201611301 조용빈
201611305 채현우
201811219 정서윤
201812627 유지원

Software V&V

CTIP 환경 구축

1. IDE

공통된 개발자 툴을 하나의 그래픽 사용자 인터페이스로 결합하는 어플리케이션을 구축하기 위한 소프트웨어
(소스 코드 편집기, 로컬 빌드 자동화, 디버거 등)

1. IDE

장점

- 다른 Testing Tool과 연동 편리
- 안정적인 Plugin 지원
- 자동완성, 스마트 코딩 등 기능이 다른 IDE 비교 -> 우수

단점

- 유료 프로그램 ⇒ But , 커뮤니티 버전 혹은 학생 라이센스로 무료 사용 가능

1. IDE

장점

- 무료 프로그램
- 여러 프로젝트를 한 윈도우에서 볼 수 있음
- 플러그인 종류가 많고 확장이 쉬움

단점

- 플러그인 호환 문제 잦음
- 개발 언어 추가 시 플러그인 설치 해야 함

1. IDE

2. Code Configuration Management

소프트웨어의 변경사항을 체계적으로 추적하고 통제하는 것

2. Code Configuration Management

- 분산 저장소 타입
- 개발자가 자신만의 version history를 가짐
- local에 commit되고 충돌이 되지 않게 merge 후 원격 저장소에 올리게 되어 있어 충돌 위험 적음
- 다른 도구와 연동성 좋음

2. Code Configuration Management

- Client/Server 타입
- 개발자가 자신만의 version history를 가질 수 없다.
- 두 개발자가 동시에 수정하고 commit 시 충돌 위험 많다

2. Code Configuration Management

3. Requirements Management&Bugtracking

개발 도중 발생하는 이슈들을 트래킹해주는 도구, 분업 시 이슈(새 기능, 버그, 추가 기능)등 관리에 용이

3. Requirements Management&Bugtracking

- 유료 (But, 10명 이하 무료)
- 애자일 개발 방식 지원
- 작업자별로 상태 값 변경 가능
- 다양한 플러그인 지원
- 통계 및 시각화 기능 자체적 제공
- Git과의 연동 가능 -> 이슈 추적 용이
- Slack 과 연동 -> 각 사용자에 맞는 알림 제공

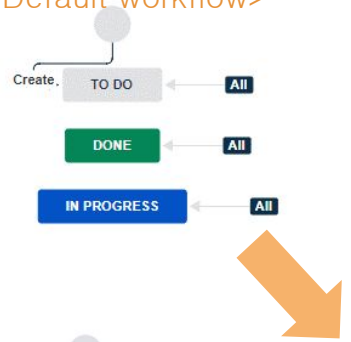
3. Requirements Management&Bugtracking

- 오픈소스이며 무료
- 다양한 모듈 제공 - 일감관리, 시간 추적, 간트 차트 등
- 플러그인 기능 지원
- 여러 데이터 베이스 지원 - Mysql, SQLite 등

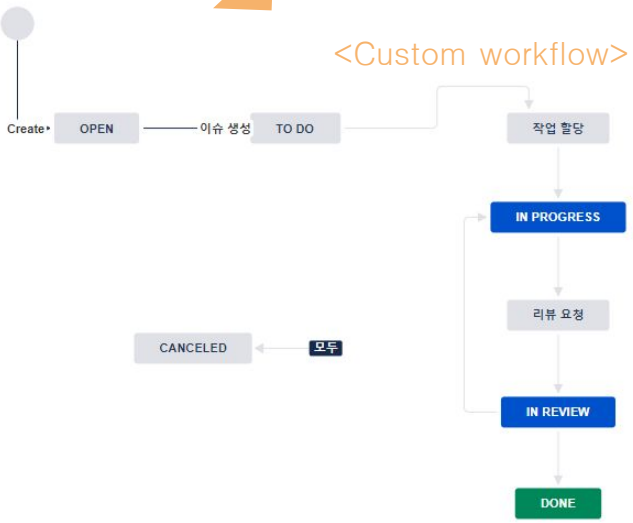
3. Requirements Management&Bugtracking

<Default workflow>

프로젝트 특성에 따른 workflow 커스텀 가능



<Custom workflow>



분리된 스프린트들을 이용하여 작업 단위별 (CTIP환경 구축, STA도구 연동, System testing, Static Analysis 등)로 이슈 관리 가능

다양한 보고서들을 이용하여 이슈 분석 / 관리, 사용자 업무량 및 예상 완료 시간 추적 가능

3. Requirements Management&Bugtracking

4. Unit Testing

컴퓨터 프로그래밍에서 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차

4. Unit Testing

- 자바 전용 Unit test를 위한 표준 framework

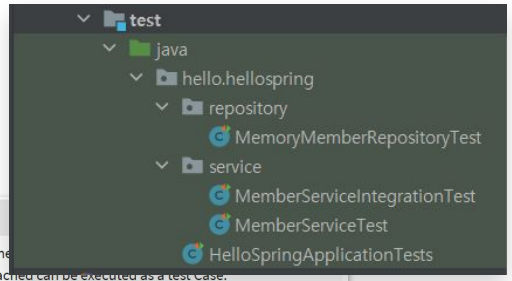
- System.out.println을 사용하지 않고, unit test가 가능하게
  (DB, 화면 등 연결하지 않아도 테스트 가능)

- 테스트 성공/실패를 자동으로 알려줌

4. Unit Testing

- 단정문으로 테스트 케이스의 수행 결과를 판별 assertThat을 이용하여 Test 진행

```
@Test
public void findByName(){
    Member member1 = new Member();
    member1.setName("spring1");
    repository.save(member1);

    //shift+F6 : rename편하게
    Member member2 = new Member();
    member2.setName("spring2");
    repository.save(member2);

    Member result = repository.findByName("spring1").get();
    assertThat(result).isEqualTo(member1);
}
```



- Class/Method 선언 시 annotation으로 간결하고 readable하게 테스트 지원

```
//test가 끝난 후 메모리 정리해주는 메소드 , 테스트들 중 어떤 것이 먼저 실
@AfterEach
public void afterEach() { repository.clearStore(); }

@Test
public void save(){
    Member member = new Member();
    member.setName("spring");

    repository.save(member);
    Member result = repository.findById(member.getId()).get();
    Assertions.assertEquals(member,result);
}
```

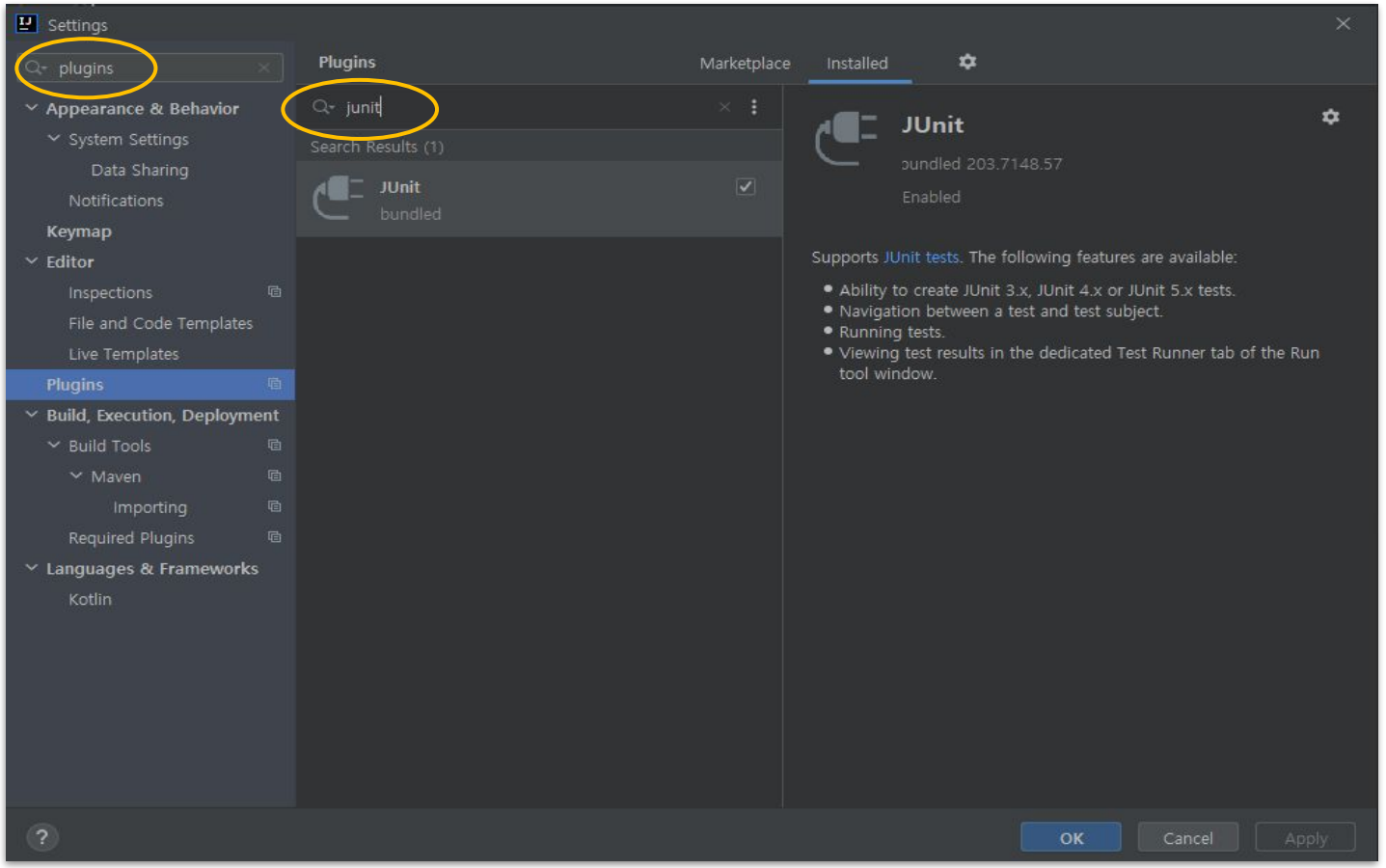
S.No.	Annotations	Description
1.	@Test	This annotation is a replaceme oid method to which it is attached can be executed as a test case.
2.	@Before	This annotation is used if you want to execute some statement such as preconditio ns before each test case.
3.	@BeforeClass	This annotation is used if you want to execute some statements before all the test c ases for e.g. test connection must be executed before all the test cases.
4.	@After	This annotation can be used if you want to execute some statements after each Tes t Case for e.g resetting variables, deleting temporary files ,variables, etc.
5.	@AfterClass	This annotation can be used if you want to execute some statements after all test c ases for e.g. Releasing resources after executing all test cases.
6.	@Ignores	This annotation can be used if you want to ignore some statements during test exe cution for e.g. disabling some test cases during test execution.
7.	@Test(timeout=5 00)	This annotation can be used if you want to set some timeout during test execution for e.g. if you are working under some SLA (Service level agreement), and tests nee d to be completed within some specified time.
8.	@Test(expected=I llegalArgumentEx ception.class)	This annotation can be used if you want to handle some exception during test exec ution. For, e.g., if you want to check whether a particular method is throwing specif ied exception or not.

https://www.guru99.com/junit-annotations-api.html

4. Unit Testing

플러그인 설치

File – settings → plugins검색 → JUnit 검색



4. Unit Testing
예시 및 실행결과

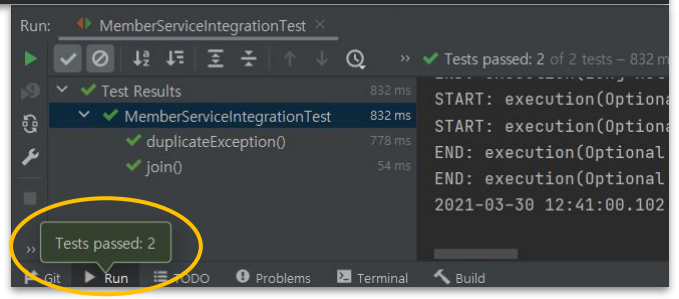
java파일

```java
public class MemoryMemberRepository implements MemberRepository{
    private static Map<Long, Member> store = new HashMap<>();
    private static long sequence = 0L;

    @Override
    public Member save(Member member) {
        member.setId(++sequence);
        store.put(member.getId(),member);
        return member;
    }

    @Override
    public Optional<Member> findById(Long id) { return Optional.ofNullable(store.get(id)); }

    @Override
    public Optional<Member> findByName(String name) {
        return store.values().stream()
                .filter(member -> member.getName().equals(name))
                .findAny();
    }

    @Override
    public List<Member> findAll() { return new ArrayList<>(store.values()); }

    public void clearStore(){
        store.clear();
    }
}
```

UnitTest

```java
//통합테스트
@SpringBootTest
@Transactional //반복적인 테스트를 위해 존재하는 기능. insert등을 db에 한 후 마지막에 Rollback시킨다.
class MemberServiceIntegrationTest {

    //test는 어차피 끝단에 존재하는 애들이기 때문에 그냥 편한 Autowired로 DI시켜도 괜찮다. (생성자 주입 등 굳이X)
    @Autowired MemberService memberService;
    @Autowired MemberRepository memberRepository;

    @Test
    //@Commit //commit태그를 사용하면 디비에 바로 반영이 가능하다.
    void join() {
        //given
        Member member = new Member();
        member.setName("spring");

        //when
        Long saveId = memberService.join(member);

        //then
        Member findMember = memberService.findOne(saveId).get();
        Assertions.assertThat(member.getName()).isEqualTo(findMember.getName());
    }

    @Test
    public void duplicateException(){
        //given
        Member member1 = new Member();
        member1.setName("spring");

        Member member2 = new Member();
        member2.setName("spring");

        //when
        memberService.join(member1);
        IllegalStateException e = assertThrows(IllegalStateException.class, () -> memberService.join(member2));
        assertThat(e.getMessage()).isEqualTo("이미 존재하는 회원입니다.");
    }
}
```

실행결과



4. Unit Testing
예시 및 실행결과

```
VMProjectTest > multiply() FAILED
    org.opentest4j.AssertionFailedError at VMProjectTest.java:11

1 test completed, 1 failed

> Task :test FAILED

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':test'.
> There were failing tests. See the report at: file:///var/lib/jenkins/workspace/Vending%2

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get m

* Get more help at https://help.gradle.org

BUILD FAILED in 14s
5 actionable tasks: 5 executed
Build step 'Invoke Gradle script' changed build result to FAILURE
Build step 'Invoke Gradle script' marked build as failure
[Slack Notifications] found #9 as previous completed, non-aborted build
[Slack Notifications] will send OnEveryFailureNotification because build matches and user prefe
Finished: FAILURE
```

Junit 테스트 실패

```
 > git config core.sparsecheckout # timeout=10
 > git checkout -f f557a0b94e4f053da7c4425072465afa03bc6888 # timeout=
Commit message: "junit test3"
 > git rev-list --no-walk 0022c645205a8c6965b72c4f2b58128351d3d629 # t
[Gradle] - Launching build.
[Vending Machine Project] $ /var/lib/jenkins/tools/hudson.plugins.grac
Starting a Gradle Daemon (subsequent builds will be faster)
> Task :clean
> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes
> Task :jar
> Task :assemble
> Task :compileTestJava
> Task :processTestResources NO-SOURCE
> Task :testClasses
> Task :test
> Task :check
> Task :build

BUILD SUCCESSFUL in 14s
5 actionable tasks: 5 executed
Build step 'Invoke Gradle script' changed build result to SUCCESS
[Slack Notifications] found #10 as previous completed, non-aborted bui
[Slack Notifications] will send OnBackToNormalNotification because bui
Finished: SUCCESS
```

Junit 테스트 성공

5. Automatic Build (Maven, Gradle)

소스코드에서 실행 가능한 어플리케이션을 자동 생성하는 프로그램
라이브러리들을 설정 파일을 통해 자동으로 다운로드 해주고 간편히 관리해주는 도구
(Compile + Packaging + Testing + Deploy + Document)

5. Automatic Build (Maven, Gradle)

- pom.xml을 이용한 정형화된 빌드 시스템 제공
- 상속 구조 멀티 프로젝트

5. Automatic Build (Maven, Gradle)

- JAVA, C/C++, Python 등 지원

- Groovy 기반의 별도 빌드 스크립트로 라이브러리 관리
  (Build-by-convention을 바탕으로 하여 스크립트 규모가 작고 읽기 쉬움)

  ⇒ Maven과 달리 동적인 빌드 행위에 xml로 제한을 가하지 않아도 됨

- 구성 주입 방식 채택 (Configuration Injection)

  ⇒ 멀티 프로젝트 구성시 재사용에 용이함

- 다양한 플러그인 사용 가능

5. Automatic Build (Maven, Gradle)

6. Team Communication

개발 도중 발생하는 이슈들을 트래킹해주는 도구, 분업 시 이슈(새 기능, 버그, 추가 기능)등 관리에 용이

6. Team Communication

- 일반적인 메신저와 크게 다르지 않은 익숙한 인터페이스.

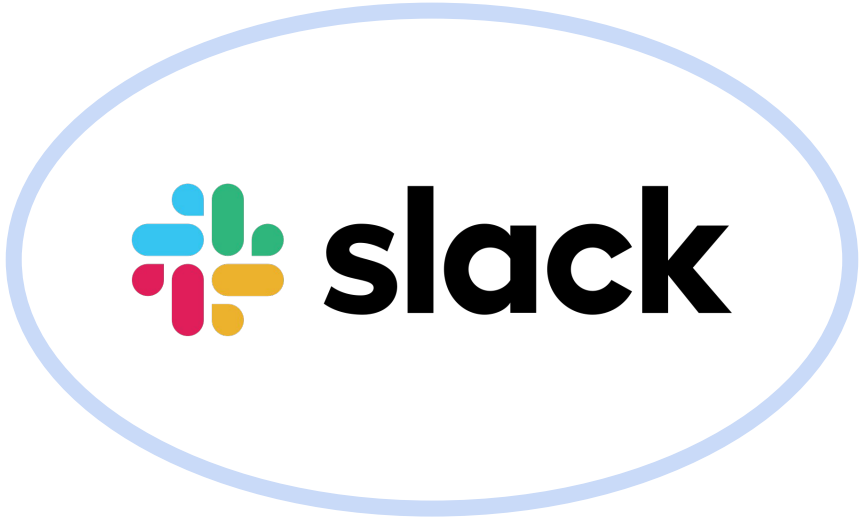
- Git, Jenkins, Jira등 CTIP 환경 내 다른 도구들과 연동가능. (다양한 플러그인 제공)

- 목적 별 채널 분리로 명확한 의사소통이 가능.

6. Team Communication

- 채팅, 통화, 화면 공유 등을 무료로 가능
- CTIP 환경 내 다른 도구들과 연동이 어려움
- Jira와 연동 불가

6. Team Communication



7. CI(Continuous Integration) Server

팀원들이 작업한 내용을 정기적으로 통합하는 것, Submit된 코드들을 정기적으로 통합하여 빌드 프로세스를 관리한다.
→ 문제의 조기 발견 및 해결 가능

7. CI(Continuous Integration) Server

Jenkins

- 오픈 소스로 무료로 이용가능
- 많은 Plugin을 통해 다른 플랫폼과 연동 가능
  (Github, Sonarqube, Discord, Slack 등)
- 클라우드 서버 구축 혹은 로컬 서버로 사용 가능

7. CI(Continuous Integration) Server

- 프로세스의 빠른 실행을 위해 병렬 빌드 설정 가능
- 모든 작업을 새 컨테이너로 실행하여 문제를 일으키는 오래된 빌드 데이터를 방지
- 성능 기반 확장 옵션 제공

7. CI(Continuous Integration) Server

Jenkins

circleci

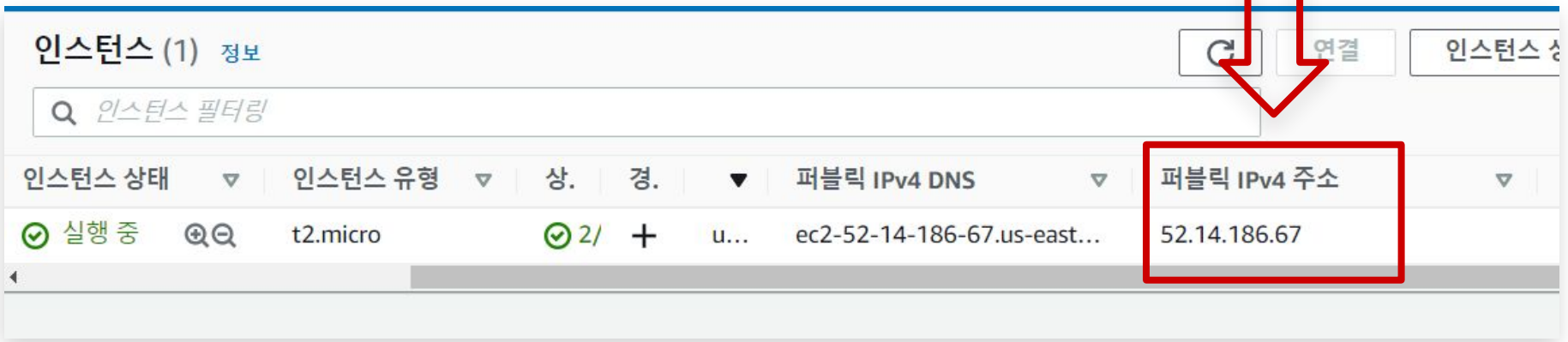
8. AWS + Jenkins

AWS free tier 를 이용해 EC2 서버를 개설하고 Jenkins 를 설치함

***특이사항**
중간에 인스턴스를 종료했다 다시 시작했더니 IP 주소가 달라져서 혼동이 있었음

```
yum update -y

# Jenkins 패키지 추가
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins.io/redhat/jenkins.repo &&
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key

# Install java, docker, git
sudo yum install -y java-1.8.0-openjdk jenkins git docker
# 자바 버전 8 로 설정
alternatives --config java
service jenkins start
```

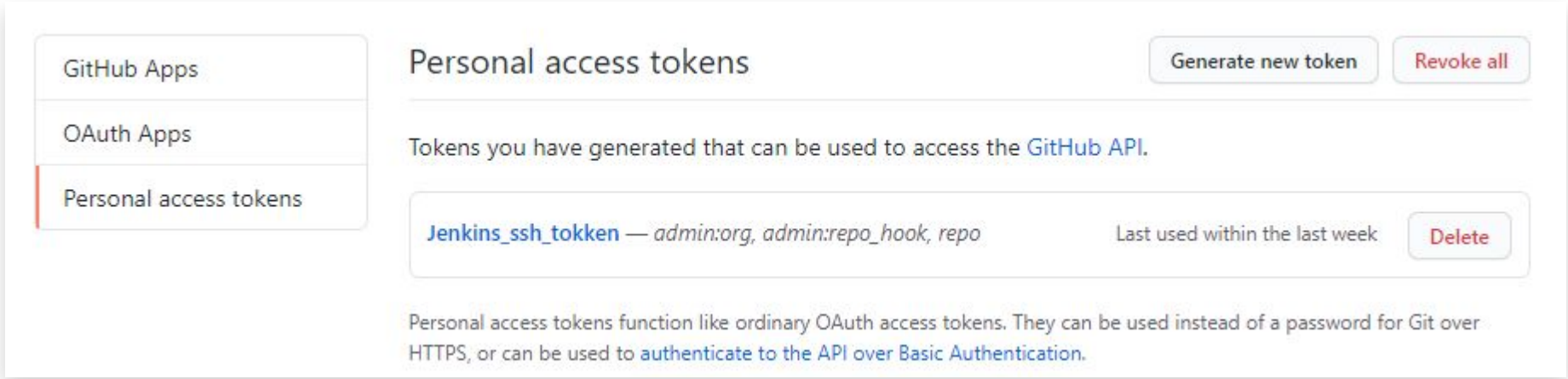
http://18.224.52.132:8080



9. Jenkins + Git

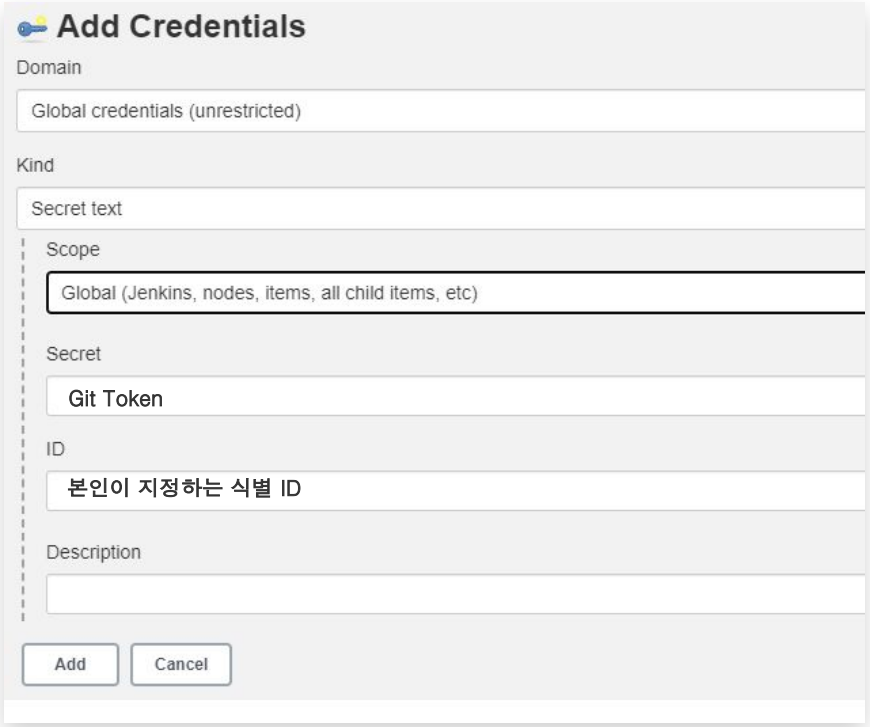
Git Token 생성

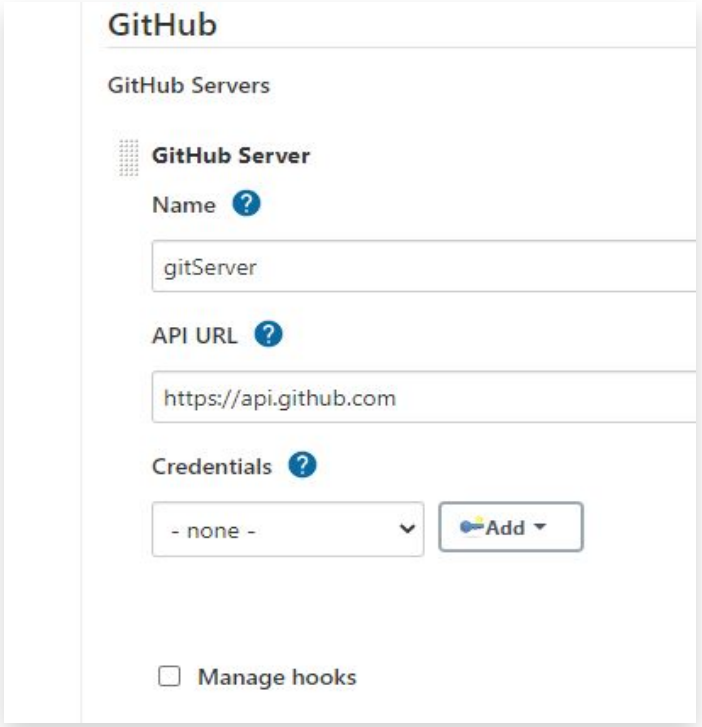
github -> Settings -> Developer Settings -> Personal access tokens -> Token 생성



9. Jenkins + Git

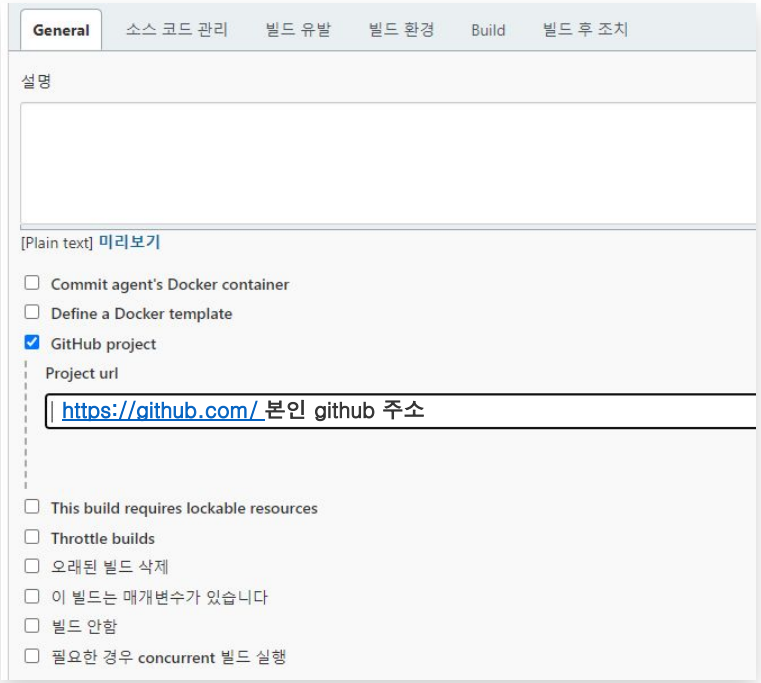
Jenkins 관리 -> 시스템 설정 -> Github Server -> 위에서 만든 Token을 사용해 설정

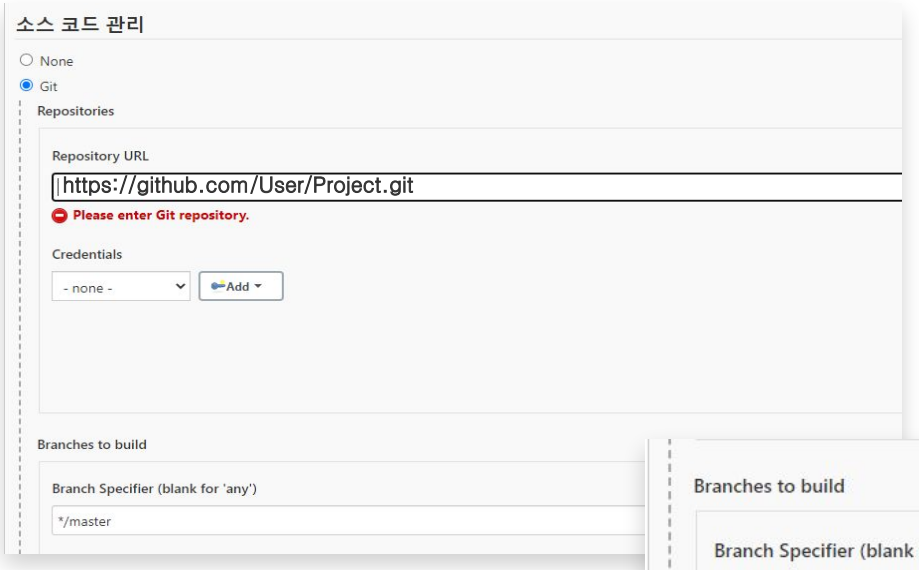




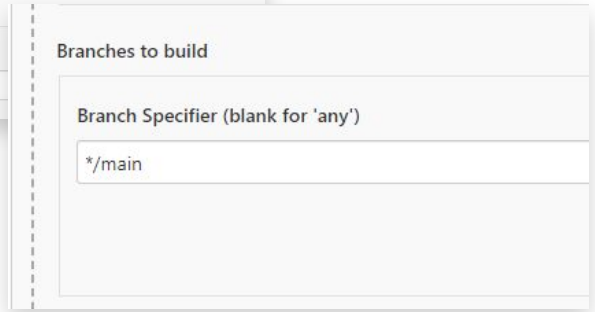
9. Jenkins + Git

Jenkins 새로운 Item 소스 코드 관리 -> git -> Repository Url(~~~.git) -> Credentials -> 이전에 생성한 Credentials 선택
FreeStyle Project -> General -> Github project -> project url -> 본인의 Git project url



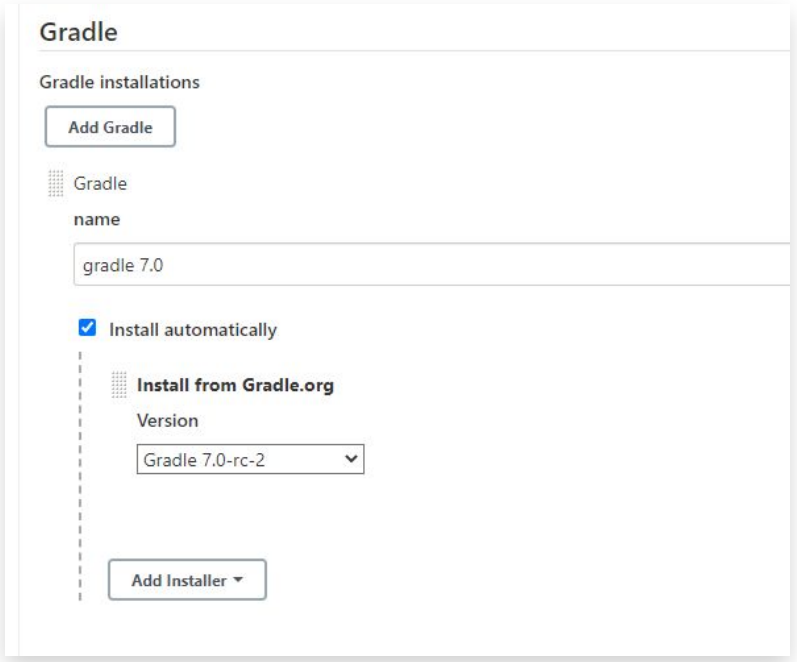


Git의 master 브랜치가 main 으로
바뀌어있었기 때문에 오류가 났었음
=> 빌드할 브랜치 이름을 꼭 확인할 것

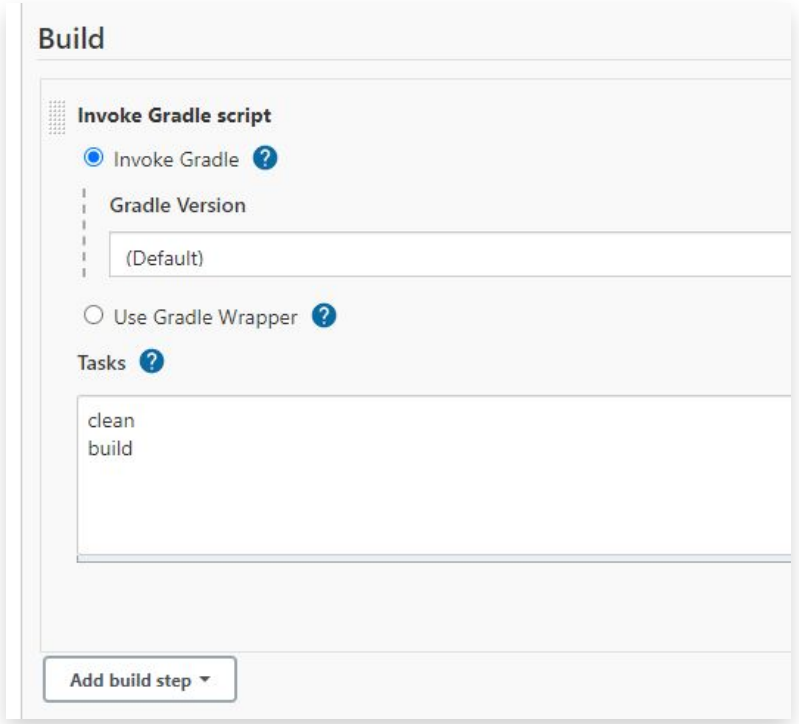


10. Jenkins + Gradle

jenkins 관리 -> Global tool configuration



item -> 소스코드 관리



10. Jenkins + Gradle

젠킨스 빌드시 메모리 누수 문제 발생
Jenkins 'Cannot allocate memory' Error

-> swap memory 추가로 해결

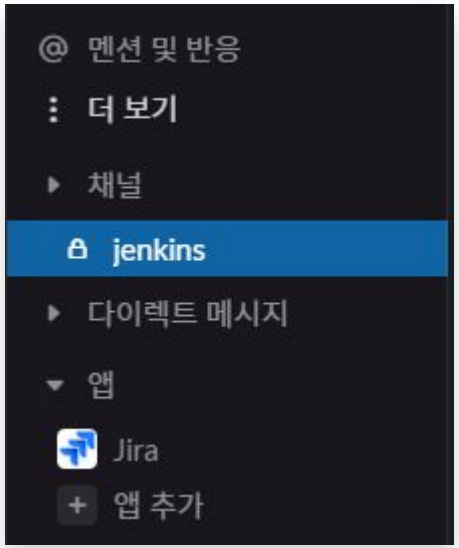
```
$ free -h
              total        used        free      shared  buff/cache   available
Mem:           983M        565M        333M         68K         84M        305M
Swap:          2.0G        114M        1.9G
```

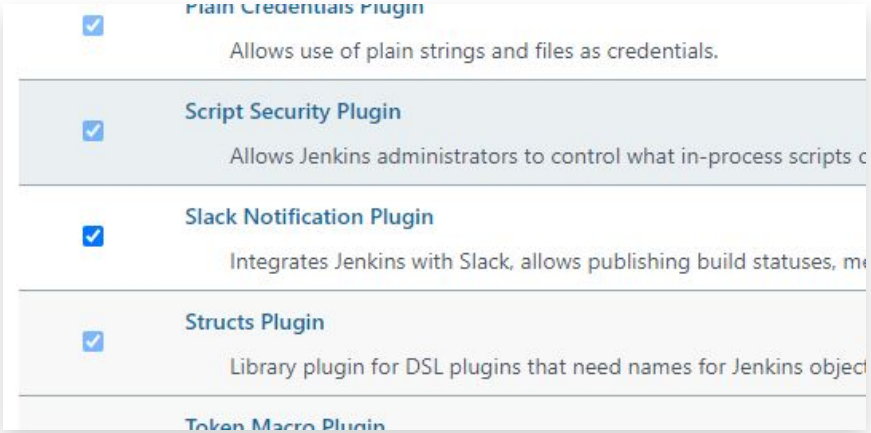
참고:https://blog.jiniworld.me/33

11. Jenkins + Slack

Slack -> 앱 추가 -> Jenkins CI 설치(알림 받을 채널 선택, team token 저장)

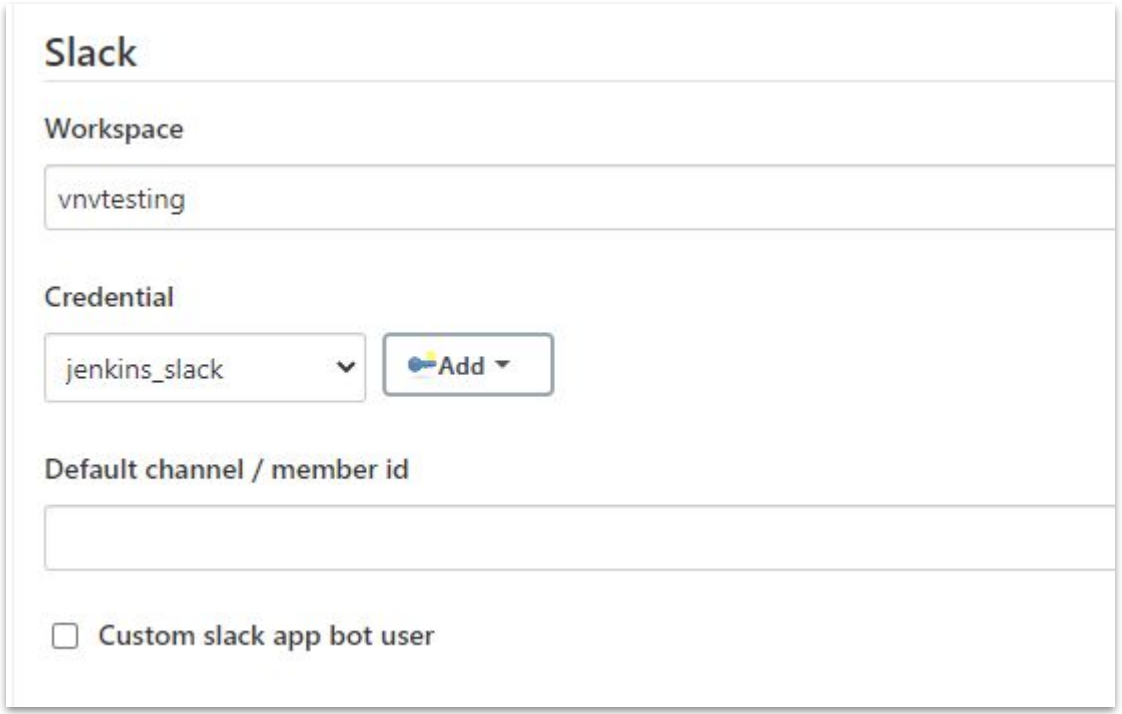
Jenkins -> 플러그인 관리 -> Slack Notification Pulgin 설치





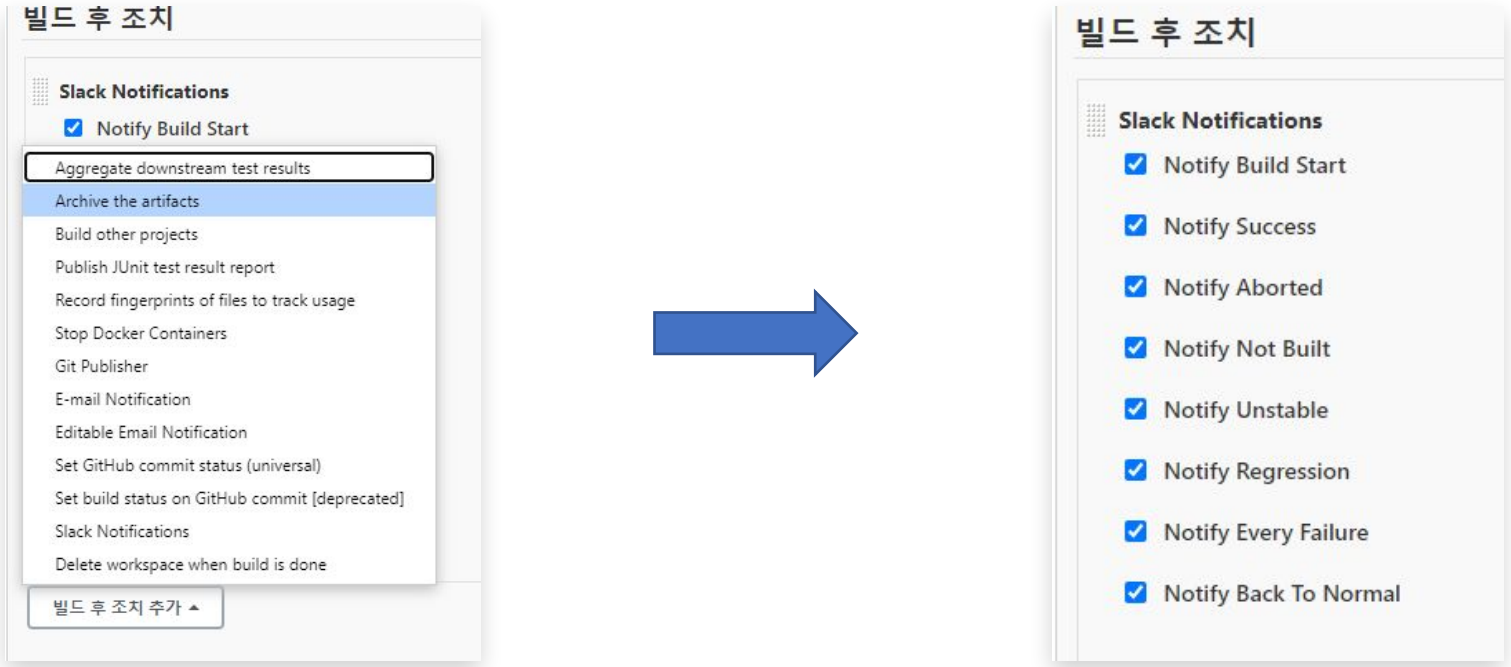
11. Jenkins + Slack

Jenkins -> Jenkins 관리 -> Configuration System -> Slack -> 알림 받을 채널 ID 입력



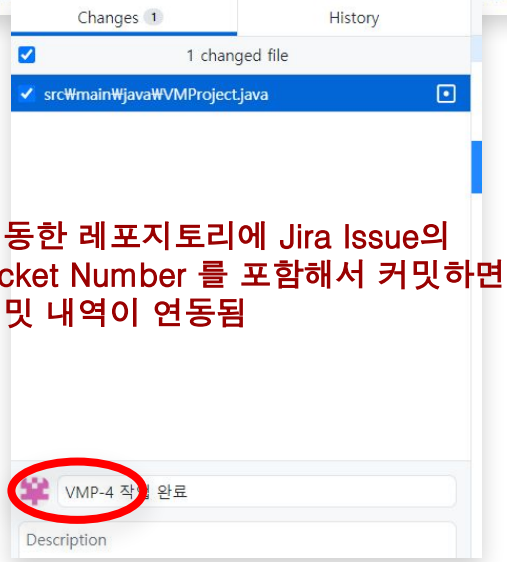
11. Jenkins + Slack

Item -> 빌드 후 조치 -> Slack Notification -> 원하는 항목 선택



12. Jira + Github

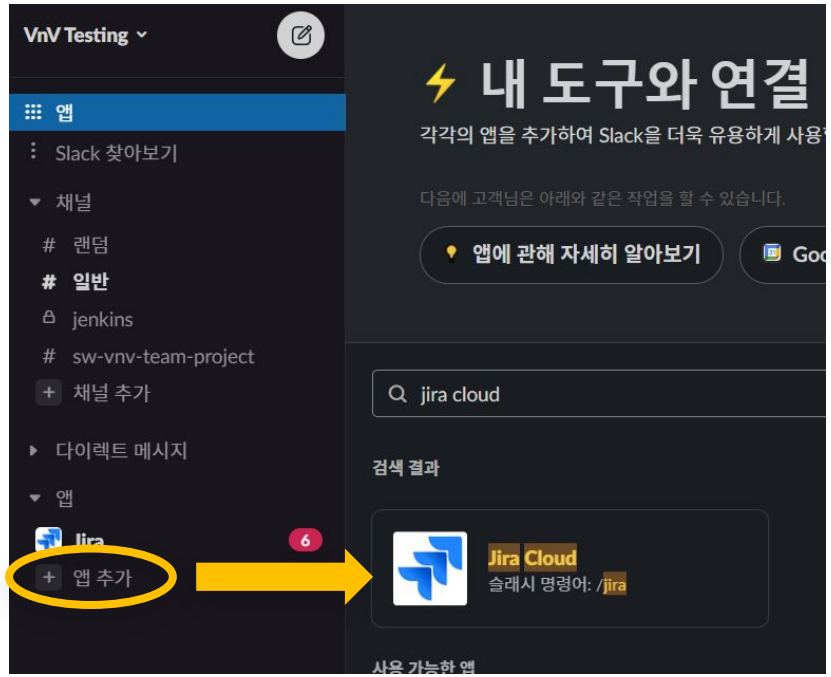
Jira 앱 > 새 앱 찾기 > GitHub for Jira > Add an Organization > Authorize Jira > Install Jira > Repository 지정 > 연동 확인

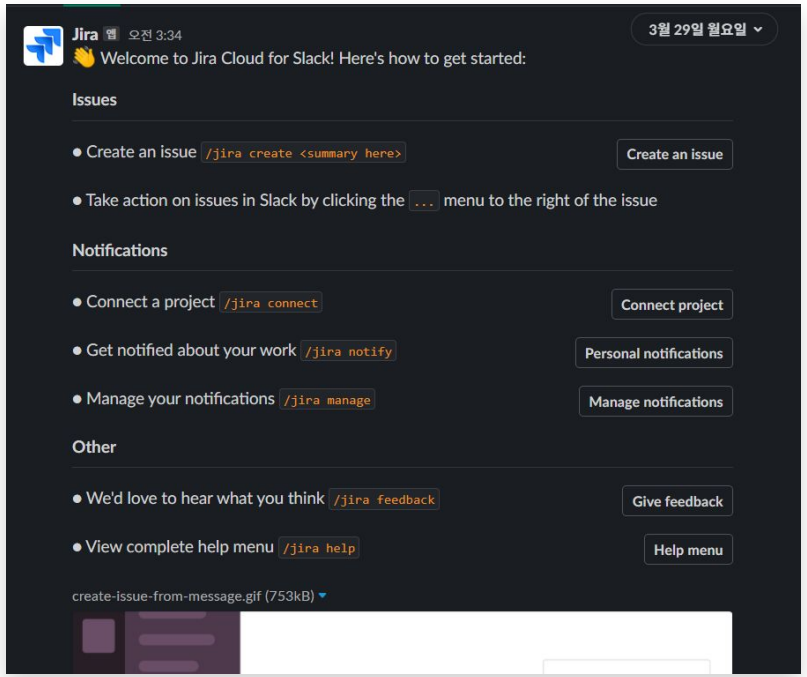


연동한 레포지토리에 Jira Issue의
Ticket Number 를 포함해서 커밋하면
커밋 내역이 연동됨

13. Jira + Slack

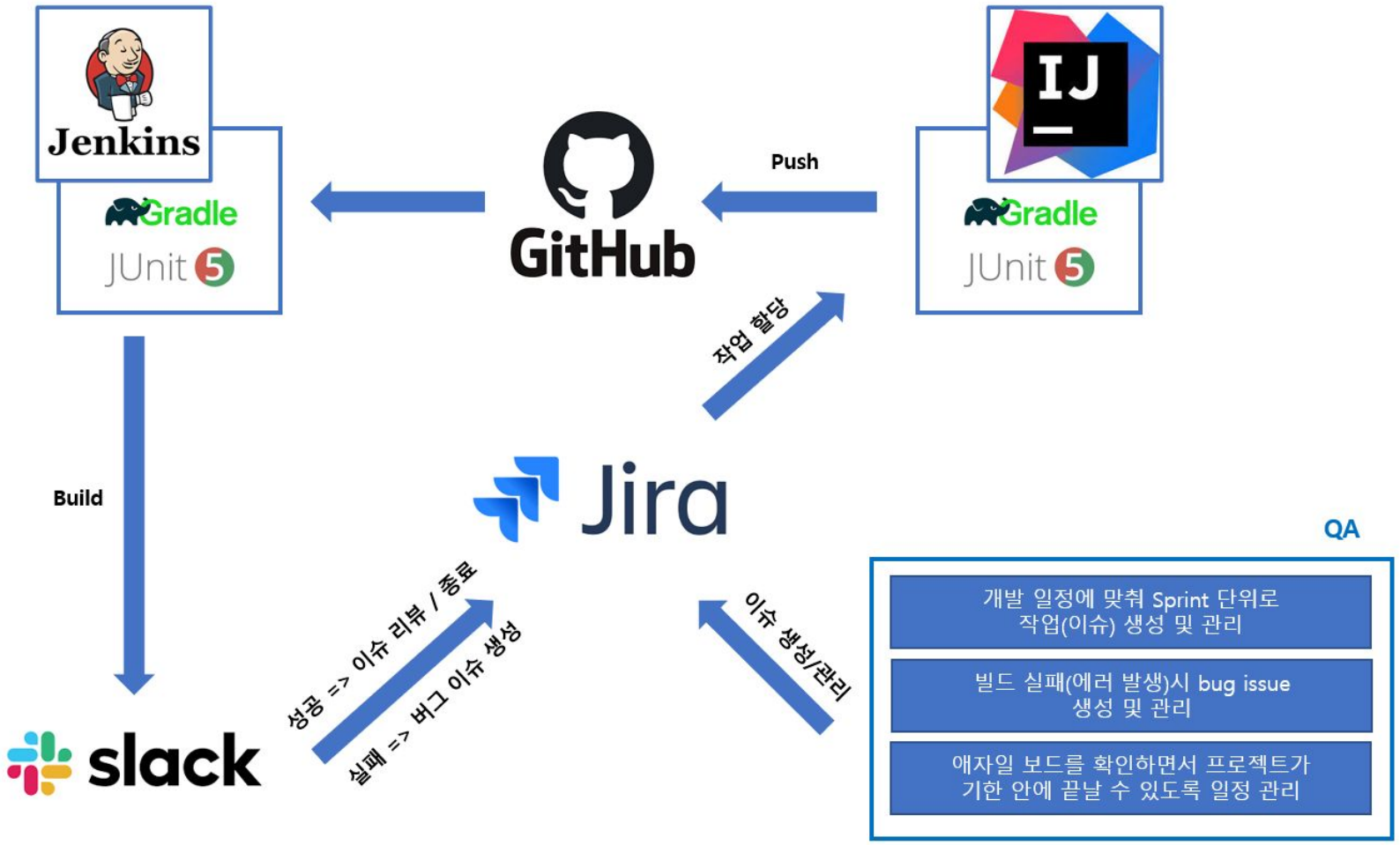
Slack 에 Jira Cloud 추가 > Attlassian 로그인 >> 연동 확인 >> 슬랙 메신저에서 Jira Project 연결





/명령어 형식을 이용해서 Slack 에서 Jira 이슈들을 관리할 수 있음

14. 최종


Jenkins
Gradle
JUnit 5
GitHub
Push
IJ
Gradle
JUnit 5
작업 할당
Build
Jira
slack
성공 => 이슈 리뷰 / 종료
실패 => 버그 이슈 생성
이슈 생성/관리
QA
개발 일정에 맞춰 Sprint 단위로
작업(이슈) 생성 및 관리
빌드 실패(에러 발생)시 bug issue
생성 및 관리
애자일 보드를 확인하면서 프로젝트가
기한 안에 끝날 수 있도록 일정 관리