# Heroku

웹서비스를 배포하는 데는 많은 자원이 필요하다. 그 중에서도 가장 중요한 자원이 웹서비스를 운영할 서버로써 과거에는 물리적 하드웨어의 구매부터 장애 관리까지 직접 감당해야했으며 나아가 비스에 필요한 의존성 및 서비스 자체에 대해서도 직접 세팅하고 관리해줄 필요가 있었다. 이러한 서버 자원에 대해서 혁신적인 변화중 하나는 Amazon의 AWS 서비스였다. 특히 Amazon의 EC2는 원하는 만큼 쉽고 빠르게 확장가능한 가상 서버를 제공해 물리적 하드웨어 관리에 대한 부담을 덜어주었다.

Heroku 역시 이러한 Amazon AWS의 강점을 그대로 이용하고 있는 서비스로, 웹서비스 배포를 지원하는 PaaS(Platform as a Service)이다. EC2가 물리적 하드웨어의 관리에서 관리자를 해방시켜준다면, Heroku는 어플리케이션 배포에 필요한 소프트웨어적 환경의 관리로부터 관리자를 해방시켜주는 서비스라고 할 수 있다. 특히 앞서 설명한 내부적으로 어플리케이션을 Git 저장소로 관리할 수 있도록 되어있어, Git 저장소에 특정 구조를 가진 어플리케이션을 가지고 있다면 Heroku Git 서버에 푸쉬하는 것만으로도 어플리케이션 배포가 가능하다. 원래는 루비 온 레일즈 프레임워크를 지원하는 서비스였으나, 현재는 Ruby, Node.js, Python, Java를 지원하며 비공식적으로는 PHP까지 지원하고 있다. Ruby 언어를 만든 마츠모토 유키히로가 치프 아키텍트로 있는 것으로도 유명하다.

## Heroku를 통한 웹 서비스 배포

간단히 Heroku를 통해서 웹 서비스를 어떻게 배포하는지 살펴보도록 하자. 여기서는 yeoman을 angularjs 프로젝트를 생성하고 이 프로젝트를 nodejs의 express 서버를 통해서 배포한다.

먼저 디렉토리를 생성하고 git 저장소를 초기화한다.

$ mkdir heroku\_angular$ cd heroku\_angular$ git init$

다음으로 npm을 통해서 generator-angular를 설치한다. 이를 통해 yo 명령어로 angularjs 프로젝트를 초기화할 수 있다.

$ npm install -g generator-angular

Yeoman을 통해서 angularjs 프로젝트를 초기화한다. 초기화 과정에서 설정을 기본값을 따르며, 각 질문에서 엔터키를 입력한다.

$ yo angularOut of the box I include Bootstrap and some AngularJS recommended modules.[?] Would you like to use Sass (with Compass)? <Enter>[?] Would you like to include Twitter Bootstrap? <Enter>[?] Would you like to use the Sass version of Twitter Bootstrap? <Enter>[?] Which modules would you like to include? (Press <space> to select)‣⬢ angular-resource.js ⬢ angular-cookies.js ⬢ angular-sanitize.js ⬢ angular-route.js <Enter>

다음으로 Grunt 명령어를 통해서 프로젝트를 빌드한다. Grunt는 개발중인 angularjs 프로젝트를 배포가능한 형태로 dist 디렉토리에 생성해준다. 아직은 Yeoman을 통해서 생성한 기본적인 요소들만 가지고 있으므로, 실제로 빌드를 거치게 되면 임시 페이지를 볼 수 있다.

$ grunt build

npm을 통해 프로젝트에 express 패키지를 추가해준다. 여기서 express는 angularjs 프로젝트를 빌드한 static 파일들을 배포하기 위한 서버로 사용된다.

$ npm install express --save

아래와 같이 express 웹 서버 구동을 위한 web.js 파일을 생성한다.

var express = require('express');var app = express();app.use(express.static(\_\_dirname + '/dist'));app.listen(process.env.PORT || 5000);

이 파일은 Heroku에서 웹서버로 구동된다. 이를 통해 Grunt로 빌드한 dist 디렉토리 아래의 파일들을 웹에서 접근할 수 있도록 해준다.

다음으로 아래와 같이 Procfile을 생성한다. 이 파일에는 Heroku에서 어플리케이션이 어떻게 실행되는지 정의한다.

web: node web.js

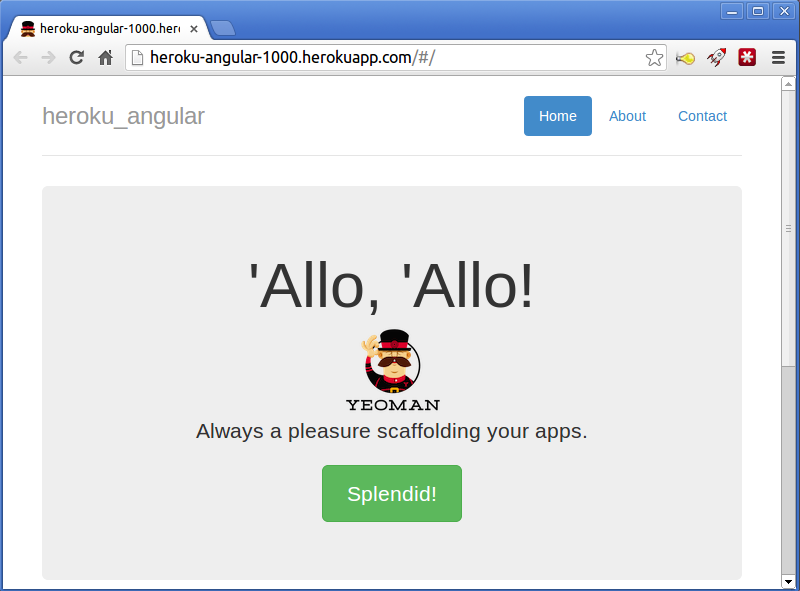
마지막으로 지금까지의 변경사항을 저장소에 commit합니다.

$ git add -f .$ git commit -m'First commit'

이제 Heroku 저장소를 생성하고 Push한다. Heroku에 Push하게 되면 Heroku에서는 자동적으로 지원되는 어플리케이션이 있는지 확인하고 어플리케이션이 작동하도록 적절하게 빌드해준다.

$ heroku create heroku-angular-1000$ git push heroku master$ heroku start$ heroku ps:scale web=1

Heroku의 어플리케이션 이름은 herokuapp.com 의 서브도메인이 되므로 중복된 이름은 사용할 수 없다. 여기서는 heroku-angular-1000 대신에 사용하고자 하는 도메인을 지정한다.



AngularJs on Heroku

# Nitrous.io

# Git

## 개요

Git은 리눅스의 리누스 토발즈가 리눅스 커널 관리를 위해 개발한 분산 버전관리 시스템(decentralized version control system)이다. Git은 단순한 diff와 버전관리를 넘어서 브랜치와 태그를 제공해 프로젝트 전체를 좀 더 체계적으로 관리하고 추적할 수 있도록 도와주며, 프로젝트에 부가된 변경이력 정보를 통해 다양한 정보를 통해 다양한 기능들을 제공한다. 또한 분산형 버전관리 어플리케이션으로서 중앙서버를 통한 관리뿐만 아니라 개개의 작업 공간을 공유하거나 분산된 작업 환경을 통합하기 위한 기능들을 제공한다. 또한 SVN과 같이 이전에 많이 쓰이던 버전관리 시스템에 스테이징 영역이 추가되어 로컬 상에서 저장소에 파일을 commit 하는 과정에 있어서 좀 더 섬세한 조작이 가능하다. 이러한 많은 장점을 비롯해 무엇보다도 Git 오픈소스로서 하나의 어플리케이션으로는 대부분의 환경에서 아무런 제약없이 사용할 수 있다.

## Git 설치하기

### 리눅스

#### 패키지 관리자를 이용한 설치

리눅스에서는 각각 배포판에서 사용하는 패키지 관리자를 사용해 Git을 바로 설치할 수 있다. 쉘에서 패키지 관리자 설치 명령어를 직접 입력하거나, GUI로 만들어져있는 패키지 매니저에서 검색해 설치할 수 있다. 아래는 Git을 설치하는 패키지 관리자 명령어이다.

# Debian / Ubuntu$ apt-get install git# Fedora$ yum install git# Gentoo$ emerge --ask --verbose dev-vcs/git# 설치 확인$ git --versiongit version 1.8.1.2

이외의 배포판에서도 각각의 패키지 관리자를 사용해 설치할 수 있다.[[1]](#footnote-31)

#### 소스를 이용한 설치

패키지 관리자를 통해서 설치할 수 있는 버전은 해당하는 운영체제에 안정화된 버전이다. 일반적으로 안정화된 버전을 사용하는 것으로도 충분히 Git을 사용할 수 있지만, 최신 버전이나 특별한 버전을 설치할 경우 소스 코드를 다운로드 받아 직접 빌드해서 설치할 수 있다.

Git의 소스코드는 Git 공식 저장소[[2]](#footnote-33)에서 다운로드 받을 수 있다. 예를 들어 아래에서는 Git 최신 버전 1.8.5.2 버전을 설치한다.

# 패키지 관리자에서 제공하는 패키지 정보를 확인$ apt-cache show gitCommend > apt-cache show gitPackage: gitVersion: 1:1.8.1.2-1# 소스코드 다운로드$ wget http://git-core.googlecode.com/files/git-1.8.5.2.tar.gz# 압축 해제$ tar xvzf git-1.8.5.2.tar.gz# Git 빌드$ cd git-1.8.5.2$ ./configure$ sudo make$ sudo make install# 설치 확인$ git --versiongit version 1.8.1.2

소스 코드를 통해 빌드하는 과정에서 다른 패키지에 대한 의존성 문제로 설치가 잘 되지 않을 수도 있다. 이럴 경우엔 에러 메시지를 확인하고 관련된 패키지를 설치해줘야한다. 이 외에도 시스템 환경에 따라서 다양한 문제가 발생할 수 있으므로 구글에 에러메시지를 검색해서 해결방법을 찾아볼 수 있다.

### MacOS

#### Xcode Commennd Line Tools를 이용한 설치

#### 패키지 관리자를 이용한 설치

MacOS에서도 리눅스와 마찬가지로 패키지 관리자를 사용해서 Git을 설치할 수 있다. 아래는 Homebrew를 사용해서 설치하는 예제이다. Homebrew는비교적 패키지 업데이트가 빠른 편이라 비교적 최신 버전을 쉽게 설치할 수 있다.

# Homebrew 설치$ ruby -e "$(curl -fsSL https://raw.github.com/Homebrew/homebrew/go/install)"# Git 설치$ brew install git$ git --versiongit version 1.8.5.1# 설치 확인

#### 소스를 이용한 설치

Mac에서도 리눅스와 마찬가지로 원하는 버전을 소스 코드를 직접 빌드해 설치할 수 있다. (자세한 내용은 Linux의 소소를 이용한 설치 절을 참조)

### 윈도우

## Git 기본 명령어

### Git 설정하기

### Git 저장소 생성하기

### Git 브랜치

## Github

Github[[3]](#footnote-44)는 원격에서 Git 저장소를 제공해주는 Git 서비스이자, 이 저장소를 웹에서 관리 및 조작할 수 있는 기능을 지원하는 웹서비스이다. Git에서는 기본적으로 저장소를 분산해서 관리하고 쉽게 공유할 수 있지만, Github는 이러한 원격 저장소를 한 곳에서 일관적으로 관리할 수 있도록 도와주며 브랜치, 태그, 코드 병합과 같은 Git의 기능을 기본적으로 지원함은 물론 추가적으로 저장소 Fork, 이슈 관리, Pull request, 접근 관리(멤버 관리), 위키 기능 등을 지원해 단순한 버전관리 시스템을 넘어서 협업 및 프로젝트 관리에 유용하게 사용할 수 있다.

Github은 공개된 코드에 대해서는 사용료가 없으며, 이러한 이유로 현재는 대부분의 오픈소스가 Github 저장소를 사용하고 있다. 공개된 코드는 오픈소스가 아니더라도 무료로 이용할 수 있으며 비공개 저장소를 만들고자 할 때는 약간의 이용료를 지불하면 된다.

이 뿐만 아니라 Github을 둘러싸고 있는 생태계 역시 주목할만하다. 어플리케이션 빌드 및 배포를 지원하는 CI서비인 Travis[[4]](#footnote-45), 테스트 커버리지 측정을 해주는 Coveralls[[5]](#footnote-46), 정적 코드 분석 도구인 Code Climate[[6]](#footnote-47) 등등, 이러한 서비스들은 Github 저장소만 있다면 언제든 바로 사용이 가능하며, 일련의 서비스들 역시 공개 저장소에 대해서는 무료로 사용할 수 있다.

### Gtihub 계정 만들기

### ssh키 생성

Github을 비롯한 외부 저장소 서비스를 이용하려면 사용자 인증을 대신해 ssh 키를 사용해야한다. ssh키를 생성하면 공개키와 비밀키가 생성이 된다. 이 때 비밀키는 인증하려는 하는 시스템에 안전하게 보관해야하며, 공개키는 Github에 등록해야한다.

#### 리눅스, MacOS

리눅스와 MacOS에서 ssh키는 ssh-keygen 명령어를 사용해 생성한다.

$ ssh-keygen -t rsa -C "<YOUR\_EMAIL>"Gnerating public/private rsa key pair.Enter file in which to save the key (/home/YOUR\_ACCOUNT/.ssh/id\_rsa): <Enter>Enter passphrase (empty for no passphrase): <YOUR\_PASSWORD>Enter same passphrase again: <YOUR\_PASSWORD>Your identification has been saved in /home/YOUR\_ACCOUNT/.ssh/id\_rsaYour public key has been saved in /home/YOUR\_ACCOUNT/.ssh/id\_rsa.pubThe key fingerprint is:...

일반적으로 ssh키 생성 경로는 홈디렉토리/.ssh를 사용한다. 패스워드를 입력하지 않으면 파일을 복사하는 것만으로 인증이 가능해지므로 패스워드를 지정하는 것이 좋다. 생성된 홈디렉토리/.ssh 디렉토리에서 확인할 수 있으면 비밀키는 id\_rsa 파일에 저장되고, 공개키는 id\_rsa.pub 파일에 저장된다.

#### 윈도우

### Github ssh키 등록

ssh키를 생성했다면 Github에 공개키를 등록해 커맨드라인이나 다른 어플리케이션에서 인증이 가능하도록 해주어야한다.

* 먼저 Github에 로그인을 하고 오른쪽 상단 메뉴 중에 Account Settings에 접속한다.
* 왼쪽 사이드 바의 "ADD SSH key"를 선택한다.
* key 필드에 공개키의 내용을 그대로 복사하고 키를 추가한다.

키 등록이 완료되었으면 ssh를 통해 인증이 정상적으로 이루어지는지 확인해볼 필요가 있다.

$ ssh -T git@github.com

처음 접속시 아래와 같은 메시지가 출력된다.

The authenticity of host 'github.com (207.97.227.239)' can't be established.# RSA key fingerprint is 16:27:ac:a5:76:5d:2d:36:63:1b:32:4d:eb:df:a6:48.# Are you sure you want to continue connecting (yes/no)?

yes를 입력하거나 엔터를 치면 아래와 같은 메시지를 볼 수 있다.

Hi YOUR\_NAME! You've successfully authenticated, but GitHub does not provide shell access.

접속에 실패하면 다음과 같은 메시지가 출력된다.

Permission denied (publickey).

이러한 경우 키 생성이나 공개키 복사 과정에서 문제가 생겼을 수 있으니 다시 한 번 확인해볼 필요가 있다. 이러한 과정이 정상적인 경우에도 문제가 발생하는 경우엔 네트워크 문제나 어플리케이션 접근하려는 ssh 키 경로가 다를 수 있다. 이를 때는 자세한 접속 로그를 확인해볼 필요가 있다.

$ ssh -vT git@github.com

-v 옵션을 사용하면 자세한 접속 로그를 확인할 수 있어 이러한 문제를 확인할 수 있어 문제 해결에 도움을 준다.

### Github에 저장소 생성하기

Github에 저장소를 생성하는 방법에는 크게 두가지 방법이 있다. 하나는 Github에서 저장소를 초기화해서 로컬으로 클론(clone)하는 방법이며, 또 하나는 로컬에서 관리중에 git 저장소를 Github에 푸쉬(push)하는 방법이다. 전자는 미리 저장소를 초기화할 필요가 있는 경우에 사용하며, 후자는 이미 로컬에서 관리중인 git 저장소가 있는 사용하는 게 보통이지만 두 접근법에 근본적인 차이가 있는 것은 아니므로 상황에 따라 편리한 방법을 사용하면 된다.

#### Github에서 저장소 초기화하기

아래와 같은 절차에 따라 Git저장소를 웹상에서 바로 초기화할 수 있다.

* github.com 에 접속해서 로그인
* 오른쪽 상단의 Create a new repo 선택
* Repository 이름 입력
* Initialize this repository with a README 옵션에 체크
* Create Repository 선택

저장소를 만들면 바로 저장소 관리 페이지를 볼 수 있다. 페이지 오른쪽에는 로컬에서 clone할 수 있는 주소를 볼 수 있다. 이 주소를 복사해 로컬에서 아래 명령어를 입력하면 저장소 이름으로 새로운 디렉토리가 생성된다.

#### 로컬 저장소를 Github에 푸쉬해서 초기화하기

### Github에 저장소 푸쉬

### hub를 통한 Git 확장

1. 각각의 리눅스 배포판에서 설치하는 명령어는 http://git-scm.com/download/linux 에서 확인할 수 있다. [↑](#footnote-ref-31)
2. http://code.google.com/p/git-core/downloads/list [↑](#footnote-ref-33)
3. http://github.com [↑](#footnote-ref-44)
4. http://travis.com [↑](#footnote-ref-45)
5. http://coveralls.io [↑](#footnote-ref-46)
6. http://codeclimate.com , 현재는 ruby와 javascript의 정적 분석을 지원한다. [↑](#footnote-ref-47)