

IDG Deep Dive

# 프로그래머 개발 도우미 '깃허브' 완전정복

소스코드를 저장, 공유하는 리포지토리가 소프트웨어 개발의 필수 툴로 부상하고 있다. 특히 오픈소스 리포지토리인 '깃허브'는 개인 개발자부터 중소기업, 대기업까지 거의 모든 개발 조직의 주목을 받고 있다. 깃허브를 도입한 한 대기업은 오랜 병폐 중 하나인 관료주의를 몰아내고 혁신을 이뤄가고 있다고 평가하기도 했다. 개발자가 사랑하는 필수 프로그래밍 도우미 '깃허브'의 활용 가치와 그 세계에 입문하는 필수 팁을 살펴보자.



## ※ Market Trend

점점 높아오르는 기업용 리포지토리 '삼국지'

## ※ Tech Guide

깃허브 처음 시작하기  
'깃 스마트!' 깃허브 사용자를 위한 20가지 필수 팁

## ※ Tech Story

월마트, 오픈소스로 관료주의를 몰아내다

## ※ Column

기업의 미래를 보는 특별한 기준 '깃허브 수용도'

무단 전재 재배포 금지

본 PDF 문서는 IDG Korea의 프리미엄 회원에게 제공하는 문서로, 저작권법의 보호를 받습니다.  
IDG Korea의 허락 없이 PDF 문서를 온라인 사이트 등에 무단 게재, 전재하거나 유포할 수 없습니다.

# 깃허브-깃랩-비트버킷 점점 달아오르는 기업용 리포지토리 '삼국지'

Paul Krill | InfoWorld

깃허브(GitHub)는 오픈소스 리포지토리(repository, 저장소) 호스팅 분야의 대표 주자이다. 그러나 기업용 프라이빗 리포지토리 호스팅 서비스 시장만 놓고 보면 상황이 조금 다르다. 깃랩(GitLab), 아틀라시안 비트버킷(Atlassian Bitbucket) 등이 깃허브의 경쟁자로 부상하고 있다. 이들 3개 업체 모두 깃(Git) 분산 버전 관리 시스템을 기반으로 구축한 플랫폼을 서비스한다. 기업 방화벽 안에서 리포지토리를 호스팅하는 온프레미스 솔루션을 제공하는 것도 닮은 꼴이다.

그러나 공통점은 여기까지다. 3개 업체는 '3인 3색' 자신만의 강점을 내세우고 있다. 먼저, 깃랩은 자사의 서비스가 자체 리포지토리 호스팅을 선호하는 기업 사이에서 인기를 끌고 있다고 주장한다. 깃랩 CEO 시드 시즈브란디즈는 "우리 솔루션은 기업 시장을 위해 태어났다. 자체 서버에서 운영하려는 기업이 주요 고객이다. 리포지토리를 자체 서버에서 호스팅하는 시장에서는 깃랩이 깃허브보다 점유율이 높다"라고 말했다. 주요 고객사로는 익스피디아, AT&T, 나스닥 등이 있다. 아틀라시안도 비슷한 설명을 내놓는다. 업체의 소프트웨어팀 총괄 임원 켄스 슈마커는 "깃허브가 오픈소스에 집중한 것과 달리 아틀라시안 비트버킷 플랫폼은 시작부터 기업 시장에 초점을 맞춰왔다"라고 강조했다.

깃랩과 아틀라시안 모두 자사의 제품이 깃허브 엔터프라이즈보다 좋은 기능을 제공한다고 주장한다. 깃랩은 5단계 접속 인증과 지속적 통합, 딜리버리를 위한 프로비저닝 기능을 예로 제시한다. 채팅 클라이언트, 칸반(Kanban) 보드와 유사한 계획 게시판 기능을 지원

하고, 사설 도커(Docker) 컨테이너와 도커 레지스트리도 쓸 수 있다. 깃랩은 앞으로 대기업이 소규모 소프트웨어 개발팀 같은 효율성을 달성할 수 있도록 소프트웨어 개발 단계 분석 기능을 추가할 예정이다.


아틀라시안 역시 자사의 비트버킷이 이러한 인증과 분산 팀 지원 기능을 제공한다고 설명했다. 리포지토리 복사본을 만드는 스마트 미러링 기능도 있다. 특히 비트버킷 서버와 별개로 고가용성과 고성능이 필요한 대기업을 위해 '비트버킷 데이터센터' 버전을 내놓았다. 비트버킷 서버 노드 클러스터를 지원하며 수백 개의 애드온을 이용해 코드 분석이나 작업 흐름 등의 기능으로 확장할 수 있다. 아틀라시안의 슈마커는 "기업이 깃허브 엔터프라이즈



이즈 대신 비트버킷을 선택하는 주요 이유 중 하나는 아틀라시안 툴 스위트와의 통합 때문이다”라고 말했다. 이 스위트에는 아틀라시안의 인기 프로젝트 관리 툴인 ‘지라(Jira)’가 포함돼 있다.

한편 깃허브도 기업 시장에서 새로 부상하는 경쟁자에 대응하고 있다. 깃허브 엔지니어링 부회장 토드 버먼은 “우리는 사용자가 어떤 상황에 있든 상관없이 소프트웨어를 구축하도록 지원할 것이다. 더구나 깃허브의 성과는 경쟁사에 오히려 도움이 된다. 예를 들어 LFS(Large File Storage) 프로토콜은 깃허브가 개발해 오픈소스화한 것으로, 현재 아틀라시안과 깃랩이 이를 사용하고 있다”라고 말했다. 버먼에 따르면, 사실 깃허브는 깃허브 엔터프라이즈를 깃허브닷컴(GitHub.com)의 별도 제품으로 보지 않는다. 기업을 위해 온프레미스 기능을 갖추기는 했지만, 오픈소스 리포지토리 사이트와 본질에서 같다는 것이다.

현재 깃허브는 방화벽 뒤에 설치해 사용하는 깃허브 엔터프라이즈와 깃허브닷컴에서 호스팅하는 프라이빗 리포지토리 서비스를 통해 매출을 올리고 있다. 단, 깃허브 엔터프라이즈 제품은 인스턴트 퍼미션, SAML, LDAP, CAS SSO 프로토콜 등을 추가로 지원한다. 버먼은 “깃허브의 매출원은 깃허브를 프라이빗으로 사용하려는 수요이다. 예를 들어 어도비의 경우 소프트웨어 엔지니어 8,000명이 깃허브 엔터프라이즈를 사용한다. 포천 50대 기업의 44%가 이 서비스를 사용하고 있다”라고 말했다.

깃허브 엔터프라이즈는 가상머신을 통해 기업 방화벽 내에 깃허브닷컴의 축소 버전을 설치하는 방식으로 구축한다. 퍼블릭 클라우드에도 설치해 사용할 수 있다. 깃허브는 최근 프로젝트 관리와 코드 리뷰 기능을 깃허브닷컴에 추가했으며, 이 기능은 향후 깃허브 엔터프라이즈에도 확대 적용될 예정이다. 클라우드와 기업 사이의 데이터 이동성 같은 기능도 추가할 계획이다. 

# 깃허브 처음 시작하기

Matthew Heusser | CIO

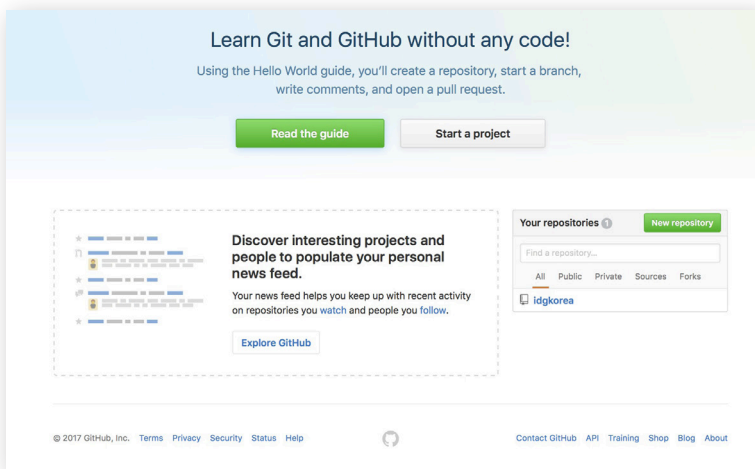
깃(Git)을 정말로 배우고 싶은데 직접 서버를 구성할 수 없다면, 깃허브(GitHub) 서비스가 안성맞춤이다. 깃허브에 주목해야 하는 이유는 또 있다. 바로 채용이다. 실제로 그루폰의 선임 인재 스카우트는 구인 사이트가 아니라 깃허브 커밋(Commit)을 검색해 채용 대상을 물색하고 있다고 말했다. 이래도 깃허브를 배워야 할 이유로 부족하다면, 무료 튜토리얼은 어떤가? 오늘은 독자들이 운 좋은 날이다. 내용을 훑어 보는 데 한 시간 정도면 충분할 것이다.

## 첫째, 깃 프로젝트를 하나 고르거나 샘플을 사용하라

깃은 이슈당 한 개의 브랜치(Branch-per-Issue) 모델을 채택하고 있다. 로컬 사본(Local Copy)이 사용자가 지금 작업하고 있는 브랜치이다. 사용자는 이 로컬 사본을 확정(commit), 롤 백(Roll Back)할 수 있으며, 주기적으로 깃허브 서버에 올릴(Push) 수 있다. 이처럼 기본 작업은 추가(Add), 확정(Commit) 그리고 올리기(Push)이다. 깃허브는 기업 사용자에게 대해서는 비용을 받지만 오픈소스 프로젝트는 무료이다.

여기서는 오픈소스 프로젝트를 만들어서 이를 깃허브에 올리는 과정을 알아보자. 단지 코딩만 조금 하면 된다. 먼저 약간의 코드와 빌드 스텝(Build Step)이 포함된 프로젝트를 선정해야 한다. 코드가 전혀 없다면, 루비(Ruby)만 있으면 되는 **팩토리 시뮬레이션**을 사용해도 된다. 'lib'이란 이름의 서브 디렉터리를 가지는 디렉터리 하나를 만들고 적절한 곳에 파일 5개를 복사하면 된다. 이제 애플리케이션을 구동하기 위해 루트 디렉터리로 가서

명령어 라인에서 'ruby factory\_multi3.rb'를 입력한다. 이를 테스트해 보려면 테스트 디렉터리로 가서 'ruby run\_all.rb'를 입력하면 된다. 이 코드는 자크 스펜서의 도움을 받아 필자가 작성해서 오픈소스 라이선스로 기부한 것이다.



화면 1 | 리포지토리 생성부터 시작한다. 쉽지 않다면 깃허브가 제공하는 초보 가이드를 참고하면 된다.

## 깃허브 계정을 생성하라

이제 github.com으로 가서 등록하고 최신 버전의 명령어 라인 도구를 내려받아 설치하라. 여기서부터는 웹 인터페이스를 통해 리포지토리(Repository)를 생성하고, 명령어 라인 도구를 사용해 우리 코드를 깃허브로 가져오기할 것이다. 로그인 후 오른쪽 가운데 있는 초록색 '+ New Reposi-

tory' 버튼을 클릭한다:

리포지토리를 생성했다면 명령어 라인을 시작하고 다음 명령어를 입력한다. 코드 디렉터리에 따라 정확하게 디렉터리를 변경해야 한다.

```
touch README.md
```

단, 윈도우 버전에는 'touch' 명령어가 없다. 대신, README.md라는 이름의 빈 텍스트 파일을 편집해 저장하면 된다.

```
git init
git add *
git commit -m "first commit"
git remote add origin https://github.com/(Username)/(Repository_name).git
git push -u origin master
```

이제 사용자명과 비밀번호를 입력하면 깃허브에 공개 코드를 올려놓는 작업이 끝났다. 지금 만든 URL([https://github.com/\(Username\)/\(Repository\\_name\)](https://github.com/(Username)/(Repository_name)))로 전 세계에서 소스 내용을 볼 수 있다.

### 방금 내가 뭘 일을 했지?

지금까지 한 일을 복기해 보자. 첫째, 'touch' 명령어를 이용해, 온라인으로 자신의 리포지토리 루트 디렉터리를 볼 때 git init이 표시하는 리드미(Readme) 파일을 생성했다. 이 파일에는 프로그램의 구동 방법을 알려주는 내용이 들어간다.

'init'은 사용자의 로컬 디렉터리에 비어있는 깃 리포지토리를 생성했다. 'add\*'는 해당 디렉터리와 로컬 리포지토리를 포함해 모든 서브 디렉터리에 파일을 추가하며, 커밋은 변경사항을 체인지셋(Changeset)에 확정했다. 명령어 라인 가장 뒤에 붙은 -m에도 주목해야 한다. 이것은 버전 주석이다. -m을 빼면, 깃은 vi를 사용해 주석을 편집하도록 한다.

그 다음, 우리는 'remote add'를 사용해서 변경사항을 깃허브에 연결했다. 자체 서버를 운용하고 있다면 URL만 바꾸면 된다. 마지막으로 git push를 사용해서 변경사항을 올려놓았다.

이제, (조금 전에 필자가 가져오기 한) <https://github.com/MattExcelon/factory6>로 가면, README.md가 비어있음을 알 수 있다. 다음 텍스트를 추가해보자. 이 PDF 문서에서 복사해 바로 붙여넣어도 된다.

```
The Factory Simulation
```

```
From CIO.com, uploaded by (name), based on the work of Matthew
```



```
Heusser  
Matt@xndev.com
```

```
Distributed under the GNU GPL 2.0 license:  
http://choosealicense.com/licenses/gpl-2.0/
```

Imagine a factory that has a number of stations. Each day, work proceeds through the stations.

The stations have high variability but are balanced. We simulate this with a six sided die. Users enter the number of stations and days and the application shows how work processes. The advanced version, `factory_multi3.rb` allows you to simulate multiple runs of the factory (run it a thousand times and take the averages) or change the number of dice.

```
## Dependencies
```

This code developed and tested under ruby 2.0.0p247. As long as you have ruby 1.9.3 or higher you should be fine.

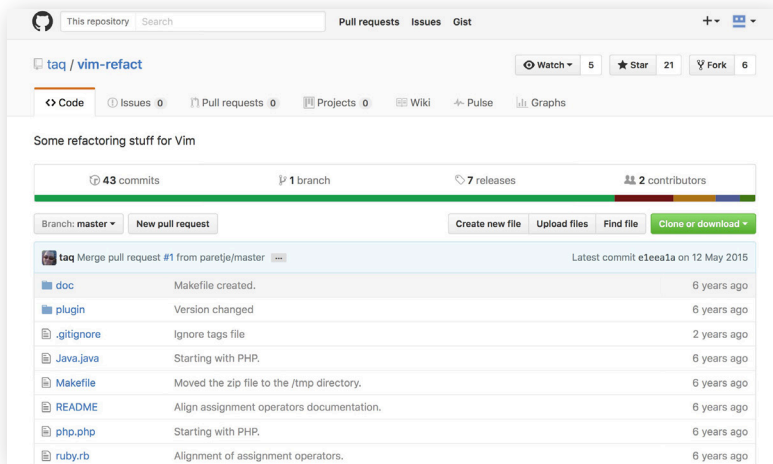
```
## Running the Simulation
```

1. ``cd this/project/directory``
2. ``ruby factory.rb`` or ``ruby factory_multi3.rb``

```
## Running the Tests
```

코드에 확정할 필요가 있는 수정을 했으므로 다음과 같은 작업해준다.

```
git add README.md  
git commit -m "Updated the Readme to provide information"  
git push
```



화면 2 | 다른 사람의 작업에 대한 사본을 만들어 기여할 수 있다.

를 팔로우하고 변경사항에 대한 알림을 받을 수 있다. 또한, 자신의 작업뿐 아니라, 다른 사람의 작업에 대한 사본을 만들어 그 작업에 기여(Fork)할 수 있다. 기여하려면 깃허브 리포지토리를 살펴보고(〈화면 2〉의 예제는 <https://github.com/taq/vim-refact>이다) 오른쪽 위에 있는 'Fork' 버튼을 클릭하면 된다. 그러면 자동으로 리다이렉트돼 나만의 버전이 생성된다.

이제는 자신의 페이지 이름 오른쪽에 'HTTPS 복제 URL(HTTPS clone URL)'이 붙어 있을 것을 볼 수 있다. 명령어 라인에서 깃 설치본의 루트로 가서 다음과 같이 입력한다.

```
git clone (복사한 URL)
```

깃은 이 리포지토리 사본을 사용자의 로컬 기기에 생성하므로 사용자가 추가, 확정 그리고 올리기를 할 수 있다. 변경사항은 사용자 자신의 리포지토리에만 국한되며 다른 누구에게도 영향을 주지 않는다. 적절한 시기가 되면, 풀 요청(Pull Request)을 해서 다른 사람에게 자신의 변경사항을 알리면 된다. 그러면 원하는 사람이 변경사항을 적용할 수 있다. 이번 글에서는 깃의 기본 사용법을 살펴보았다. 깃허브에 추가, 확정, 올리는 방법 그리고 리포지토리 생성 방법 등이다. 나만의 코드를 작성하기에는 그 정도면 충분하다. 오히려 더 어려운 것은 채용 담당자가 나의 코드를 알아보고 연락할 만큼 흥미로운 개인 프로젝트를 만드는 것이다. 이것은 독자 여러분의 몫이다. ITWORLD

이렇게 한 후 브라우저에서 [https://github.com/\(Username\)/\(Repository\\_name\)](https://github.com/(Username)/(Repository_name))을 시프트 리프레시(Shift-Refresh, 캐시된 콘텐츠를 무시하고 현재 페이지를 새로 고침)하면 README 내용이 나타난다. 참고로 여기서 # 기호는 일종의 마크다운(Markdown) 언어로 제목을 생성한다.

### 자신의 리포지토리를 생성한 이후 작업

깃허브는 웹사이트 이상이다. 일종의 커뮤니티이다. 다른 사람이나 다른 프로젝트를

# ‘깃 스마트!’ 깃허브 사용자를 위한 20가지 필수 팁

Martin Heller | Infoworld

**깃**(Git)과 깃허브(GitHub)는 프로그래밍 세계에서 가장 널리 사용되는 분산 버전 관리 시스템(Distributed Version Control System)이다. 이를 이용하면 사용자가 코드를 탐색, 공유 그리고 개선할 수 있다. 깃에 대한 수십 가지의 시작 안내서가 나와 있고 깃허브 사용자는 GitHub.com을 새로 고칠 때마다 ‘팝업 팁’을 볼 수 있다. 그러나 깃과 깃허브를 이용해 더 스마트하게 일하고자 하는 개발자를 위한 유용한 팁을 찾는 것은 여전히 어렵다. 이 글을 쓴 이유이기도 하다. 처음 몇 단락은 깃 또는 깃허브에 익숙하지 않은 사람을 위한 배경 지식이다. 이후에 본격적으로 팁을 살펴본다.

깃은 원래 리눅스 커널 커뮤니티를 위한 분산 버전 관리 시스템으로, 지난 2005년에 리누스 토발즈가 개발했다. 이 글의 목적은 깃을 홍보하는 것이 아니므로 깃이 얼마나 빠르고 가벼우며 유연하고 인기가 많은지 장황하게 설명하지 않겠다. 그러나 깃 리포지토리(Repository, 짧게 ‘리포(Repo)’)를 복제할 때 자신의 컴퓨터로 단지 한 번에 한 브랜치(Branch)의 스냅샷(Snapshot)만 가져오는 것이 아니라 전체 버전 이력이 넘어온다는 것만 봐도 얼마나 유용한 툴인지 알 수 있다.

깃은 리눅스 커널 커뮤니티에 뿌리를 두고 있으므로 명령줄(Command-line) 도구로 시작했다. 원한다면, 여전히 깃 명령줄도 사용할 수 있지만 꼭 그럴 필요는 없다. 특히, 깃허브 온라인 서비스를 사용한다면, 윈도우나 맥에서 무료 깃허브 클라이언트를 내려받아 함께 사용하는 것이 편하다. 깃 명령줄은 어떤 호스트에서도 작동하고 맥과 리눅스 시스템 대부분에 이미 설치돼 있다.

명령줄을 사용할지 아니면 그래픽 사용자 인터페이스(GUI)를 가진 기본 클라이언트를 사용할지 여부는 오롯이 사용자의 결정이다. GUI를 선호한다면 윈도우 또는 맥용 깃허브 클라이언트 외에, 소스 트리(Source-Tree, 윈도우와 맥 지원, 무료), 토터스깃(TortoiseGit, 윈도우 전용, 무료), 그리고 깃박스(맥 전용, 14.99 달러)를 눈여겨볼 필요가 있다. 문서편집기나 내부에서 깃을 지원하는 통합개발환경(IDE)을 사용할 수도 있다.

## 깃/깃허브 팁 1 거의 모든 것을 복제하라

깃허브와 다른 공개 깃 리포지토리에는 사용자가 자신의 컴퓨터로 자유롭게 복제할 수 있는 흥미로운 프로젝트가 많다. 이렇게 사용해야 하는 이유가 무엇일까? 한 가지는 커밋 로그(Commit Log, 확정 로그) 주석 달기 스타일을 포함해서 코딩 방식, 관례, 그리고 본인이 관심 있는 언어로 작성된 도구에 대해 배울 수 있기 때문이다. 두 번째 이유는 해당 프



로젝트가 목표를 어떻게 달성했는가를 배우기 위함이다. 세 번째 이유는 라이선스가 그렇게 할 수 있게 허용하고 해당 프로젝트가 자신의 작업이나 제품에 유용하기 때문이다(그러나 라이선스를 재확인해서 나중에 발생할 수 있는 문제에 방지하는 것이 좋다).

매뉴얼 페이지의 git clone에 대한 정의는 다음과 같다:

*새로 생성한 디렉터리로의 저장소 복제는, 복제된 저장소에 있는 각 브랜치에 대한 리모트 트래킹 브랜치를 생성하고(git branch -r을 사용해서 볼 수 있음), 복제된 저장소의 현재 활성(Active) 브랜치에서 포크(Fork, 분기)된 초기(Initial) 브랜치를 생성하고 점검한다. 복제 이후 인수가 없는 평범한 git fetch 명령은 모든 리모트 트래킹 브랜치를 업데이트한다. 인수가 없는 git pull은 마스터 브랜치가 존재할 경우 추가로 리모트 마스터 브랜치를 현 마스터 브랜치에 병합(Merge)한다.*

### 깃/깃허브 팁 2 자주 풀(Pull)하라

깃을 사용하면서 (그리고 다른 어떤 버전 관리 시스템도 마찬가지로) 빠지는 가장 일반적인 실수가 파일을 동기화(Sync)하지 않고 내버려 두는 것이다. 따라서 종종 git pull을 실행해 저장소의 사본을 최신으로 유지하는 것이 좋다. 또한, 자신이 변경한 코드를 다른 사람의 수정사항과 쉽게 병합할 수 있다. 때로는 너무 쉬워서 자동으로 할 수 있다고 생각할 정도이다. 이 팁의 핵심은 자신의 프로젝트 상태를 항상 살펴보라는 것이다. 깃 클라이언트 대부분이 최신 상태를 유지하기 위해 업데이트가 필요한 시점을 자동으로 알려준다.

### 깃/깃허브 팁 3 조기에 그리고 자주 커밋하라

커밋은 프로젝트에 대한 정교한 업데이트로, 하나 또는 그 이상의 파일에 대한 한 가지 또는 그 이상의 변경사항을 포함한다. 완결된 작업 단위에 대한 하나의 기록으로, 논리적 완전체인 프로젝트에 적용하거나 제거할 수 있는 것으로 생각해도 좋다. 테스트 전이라도 완료한 모든 논리적 변경사항은 커밋하라. 참고로 커밋은 자신의 로컬 저장소에만 적용된다.

매뉴얼 페이지의 git commit에 대한 정의는 다음과 같다.

*변경사항을 정의한 사용자로부터의 로그 메시지와 함께 신규 커밋에 있는 인덱스(Index)의 현재 콘텐츠를 저장한다.*

### 깃/깃허브 팁 4 다른 사람에게 바라는 것처럼 자신의 커밋에도 주석을 달라

세상에는 두 종류의 코더가 존재한다. 자신의 커밋에 주석을 다는 사람, 그리고 그렇지 않은 사람이다. 필자는 여기에 약간의 결벽증이 있다는 것을 기꺼이 인정한다. 모든 커밋에 반드시 메시지를 넣도록 설정했으며, 몇 가지 좋지 않은 로그가 포함된 커밋이 도착하면 한밤중이라고 해도 짜증 나는 메시지를 보내 버린다. 만약 코드는 명확해야 하고, 인라인(In-line) 주석이 변경 로그(Change Log)보다 훨씬 더 중요하다고 생각한다면, 다른 사람의 저장소를 하나 임의로 복제해서, 코드 전체를 읽지 않고 게시된 최신 이슈 관련 커밋을 찾아보라. 주석이 얼마나 중요한지 바로 알 수 있을 것이다. 즉, 정확한 커밋 로그는

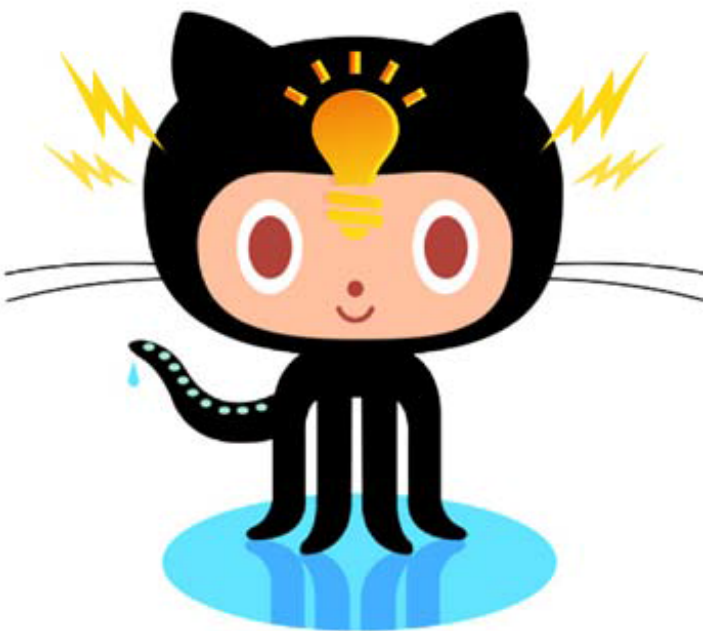
더할 나위 없이 좋은 것이다.

**깃/깃허브 팁 5 변경사항을 테스트했으면 올려라(Push)**

필자가 아는 최악의 깃 관련 실패 사례는 아웃소싱 회사가 서브 버전(Subversion)에서 전환했으나, 불행하게도 소속 개발자에게 분산 소스 관리와 중앙집중형 소스 관리의 차이를 교육하지 않은 것이었다. 약 1개월 뒤 이 프로젝트는 누구도 찾아낼 수 없을 것 같은 희한한 버그를 만들어냈다. 일일 현장 미팅에서 해당 버그 부분의 총괄 개발자는 “그 버그는 이미 2주 전에 수정했다”며 변경 사항을 풀(Pull)하지 않은 다른 개발자를 비난했다. 결국, 누군가가 문제를 밝혀냈고 모든 개발자에게 커밋을 올리는 방법과 시기를 공지했다. 요약하면, 로컬 빌드(Build)에서 커밋이 성공적으로 테스트를 통과할 때마다 풀하라는 것이다. 이후 이 회사는 통합한 제품을 빌드해서 배포할 때까지 이틀간의 병합 기간을 가졌다.

**깃/깃허브 팁 6 자유롭게 브랜치하라**

깃이 다른 버전 관리 시스템보다 좋은 점 중 하나는 병합용으로 사용할 최상의 공통 조상(common ancestor)을 자동으로 선택하므로, 병합작업이 대체로 잘 작동한다는 것이다. 따라서 소프트웨어 개발자는 자신의 프로젝트에서 자주 브랜치를 생성하는 것이 좋다. 개발팀 전원이 참석하는 괴로운 전략 미팅의 결과가 아니라 일상적으로 매일 발생하는 일이어야 한다. 브랜치 프로젝트가 완결되고 받아들여져 주(Main) 프로젝트로 옮겨갈 준비를 마치면, 병합 과정엔 아무런 문제도 없을 것이다. 특히, 버전 관리 시스템을 사용해 소스 코드를 관리하는 회사라면 더 그렇다. 물론 이런 과정은 쉽지 않다. 그러나 갑작스러운 버그로 인해 트렁크(Trunk) 프로젝트를 공개해야 할 때 고객이 우연히 미완성의 실험적 코드를 보게 되는 것보다는 훨씬 더 나은 상황이 아닌가?



**깃/깃허브 팁 7 조심스럽게 병합하라**

깃을 사용한 병합은 대개 잘 작동하지만, 아무 생각 없이 병합 작업을 한다면, 때로 어려움에 직면할 수 있다. 1단계는 커밋하지 않은 변경사항이 없는지 확인하는 것이다. git merge 매뉴얼을 보면 다음과 같은 설명이 있다.

외부 변경사항을 적용하기 전에 자신의 작업부터 정리하고 로컬에서 커밋하는 것이 좋다. 이렇게 해야 충돌이 있어도 피해를 최소화할 수 있다. 복잡한 충돌을 초래한 병합을 시도했다가 처음부터 다시 시작해야 하는 상황이라면 git merge -abort를 사용해서 복구할 수 있다.

사용자가 병합작업을 하는 데 GUI를 선호한다면 git merge의 뒤를 잇는 명령은 대개 git mergetool이다. 전통적인 방법이 더 익숙하다면 충돌하는 파일을 프로그램 편집기로 열어서 <<<<<<<<, =====, 그리고 >>>>>>>> 줄을 완전히 삭제하고, 수정된 파일을 저장한 후 수정한 각 파일을

'git add'하면 된다.

### 깃/깃허브 팁 8 다른 브랜치로 이동하기 전에 정리부터 하라

소프트웨어 개발자의 워크플로우는 1차원적인 경우가 드물다. 사용자는 억울한 버그를 알려오고, 관리자는 개발자가 작업하려고 했던 것이 아닌 다른 장애처리 티켓에 더 높은 우선순위를 부여한다. 때로는 개발자 스스로 마음을 바꿔 하려는 일을 바꿀 수도 있다.

이런 상황을 가정해 보자. 릴리즈를 위해 3개의 파일을 커밋했는데, 4번째 파일은 변경됐지만 작동하지 않는다(git status 명령어는 사용자가 자신의 현재 상태를 기억하지 못할 때 이 모든 상황을 알려준다). 동시에 갑자기 이 프로덕션 버전에 대한 버그 수정 작업을 해야 해서 재빨리 브랜치를 전환해야 한다. 문제는 개발자의 작업 디렉터리가 정리되지 않았고 더 하고 싶은 2시간 분량의 작업이 있다는 것이다. 이럴 때는 'git stash'를 입력하라. 모든 변경사항을 WIP(Work in Progress)에 저장하고, 깔끔해진 디렉터리에서 생산 브랜치로 전환할 수 있다. 수정 작업을 마치면 'git stash apply' 명령으로 원래 있던 위치로 돌아올 수 있다.

### 깃/깃허브 팁 9 gist를 사용해 스니펫과 페이스트를 공유하라

깃허브의 지스트(gist, 공유된 스니펫(Snippet, 코드 조각))는 깃을 사용한다. 모든 지스트는 깃 저장소이며, 깃허브 지스트는 더 쉽게 지스트를 공유할 수 있게 해준다. 사용자는 지스트에서 주제, 프로그래밍 언어, 포크된 상태, 그리고 공유 상태별로 검색할 수 있다. 비밀 지스트(Secret gist)를 생성해 URL로 공유할 수도 있다.

### 깃/깃허브 팁 10 익스플로어 깃허브

수많은 흥미로운 오픈소스 프로젝트가 깃허브에 저장소를 가지고 있다. 이 중 하나가 프로젝트를 찾는 브라우징 인터페이스인 익스플로어 깃허브(Explore GitHub)이다. 프로젝트의 이름 몇 글자를 검색 상자에 입력해 찾을 수 있다. 예를 들어, 주요 오픈소스 자바스크립트 프레임워크 중 3가지를 찾으려면 jq 또는 back 혹은 ang를 입력하면 된다.

### 깃/깃허브 팁 11 오픈소스 프로젝트에 기여하라

오픈소스 프로젝트를 둘러보는 것 외에 직접 기여해보는 것은 어떨까? 생각보다 어렵지 않고 많은 것을 배울 수 있다. 예를 들면, jquery/jquery(j쿼리 코어)를 복제해, README.MD를 훑어보면 상단에 다음과 같은 내용이 나온다.

*오픈소스 개발 정신에 따라 j쿼리는 늘 커뮤니티 코드 기부를 권장하고 있다. 시작하는 데 도움이 되지 않더라도 코드 작성에 들어가기 전에, 기여 지침을 반드시 빠짐없이 읽어보기 바란다.*

그리고 뒤이어 3개의 링크가 나오는데, 특히 이 중 첫 번째 링크는 사용자가 빠르게 공헌을 시작할 수 있도록 도와준다. 물론 모든 오픈소스 프로젝트가 이처럼 명확하게 방법을 제시하는 것은 아니다. 그러나 나름대로 다들 노력은 하고 있다.

이를 위해서는 컨트리뷰터(Contributor, 기여자)와 커미터(Committer, 소스코드 개발자)의 차이점을 아는 것이 중요하다. 컨트리뷰터는 필수 합의서(Required Agreement)에 서명하고 해당 프로젝트에 기여하는 사람이다. 커미터는 해당 프로젝트 저장소에 제공된 기여사항을 실제로 커밋할 수 있는 권한을 갖고 있다. 커미터가 기여사항을 테스트하는 동안 지연이 발생하고 마스터 브랜치가 묶일 수 있으므로, 풀 요청(Pull Request)을 발송하기 전에 변경사항을 다른 브랜치에 복사하는 것이 좋다.

### 깃/깃허브 팁 12 깃을 지원하는 편집기나 IDE를 사용하라

신나게 편집하고 체크인을 하러 가보니 누군가가 같은 코드에 작업하고 있었다면 어떤 기분일까. 이런 좌절을 피하려면 깃과 통합하는 편집기 혹은 통합개발환경(IDE)을 사용하는 것이 좋다. 현재 작업 중인 코드에 대한 새로운 커밋이 있는지, 풀 해야 하는지, 신규 커밋이 무엇을 하려는 지 등을 알려준다.

### 깃/깃허브 팁 13 저장소를 포크하라

저장소 포크(fork)란 어떤 저장소에 대해 자신이 코드를 작성할 수 있는 서버 사본을 생성하는 것을 의미한다. 일종의 갈림길을 만드는 셈이다. 앞서 어떤 저장소에 대해 우리의 클라이언트 사본을 만들기 위해 저장소를 복제하는 팁을 살펴봤다. 이 저장소가 커밋 권한이 없는 공개 저장소라면, 작업한 변경사항을 기여할 수 있는 가장 쉬운 방법은 원래 깃허브 프로젝트에 있는 포크 단추를 통해 우리 포크에 변경사항을 커밋하는 것이다. 그 후에 포크한 저장소 소유자에게 풀 요청을 하면 된다. 그들이 테스트한 후 기여사항을 사용할 수 있게 할 것이다. 처음에는 다소 복잡한 이야기처럼 들리겠지만 몇 번 해보면 간단한 작업이라는 것을 알 수 있다.

### 깃/깃허브 팁 14 프로젝트를 워치하라

어떤 프로젝트를 포크할 때, 원래(Upstream) 프로젝트에서 어떤 일이 벌어지고 있는지 알고 싶을 것이다. 그렇다면, 해당 저장소를 워치(Watch)하라. 업데이트 알림이 본인을 성가시게 한다면 언워치(Unwatch)하면 된다. 현재 하는 작업에 영향을 주는 변경사항이 있으면 원래 프로젝트의 커밋을 가져와 병합하면 된다.

### 깃/깃허브 팁 15 친구를 팔로우하라

깃허브는 깃허브 구성원을 팔로우하라고 적극 권고하고 있다. 흥미 있는 프로젝트 관련 개발자를 팔로우하면 또 다른 흥미로운 프로젝트를 발견할 수 있다. 예를 들어 필자는 깃허브에서 dmethvin을 팔로우했다. PC 테크 저널 출신이고 현재는 j쿼리 재단의 대표이다.

### 깃/깃허브 팁 16 풀 요청을 발송하라

13번째 팁에서 우리는 깃허브 저장소 포크 작업에 대해 살펴봤다. 원래 저장소(자신의 코드를 작성하기 위해 포크했던 저장소)에 본인의 변경사항 일부 또는 전부를 포함하려면 [깃허브 가이드](#)에 따라 풀 요청을 하면 된다.

**깃/깃허브 팁 17 이슈를 제기하고 해결하라**

모든 소프트웨어에는 버그가 있다. 그래서 많은 개발자가 별도의 소프트웨어 버그 추적 시스템을 사용하지만, 깃허브의 이슈(Issue) 기능도 꽤 유용하다. 이슈를 발행해 프로젝트에 기여할 수 있고, 이를 해결해 더 실질적인 도움이 될 수도 있다.


**깃/깃허브 팁 18 유익한 리드미(README) 페이지를 작성하라**

11번째 팁에서 프로젝트에 대해 알아보기 위해 jquery/jquery의 리드미(README) 페이지를 살펴봤다. 이번에는 자신의 프로젝트에 대해 충실한 안내 페이지를 만들어보자. 머지않아 진가를 확인할 것이다. 개인적으로 안내 페이지의 중요성은 필자가 프로그래밍에 눈을 뜬 1960년 이후 아무리 강조해도 지나치지 않다고 생각한다.

**깃/깃허브 팁 19 마크다운(Markdown)을 사용하라**

초기 리드미 파일은 사실상 그냥 텍스트였다. 그러나 현재의 README 파일 형식 표준은 마크다운 특히, 깃허브 취향의 마크다운이다. HTML 포맷의 README 파일을 사용하던 시절도 있었지만 지금은 거의 사라지는 추세이다.

**깃/깃허브 팁 20 오래된 저장소를 깃으로 변환하라**

여기서 살펴본 모든 팁 중에서 이번 팁이 기술적으로나 정치적으로 가장 어려울 것이다. 특히 프로그래머는 천성적으로 자신이 쓰는 도구에 대해 보수적이기 때문에 이런 변화는 '정치적으로' 어렵다. 대안은 교육이다. 결국, 배우는 수밖에 없다. 수만 줄의 코드, 수천수만의 주석, 그리고 수천 개의 태그(Tag)가 붙어있는 커다란, 구형 저장소를 변환하는 것은 엄청난 작업이고 기술적으로도 까다롭다. 필자는 대형 아마존 EC2 인스턴스 상에서만 변환할 수 있었을 10년 묵은 버전 관리 시스템 저장소를 가지고 있었는데, 이를 깃으로 변환하는 데만 며칠이 걸렸다. 서버 버전에서 변환하는 경우라면 svn2git을 사용해보라. 버전 관리 시스템에서 변환한다면, git -cvsimport와 cvs2git도 유용할 것이다. 



# 월마트, 오픈소스로 관료주의를 몰아내다

Clint Boulton | CIO



**월**마트(Wal-Mart)의 소프트웨어 엔지니어를 만나면 오픈소스 기술이 그들의 프로젝트에 실제로 어떤 영향을 주었는지 솔직한 이야기를 들을 수 있다. 이 거대 유통업체는 전자상거래, 검색, 모바일 결제, 기타 디지털 기능을 서비스하기 위해 코드를 공유하는 오픈소스 정신을 받아들였다. ‘엔터프라이즈 소스(Enterprise Source)’, ‘내부 오픈소스’, ‘이너소스(Innersource)’라고 불리는 이런 흐름은, 전통적 오픈소스의 파생물이다. 오토데스크나 캐피탈 원(Capital One), 페이팔, 블룸버그 등 다른 기업에서도 사례를 찾을 수 있다.

월마트의 글로벌 전자상거래 CTO 겸 월마트랩스(@Walmart-Labs)의 책임자인 제레미 킹은 “내부 오픈소스를 더 활용할수록 더 유연하고 빠르게 코드를 공개할 수 있다. 우리가 오픈소스를 선호하는 것은 더 빠르고 저렴하며 더 좋은 방법이기 때문이다. 동시에 오

픈소스 커뮤니티에도 기여하고 싶다. 그것이 사회를 위해 옳은 일이기 때문이다”라고 말했다.

최근 구글, 페이스북을 비롯해 여러 실리콘 밸리의 기업이 대규모로 코드를 공개하면서 이제 오픈소스는 일부 기업이 아닌 거의 업계 공통의 흐름으로 자리 잡았다. 월마트는 자체적으로 이 모델을 도입해 그 성과물을 외부에 공유하고 있다. 다양한 클라우드 서비스를 테스트하고 전환할 수 있는 플랫폼인 ‘원옵스(OneOps)’를 공개한 것이 대표적이다. 이는 하이브리드 클라우드 모델을 도입하려는 모든 기업에 도움이 됐다.

## 이너소스는 관료제 개선에 도움

이너소스는 기본적으로 방화벽 안에서 개발해 엔지니어링 그룹 간에 공유하는 오픈소스 소프트웨어이다. 이런 방식은 직원이 수천 명에 달해 관료주의 문화를 없애기 힘든 대기업 조직에서 특히 유용하다. 월마트의 IT팀 규모는 상당하다. 랩스에서만 매달 3만 줄의 코드를 생산한다. 그러나 여느 기업처럼 단일 기능 개발을 위해 1,000명을 투입하지는 않는다. 랩스는 100개의 팀에 각각 10~20명씩 전략적으로 배치한다. 이들 엔지니어는 데브옵스(DevOps) 방식에 따라 코드를 작성, 테스트해 공개한다. 그래서 킹은 월마트를 ‘세계에서 가장 큰 스타트업’이라고 표현한다.

킹은 “물론 개발자가 지속적인 통합/연속 배치 방식으로 코드를 만든다고 해도 다른 팀이 대기 중인 프로젝트가 너무 많으면 병목 현상이 발생할 수 있다. 그래서 우리는 엔지니




어에게 다른 팀의 코드를 작성하고 배포하는 자율성을 허용했다. 이를 통해 병목을 줄일 수 있었다”라고 말했다. 예를 들어 결제 게이트웨이(Gateway)에 새로 연결하는 작업이 있는데 이를 지원하는 팀에 5개 프로젝트가 걸려 있다고 가정해 보자. 이때 해당 엔지니어는 자신의 연결 소스를 작성한 후 검토, 승인되도록 할 수 있다(물론 기준에 미치지 못하면 거부될 수 있다). 아니면 쇼핑 카트/체크아웃(Checkout) 개발팀 구성원이 새로운 운송 경로를 개발하는 다른 팀의 구성원이 될 수도 있다. 이때 최종 권한을 갖는 해당 팀이 코드를 테스트하고 검토해 코드 저장소 ‘깃허브(GitHub)’를 통해 공유한다. 킹은 “모든 코드가 이런 식으로 처리되는 것이 이상적이다”라고 말했다.

하지만 결과적으로 이너소스는 모든 랩스 프로젝트에 완벽한 접근방식이 아니었다. 킹은 “이 플랫폼은 매우 복잡해 박사급 데이터 공학자와 다른 전문가 정도만 소프트웨어를 다듬을 수 있었다. 예를 들면 검색팀 업무의 경우 그 구성원이 아닌 사람이 검색 알고리즘에 기여하기가 어려웠다. 더 많은 사람이 기여했으면 좋겠지만 너무 복잡했다. 그냥 깃허브에 들어가서 기여할 수 있는 것이 아니었다”라고 말했다.

### 프로그래밍 측면에서 오픈소스를 선호

현재 월마트는 ‘깃허브 엔터프라이즈(Github Enterprise)’를 이용해 이너소스를 운영한다. 이는 오픈소스 깃허브와 비슷하지만 한 가지 차이점이 있다. 바로 기업 방화벽 안에서 관리한다는 점이다. 덕분에 IT 임원은 마음의 평화를 얻었다. 특히 지극히 내부적인 검색 알고리즘의 경우에 더욱 그랬다. 현재 월마트가 사용하는 깃허브 엔터프라이즈는 원오피스 공개 당시 사용했던 버전의 깃허브와 다르다. 킹의 부서는 개발자가 모바일 웹과 모바일 앱용 웹페이지를 더 신속하게 개발할 수 있는 툴인 ‘일렉트로드(Electrode)’도 공개할 예정이다.

월마트의 오픈소스 사랑은 지난 2011년 킹이 이베이(eBay)에서 월마트로 이직하면서 본격화됐다. 그는 랩스를 더 민첩하게 운영하기 위해 10년 이상 된 많은 레거시 소프트웨어를 카산드라(Cassandra) 기술 등의 NoSQL 데이터베이스, 메시징 버스를 위한 카프카(Kafka) 대기 행렬, 서비스지향 자바스크립트 환경인 Node.js 등 오픈소스 프로그래밍 툴로 교체했다.

그 결과 현재 월마트는 오픈스택(OpenStack)을 이용해 재구축한 클라우드 플랫폼을 최대 규모로 운영하는 기업 중 하나가 됐다. 이런 성과의 이면에는 오픈소스를 공개적으로 적극 활용한 것이 한몫했다. 이뿐만이 아니다. 월마트는 오픈소스를 도입한 후 최고 수준의 기술 인재를 영입할 수 있었다. 킹은 “원했던 유망주의 절반은 애플, 구글, 페이스북, 링크드인 등 유명 기업의 입사 제안을 동시에 받고 있었지만 70% 정도가 우릴 택했다. 월마트라는 브랜드와 도전적인 업무에 대한 약속, 오픈소스 덕분이다. 이들은 안정적인 회사이면서도 스타트업처럼 일할 수 있는 직장을 원했다”라고 말했다. 

# 기업의 미래를 보는 특별한 기준 '깃허브 수용도'

Steven Max Patterson | Network World

**최** 근 페이스북과 구글의 모기업 알파벳(Alphabet)은 놀라운 모바일 광고 수익 성장률을 기록했다. 그러나 그 이면에는 충격적인 반전이 자리 잡고 있다. 많은 기업이 소비자와 제2, 제3의 관계를 쌓기 위해 광고 지출을 높이고 있다는 것이다. 즉 자사의 앱으로 고객과 1차 관계를 수립하는 기업이 거의 없다는 의미이다. 실제로 아마존, 에어비앤비(Airbnb), 우버(Uber) 등의 기업은 모바일 중심적 클라우드 플랫폼을 성공적으로 구현해 비즈니스 지형을 바꾸고 있지만, 이들 극소수 기업을 빼면 다른 성공 사례를 찾기가 쉽지 않다. 그래서 승자는 이러한 변화를 '디지털 혁신'이라 부르지만, 패자에게는 '디지털 파괴'라는 말이 더 어울린다.

지난 10년 동안 출판, 유통, 통신, 여행 및 숙박, 교통 등의 산업에 끼친 디지털 파괴가 시사하는 바는 혁신하지 못하면 존속할 수도 없다는 것이다. 반면 성공적으로 변혁을 달성한 기업은 범용화된 하드웨어와 오픈소스 소프트웨어를 이용해 모바일 주도적 클라우드를 구축하고 각자의 비즈니스를 영위하고 있다. 알파벳과 페이스북이 온몸으로 웅변하는 사실이다.

최근 깃허브(Github)의 제품 관리 책임자 카쿨 스리바스타바는 일부 기업이 깃허브와 개방형 혁신 모델을 통해 어떻게 혁신하고 사업을 변화시키는지 발표했다. 그에 따르면 디지털 혁신은 기업이 어느 업종에 속해있는지에 상관없이 스스로가 소프트웨어 기업임을 인식하는 데에서 시작된다. 오늘날의 기업은 혁신 한계를 빠르게 넘어서기 위해 오픈소스 소프트웨어로 눈을 돌리고 있다. 그리고 깃허브는 이 오픈소스 프로젝트 저장소로서 독점적인 지위를 보유한 덕분에 이런 혁신의 중심에서 있다. 이는 깃허브가 기업의 변화를 관찰할 수 있는 특별한 기준점이라는 의미이기도 하다.

스리바스타바에 따르면 월마트(Walmart) 등의 대기업이건 수많은 신생기업이건 모두 깃허브의 소스코드 저장소가 제공하는 협업 방식이 필요하다. 또 각자의 오픈소스 전략을 위해 깃허브에 도움을 구하고 있다. 그는 개방형 혁신 모델을 이용한 엔드 투 엔드 핵심 소프트웨어 역량을 구축하는 과정이 다음과 같이 단계적으로 형성된다고 설명했다. 우선 일반적인 질문에서 시작된다. 우리에게 중요한 오픈소스 커뮤니티는 무엇인가? 둘째, 우리는 어디에 가입해 기여해야 할까? 깃허브의 오픈소스 책임자인 브랜드 키퍼스는 이런 고민이 빠른 속도로 기업의 핵심 과제로 자리 잡고 있다고 분석했다.

키퍼스는 페이스북의 리액트(React) 오픈소스 프레임워크를 예로 들었다. 자바스크립트 개발자가 혁신적인 웹 애플리케이션을 개발할 수 있도록 지원한다. 그에 따르면 페이스북



의 웹 및 데스크톱 혁신의 성공 여부는 최고의 자바스크립트 개발자를 확보하느냐에 달려 있다. 깃허브의 리액트 오픈소스 개발은 페이스북 내외부의 크고 다양한 개발자 커뮤니티와 협업해 리액트를 개발하는 것 이상의 역할을 했다. 바로 깃허브를 통해 뛰어난 페이스북 개발자 채용 후보를 찾을 수 있었다.

실제로 깃허브는 IT 기업의 인력 수급과 관련해 중심적인 역할을 하고 있다. 인바운드 마케팅 서비스형 플랫폼 기업 허브스팟(Hubspot)의 CPO(Chief People Officer) 짐 오네일은 “우리 팀에 누군가를 새로 합류시킬 때 링크드인 프로필보다 그의 깃허브 계정에서 더 많은 것을 확인할 수 있다. 기존 인력 역시 오픈소스 소프트웨어와 범용 인프라를 이용해 서비스로 서플랫폼(PaaS)을 구축하는 과정에서 혁신자적 관점을 얻을 수 있었다”라고 말했다.

전직 CIO이자 디지털 인재를 채용하고 유지하는 HR 부문을 책임지고 있는 오네일도 이 의견에 동의했다. 혁신은 새로운 소프트웨어 개발 기술을 갖춘 최고 인재 확보에 달려 있다는 것이다. 혁신에 집중하려면 오픈소스 커뮤니티의 관찰자로 은둔할 수 없다. 일어나서 커뮤니티의 리더가 돼야 한다. 그렇지 않으면 재능 있는 개발자는 기업에 합류하지 않거나 오픈소스 프로젝트에 기여하지 않을 것이다. 스리바스타바는 “기업이 각자의 목표를 달성하기 위해 오픈소스 커뮤니티 구축에 관해 진지하게 생각하기 시작했다”라고 말했다.

깃허브에서의 협업은 기업 내/외부에 영향을 끼친다. 재능이 있는 개발자는 평생 공부한다. 이들은 자신의 개인적인 시간을 투자해 새로운 언어로 프로그램을 개발하는 방법과 새로운 프레임 워크, 방법론을 탐구한다. 이들은 그 과정에서 깃허브와 만나게 된다. 개방형 혁신을 도입하는 기업은 독점 개발을 위한 배타적인 저장소와 오픈소스 커뮤니티를 모두 포용하는 플랫폼이 필요하다. 그리고 개발자의 관심이 이 오픈소스 커뮤니티를 포용하는 협업적인 기업 저장소와 만나 기업의 깃허브 도입을 자극하게 된다.

사유 소프트웨어 기업인 애플과 마이크로소프트의 최근 행보를 봐도 개방형 혁신의 중요성을 알 수 있다. 키퍼스에 따르면 애플은 독립적인 개발자가 혁신적인 앱을 개발하면서, 동시에 애플과의 연계성을 높이는 2가지 목표를 달성하기 위해 스위프트(Swift) 프로그래밍 언어를 오픈소스화했다. 마이크로소프트는 새로운 오픈소스 소프트웨어를 개발하고 닷넷(.Net)과 비주얼 스튜디오(Visual Studio) 등의 자체 기술을 오픈소스화해 새로운 고객을 유치하고 애저(Azure) 클라우드로 유도하고 있다.

스리바스타바는 기업 고객인 월마트에 관해서도 이야기했다. 이 기업은 내부적으로 전자 상거래 인프라를 운영해 기업이 아마존의 프라이빗 클라우드 없이 클라우드 앱을 손쉽게 개발할 수 있도록 하는 클라우드 관리 툴 ‘원옵스(OneOps)’를 오픈소스화했다. 이밖에 독일에 있는 존 디어(John Deere)의 혁신 연구소 같은 다른 사례도 있다. 현재 기업이 겪는 디지털 혁신과 디지털 파괴 사이의 갈등은 아직 초기 단계이다. 그러나 분명한 것은 혁신에 성공하는 기업과 파괴의 피해자가 되는 기업은 깃허브와 오픈소스/개방형 혁신 커뮤니티에의 참여 여부에 따라 극명하게 갈릴 것이라는 점이다. 