한다. 2. User Story를 구현하기 위해 실제 Task들을 해당 User Story 아래 생성한다. 3. 생성된 Task 들을 개발자에게 지정(Assign)한다. 4. Assign 되지 않은 작업에 대해 개발자가 스스로 작업을 가지고 가서 작업을 진행한다.   **Issue Tracking 시스템**   * 프로젝트에서 예상되거나 이미 발생한 이슈들의 관리하는 시스템 * 장점   1. 특정 이슈를 누가 발견했는지, 누가 해결해야하는지, 이슈의 현재 상태 파악하고 한눈에 해결 및 관리 가능   2. 한눈에 팀원들의 작업현황 확인 가능   3. 스케쥴, 우선순위 조절   4. 이슈에 대한 역활과 임무를 분명하게 할 수 있음   5. 불필요한 커뮤니케이션 비용 절약   6. 비슷한 이슈가 발생했을 때 처리 과정 되짚어 볼 수 있음 * 이슈관리 방법   1. 목록으로 정리   2. 우선순위 부여   3. 담당자 지정   4. 진행 과정 추적 * 유사 솔루션   1. Redmine   2. Mantis   3. Trac   4. Bugzila   **Project 관리 도구**   * 프로젝트 진행에 필요한 기간, 리소스 등에 대한 계획 수립,이슈들의 진행상태 추적, 통합된 리포트 및 대시 보드 제공 * 프로젝트 참여자의 업무 과정(태스크의 할당, 작업 로그 등)을 기록 * 프로젝트 전반에 걸친 모니터링 및 플래닝 도구 제공 * 개발 과정이나 업무 처리에 필요한 다양한 도구와 연동   **JIRA Project**   * 실제 조직의 프로젝트와 mapping * 프로젝트 단위로 이슈들을 관리 * 멤버 관리 * 이슈 유형 관리 * 워크플로우 관리   + 정의     - 생성에서 완료까지 문제가 발생하는 순차적 경로를 나타냄   + 효과     - 진행중인 작업의 상태를 명확하게 알려 줌       * 작업 상태를 알기 위한 커뮤니케이션 불필요     - 전체프로젝트의 상태 파악 및 리소스 계획 수립 가능     - 새로운 팀원을 포함한 전체 팀원들 간의 협업   + 기본 제공 workflow     - open : 이슈가 최초 생성되어 아직 작업에 들어가기 전 상태     - In Progress : 담당자가 정해지고 이슈에 대한 작업이 진행 중인 상태     - Resolved : 담당자가 작업을 완료한 상태     - Closed : 이슈를 요청했던 요청자가 담당자의 작업 결과에 만족하여 이슈의 완료에 동의한 상태     - Reopened : 완료된 이슈에 문제가 발견되어 다시 작업을 진행하는 상태   **Agile 보드**   1. **Kanban 보드**    * 신호, 지시카드를 나타내는 일본어    * 지속적으로 이루어지는 유지보수 업무에 적합 2. **Scrum 보드**    * 전체 프로젝트 기간을 Sprint라는 정해진 기간으로 분리    * 프로젝트 전체적/으로 해/야할 작업내용을 백로그에 저장 및 생성    * Sprint 별로 할 수 있는 분량만큼의 작업(백로그)를 선택해서 Sprint에 넣고 작업    * Sprint 기간 동안 처리 불가능한 작업들을 리뷰를 거쳐 다음 Sprint로 이관    * 백로그를 관리하고 Sprint 기간의 통제, 리뷰를 주도할 스크럼 마스터 역할의 필요 | | | |

학습 정리

|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **5678** | **구성원** | **김호영, 박수현, 윤성철, 배민수** |
| **주제** | **GIT** | | |
| **주요 내용** | | | |
| **GIT**  소스코드의 버전 관리와 이력 관리가 가능한 분산형버전관리시스템(DVCS - Distributed Version Control System)  **장점**  소스 코드가 변경된 이력을 쉽게 확인 가능  특정 시점에 저장된 버전과 비교하거나 특정시점으로 되돌아갈 수 있음  다른 사람과 동시에 편집 시 경고가 발생하기에 merge를 쉽게할 수 없음   1. **repository(저장소)**   파일이나 폴더를 저장해두는 장소로 하나의 파일이 변경 이력 별로 구분되어 저장됨    기존에 저장소에 저장된 파일에 변경사항이 발생하면 다른 파일로 인식하기 때문에 파일을 변경사항 별로 구분하여 저장 가능  원격 저장소와 로컬 저장소  원격저장소(Remote repository): 여러 사람과 공유하기 위한 저장소로, 파일이 원격 저장소 전용 서버에서 관리됨  로컬 저장소(Local Repository): PC의 개인 전용 저장소로, 작업한 내용을 원격 저장소에 업로드   1. **변경 기록** 2. Git add .   현재 working directory 에서 commit할 목록에 담아놓는 것 ($ git add .)  로컬 디렉토리에서 작업한 파일을 스테이징 영역에 올려놓는 것으로, 파일 선언하는 단계  // git status <– git의 현재 상태 확인 명령어     1. Git commit   현재 소스 코드 상태를 스냅샷 찍는 것과 동일한 느낌으로, staging area에 담겨 있는 내용을 이력으로 기록  변경된 파일(staging 된 파일)을 로컬 저장소에만 반영하는 것(! 원격 서버에는 올라가지 않았음)  ($ git commit –m “ 커밋 메시지”)   1. Git push   원격 저장소로 보내기 ($ git push –u 000(저장소 url 이름) master)     1. **작업트리와 인덱스**   index스: 커밋을 실행하기 전, 저장소와 작업 트리 사이에 존재하는 공간  Commit은 작접 트리에 있는 변경내용을 저장소에 바로 기록하는 것이 아니라, 사이 공간인 index에 파일 상태를 staging 하는 것   1. **Diff**  * 리눅수에서 diff 명령어는 두 파일의 콘텐츠를 한줄씩 비교하여 서로 차이가 나는 부분을 요약해서 보여줌 * git에서도 diff명령이 존재 * git에서의 diff는 세가지 기본 형태의 트리객체나 그와 유사한 객체를 활용함   + 전체 커밋 그래프 내에 있는 임의의 트리 객체   + 사용자의 작업디렉토리   + 인덱스 * git diff   $ git diff  작업디렉토리와 인덱스 간의 차이점을 보여준다. 작업디렉토리에서 변경된사항으로, 다음커밋동안 준비된 상태(onstage)가 될 후보 항복들을 보여주는 것.  $ git diff HEAD  작업 디렉토리와 HEAD를 비교해준다.  $ git diff --cached  인덱스와 HEAD를 비교해준다.  $ git diff HEAD^ HEAD  이전 헤드와 현재 헤드의 비교  $ git diff --cached [ commit ]  onstage상황과 지정된 커밋간의 차이점  $ git diff [ commit 1 ] [ commit 2 ]  두 커밋과의 차이점   1. **Bisect**   git을 활용하다가, 프로그램이 뭔가 결함이 있어 작동하지 않는다고 가정하자. 이전의 작동하는 나쁜버전이 아니라, 좋은 버전을 찾아야 할 때, 해당 명령어로 확인 가능  bisect는 한쪽 끝에서는 좋은 동작을 기준으로, 한쪽 끝에서는 나쁜 동작을 기준으로 범위를 좁혀가면서 새 커밋을 선택하게 되는 과정이고, 결과적으로 잘못된 동작이 도입된 커밋에 도달 가능 한마디로, 이분검색을 통해 디버깅해가는 과정을 의미함  $ git bisect start  이분 검색을 위한 새로운 브랜치가 생성되는 순간  $ git bisect bad  현재 헤드(HEAD)에 문제가 있음을 알려주거나, bad명령어 뒤에 커밋id를 남겨서 bad 커밋이 무엇인지 확인 가능  $ git bisect good [ commit\_ID ]  오류가 없는 커밋을 알려줌  git은 자동으로 사용자의 디렉토리를 good커밋과 bad 커밋사이의 특정 리비전으로 바꿔주기에, 사용자는 해당 버전이 좋은 버전인지 나쁜 버전인지 테스트 한 다음, git에게 결과를 알려주면 됨  $ git bisect [good/bad]  이분검색을 최대로 실행할 경우 단 하나의 커밋에 도달할때까지 검색이 반복되며, 하나의 good을 찾는 경우 이분검색이 끝났음을 알리는 커밋메시지가 뜬다.  Good, bad 선택 시 잘못되었거나, 다시 시작하고 싶으면 해당 명령어로 한단계 돌아갈 수 있음  $ git bisect reset  $ git bisect replay   1. **Reset** 2. Soft : 지정된 commit을 가리키도록 head 참조 변경하는 명령어, 인덱스와 작업중인 컨텐츠 그대로 유지됨 ($git reset –soft [commit]) 3. Mixed: 지정된 commit을 가리키도록 head를 변경하는 명령어. 트리구조에 맞게 인덱스의 컨텐츠가 변경됨 ($ git reset –mixed [commit]) 4. Hard: 지정된 commit을 가리키도록 head를 변경하는 명령어, 인덱트 컨텐츠, 작업 디렉토리 컨텐츠 모두 변경됨 | | | |

학습 정리

|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **5678** | **구성원** | **김호영, 박수현, 윤성철, 배민수** |
| **주제** | **Azure(Cloud)** | | |
| **주요 내용** | | | |
| **Microsoft Azure**  [Azure란?]  Windows Azure는 전 세계의 Microsoft 데이터 센터에서 응용 프로그램을 빌드하고, 배포하고, 관리할 수 있는 유연한 개방형 클라우드 플랫폼이다. 사용자는 모든 언어, 도구 또는 프레임워크를 사용하여 응용 프로그램을 빌드 할 수 있다. 또한 공용 클라우드 응용 프로그램을 기존의 IT 환경과 통합할 수 있다.  Azure는 기본적으로 Paas 형의 클라우드다. Iaas 형태의 클라우드가 OS까지만 제공한다면, 애플리케이션을 수행 할 수 있는 Runtime과 각종 Middleware (Web Application Server, DBMS, Service Bus etc)등을 포함해서 서비스로 제공한다.Iaas의 경우에는 이렇게 사전 제공되는 형태의 플랫폼이 없기 때문에 고객 마음대로 플랫폼을 설치 구성할 수 있기 때문에 요구사항에 대한 유연성이 매우 높지만, 반대로 플랫폼에 대한 구입,설치 및 설정과 유지보수에 대해서는 모두 고객이 책임져야 하기 때문에 그만큼 비용과 관리 인력,시간이 소요된다. Azure는 Paas형태의 클라우드 서비스로 .NET 기반의 플랫폼과 Microsoft 기반의 미들웨어는 물론이고, PHP,JAVA와 같은 다른 기술을 지원할 수 있는 플랫폼 서비스를 제공한다.  [Azure의 장점]   1. 시스템 사용시 중단이 없다.   99.95%의 월별 SLA를 제공하므로 인프라에 신경 쓰지 않고 고가용성 응용 프로그램을 빌드하고 실행할 수 있다. 또한 OS 및 서비스 자동 패치, 네트워크 부하 분산, 하드웨어 실패 복구 기능을 제공합니다. 가동 중단 시간 없이 응용 프로그램을 업그레이드 할 수 있는 배포 모델을 지원한다.   1. 확장의 개방성   개발자 또는 IT 전문가는 Azure로 생산성을 높일 수 있다. 통합 도구, 미리 작성된 템플릿 및 관리 서비스를 사용하는 경우 자신이 이미 보유하거나 알고 있는 기술을 활용하여 엔터프라이즈, 모바일, 웹 및 IoT 앱을 더 쉽고 빠르게 빌드하고 관리활 수 있다.   1. 사용 요금   Azure의 종량제 서비스는 수요에 맞춰 신속하게 확장하거나 축소할 수 있으므로 사용한 만큼만 지불하면 된다. 계산, 저장소, 대역폭 등 자주 사용하는 인프라 서비스에 대해 경쟁업체 가격과 동일하도록 보장하고 분당 청구 기능도 제공하므로 항상 성능 대비 가장 저렴한 요금이 청구된다.   1. 보안   대부분의 고객은 클라우드 환경이라는 부분에 대해서 좋은 인식만을 가지고 있지는 않다.  그렇기에 데이터 보호 및 개인 정보 보호에 대해 선도적인 노력을 기울이고 있으며, Microsoft는 유럽 연합 데이터 보호 당국으로부터 엄격한 EU 개인 정보 보호법을 제대로 준수하고 있다고 인정받은 최초의 클라우드 공급자다. 뿐만 아니라, 새로운 국제 클라우드 개인 정보 보호 표준인 ISO 27018을 채택한 최초의 주요 클라우드 공급자이기도하다.   1. 어디서나 실행가능한 앱   Microsoft에서 최고의 클라우드 서비스로, 아마존 웹 서비스와 Google 클라우드보다 22 지역-더 많은 국가와 지역에 걸쳐 마이크로 소프트 관리 데이터 센터의 전세계 네트워크에서 실행된다. 이 빠른 속도로 성장하는 글로벌 시대에 응용 프로그램을 실행하고 훌륭한 고객 성능을 보장하기 위한 많은 옵션을 제공한다. 또한 중국 본토에서 처음으로 다국적 클라우드 제공 업체다.  [기능]   1. Portal 기능   웹서비스로 다양하고 직관적인 UI를 제공한다. 데시보드나, 사용량이나 이력을 확인하거나 현재 서비스중인 클라우드들을 모두 모아서 볼 수도 있다.  1> CLI  흔히 말하는 커맨드라인 인터페이스이며 복수의 서비스를 명령어기반으로 관리하고자 할때사용한다. 클라우드 서비스도 제공하지만 VM을 하나 생성해야하는것이 흠 이다. REST API를 제공하고 있고 서비스 생성은 템플릿쪽에서 해결하는편이 합리적이다. 서비스삭제는 직관적인 포털이 낫다.사실 CLI는 REST API를 기반으로 작동한다. 이는 웹쿼리기반 자동제어에서도 사용 할 수 있다는 얘기가 된다. 파워셸로도 제공하고 있다. CLI 만큼 직관적이지 않다.   1. Template 기능   자동화를 위한 템플릿이다. 애저의 서비스 게시를 시작할때 이 템플릿을 기반으로 만든다. 복수의 리소스와 리소스 연관성을 지정하여 일렬의 자동화도 가능하다. 예를들면 100개의 서비스를 직접 생성하면 포털 탐색에 질려버릴테지만 템플릿으로 가서 템플릿을 저장한다음 배포하면 모든 게시를 한번에 할 수 있다.   1. Compute 기능   애플리케이션을 새로 구축할 때나 기존 애플리케이션을 배포할 때, Azure는 앱을 실행하는 데 필요한 인프라를 제공한다. Compute 기능에서 제공하는 것들을 살펴보자.  1> Virtual Machines  가상 컴퓨터를 Azure에 배포하는 핵심 기능으로, Windows, Linux 및 Windows 가상머신 위에 돌아가는 SQL Server, BizTalk Server, SharePoint(물론 가격은 따로 책정해 판다), Oracle 소프트웨어가 깔린 Windows Server 가상머신(윈도우 라이선스, 오라클 라이선스 가격이 별도로 청구된다)을 제공한다. AWS의 EC2와 유사하다. 리소스 관리자 템플릿을 통해 Azure 포털에서 미리 배포 계획을 짠 후 원클릭으로 여러 응용프로그램이 설치된 가상머신을 Azure에 배포할 수 있다. 마이크로소프트에서 제공하는 클라우드이므로 당연히 SharePoint, Dynamics, System Center 같은 퍼스트 파티 윈도우 서버 프로그램과 통합이 잘 되어 있다. 또한 디스크 암호화와 여러 보안 업체를 통한 보안 규제 및 규정 요구 사항을 검증받고 있다. 마이크로소프트가 윈도우 서버만 제공할 것 같지만 Oracle 소프트웨어와 함께 우분투, openSUSE 등 리눅스 배포판과 Chef, Puppet, Docker 솔루션도 제공한다. SQL Server 이미지를 통해 SQL Server가 깔린 윈도우 가상머신을 쉽게 여러 Azure 지역에 배포할 수 있다. VPN을 통해 DNS IP 주소를 정의해 네트워크의 모든 요소를 제어, 구성하고, 부하 분산과 통신사 파트너를 통한 데이터 센터 직접 연결 기능도 있다. 여러 종류의 가상머신을 통해 원하는 기능에 특화된 가상머신(예: 저장소 특화, HPC 기능 특화, 그냥 최신 가상 머신 등)을 제공받을 수 있다. 또, 백업 기능으로 가상머신을 보호할 수 있다.  2> Batch  자체 슈퍼컴퓨터가 없는 곳을 위한 기능으로, 워크스테이션과 클러스터에서 쓰는 응용프로그램을 클라우드에서 더 많은 자원으로 빠르게 돌리는 기능이다. 스케줄러를 통해 작업을 예약한다. 또한 자동으로 필요 프로그램을 설치하고, 실행하고, 실패한 곳이 있으면 큐를 조정하고, 완료하면 축소한다. 안써도 전기가 나가는 기존의 슈퍼컴퓨터와 달리 쓴 만큼 요금을 지불하기 때문에 더 저렴할 수도 있다. 또한, 작업을 미리 정의한 일정이 아니라 주문형으로 바로 처리할 수도 있다. 배치를 통해 클라우드에서 렌더링 작업을 실행할 수도 있다.  3> Functions  서버리스 아키텍처를 제공하는 서비스이다. (AWS Lambda와 유사하다.)   1. 웹 및 모바일 기능   1> App Service  -Web Apps : .NET, Node.js, PHP, Python, Java 등으로 개발된 웹 앱을 지원한다. Visual Studio나 IntelliJ IDEA 등의 IDE에 플러그인으로 제공되는 Azure SDK를 이용하여 개발이 가능하다. Office 365, Salesforce.com과 같은 SaaS 서비스에 쉽게 연결할 수 있고, Visual Studio Team Services, GitHub 등과 연동해 개발과 배포를 쉽게 할 수 있다. Azure Active Directory와 통합해 업무용 웹을 지원하고, 여러 지역의 데이터센터에 배포해 전세계에 안정적으로 서비스를 제공할 수 있다. 또한 인프라에 신경쓰지 않고 개발→테스트→배포 프로세스를 진행할 수 있다.  -Mobile Apps : 모바일 앱 백엔드 기능으로, 백엔드 논리 외에도 데이터 저장, 사용자 인증, 푸시 알림 등을 보낼 수 있다. Active Directory를 통해 사용자를 인증하고, SAP, Oracle, SQL Server, SharePoint에 접근하고, Xamarin, PhoneGap 등 크로스플랫폼 프레임워크를 통해 iOS, Android, Windows, macOS에서 돌아가는 기업용 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. 또한, 밑의 알림 허브를 통해 수백만 건의 푸시 알림을 전송할 수 있다. 역시 자동 확장 기능을 제공하므로 트래픽이 몰려도 자동으로 서버가 확장되어 서비스가 중단되는 일이 적어진다.  -API Management : 대규모로 백엔드 API를 게시하고 관리할 때 쓰이는 기능이다. API 설명서와 대화식 콘솔을 제공하고, API 속도 제한 및 할당량을 설정할 수 있다. 또, API 상태를 지속적으로 모니터링 해 오류를 신속하게 잡아낼 수 있다. JSON, REST 형식도 지원하고, 백엔드 여러 부분의 API를 모아서 보여줘 API 셋에 대해 넓은 그림으로 볼 수 있다. SmartBear 사의 Ready! 플러그인을 통해 API 전체를 빠르게 테스트하고 생성할 수 있다. API를 통한 인증도 지원해 보안도 꾀할 수 있다. API가 어떻게 활용되는지 분석하고, 성능 오류 비율 등을 알 수 있다. API 검색 기능을 통해 API 수천개의 API를 쉽고 빠르게 찾아볼 수 있다.  -알림 허브  -모바일 고객 관리   1. 데이터 저장소 기능   1> CosmosDB  2> SQL 데이터베이스  3> Data Lake Storage  4> DocumentDB  5> Azure Redis Cache  6> 저장소  7> StorSimple  8> 검색   1. 분석 기능   1> HDInsight  2> 기계 학습  3> 스트림 분석  4> 데이터 팩터리   1. IoT   1> Azure IoT Suite  2> 이벤트 허브   1. 네트워킹   1> 가상 네트워크  2> Express 경로  3> 트래픽 관리자  4> 부하 분산 장치  5> VPN 게이트웨이  6> 응용 프로그램 게이트웨이   1. 미디어 및 CDN   9-1> 미디어 서비스  -인코딩  -Azure Media Player  -미디어 인덱서  -콘텐츠 보호  -라이브 및 주문형 스트리밍  9-2> CDN   1. 하이브리드 통합   1> BizTalk 서비스  2> Service Bus  3> Azure 백업  4> 사이트 복구   1. ID 및 액세스 관리   1> Azure Active Directory  2> Azure 다단계 인증   1. 개발자 서비스   1> Azure DevOps  GitHub와 비슷하게 프로젝트 관리를 해주는 서비스. Microsoft Azure 플랫폼을 이용한 개발을 할 수 있게 해 준다. 서브스크립션 레벨에 따라 제공하는 기능이 달라지며, 무료 버전도 5유저까지 지원한다. 무료 계정의 경우에도 온라인 방식으로 제공되는 Team Foundation Server를 쓸 수 있고 (클라우드식 TFS라 봐도 무방) 무료 Visual Studio Community에 그대로 연동해서 쓸 수 있기 때문에, 취미 목적의 소규모 개발에는 사실상 걸림돌이 없어진 셈. 이 링크를 통해 사용해볼 수 있다. 단순히 TFS를 클라우드로 제공하는 것을 넘어서 클라우드 기반의 빌드, 테스팅, 실시간 서비스 모니터링 기능, 웹기반 소스코드 에디팅 등을 제공한다. 비주얼 스튜디오 최신 버전의 동향을 그대로 따르기에 Git 등의 다른 서비스 연동기능과 크로스 플랫폼 지원도 점점 강화되고 있다. 팀 서비스의 에디터(코드네임 '모나코')는 아래의 Visual Studio Code의 기반이 되었다.   1. 관리   1> Azure 포털  2> 공정 관리자  3> 자동화  4> Operational Insights  5> 주요 자격 증명 모음   1. 종료된 서비스   1> Mysql : 무료로 20MB도 제공했으나 순차적으로 사라졌다.  2> RemoteApp : 클라우드를 통해 윈도우 앱을 제공하는 기능으로, 여러 기업이나 PC방에서 적용하고 있는 VDI 기능의 공용 클라우드 버전이다. 사내 서버와 달리 하드웨어를 업그레이드하거나 관리하는데 시간과 돈을 들이지 않아도 된다. 또, 앱을 다시 만들 필요 없이 바로 배포할 수 있다. 앱과 데이터가 클라우드에서 유지되기 때문에 장치를 분실, 도난당해도 정보가 손실될 위험이 줄어든다. 또, 윈도우 말고도 OS X, iOS, Android 등 다양한 장치에 응용프로그램을 제공할 수 있다. | | | |

학습 정리

|  |  |  |  |
| --- | --- | --- | --- |
| **팀** | **5678** | **구성원** | **김호영, 박수현, 윤성철, 배민수** |
| **주제** | **Jenkins(CI/CD)** | | |
| **주요 내용** | | | |
| **젠킨스**(Jenkins)는 소프트웨어 개발 시 [지속적 통합](https://ko.wikipedia.org/wiki/%EC%A7%80%EC%86%8D%EC%A0%81_%ED%86%B5%ED%95%A9)(continuous integration) 서비스를 제공하는 툴입니다. 다수의 개발자들이 하나의 프로그램을 개발할 때 버전 충돌을 방지하기 위해 각자 작업한 내용을 공유 영역에 있는 저장소에 빈번히 업로드함으로써 지속적 통합이 가능하도록 해줍니다.  주  https://lh5.googleusercontent.com/u0snZg55NDKM4vBQUIQ7mqWGN2y2aS0C0X7Cq4K8uoZjkbfU1A1v2zx954NLwo6b2gykxKgZQrfUBt_UibpSPEuQqcfEhZmdv-1D0qBBO7y8KhFX0oOW3y9EORTTb-SvEiw60QG6    github와 연동하여 master branch를 빌드한 모습    빌드외에도 자동화 테스트를 자동으로 실행할 수 있습니다. 또한 배포 역시 진행할 수 있습니다.  주기적으로 빌드를 수행하기도 하며 git 혹은 github와 연동하여 commit 또는 push가 발생할 때마다 자동으로 빌드를 수행합니다.  github의 hook을 이용하여 git의 push 내용을 탐지하여 빌드할 수 있습니다.    Jenkins 2.0이상 버전부터 추가된 pipe라인을 통하여 명령을 작성할 수 있습니다.  groovy를 통하여 작성합니다. 다양한 명령을 순차, 병렬적으로 실행할 수 있으며 각 job에 대한 예외처리를 통해 유연한 작업을 할 수 있습니다.      https://lh3.googleusercontent.com/QopM48oQy8lD-GB_-YmFBB8sZ7MpaFFKmGuq24DwOndC_eJ41dJ5eIad9uPiwwOpINzr74ZNN3msDX6zTJonz2RaCoW5kTGPN9DBbl4EGCAnZmKiX4uhjNVxUfRZvWoV9deyvd3Z  github와 연동되어 소스자료를 바로 확인, 다운로드 받을 수 있습니다.    Jenkins의 다양한 기능을 활용하여 팀단위 프로젝트를 수월하게 진행하게 합니다.  다양한 언어의 빌드와 관련된 환경설정을 진행하여 다양한 빌드환경에 대응할 수 있습니다.  자동화된 빌드 실패시 Slcak과 같은 App과 연동하여 알람을 받는다던가 개발자가 작성한 코드가 표준을 지키지 않은 것을 알려준다던가 하는 기능 등을 활용할 수 있고 다양한 플러그인을 설치하여 기본기능 외에도 다양한 기능을 이용할 수 있습니다. | | | |