<http://insanehong.kr/post/gitlab5.1-install-guide/>

<https://eastskykang.wordpress.com/2015/01/19/ubuntu-14-04-lts-%EC%84%9C%EB%B2%84%EC%97%90-gitlab-%EC%84%A4%EC%B9%98/>

**Ubuntu 14.04 LTS 서버에 GitLab 설치**

[January 19, 2015February 8, 2015](https://eastskykang.wordpress.com/2015/01/19/ubuntu-14-04-lts-%ec%84%9c%eb%b2%84%ec%97%90-gitlab-%ec%84%a4%ec%b9%98/)[east0822](https://eastskykang.wordpress.com/author/east0822/)

GitLab은 서버에 설치해서 Git Remote 서버로 쓸수 있게 해주는 동시에, 웹브라우저로 접속하여 Git 관리를 할 수 있게 해주는 툴이다. [GitLab.com](https://gitlab.com/users/sign_in)에서 호스팅해주는 서비스를 사용해도 되지만 오픈소스로 배포되고 있어서 직접 서버에 구성해서 사용할 수 있다.

물론 Git repository 관리를 위해서 [GitHub](https://github.com/)를 쓰거나 GitLab.com 의 서비스를 사용해도 되지만 GitHub 같은 경우에는 프로젝트를 비공개로 설정하려면 유료 계정으로 전환해야 한다.

만약 무료로 개인용 서버에 설치해서 비공개 repository를 운영하거나, 팀단위에서 cooperation을 하기 위해 Git Remote를 운영하고 싶다면 오픈소스로 배포되는 [GitLab Community Edition](https://gitlab.com/gitlab-org/gitlab-ce)이 좋은 옵션이 될 수 있다.

이 포스팅은 Ubuntu 14.04 LTS에 GitLab 7.6 버젼을 설치하는 방법에 대한 내용이다.:

**GitLab**

* Web에서 Browser로 접속할 수 있는 Git 관리 툴.
* Ruby 로 개발됨.
* 오픈소스로 프로젝트를 공개하여, 개인 서버에도 구성이 가능함. (필자는 서버 컴퓨터에서 VMWare로 Linux 가상머신을 만들고 GitLab을 설치했다. 과정은 크게 다르지 않다.)
* 자세한 내용은 [GitLab – About GitLab](https://about.gitlab.com/) 참조

**GitLab 설치**

* GitLab 홈페이지에서 다운로드 및 설치 메뉴얼을 참조  
  — Omnibus 배포판을 통해 Community Edition을 설치할 수 있다.

[GitLab – Download and Installation](https://about.gitlab.com/downloads/)  
[GitLab – Omnibus GitLab Documentation](https://gitlab.com/gitlab-org/omnibus-gitlab/blob/master/README.md)

* Ubuntu 14.04 에서는 아래 명령어로 설치 가능하다.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **wget https://downloads-packages.s3.amazonaws.com/ubuntu-14.04/gitlab\_7.6.2-omnibus.5.3.0.ci.1-1\_amd64.deb**  **sudo apt-get install openssh-server**  **sudo apt-get install postfix** # Select 'Internet Site', using sendmail instead also works, exim has problems  **sudo dpkg -i gitlab\_7.6.2-omnibus.5.3.0.ci.1-1\_amd64.deb**    # 위 명령어를 실행하면 email 설정 화면으로 넘어간다. Internet site 로 설정하고 관리자의 email 을 입력하면 된다.    **sudo dpkg -i gitlab\_7.6.2-omnibus.5.3.0.ci.1-1\_amd64.deb** |

* GitLab 설정을 한다. [GitLab – Omnibus Configuration](https://gitlab.com/gitlab-org/omnibus-gitlab/blob/master/README.md#configuration) 참조  
  — 설정 파일은 /etc/gitlab/gitlab.rb

— sudo vim /etc/gitlab/gitlab.rb로 수정. (vim editor 가 사전에 설치되어 있는지 확인할 것.)  
— 포트를 바꾸거나 GitLab의 주소를 바꾸고 싶으면 **interal\_api\_url** 을 입력해준다.

|  |  |
| --- | --- |
| 1 | gitlab\_rails['interal\_api\_url'] = 'http://IP:PORT'; |

* 도메인을 연결하거나 외부에서 접속할때의 주소를 지정하고 싶으면 **exterunal\_url**을 입력해준다.

|  |  |
| --- | --- |
| 1 | external\_url '[http://address](http://address/)'; |

* 설정이 완료되면 터미널에서 sudo gitlab-ctl reconfigure 명령어로 설치가 제대로 됐는지 확인한다.

sudo gitlab-ctl status 를 입력했을 때 아래처럼 뜨면 정상

|  |  |
| --- | --- |
| 1  2  3  4  5 | run: nginx: (pid 972) 7s; run: log: (pid 971) 7s  run: postgresql: (pid 962) 7s; run: log: (pid 959) 7s  run: redis: (pid 964) 7s; run: log: (pid 963) 7s  run: sidekiq: (pid 967) 7s; run: log: (pid 966) 7s  run: unicorn: (pid 961) 7s; run: log: (pid 960) 7s |

* 클라우드가 아니라 개인 서버나 PC에 GitLab을 구성하는 경우 **방화벽**이 문제되는 경우가 있다.
* 해당 포트의 방화벽을 해제해준다.  
  — GitLab은 80 포트를 사용하여 웹호스팅한다.

**VMWare에 GitLab 구성하는 경우**

<li><strong>VMware의 Virtual Network Editor 에서 포트 포워딩 해줘야 함.</strong></li>

1. 도구막대의 Edit-Virtual Network Editor  
2. NAT 을 선택한 후 NAT Settings 버튼을 누른다.  
3. Port Forwarding 에서 Add.  
4. Host Port 는 Guest와 연결될 Host의 포트 (여기선 8080으로 설정함.)  
5. Guest Port 는 Guest에 설치된 GitLab이 사용하는 포트. 별다른 설정을 하지 않았으면 80.  
6. IP는 Guest의 IP 입력. (여기선 192.168.46.128)  
7. Description 입력. (e.g) GitLab)  
8. Host의 방화벽 설정에서 Port Forwarding 에서 입력한 Host Port 를 열어준다. (자세한 내용은 아래 링크 참조)

링크 참조 [MS Windows – Open Port Firewall](http://windows.microsoft.com/ko-kr/windows/open-port-windows-firewall#1TC=windows-7)

* 학내 네트워크 정책으로 인해서 **GitLab 유저는 email 을 반드시 @snu.ac.kr 도메인으로 설정**해야 confirmation email 등 GitLab에서 보내는 email 을 받을 수 있다.
* 학내 네트워크 정책으로 80 포트가 방화벽으로 인해 학외에서는 접속이 불가. 따라서 8080 포트로 우회하도록 포트를 8080으로 설정했다.

**GitLab 사용**

* 관리자는 Gitlab에 멤버를 초대할 수 있음
* 관리자에게 계정 생성을 요청하거나 GitLab 에 접속하여 계정 생성
* GitLab Web에서 관리자의 초대 없이 사용자가 직접 계정을 생성할 수 있도록 하기 위해선 GitLab 설정 (서버에서 sudo vim /etc/gitlab/gitlab.rb로 설정 파일 수정)에서 아래를 입력.

|  |  |
| --- | --- |
| 1 | gitlab\_rails['gitlab\_signup\_enabled'] = true |

* 계정을 생성하면 생성한 계정의 이메일로 확인 메일이 발송된다. 확인 메일의 링크를 클릭해야 계정 생성 완료.
* 허용된 계정으로 Gitlab 에 git commit 이나 push 등의 명령을 하려면 ssh key 를 등록해야 함. 자세한 내용은 프로필의 ssh keys의 “generate it” 링크 확인.

<http://sanghyukchun.github.io/80/>

**Gitlab을 깔자**

버전관리의 중요성은 몇 번을 강조해도 부족하지 않다. 버전관리를 하기 위한 여러 도구들이 존재하지만, 난 분산 버전 관리 (DVCS) 를 좋아한다. Mercurial 로 처음 버전관리를 공부했고, 지금은 주로 Github에 소스를 올리기 때문에 git 을 많이 쓰고 있다. 콘솔에서는 큰 차이가 있지만, 정작 둘 다 [SourceTree](http://www.sourcetreeapp.com/)로만 관리하기 때문에.. 예전에는 맥버전만 있었지만, 이제는 윈도우 버전도 생겨서 다른 사람들도 용이하게 쓸 수 있다. 이런 이유로 최근에는 거의 모든 레포지토리를 git 레포지토리로 만들게 되는데, 연구실에서 내가 쓰고 있는 논문이나 프로젝트용 코드 같은 경우는 공개된 레포지토리로 올려도 곤란하고, 연구실 단위로 관리를 해야할 필요를 느껴서 [GitLab](https://about.gitlab.com/)이라는 놈을 깔아보기로 헀다.

Gitlab을 한 마디로 요약하자면 ‘개인용 github’ 라고 할 수 있다. 코드를 훑어보니 rails 기반에 nginx를 서버로 사용하고 있어서 내가 친숙한 환경이기도 해서 좋더라. 늘 깔아야지 깔아야지했는데 이게 생각보다 설치가 머리가 아파서.. 나중에 시간이 넉넉해지면 설치하려 했으나 어쩌다보니 갑작스럽게 설치를 하게 되었다.

내 설치환경은 Ubuntu 14.04, [다운로드 페이지](https://about.gitlab.com/downloads/)에서 ubuntu 14.04를 선택하는 아래와 같은 명령어를 실행하면 알아서 옴니버스 버전을 깔아준다고 한다.

|  |  |
| --- | --- |
|  | **wget https://downloads-packages.s3.amazonaws.com/ubuntu-14.04/gitlab\_7.4.3-omnibus.5.1.0.ci-1\_amd64.deb**  **sudo apt-get install openssh-server**  **sudo apt-get install postfix** # Select 'Internet Site', using sendmail instead also works, exim has problems  **sudo dpkg -i gitlab\_7.4.3-omnibus.5.1.0.ci-1\_amd64.deb** |

난 openssh-server, postfix는 깔려있으니 생략했다.

다음으로는 **sudo vim /etc/gitlab/gitlab.rb** 를 실행해 **gitlab.rb** 를 수정해야한다. 다시 한 번 말하지만, gitlab은 rails로 돌아가기 떄문에 rails setting 을 해줘야한다. 원래 레일즈 프로젝트의 설정을 바꾸기 위해서는 yml 파일이나 다른 rb 파일들을 직접 수정해주어야 하는데, gitlab 옴니버스 버전에서는 친절하게 이 루비 파일 하나만 바꾸면, 알아서 yml 등을 generate해준다. 아 편하고 좋다! 라고 생각했지만 이것이 그 모든 재앙의 시작이었다..

편하게 해주려고 만든 ruby setting 파일이 왜 문제가 되었느냐, 사실 이 대부분은 문제가 생기지 않는다. 예를 들어서 **www.example.com** 이라는 도메인을 가지고 있고, gitlab 의 접속 경로를 **gitlab.example.com** 으로 사용한다면 옴니버스 버전을 바로 사용하면 된다. 하지만 내가 사용하는 서버는 학교 도메인에서 서브도메인을 받아서 사용하기 때문에 sanghyuk.kaist.ac.kr 이런 식의 도메인을 가지고 있다. 따라서 위와 같은 경로를 취하게 되면 gitlab.sanghyuk.kaist.ac.kr 이라는 기괴한 경로가 생기게 되고, 당연하지만 이런 경로는 허용되지 않는다. 따라서 sanghyuk.kaist.ac.kr/gitlab 같은 relative domain을 사용해야한다. nginx에서 이런 서브 도메인을 루트로 삼는 것은 규칙에 위반되지만 사용하는 것이 가능하긴하다. rails에서도 root url을 relative url로 바꾸고 하면 돌릴 수 있지만.. 이건 내가 rails파일을 직접 바꿀 때 얘기였다.

옴니버스 버전에서 이런저런 삽질을 하다가 찾아낸 이슈.. [옴니버전에서는 Relative URL root를 지원할 계획이 없다](https://gitlab.com/gitlab-org/omnibus-gitlab/issues/238) 아… gitlab.rb 에서 삽질을 하고 있었는데 다 쓸데 없는 짓에 불과했던 것이다.

그래서 relative url은 포기하고 다른 쪽으로 알아보니 포트를 바꿔서 접속을 하는 방법이 있더라. 이건 그나마 훨씬 할 만 헀다. **/etc/gitlab/gitlab.rb** 에서 다음과 같이 설정해준다.

|  |  |
| --- | --- |
|  | **external\_url = 'http://sanghyuk.kaist.ac.kr:1234'** |

아 참고로, 내가 깐 버전은 = 을 안쓰면 에러가 나서 내가 = 을 따로 넣어줬다. 옴니버스 버전별로 다른 모양

이렇게 하고 **sudo gitlab-ctl reconfigure** 을 실행시켜서 yml 등을 자동으로 generate시키기만 하면 문제 해결! … 이 아니었다. 아예 접속 자체가 되지를 않아서 이제 여기에서 다시 삽질을 시작했는데, 먼저 netstat으로 포트는 열려있나 봤다. 안타깝게도 아예 포트가 열려있지도 않았다. 돌아버리겠는건 **gitlab-ctl status** 에서는 잘 실행되는 것으로 나오는 것.

이제 또 한참 삽질을 하다가, 아예 nginx 조차 돌아가는 것 같지 않아서 nginx 세팅을 수동으로 뜯어고치기로 결정했다. 이게 상당히 위험한 짓인데, 나중에 내가 생각없이 또 reconfigure를 때리면 내가 고친 파일들이 새로운 파일들로 덮어씌일 것이기 때문이다. 때문에 이렇게 설정을 시작한다면 reconfigure 대신 **gitlab-ctl restart** 로 configure 파일을 다시 생성시키지 말고 레일즈랑 nginx 등만 내렸다가 올려야한다. 이제 세팅을 바꿔보자.

가장 먼저 nginx 설정을 찾아봤다. **/var/opt/gitlab/nginx/conf/gitlab-http.conf** 에 설정 파일이 있는데, listen과 server name이 엉망으로 되어있더라. 당연히 nginx가 설정이 잘못되었으니 서버에 접속도 못하고 포트도 안열려 있던 것. 이 둘을 제대로 바꿔주고 restart를 했다.

드디어 ‘페이지를 찾을 수 없습니다’ 창말고 다른 창을 볼 수 있었다. 그러나 여기에서 또 502 에러가 발생했는데, 아마도 내부 rails가 제대로 올라오지 않은 모양인가보다. 그래서 혹시 레일즈 세팅도 이상한가 싶어서 yml 파일을 찾아봤다.

**/var/opt/gitlab/gitlab-rails/etc/gitlab.yml** 을 열어봤더니 여기도 host랑 port가 엉망이었다. 이 부분을 바꿔주고 나서 다시 **sudo gitlab-ctl restart**

아 드디어 잘 실행된다. 기존에 쓰던 다른 레포지토리를 추가해주기 위해서 [remote](http://git-scm.com/book/ko/v1/Git%EC%9D%98-%EA%B8%B0%EC%B4%88-%EB%A6%AC%EB%AA%A8%ED%8A%B8-%EC%A0%80%EC%9E%A5%EC%86%8C)를 해주려고 보니 ssh key를 생성해서 넣어줘야 하더라. 이건 [이 글](http://git-scm.com/book/ko/v1/Git-%EC%84%9C%EB%B2%84-SSH-%EA%B3%B5%EA%B0%9C%ED%82%A4-%EB%A7%8C%EB%93%A4%EA%B8%B0)을 보면 된다. 이건 예전에 다 등록해뒀던거라서 금방금방했다. 조금 가지고 놀아보니 아직까지는 매우 만족스럽다.

이렇게 일단 주먹구구식으로 깃랩을 돌리는 것에는 성공했지만, 아직 reconfigure를 함부로 하면 안된다. 내가 바꾼 **/var/opt/gitlab/gitlab-rails/etc/gitlab.yml**, **/var/opt/gitlab/nginx/conf/gitlab-http.conf** 는 **gitlab-ctl reconfigure** 를 할 때 자동으로 generate 되는 파일이다. 따라서 내가 바꾼 설정이 저장이 되지 않기 때문에 함부로 reconfigure를 했다가는… 나중에 시간이 나면 이 부분을 gitlab.rb 만 바꿔서 수정할 수 있는지 알아보자.

**4.3 Git 서버 - SSH 공개키 만들기**

**SSH 공개키 만들기**

이미 말했듯이 많은 Git 서버들은 SSH 공개키로 인증한다. 공개키를 사용하려면 일단 공개키를 만들어야 한다. 공개키를 만드는 방법은 모든 운영체제가 비슷하다. 먼저 키가 있는지부터 확인하자. 사용자의 SSH 키들은 기본적으로 사용자의 ~/.ssh 디렉토리에 저장한다. 그래서 만약 디렉토리의 파일을 살펴보면 공개키가 있는지 확인할 수 있다:

$ cd ~/.ssh

$ ls

authorized\_keys2 id\_dsa known\_hosts

config id\_dsa.pub

something, something.pub이라는 형식으로 된 파일을 볼 수 있다. something은 보통 id\_dsa나 id\_rsa라고 돼 있다. 그중 .pub파일이 공개키이고 다른 파일은 개인키이다. 만약 이 파일이 없거나 .ssh 디렉토리도 없으면 ssh-keygen이라는 프로그램으로 키를 생성해야 한다. ssh-keygen 프로그램은 리눅스나 Mac의 SSH 패키지에 포함돼 있고 윈도는 MSysGit 패키지 안에 들어 있다:

$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/Users/schacon/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /Users/schacon/.ssh/id\_rsa.

Your public key has been saved in /Users/schacon/.ssh/id\_rsa.pub.

The key fingerprint is:

43:c5:5b:5f:b1:f1:50:43:ad:20:a6:92:6a:1f:9a:3a schacon@agadorlaptop.local

먼저 키를 어디에 저장할지 경로를(.ssh/id\_rsa) 입력하고 암호를 두 번 입력한다. 이때 암호를 비워두면 키를 사용할 때 암호를 묻지 않는다.

사용자는 그 다음에 자신의 공개기를 Git 서버 관리자에게 보내야 한다. 사용자는 .pub 파일의 내용을 복사하여 메일을 보내기만 하면 된다. 공개키는 아래와 같이 생겼다:

$ cat ~/.ssh/id\_rsa.pub

ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAklOUpkDHrfHY17SbrmTIpNLTGK9Tjom/BWDSU

GPl+nafzlHDTYW7hdI4yZ5ew18JH4JW9jbhUFrviQzM7xlELEVf4h9lFX5QVkbPppSwg0cda3

Pbv7kOdJ/MTyBlWXFCR+HAo3FXRitBqxiX1nKhXpHAZsMciLq8V6RjsNAQwdsdMFvSlVK/7XA

t3FaoJoAsncM1Q9x5+3V0Ww68/eIFmb1zuUFljQJKprrX88XypNDvjYNby6vw/Pb0rwert/En

mZ+AW4OZPnTPI89ZPmVMLuayrD2cE86Z/il8b+gw3r3+1nKatmIkjn2so1d01QraTlMqVSsbx

NrRFi9wrf+M7Q== schacon@agadorlaptop.local

다양한 운영 체제에서 SSH 키를 만드는 방법이 궁금하면 http://github.com/guides/providing-your-ssh-key에 있는 Github 설명서를 찾아보는 게 좋다.

우분투에 GitLab 설치

By [sjune](http://www.sjune.net/archives/author/admin) [2014/01/15](http://www.sjune.net/archives/1726) [Dev](http://www.sjune.net/archives/category/dev) [3 Comments](http://www.sjune.net/archives/1726#comments)

우분투(v13.10) 에 [GitLab](http://gitlab.org/)(v6.4-stable)을 설치해본다. 설치하는데 아래와 같은 구성요소들이 필요하며 순서대로 진행된다.

Components [[hide](http://www.sjune.net/archives/1726)]

* [1.Packages / Dependencies](http://www.sjune.net/archives/1726#1Packages_Dependencies)
* [2.Ruby](http://www.sjune.net/archives/1726#2Ruby)
* [3.System Users](http://www.sjune.net/archives/1726#3System_Users)
* [4.GitLab shell](http://www.sjune.net/archives/1726#4GitLab_shell)
* [5.Database](http://www.sjune.net/archives/1726#5Database)
* [6.GitLab](http://www.sjune.net/archives/1726#6GitLab)
  + [6.1. Clone the Source](http://www.sjune.net/archives/1726#61_Clone_the_Source)
  + [6.2. Configure it](http://www.sjune.net/archives/1726#62_Configure_it)
  + [6.3. Configure GitLab DB settings](http://www.sjune.net/archives/1726#63_Configure_GitLab_DB_settings)
  + [6.4. Install Gems](http://www.sjune.net/archives/1726#64_Install_Gems)
  + [6.5. Initialize Database and Activate Advanced Features](http://www.sjune.net/archives/1726#65_Initialize_Database_and_Activate_Advanced_Features)
  + [6.6. Install Init Script](http://www.sjune.net/archives/1726#66_Install_Init_Script)
  + [6.7. Set up logrotate](http://www.sjune.net/archives/1726#67_Set_up_logrotate)
  + [6.8. Check Application Status](http://www.sjune.net/archives/1726#68_Check_Application_Status)
  + [6.9. Start Your GitLab Instance](http://www.sjune.net/archives/1726#69_Start_Your_GitLab_Instance)
  + [6.10. Compile assets](http://www.sjune.net/archives/1726#610_Compile_assets)
* [7.Nginx](http://www.sjune.net/archives/1726#7Nginx)
  + [7.1. Installation](http://www.sjune.net/archives/1726#71_Installation)
  + [7.2. Site Configuration](http://www.sjune.net/archives/1726#72_Site_Configuration)

1.Packages / Dependencies

먼저 패키지와 의존성 설치를 해야한다. root권한이 없으면 sudo를 이용해 명령어를 실행한다.



|  |  |
| --- | --- |
| 1  2  3 | apt-get update -y  apt-get upgrade -y  apt-get install sudo -y |

vim을 설치하고 기본에디터로 세팅한다. 이미 사용중이라면 스킵해도 상관없다.



|  |  |
| --- | --- |
| 1  2 | sudo apt-get install -y vim  sudo update-alternatives --set editor /usr/bin/vim.basic |

의존성있는 패키지들을 설치한다.



|  |  |
| --- | --- |
| 1 | sudo apt-get install -y build-essential zlib1g-dev libyaml-dev libssl-dev libgdbm-dev libreadline-dev libncurses5-dev libffi-dev curl openssh-server redis-server checkinstall libxml2-dev libxslt-dev libcurl4-openssl-dev libicu-dev logrotate |

다음은 파이썬의 버전을 맞춘다. 우분투 13.10버전에서는 python 2.7이 기본적으로 깔려있어서 아래 내용은 스킵하였다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | # Install Python  sudo apt-get install -y python    # Make sure that Python is 2.5+ (3.x is not supported at the moment)  python --version    # If it's Python 3 you might need to install Python 2 separately  sudo apt-get install -y python2.7    # Make sure you can access Python via python2  python2 --version    # If you get a "command not found" error create a link to the python binary  sudo ln -s /usr/bin/python /usr/bin/python2    # For reStructuredText markup language support install required package:  sudo apt-get install -y python-docutils |

설치된 git의 버전이 올바른지 확인한다.



|  |  |
| --- | --- |
| 1  2  3  4  5 | # Install Git  sudo apt-get install -y git-core    # Make sure Git is version 1.7.10 or higher, for example 1.7.12 or 1.8.4  git --version |

git이 오래된 버전이라면 아래 명령어들을 사용해서 제거한뒤 1.8.4.1버전으로 재컴파일한다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | # Remove packaged Git  sudo apt-get remove git-core    # Install dependencies  sudo apt-get install -y libcurl4-openssl-dev libexpat1-dev gettext libz-dev libssl-dev build-essential    # Download and compile from source  cd /tmp  curl --progress https://git-core.googlecode.com/files/git-1.8.4.1.tar.gz | tar xz  cd git-1.8.4.1/  make prefix=/usr/local all    # Install into /usr/local/bin  sudo make prefix=/usr/local install    # When editing config/gitlab.yml (Step 6), change the git bin\_path to /usr/local/bin/git |

메일서버는 postfix를 추천하는 것 같다. 필요하면 설치한다.



|  |  |
| --- | --- |
| 1 | sudo apt-get install -y postfix |

2.Ruby

ruby 1.8 이전 버전을 사용하고 있다면 제거한다.



|  |  |
| --- | --- |
| 1 | sudo apt-get remove ruby1.8 |

ruby를 다운로드하고 컴파일한다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | mkdir /tmp/ruby && cd /tmp/ruby  curl --progress ftp://ftp.ruby-lang.org/pub/ruby/2.0/ruby-2.0.0-p353.tar.gz | tar xz  cd ruby-2.0.0-p353  ./configure --disable-install-rdoc  make  sudo make install |

Bundler Gem을 설치한다.



|  |  |
| --- | --- |
| 1 | sudo gem install bundler --no-ri --no-rdoc |

3.System Users

GitLab을 위해 git 사용자를 추가한다. 비로그인 사용자로 생성하므로 –disabled-login 옵션을 붙여준다.



|  |  |
| --- | --- |
| 1 | sudo adduser --disabled-login --gecos 'GitLab' git |

4.GitLab shell

GitLab shell은 ssh 접근 및 repository를 관리 소프트웨어이며 GitLab API를 제어한다. 이것 역시 설치한다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | # Go to home directory  cd /home/git    # Clone gitlab shell  sudo -u git -H git clone https://github.com/gitlabhq/gitlab-shell.git -b v1.8.0    cd gitlab-shell    sudo -u git -H cp config.yml.example config.yml    # 아래의 config.yaml에서 gitlab\_url 정보를 바꿔준다.  # 본 설정에서는 'http://domain.com/gitlab' 와 같은 URL을 사용하도록 변경했다.  sudo -u git -H editor config.yml    # Do setup  sudo -u git -H ./bin/install |

5.Database

Database는 MySQL과 PostgreSQL을 지원하는데 본 문서에서는 MySQL로 설치한다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | # Install the database packages  sudo apt-get install -y mysql-server mysql-client libmysqlclient-dev    # Pick a database root password (can be anything), type it and press enter  # Retype the database root password and press enter    # Secure your installation.  sudo mysql\_secure\_installation    # Login to MySQL  mysql -u root -p    # Type the database root password    # Create a user for GitLab  # do not type the 'mysql>', this is part of the prompt  # change $password in the command below to a real password you pick  mysql> CREATE USER 'git'@'localhost' IDENTIFIED BY 'git';    # Create the GitLab production database  mysql> CREATE DATABASE IF NOT EXISTS `gitlabhq\_production` DEFAULT CHARACTER SET `utf8` COLLATE `utf8\_unicode\_ci`;    # Grant the GitLab user necessary permissions on the table.  mysql> GRANT SELECT, LOCK TABLES, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER ON `gitlabhq\_production`.\* TO 'git'@'localhost';    # Quit the database session  mysql> \q    # Try connecting to the new database with the new user  sudo -u git -H mysql -u git -p -D gitlabhq\_production    # Type the password you replaced $password with earlier    # You should now see a 'mysql>' prompt    # Quit the database session  mysql> \q |

6.GitLab

GitLab을 설치하기 위해 git 사용자의 홈디렉토리로 이동한다.



|  |  |
| --- | --- |
| 1 | cd /home/git |

6.1. Clone the Source

GitLab 소스를 git clone 한다.



|  |  |
| --- | --- |
| 1  2  3  4  5 | # Clone GitLab repository  sudo -u git -H git clone https://github.com/gitlabhq/gitlabhq.git -b 6-4-stable gitlab    # Go to gitlab dir  cd /home/git/gitlab |

6.2. Configure it

GitLab 환경정보를 수정한다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | cd /home/git/gitlab    # Copy the example GitLab config  sudo -u git -H cp config/gitlab.yml.example config/gitlab.yml    # Make sure to change "localhost" to the fully-qualified domain name of your  # host serving GitLab where necessary  #  # If you installed Git from source, change the git bin\_path to /usr/local/bin/git  # 호스트 정보와 포트정보 등 서비스에 필요한 정보를 수정한다. http://domain.com/gitlab처럼 접근하려면 relative\_url\_root의 값을 /gitlab 으로 변경한다.  sudo -u git -H editor config/gitlab.yml    # Make sure GitLab can write to the log/ and tmp/ directories  sudo chown -R git log/  sudo chown -R git tmp/  sudo chmod -R u+rwX  log/  sudo chmod -R u+rwX  tmp/    # Create directory for satellites  sudo -u git -H mkdir /home/git/gitlab-satellites    # Create directories for sockets/pids and make sure GitLab can write to them  sudo -u git -H mkdir tmp/pids/  sudo -u git -H mkdir tmp/sockets/  sudo chmod -R u+rwX  tmp/pids/  sudo chmod -R u+rwX  tmp/sockets/    # Create public/uploads directory otherwise backup will fail  sudo -u git -H mkdir public/uploads  sudo chmod -R u+rwX  public/uploads    # Copy the example Unicorn config  sudo -u git -H cp config/unicorn.rb.example config/unicorn.rb    # Enable cluster mode if you expect to have a high load instance  # Ex. change amount of workers to 3 for 2GB RAM server  # config.yml에서 releative\_url 정보를 변경했으므로 마찬가지로 ENV['RAILS\_RELATIVE\_URL\_ROOT'] 부분의 주석을 해제하고 값을 "/gitlab" 으로 수정한다.  sudo -u git -H editor config/unicorn.rb    # Copy the example Rack attack config  sudo -u git -H cp config/initializers/rack\_attack.rb.example config/initializers/rack\_attack.rb    # Configure Git global settings for git user, useful when editing via web  # Edit user.email according to what is set in gitlab.yml  sudo -u git -H git config --global user.name "GitLab"  sudo -u git -H git config --global user.email "gitlab@localhost"  sudo -u git -H git config --global core.autocrlf input |

6.3. Configure GitLab DB settings

데이터베이스 환경 정보도 수정한다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # Mysql  sudo -u git cp config/database.yml.mysql config/database.yml    # Make sure to update username/password in config/database.yml.  # You only need to adapt the production settings (first part).  # If you followed the database guide then please do as follows:  # Change 'secure password' with the value you have given to $password  # You can keep the double quotes around the password  sudo -u git -H editor config/database.yml |

6.4. Install Gems

mysql을 위한 Gems를 설치한다. 다소 시간이 걸린다.



|  |  |
| --- | --- |
| 1  2  3  4 | cd /home/git/gitlab    # For MySQL (note, the option says "without ... postgres")  sudo -u git -H bundle install --deployment --without development test postgres aws |

6.5. Initialize Database and Activate Advanced Features

MySQL 정보를 초기화한다.



|  |  |
| --- | --- |
| 1  2  3  4  5 | sudo -u git -H bundle exec rake gitlab:setup RAILS\_ENV=production    # Type 'yes' to create the database.    # When done you see 'Administrator account created:' |

초기화가 끝나면 아래처럼 admin 초기 비밀번호가 출력되므로 메모를 잘 해놓는다.



|  |  |
| --- | --- |
| 1  2  3 | Administrator account created:  login.........admin@local.host  password......5iveL!fe |

6.6. Install Init Script

init 스크립트를 다운로드한다. /etc/init.d/gitlab 파일은 업그레이드때마다 변동될 가능성이 있으므로 수정하면 안된다.



|  |  |
| --- | --- |
| 1 | sudo cp lib/support/init.d/gitlab /etc/init.d/gitlab |

그리고 init 스크립트의 설정파일도 /etc/default/ 경로 아래복사한다. 만약 GitLab을 default 설치로 진행하지 않았다면(default folder에 설치하지 않았거나 user copy를 했다는둥) 환경에 맞게 /etc/default/gitlab을 수정해야 한다.



|  |  |
| --- | --- |
| 1 | sudo cp lib/support/init.d/gitlab.default.example /etc/default/gitlab |

서버가 부팅될 때 gitlab이 시작하도록 스크립트를 update-rc.d에 등록해준다.



|  |  |
| --- | --- |
| 1 | sudo update-rc.d gitlab defaults 21 |

6.7. Set up logrotate

로그파일을 관리하기 위해 logrotate 를 등록한다.



|  |  |
| --- | --- |
| 1 | sudo cp lib/support/logrotate/gitlab /etc/logrotate.d/gitlab |

6.8. Check Application Status

설치가 거의 마무리 되었다. GitLab의 설정 환경이 제대로 설정되었는지 확인한다.



|  |  |
| --- | --- |
| 1 | sudo -u git -H bundle exec rake gitlab:env:info RAILS\_ENV=production |

6.9. Start Your GitLab Instance

GitLab 인스턴스를 시작한다.



|  |  |
| --- | --- |
| 1  2  3 | sudo service gitlab start  # or  sudo /etc/init.d/gitlab restart |

6.10. Compile assets

assets을 컴파일한다.



|  |  |
| --- | --- |
| 1 | sudo -u git -H bundle exec rake assets:precompile RAILS\_ENV=production |

7.Nginx

GitLab은 Nginx을 공식적인 웹서버로 사용한다. 다른 웹서버에서 사용하려면 [gitlab-recipes](https://github.com/gitlabhq/gitlab-recipes) 를 참고한다.

7.1. Installation

nginx를 설치한다.



|  |  |
| --- | --- |
| 1 | sudo apt-get install -y nginx |

7.2. Site Configuration

사이트 설정 예제를 다운로드한다.



|  |  |
| --- | --- |
| 1  2 | sudo cp lib/support/nginx/gitlab /etc/nginx/sites-available/gitlab  sudo ln -s /etc/nginx/sites-available/gitlab /etc/nginx/sites-enabled/gitlab |

환경에 맞춰 설정 파일을 수정한다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | # Change YOUR\_SERVER\_FQDN to the fully-qualified  # domain name of your host serving GitLab.  # server의 listen 부분을 서비스할 ip나 도메인으로 변경해준다.  sudo editor /etc/nginx/sites-available/gitlab  #### Restart  sudo service nginx restart |

설치가 완료되었다. 어플리케이션 상태를 다시 체크해보려면 아래의 명령어를 실행한다.



|  |  |
| --- | --- |
| 1 | sudo -u git -H bundle exec rake gitlab:check RAILS\_ENV=production |

모든 아이템의 색상이 녹색으로 나오면 GitLab이 성공적으로 설치된 것이다.

이상 우분투 서버에서 GitLab을 서비스하기 위한 기본적인 설치해보았다. 기본설치 이외에도 Redis의 Resque 설정, SSH, LDAP, omniauth 인증 기능도 제공하므로 관련된 정보는 [advanced-setup-tips](https://github.com/gitlabhq/gitlabhq/blob/6-4-stable/doc/install/installation.md#advanced-setup-tips) 를 참고하면 된다.

**References**

* https://github.com/gitlabhq/gitlabhq/blob/6-4-stable/doc/install/installation.md

# GITLAB 설치

GitLab 솔루션을 설치하는 과정을 상세하게 기술려 했으나, 이미 공식 홈페이지에 설치 절차가 상세하게 설명되어 있으므로 링크만 기록해 두도록 하겠습니다.

**[GitLab 설치 가이드]** <https://github.com/gitlabhq/gitlabhq/blob/5-3-stable/doc/install/installation.md>

# Apache와 GITLAB 연동

공식 홈페이지의 GITLAB 설치 가이드는 Nginx 웹 서버와 연동하는 방법을 기준으로 되어 있습니다. 다른 종류의 웹 서버를 사용해도 GITLAB과 연동할 수 있으나, 이 경우 마지막 웹 서버 세팅 부분을 다음 절차로 진행해야 합니다.

Apache 서버가 정상적으로 동작하고 있는 상태에서, Ruby on Rails와 연동하기 위한 모듈을 새로 설치하고 사용할 수 있도록 설정해 주어야 합니다.

## Passenger 및 Apache 연동 모듈 설치

[?](http://tuwlab.com/ece/10476)

|  |
| --- |
| sudo gem install passenger --no-rdoc --no-ri  sudo passenger-install-apache2-module |

## Apache 추가모듈 설치

[?](http://tuwlab.com/ece/10476)

|  |
| --- |
| apt-get install apache2-threaded-dev  apt-get install libapr1-dev  apt-get install libaprutil1-dev |

## Site 파일 작성 및 활성화

/etc/apache2/sites-available/gitlab 파일을 다음과 같이 작성합니다.

※ Gem이나 Passenger의 설치 버전에 따라서 경로명이 조금씩 다를 수 있습니다. passenger-x.y.z 부분은 자신이 설치한 버전에 맞게 고쳐 주도록 합니다.

[?](http://tuwlab.com/ece/10476)

|  |  |
| --- | --- |
| gitlab | |
| 1  2  3  4  5  6  7  8 | LoadModule passenger\_module /usr/local/lib/ruby/gems/1.9.1/gems/passenger-4.0.8/buildout/apache2/mod\_passenger.so  PassengerRoot /usr/local/lib/ruby/gems/1.9.1/gems/passenger-4.0.8  PassengerRuby /usr/local/bin/ruby    <VirtualHost \*:80>      ServerName      [GITLAB 접속 주소]      DocumentRoot    [GITLAB 설치 위치]/public  </VirtualHost> |

작성을 완료했으면 이제 사이트를 활성화 시키고 Apache를 재시작합니다.

[?](http://tuwlab.com/ece/10476)

|  |
| --- |
| sudo a2ensite gitlab  sudo service apache2 restart |

재시작할 때 mod\_passenger.so 파일을 찾을 수 없다는 경고가 나오면, passenger 모듈의 경로가 잘못 지정된 것이므로 경로를 올바르게 지정해 주도록 합니다.