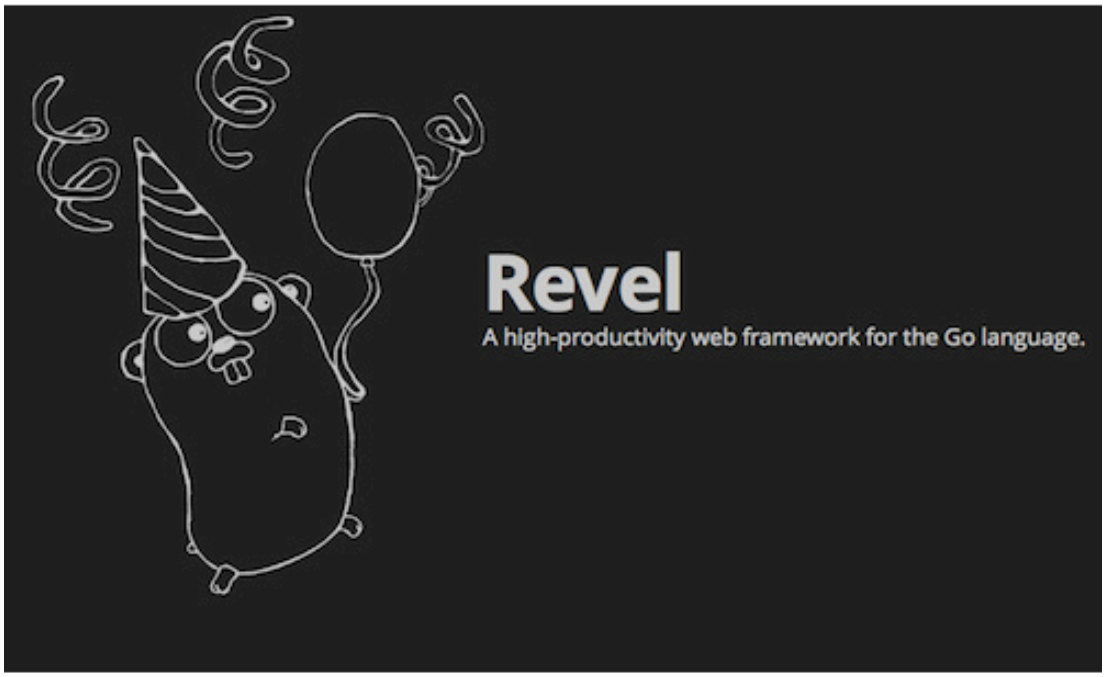
Revel
A high-productivity web framework for the Go language.

Revel Web Framework 入門

堀田直孝 著

2014-05-03 版　　Hungry Foolish Gray Brain.　発行

はじめに (仮)

この本について

Background（背景)
■プロジェクトの背景。これが無ければ始まらない。

Objective（目的)
■プロジェクトの目的。目指すゴール地点を定義する。

Scope（範囲）
■プロジェクトの範囲。範囲外についても言及する。線引はキチッとする。

Constraint（制約）
■プロジェクト実行側が守る制約。期限、予算、リソースなど。

Assumption（前提）
■プロジェクト依頼側が守ること。情報提供など。ないがしろになりがち。

Report（成果）
■必要な成果物。

ご協力のお願い:

私は、日本語が得意ではありません。誤字脱字については、是非ご連絡頂けると幸いです。又用語の定義違いその他、お気づきの点は是非ご連絡頂けると幸いです。

この本のために使ったソースコードは、Github で公開しています。pull request で修正を頂けますと、ソースの修正から再公開までの作業がスムーズに進むと思われます。是非、Github での pull request 申請による修正も活用していただけると幸いです。

■Github Repository: https://github.com/jhotta/revel-book

謝辞:

この本のコンテンツは、Ruby on Rails チュートリアル (http://railstutorial.jp/) にインスパイアされて作成しました。このような素晴らしいチュートリアルを公開していてくれることに感謝します。更にこの本の査読を手伝ってくれたコーワーキングスペース mitacafe のメンバーの皆さんに感謝します。

最後に、無職の私に本を書く動機とチャンスを与えてくれた妻と娘に感謝します。

筆者について

図: @jhotta

略歴

略歴は、肯定的に書こうと思っているのですが何を書いたらいいのか思い当たりません。「自分の良いところをきちんと認知しないと…」と常々思っているのですが、妄想ばかりのようなきがします。

- DevOps に関して真剣の考えたことのある人材
- プログラマーの地位を向上させたいと思っている人材
- 妻のことが大好きな人材

協力者リスト

以下の協力者を紹介します。

- aaaaa
- bbnbbbb

準備する環境

この本の対象とする読者にはマイクロソフト社製の OS を利用している人も多々いると思うが、私自身 Windows XP 以降マイクロソフト社製 OS を利用していない。現状それらの OS がどのような状況になっているか把握できていないのだ。従ってこの本では、インストールや実行確認作業を仮想 OS 環境上の Linux OS で進めていくことにする。

又、基本的な操作は Apple 社製の OS から実施している。万が一この本で使っているツールがマイクロソフト社製の OS 向けに存在しない場合は、操作の目的に合ったツールに切り替え、操作を完結して欲しい。

仮想 OS 環境を実現するソフト

■VirtualBox: https://www.virtualbox.org/

VirtualBox は、x86 仮想化ソフトウェア・パッケージの一つで、米国オラクル社によって開発がすすめられています。サポートされているホスト OS は Linux、Mac OSX、Windows、Solaris です。ゲスト OS としては、FreeBSD、Linux、OpenBSD、OS/2 Warp、Windows、Mac OS X Server、Solaris など x86/x64 アーキテクチャの OS であれば基本的には起動できます。GPL ver.2 で公開されている FOSS なので、無料で使用することが出来ます。

仮想 OS 環境で利用する Linux OS

■Ubuntu 13.10: server 64bit http://www.ubuntu.com/download/server

Ubuntu は、Debian GNU/Linux をベースとした Linux ディストリビューションの 1 つです。Ubuntu は、「誰にでも使いやすい最新かつ安定した OS」を目標にカノニカル社から支援を受けて開発されています。毎年 4 月, 10 月に更新版がリリースされ、LTS(長期メンテナンス) バージョンは 2 年に一度リリースされ、12.04 が直近バージョンになります。

仮想 OS 環境と Provision(自動設定) ソフトの間を取り持つソフト

■Vagarant: http://www.vagrantup.com/

Vagrant は、仮想 OS 環境を設定したり、設定後の仮想マシンのイメージを作成指定くれるオープンソースのソフトウェアです。VirtualBox や VMware などの仮想 OS 環境と Puppet, Chef, Ansible などの構成管理ツールの間を取り持って、再現性の高い管理環境を提供してくれます。

又、仮想 OS の起動, 設定,SSH 通信, マシンイメージ作成などをコマンドラインから操作することができ、開発者の操作性に高い操作性を提供しています。

仮想 OS を Provision(自動設定) するソフト

■Ansible: https://github.com/ansible/ansible/

Puppet, Chef と同等な Provision ソフト。Puppet, Chef と異なり、クライアントソフトをインストールすること無く、設定対象環境の Provision 作業を進めることができる。設定内容は、yaml 形式で記述されている。

目次

第 1 章

Revel で開発するための環境の準備

Go 言語を使って Revel が、起動できる環境を構築していきます。

1.1 各種ソフトウェアのインストール

1.1.1 VitualBox のインストール

この本では、VirtualBox を利用してベースとなる LinuxOS を起動することにします。VirtualBox は、次のランディングページからダウンロードしてインストールします。

```
https://www.virtualbox.org
```

実際のダウンロードページの URL は、次になります。

```
https://www.virtualbox.org/wiki/Downloads
```

OSX に適応した VirtualBox のパッケージを、このサイトからダウンロードします。OSX hots の右側にあり"x86/amd64"の文字列をダブルクリックするとダウンロードがスタートします。

図 1.1　VirtualBox のダウンロードページ

先にダウンロードした VirtualBox のインストールファイルをダブルクリックし、ポップアップメニューの指示に従って VirtualBox をインストルします。

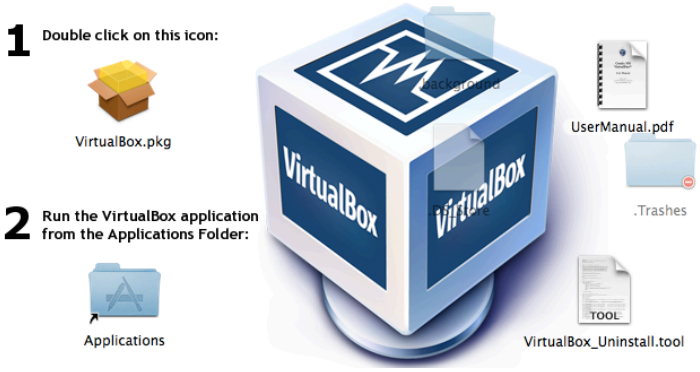

図 1.2　インストール図 1

ダウンロードしたファイルをダブルクリックし、ポップアップメニューの指示に従い VirtualBox をインストルします。

図 1.3　インストール図 3

ダウンロードしたファイルをダブルクリックし、ポップアップメニューの指示に従い VirtualBox をインストルします。

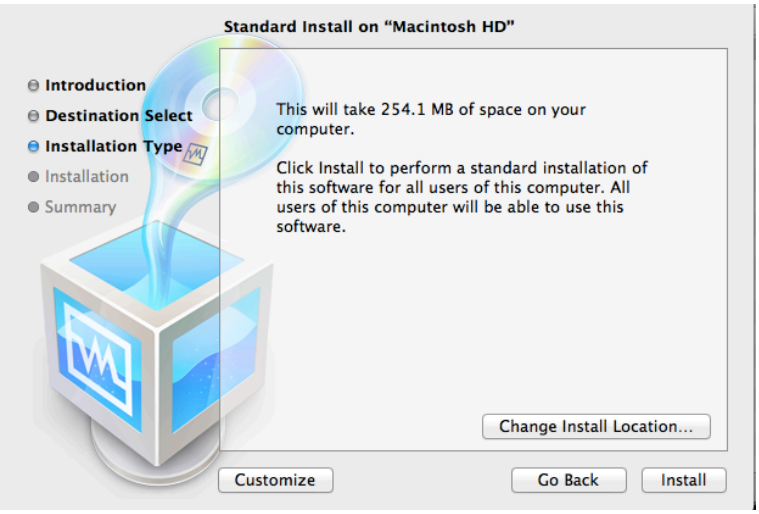

図 1.4　インストール図 3

ダウンロードしたファイルをダブルクリックし、ポップアップメニューの指示に従い VirtualBox をインストルします。

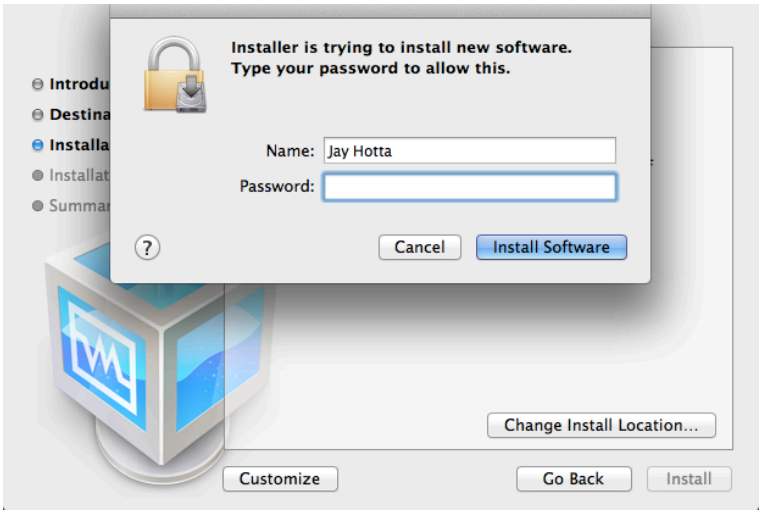

図 1.5　インストール図 3

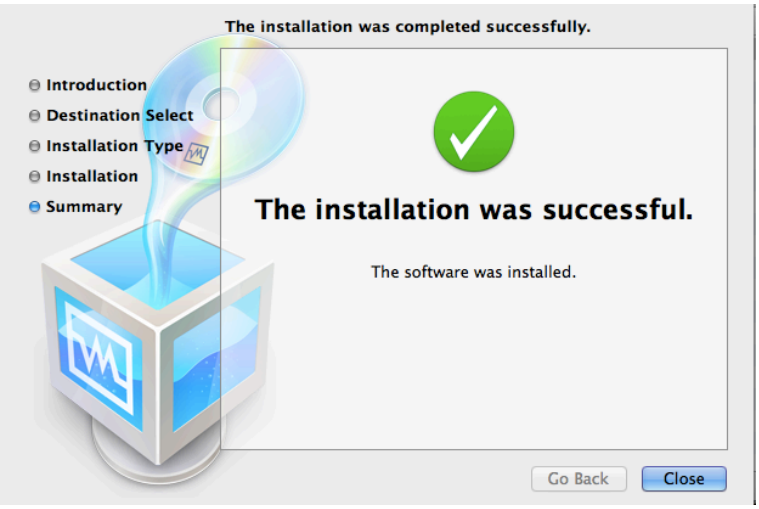

図 1.6　インストール図 3

1.1.2　Vagrant のインストール

VirtualBox は直接操作しても特に問題はなのですが、仮想 OS 環境を効率よく操作すると共に基本環境の構築の再現性を上げるために、今回は Vagrant を使用することにします。

Vagrant は、次のランディングページからダウンロードできます。

```
http://www.vagrantup.com/
```

Download タブをクリックして、パッケージの選択画面に移動します。

```
http://www.vagrantup.com/downloads.html
```

OSX に適応した VirtualBox のパッケージを、このサイトからダウンロードします。

--[[path = (not exist)]]--

Vagrant のダウンロードページ

先にダウンロードした VirtualBox のインストールファイルをダブルクリックし、ポップアップメニューの指示に従って VirtualBox をインストルします。

図 1.7　インストール図 1

図 1.8　インストール図 1

```
--[[path =  (not exist)]]--
```

インストール図 1

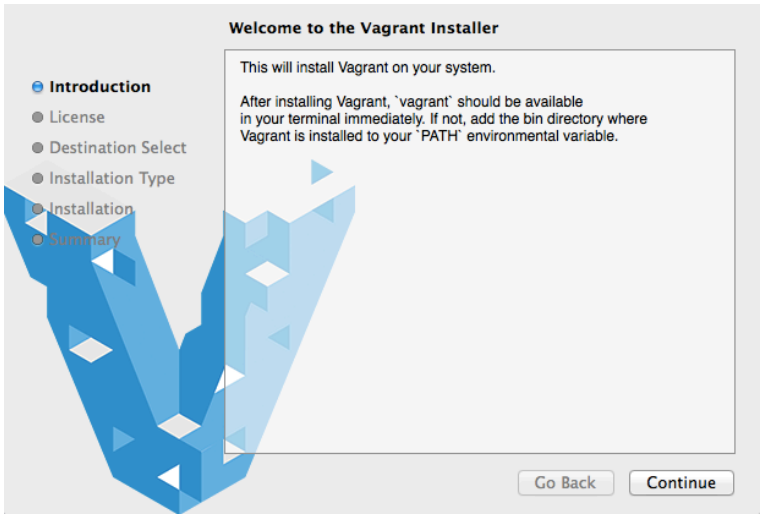

図 1.9　インストール図 1

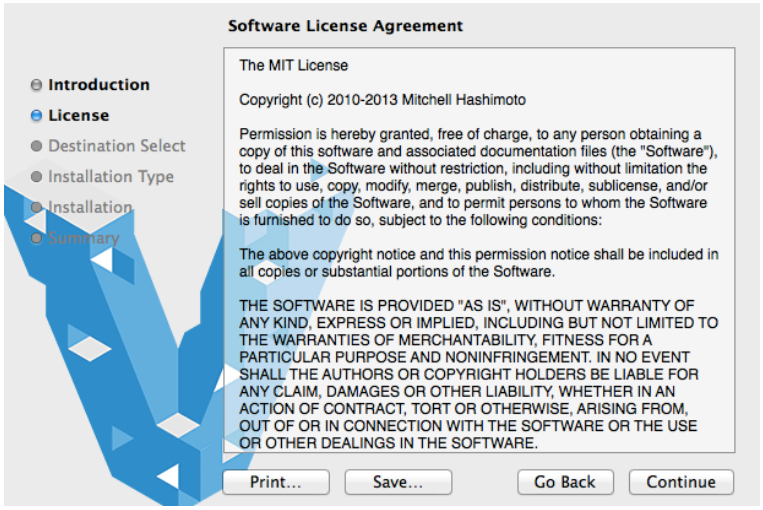

図 1.10　インストール図 1

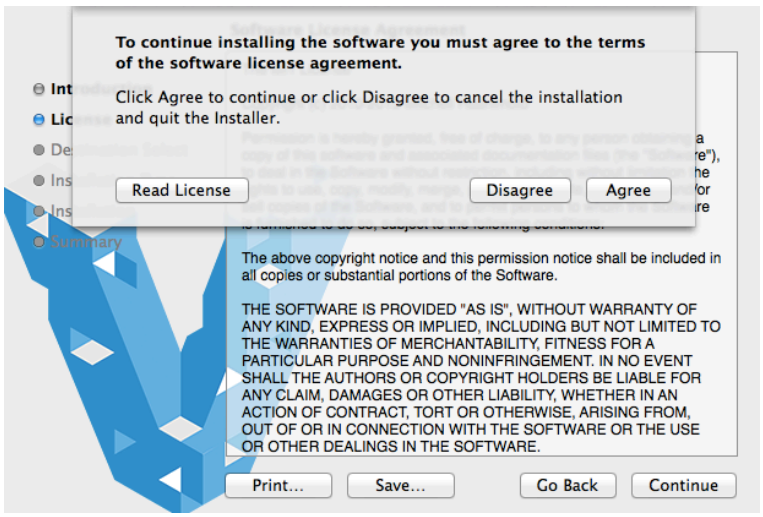

図 1.11　インストール図 1

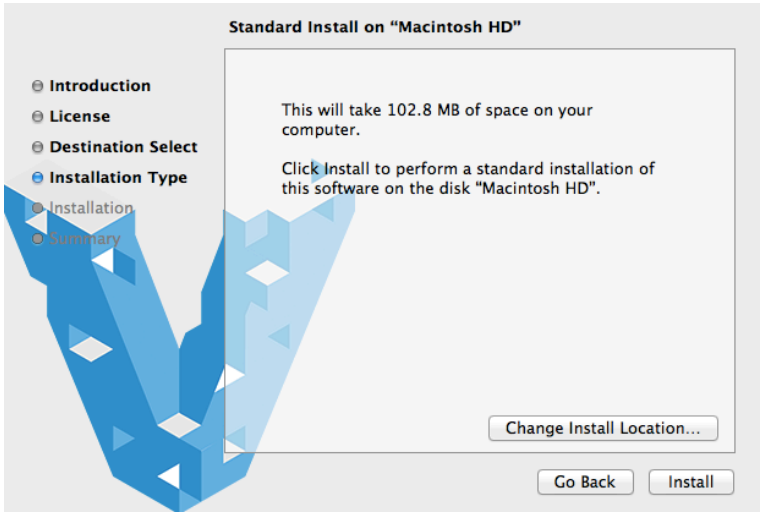

図 1.12　インストール図 1

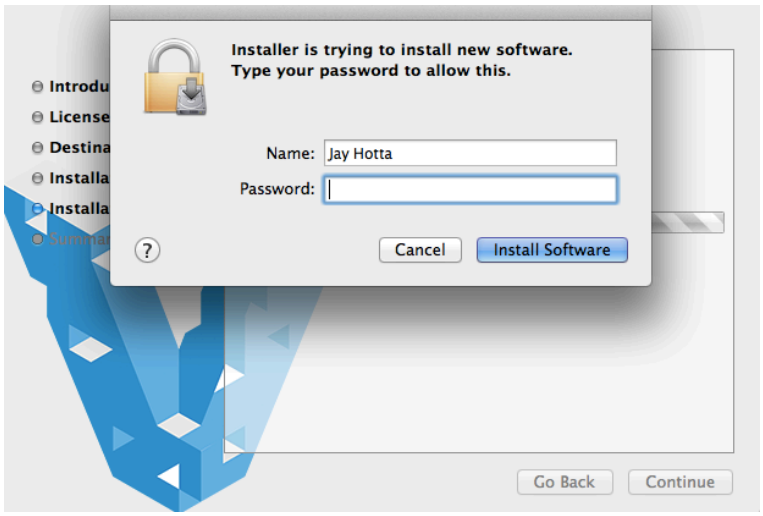

図 1.13　インストール図 1

ダウンロードしたファイルをダブルクリックし、ポップアップメニューの指示に従い Vagrant をインストルします。

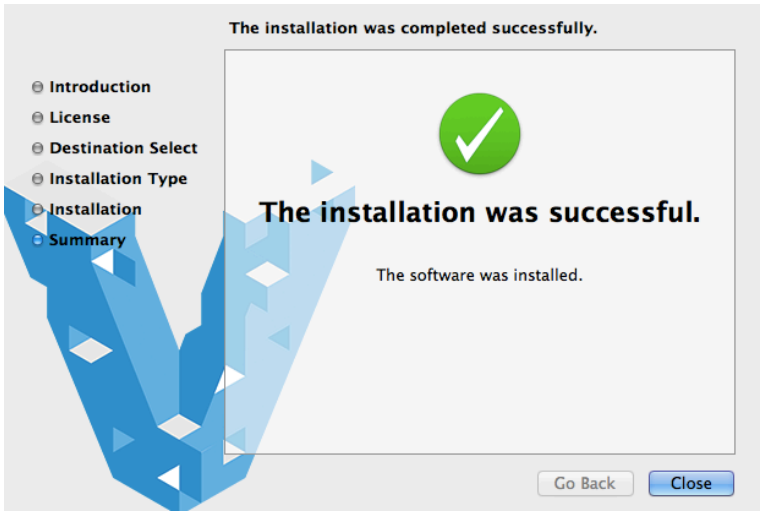

図 1.14　インストール図 1

ダウンロードしたファイルをダブルクリックし、ポップアップメニューの指示に従い Vagrant をインストルします。

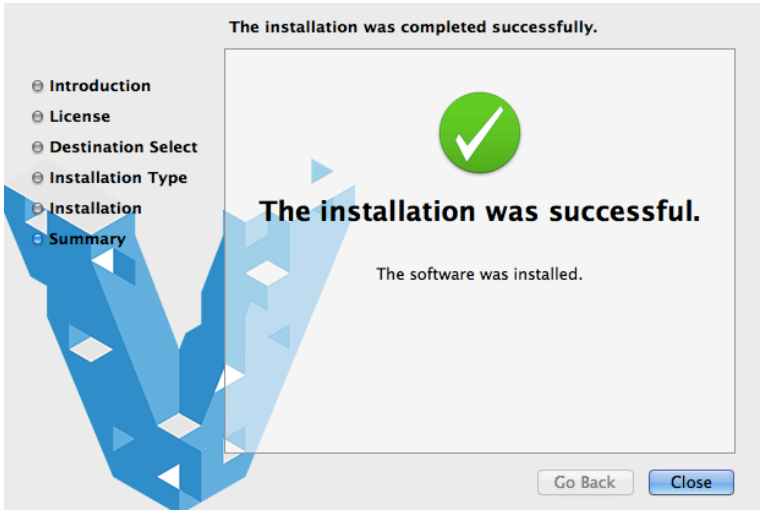

図 1.15　インストール図 1

ダウンロードしたファイルをダブルクリックし、ポップアップメニューの指示に従い Vagrant をインストルします。

インストールが完了したところで、Vagrant の操作をするするためにコンソールを起動します。Terminal を実行し、コマンドを入力することになります。

```
$ vagrant --version
```

以下の内容が表示されれば、Vagrant のインストールは終了です。

```
Vagrant 1.4.3
```

1.1.3　仮想マシンイメージの準備

http://www.vagrantbox.es/

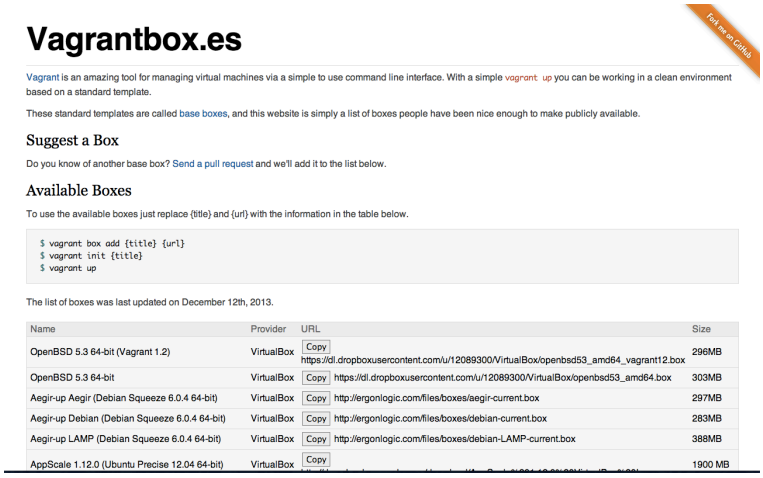

図 1.16　インストール図 3

今回は、"Official Ubuntu 13.10 daily Cloud Image amd64 (Development release, No Guest Additions)"と書かれている仮想マシンメージを Vagrant の管理下にインポートして使うことにします。

コマンドの基本は以下のフォーマットになります。

```
$ vagrant box add {title} {url}
```

今回は、ubuntu コミュニティーが提供している 先の仮想イメージの url を指定し、ubuntu を title に指定し、Vagrant 管理下にダウンロードしてきます。

```
$ vagrant box add ubuntu \
http://cloud-images.ubuntu.com/vagrant/saucy/current/\
saucy-server-cloudimg-amd64-vagrant-disk1.box
```

上記のコマンドを実行すると、下記の内容がターミナルウインドーに表示され、仮想 OS 用のイメージの登録が終了します。

```
Downloading box from URL: http://cloud-images.ubuntu.com/vagrant/saucy/current/saucy-serve
Extracting box...te: 1131k/s, Estimated time remaining: --:--:--)
Successfully added box 'ubuntu' with provider 'virtualbox'!
```

万が一、仮想 OS のイメージ登録に失敗した場合やイメージが不要になった場合は、以下の list コマンドで確認の上、削除できます。

```
$ vagrant box list
$ vagrant box revmove {title}
```

それでは、vagrant に必要なファイルを置いておくディレクトリーを作ることにします。

```
$ mkdir ~/vagrant-env
$ cd ~/vagrant-env
```

下記のフィーマットのコマンドを使って Vagarantfile を生成します。

```
$ vagrant init {title}
```

{title}の部分には、先ほど指定した仮想 OS イメージ名を指定しコマンドを実行します。

```
$ vagrant init ubuntu
```

下記のような実行か結果が表示されている確認してください。

```
A ‘Vagrantfile‘ has been placed in this directory. You are now
ready to ‘vagrant up‘ your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
‘vagrantup.com‘ for more information on using Vagrant.
```

それでは、ここで仮想 OS を起動してみます。

```
$ vagrant up
```

画面は、流れるように進んでいきます。最終的には以下のような内容の画面が表示されていることを確認してください。

```
Bringing machine 'default' up with 'virtualbox' provider...
[default] Importing base box 'ubuntu'...
[default] Matching MAC address for NAT networking...
[default] Setting the name of the VM...
[default] Clearing any previously set forwarded ports...
[default] Clearing any previously set network interfaces...
[default] Preparing network interfaces based on configuration...
[default] Forwarding ports...
[default] -- 22 => 2222 (adapter 1)
[default] Booting VM...
[default] Waiting for machine to boot. This may take a few minutes...
[default] Machine booted and ready!
[default] The guest additions on this VM do not match the installed version of
VirtualBox! In most cases this is fine, but in rare cases it can
prevent things such as shared folders from working properly. If you see
shared folder errors, please make sure the guest additions within the
virtual machine match the version of VirtualBox you have installed on
your host and reload your VM.

Guest Additions Version: 4.2.16
VirtualBox Version: 4.3
[default] Mounting shared folders...
[default] -- /vagrant
```

仮想 OS が起動している夜なら、ssh で仮想環境にアクセスすることにします。

```
$ vagrant ssh
```

以下のコマンドが、表示されているでしょうか？

```
Please report a bug if this causes problems.
Welcome to Ubuntu 13.10 (GNU/Linux 3.11.0-17-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Wed Feb 26 04:07:30 UTC 2014

  System load:  0.98              Processes:           86
  Usage of /:   2.6% of 39.34GB   Users logged in:     0
```

```
  Memory usage: 28%               IP address for eth0: 10.0.2.15
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

vagrant@vagrant-ubuntu-saucy-64:~$
```

ここまでで、Vagarant によってコントロールできる仮想環境ができました。それでは、仮想 OS 環境のシェルから、本体のシェルも戻ります。

```
$ exit
```

本体のシェルに戻ったところで、仮想 OS の環境を停止します。

```
$ vagrant halt
```

■コラム:

コマンドプロンプトの直前に下記のような WARNING が表示された場合:

```
------------------------------------------------------------------
WARNING! Your environment specifies an invalid locale.
 This can affect your user experience significantly, including the
 ability to manage packages. You may install the locales by running:

   sudo apt-get install language-pack-UTF-8
     or
   sudo locale-gen UTF-8

To see all available language packs, run:
   apt-cache search "^language-pack-[a-z][a-z]$"
To disable this message for all users, run:
   sudo touch /var/lib/cloud/instance/locale-check.skip
------------------------------------------------------------------
```

Mac 側で Terminal の設定の確認:

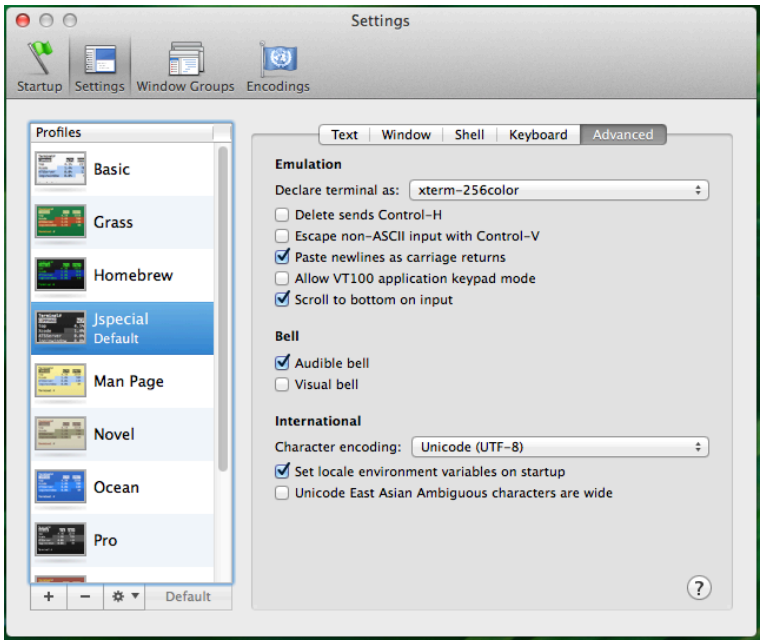

図 1.17　OSX terminal の設定

概要:

Mac 側で Terminal の locale に LC_CTYPE=UTF-8 が設定がされ、Ubuntu が解釈できないので以下のような警告が場合があります。

解決策:

"Set locale environment variables on startup"の前にあるチェエクマークを外してください。

1.1.4　ansible のインストール

OSX に Homebrew が事前にインストールされていることを前提に ansible のインストールを進めていきます。

```
$ brew install ansible
```

下記のようなインンストールが進行しビールジョッキの行が表示されれば、インストールは順調に完了したと推測できます。

```
==> Downloading https://github.com/ansible/ansible/archive/v1.4.5.tar.gz
Already downloaded: /Library/Caches/Homebrew/ansible-1.4.5.tar.gz
==> Downloading https://pypi.python.org/packages/source/p/pycrypto/pycrypto-2.6.
Already downloaded: /Library/Caches/Homebrew/ansible--pycrypto-2.6.tar.gz
==> python setup.py install --prefix=/usr/local/Cellar/ansible/1.4.5/libexec
==> Downloading https://pypi.python.org/packages/source/P/PyYAML/PyYAML-3.10.tar
Already downloaded: /Library/Caches/Homebrew/ansible--pyyaml-3.10.tar.gz
==> python setup.py install --prefix=/usr/local/Cellar/ansible/1.4.5/libexec
==> Downloading https://pypi.python.org/packages/source/p/paramiko/paramiko-1.11
Already downloaded: /Library/Caches/Homebrew/ansible--paramiko-1.11.0.tar.gz
==> python setup.py install --prefix=/usr/local/Cellar/ansible/1.4.5/libexec
==> Downloading https://pypi.python.org/packages/source/M/MarkupSafe/MarkupSafe-
Already downloaded: /Library/Caches/Homebrew/ansible--markupsafe-0.18.tar.gz
==> python setup.py install --prefix=/usr/local/Cellar/ansible/1.4.5/libexec
==> Downloading https://pypi.python.org/packages/source/J/Jinja2/Jinja2-2.7.1.ta
Already downloaded: /Library/Caches/Homebrew/ansible--jinja2-2.7.1.tar.gz
==> python setup.py install --prefix=/usr/local/Cellar/ansible/1.4.5/libexec
==> python setup.py install --prefix=/usr/local/Cellar/ansible/1.4.5
==> Caveats
Set PYTHONPATH if you need Python to find the installed site-packages:
  export PYTHONPATH=/usr/local/lib/python2.7/site-packages:$PYTHONPATH
==> Summary
^^f0^^9f^^8d^^ba  /usr/local/Cellar/ansible/1.4.5: 763 files, 8.8M, built in 13 seconds
```

ここで、ansible の動作確認をしてみます。

```
$ ansible --version
```

以下のようにバージョンが表示されれば、インストールは成功しています。

```
ansible 1.4.5
```

1.2 仮想マシンイメージの設定

1.2.1 Vagrantfile の準備

VirtualBox の動作の設定、仮想 OS の設定変更、アプリの自動でインストールのために、Vagrant を実行するディレクトリを初期化した際に出来上がった Vagrantfile を編集していきます。

完成後の Vagrantfile は、次のようになります。

リスト 1.1: 完成した Vagrantfile

```
 1: # -*- mode: ruby -*-
 2: # vi: set ft=ruby :
 3:
 4: VAGRANTFILE_API_VERSION = "2"
 5:
 6: Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
 7:
 8:   config.vm.box = "ubuntu"
 9:   config.vm.network :forwarded_port, guest: 9000, host: 9000
10:
11:   config.vm.provision "ansible" do |ansible|
12:     ansible.playbook = "playbook.yml"
13:   end
14:
15: end
```

VitrualBox の動作設定

自動で出来上がった Vagrantfile のコメント部分を削除すると次のようになるはずです。

リスト 1.2: オリジナル Vagrantfile

```
 1: # -*- mode: ruby -*-
 2: # vi: set ft=ruby :
 3:
 4: VAGRANTFILE_API_VERSION = "2"
 5:
 6: Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
 7:   config.vm.box = "ubuntu"
 8: end
```

次の行は、Vagrantfile が利用する API のバージョンの指定をしています。

```
VAGRANTFILE_API_VERSION = "2"
```

ここで示した do ~ end の間に VirtualBox が起動してくる際にオプションとして指定する項目を記述していきます。

```
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

end
```

"vagarant init"で生成した素の Vagrantfile では、生成時に指定した"ubuntu"のマシンイメージ

のみが指定されています。

```
config.vm.box = "ubuntu"
```

オリジナルの Vagrantfile の内容が把握出来たところで、追加の設定を記述していくことにします。

待ち受け Port の forwading 設定

仮想 OS 上に起動されてた Revel がブラウザーのリクエストに反応するために待ち受けている Port 番号を、VirtualBox の Port 番号と関連付けをすることにします。

```
config.vm.network :forwarded_port, guest: 9000, host: 9000
```

OS の設定変更

ポート番号の関連付けが済んだ後の設定作業は、ansibile を使って進めることにします。

```
config.vm.provision "ansible" do |ansible|
  ansible.playbook = "playbook.yml"
end
```

provision ツールの ansible を利用することを宣言し、ansible によって仮想 OS を設定する内容が playbook.yml に記述されていることを宣言しています。

1.2.2　playbook.yaml の準備

Vagrantfile に設定内容の詳細を記述するファイルとして宣言しておいた playbook.yaml の準備することにします。

完成後の playbook.yaml は、次の様な内容になります。ansible が、このファイルの内容を上から順番に処理し、仮想 OS 環境に設定変更実施しアプリケーションをインストールします。

リスト 1.3: playbook.yaml

```
1: ---
2: - hosts: all
3:
4:   user: vagrant
5:   tasks:
6:
```

```
 7:     - name: "update hosts"
 8:       copy: src=files/hosts
 9:             dest=/etc/hosts
10:             owner=root
11:             group=root
12:             mode=644
13:             backup=yes
14:       sudo: yes
15:
16:     - name: "apt-get install golang"
17:       apt:  pkg=golang
18:             update_cache=yes
19:             cache_valid_time=3600
20:       sudo: yes
21:
22:     - name: "apt-get install git"
23:       apt:  pkg=git
24:             update_cache=yes
25:             cache_valid_time=3600
26:       sudo: yes
27:
28:     - name: "apt-get install mercurial"
29:       apt:  pkg=mercurial
30:             update_cache=yes
31:             cache_valid_time=3600
32:       sudo: yes
33:
34:     - name: "apt-get install sqlite"
35:       apt:  pkg=sqlite
36:             update_cache=yes
37:             cache_valid_time=3600
38:       sudo: yes
39:
40:     - name: "apt-get install slibsqlite3-dev"
41:       apt:  pkg=libsqlite3-dev
42:             update_cache=yes
43:             cache_valid_time=3600
44:       sudo: yes
45:
46:     - name: "apt-get install language-pack-en-base"
47:       apt:  pkg=language-pack-en-base
48:       sudo: yes
49:
50:     - name: "apt-get install language-pack-ja-base"
51:       apt:  pkg=language-pack-ja-base
52:             update_cache=yes
53:             cache_valid_time=3600
54:       sudo: yes
55:
56:     - name: "apt-get install tree"
57:       apt:  pkg=tree
58:             update_cache=yes
59:             cache_valid_time=3600
60:       sudo: yes
61:
62:     - name: "update bashrc"
63:       copy: src=files/.bashrc
```

```
64:             dest=/home/vagrant/.bashrc
65:             owner=vagrant
66:             group=vagrant
67:             mode=644
68:             backup=yes
69:
70:     - name: "mkdir gocode"
71:       command:  mkdir /home/vagrant/gocode
72:                 creates=/home/vagrant/gocode
73:
74:     - name: "go get github.com/robfig/revel"
75:       shell:  export GOPATH=/home/vagrant/gocode && cd /home/vagrant/gocode
76:               && go get github.com/robfig/revel
77:               creates=/home/vagrant/gocode/src/github.com/robfig/revel
78:
79:     - name: "go get github.com/robfig/revel/revel"
80:       shell:  export GOPATH=/home/vagrant/gocode && cd /home/vagrant/gocode
81:               && go get github.com/robfig/revel/revel
82:               creates=/home/vagrant/gocode/bin/revel
83:
84:     - name: "go get github.com/coopernurse/gorp"
85:       shell:  export GOPATH=/home/vagrant/gocode && cd /home/vagrant/gocode
86:               && go get github.com/coopernurse/gorp
87:               creates=/home/vagrant/gocode/src/github.com/coopernurse/gorp
88:
89:     - name: "go get github.com/mattn/go-sqlite3"
90:       shell:  export GOPATH=/home/vagrant/gocode && cd /home/vagrant/gocode
91:               && go get github.com/mattn/go-sqlite3
92:               creates=/home/vagrant/gocode/src/github.com/mattn/go-sqlite3
93:
94:     - name:  "revel new myapp"
95:       shell:  export GOPATH=/home/vagrant/gocode && && cd /home/vagrant/gocode
96:               && revel  run myapp
97:               creates=/home/vagrant/gocode/src/myapp
```

ansbile では、各設定項目をタスクと呼びます。タスクは、次のような書式になります。

```
- name: [タスクの名前]
  [使用するモジュール]: [モジュールで規定された書式]
  [オプション]: [オプションで指定されて書式]
```

タスクの名前は、自由に付けても大丈夫です。使用するモジュールやオプションの詳細は、ansible ドキュメントサイト URL:http://docs.ansible.com/ を参考にしてください。

hosts の設定

この本を執筆している期間中何度か、ubuntu の package をアップデートするためのリポジトリーサーバの IP アドレスの解決が DNS 経由ではできなくなるケースがありました。期間中 DNS では安定しなかったため、ubuntu のリポジトリーサーバーの IP アドレス情報を/etc/hosts に静的に追

記し、その情報を基に apt-get のコマンドがリポジトリーサーバの名前解決をするようにしました。

仮想 OS 環境の/etc/hosts を上書きするためのファイルを転送するために ansible の copy モジュール使用します。

```
- name: "update hosts"
  copy: src=files/hosts
        dest=/etc/hosts
        owner=root
        group=root
        mode=644
        backup=yes
  sudo: yes
```

/etc/hosts は上書きは、root 権限が必要なので sudo の権限オプションも設定してます。

Go 言語のインストール

Go 言語のインストールは、ubuntu 標準のパッケージを使用することにします。従って、ansbile の apt モジュールを使ってインストールすることにします。

```
    - name: "apt-get install golang"
      apt:  pkg=golang
            update_cache=yes
            cache_valid_time=3600
      sudo: yes
```

update_cache, cache_valid_time はオプションの設定でパッケージの update の頻度をコントロールしています。

DB その他の deb パッケージのインストール

apt-get で Go 言語のパッケージをインストールした後は、Revel を利用するのに必要になるソフトウェアのパッケージをインストールしています。

各タスクの name の部分に設定した内容の作業を、それぞれ実行しています。

- apt-get install git
- apt-get install mercurial
- apt-get install sqlite
- apt-get install slibsqlite3-dev
- apt-get install language-pack-en-base(locale によるエラーの対策)
- apt-get install language-pack-ja-base(locale によるエラーの対策)
- apt-get install tree (ディレクトリの tree 表示させるため)

Revel インストール

Revel のインストールは、ドキュメントサイトの Getting Started ページを参考に進めていくことにします。

次は、仮想 OS に ssh でアクセスした際にの起動 Shell に GO 言語の環境 PATH が設定されいるように~/.bashrc を上書記します。.bashrc も、hosts と同様に copy モジュールを使って転送することにします。~/vagrant-env/files/ディレクトリに仮想 OS 環境へ転送するための.bashrc を設置しておきます。

```
- name: "update bashrc"
  copy: src=files/.bashrc
        dest=/home/vagrant/.bashrc
        owner=vagrant
        group=vagrant
        mode=644
        backup=yes
```

転送元の.bashrc は、ubuntu の一般的な.bashrc に下記の 3 行を追記したもになります。実際に変更した.bashrc のサンプルは付録 A に掲載してあります。

GOPATH は、GO 言語の環境 PATH で自由に決めることが出来ます。ここで決めた PATH に合わせて GO 言語用のディレクトリを作ることに成ります。又、$GOPATH/bin には、GO 言語のビルド後のバイナリーファイルが保存されます。従って、Shell の実行 PATH に、この bin ディレクトリーも追加しておきます。

```
# golang env
export GOPATH=/home/vagrant/gocode
export PATH=$PATH:/home/vagrant/gocode/bin
```

先に指定した GOPATH に基づいてディレクトリを作ります。既に該当のある場合は再度ディレクトリを作らないように、creates オプションを指定しておきます。

```
- name: "mkdir gocode"
  command:  mkdir /home/vagrant/gocode
            creates=/home/vagrant/gocode
```

次のブロックは、"go get github.com/robfig/revel"のシェルコマンドを実行しています。このシェルコマンドは、github より revel のソースを取得し、$GOPATH/src 以下に保存してくれます。既に該当のある場合は再度ディレクトリを作らないように、creates オプションを指定しておきます。

```
- name: "go get github.com/robfig/revel"
  shell:  export GOPATH=/home/vagrant/gocode && cd /home/vagrant/gocode
          && go get github.com/robfig/revel
          creates=/home/vagrant/gocode/src/github.com/robfig/revel
```

次のブロックは、"go get github.com/robfig/revel/revel"のシェルコマンドを実行しています。このシェルコマンドは、$GOPATH/bin 以下に、revel のバイナリを生成してくれます。既に該当のある場合は再度ディレクトリを作らないように、creates オプションを指定しておきます。

```
- name: "go get github.com/robfig/revel/revel"
  shell:  export GOPATH=/home/vagrant/gocode && cd /home/vagrant/gocode
          && go get github.com/robfig/revel/revel
          creates=/home/vagrant/gocode/bin/revel
```

BD を使用しない場合は、ここまで Revel の終了。

DB(Sqlite) を使用するための GO 言語関連パッケージのインストール

今回は、DB も使用したいので引き続き DB 関連パッケージをインストールすることにします。

次のブロックは、"go get github.com/coopernurse/gorp"のシェルコマンドを実行しています。このシェルコマンドは、GO 言語から DB を操作するための ORM のような gorp モジュールを github より取得し、$GOPATH/src 以下に保存してくれます。既に該当のある場合は再度ディレクトリを作らないように、creates オプションを指定しておきます。

```
- name: "go get github.com/coopernurse/gorp"
  shell:  export GOPATH=/home/vagrant/gocode && cd /home/vagrant/gocode
          && go get github.com/coopernurse/gorp
          creates=/home/vagrant/gocode/src/github.com/coopernurse/gorp
```

次のブロックは、"go get github.com/mattn/go-sqlite3"のシェルコマンドを実行しています。このシェルコマンドは、GO 言語から sqlite を使用するためのドライバーモジュールを github より取得し、$GOPATH/src 以下に保存してくれます。既に該当のある場合は再度ディレクトリを作らないように、creates オプションを指定しておきます。

```
- name: "go get github.com/mattn/go-sqlite3"
  shell:  export GOPATH=/home/vagrant/gocode && cd /home/vagrant/gocode
          && go get github.com/mattn/go-sqlite3
          creates=/home/vagrant/gocode/src/github.com/mattn/go-sqlite3
```

これで、DB も扱える環境も含めてインストールするための、Vagrantfile が完成しました。

1.2.3 Revel を起動

今回は、プロビジョンのプションを付けて仮想 OS を vagrant から起動します。

```
$ vagrant up --provision
```

起動の途中に、ssh で使用する RSA key に関する問合せがありますので、"yes"と入力してください。プロビジョンのプロセスが終了するまでには、しばらく時間がかかります。次のような実行結果が表示されれば終了です。

```
Bringing machine 'default' up with 'virtualbox' provider...
[default] Clearing any previously set forwarded ports...
[default] Clearing any previously set network interfaces...
[default] Preparing network interfaces based on configuration...
[default] Forwarding ports...
[default] -- 22 => 2222 (adapter 1)
[default] -- 9000 => 9000 (adapter 1)
[default] Booting VM...
[default] Waiting for machine to boot. This may take a few minutes...
[default] Machine booted and ready!
[default] Mounting shared folders...
[default] -- /vagrant
[default] Running provisioner: ansible...

PLAY [all] *********************************************************************

GATHERING FACTS ****************************************************************
The authenticity of host '[127.0.0.1]:2222 ([127.0.0.1]:2222)' can't be established.
RSA key fingerprint is e7:a6:12:c6:ef:17:90:c9:69:46:c0:4a:83:b8:fb:0c.
Are you sure you want to continue connecting (yes/no)? yes
ok: [default]

TASK: [update hosts] ***********************************************************
changed: [default]

TASK: [apt-get install golang] *************************************************
changed: [default]

TASK: [apt-get install git] ****************************************************
changed: [default]

TASK: [apt-get install mercurial] **********************************************
changed: [default]

TASK: [apt-get install sqlite] *************************************************
changed: [default]

TASK: [apt-get install slibsqlite3-dev] ****************************************
```

```
changed: [default]

TASK: [apt-get install language-pack-en-base] *********************************
changed: [default]

TASK: [apt-get install language-pack-ja-base] *********************************
changed: [default]

TASK: [apt-get install tree] **************************************************
changed: [default]

TASK: [update bashrc] *********************************************************
changed: [default]

TASK: [mkdir gocode] **********************************************************
changed: [default]

TASK: [go get github.com/robfig/revel] ****************************************
changed: [default]

TASK: [go get github.com/robfig/revel/revel] **********************************
changed: [default]

TASK: [go get github.com/coopernurse/gorp] ************************************
changed: [default]

TASK: [go get github.com/mattn/go-sqlite3] ************************************
changed: [default]

PLAY RECAP ********************************************************************
default                    : ok=16   changed=15   unreachable=0    failed=0
```

プロンプトが戻ってきたら、本体 OS から仮想 OS に ssh で接続します。

```
$ vagrant ssh
$ ls
```

先ほど確認した ubuntu のログイン画面が表示され、ls マンドを実行したところで gocode のディレクトリーが表示されていることを確認します。

gocode ディレクトリーに移動し、Revel を使って開発していくためのディレクトリー構成の雛形を生成することにします。

```
$ cd ~/gocode
$ revel new myapp
```

~/gocode/src/myapp ディレクトリ以下に次のような雛形が生成されました。

```
.
├── app
│   ├── controllers
│   │   └── app.go
│   ├── init.go
│   └── views
│       ├── App
│       │   └── Index.html
│       ├── debug.html
│       ├── errors
│       │   ├── 404.html
│       │   └── 500.html
│       ├── flash.html
│       ├── footer.html
│       └── header.html
├── conf
│   ├── app.conf
│   └── routes
├── messages
│   └── sample.en
├── public
│   ├── css
│   │   └── bootstrap.css
│   ├── img
│   │   ├── favicon.png
│   │   ├── glyphicons-halflings.png
│   │   └── glyphicons-halflings-white.png
│   └── js
│       └── jquery-1.9.1.min.js
└── tests
    └── apptest.go

12 directories, 18 files
```

myapp を起動してみます。

```
$ revel run myapp
```

本体 OS でブラウザーを起動し、http://localhost:9000 にアクセスします。次のような画面がブラウザーに表示されれば、始めての Revel は完成です。

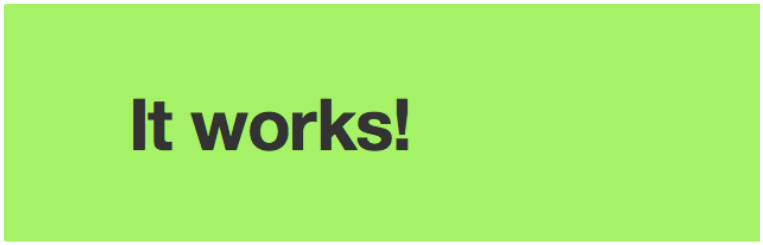

図 1.18　myapp 完成ページ

1.3　Github でソースコードを管理するための準備

仮想 OS 環境へのコードの転送や本体環境でのバージョン管理の利便性のために、Git によるコード管理をすることにします。

1.3.1　Git のインストール

次のサイトから OSX 用の Git のパッケージをダウントードしてください。ダウンロードしたパッケージをダブルクリックするとインストーラー画面が表示されます。画面の指示に従ってインストールを進めます。

```
http://code.google.com/p/git-osx-installer
```

Download タブをクリックすると、次の様なページが表示されますので、最新の Git インストーラを選択し、ダウンロートしてください。パッケージ毎のダウンロードページに移動します。

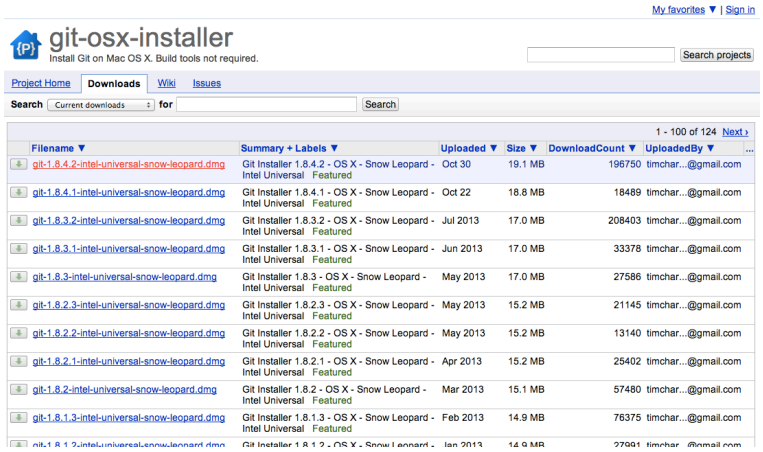

図 1.19　Git パッケージの選択ページ

パッケージのバージョンを確認し、名前をダブルクリックすることによってダウンロードを開始します。

図 1.20　Git パッケージのダウンロード確認

パッケージのダウンロードが完了したら、今取得したパッケージをダブルクリックしてインストールを開始します。

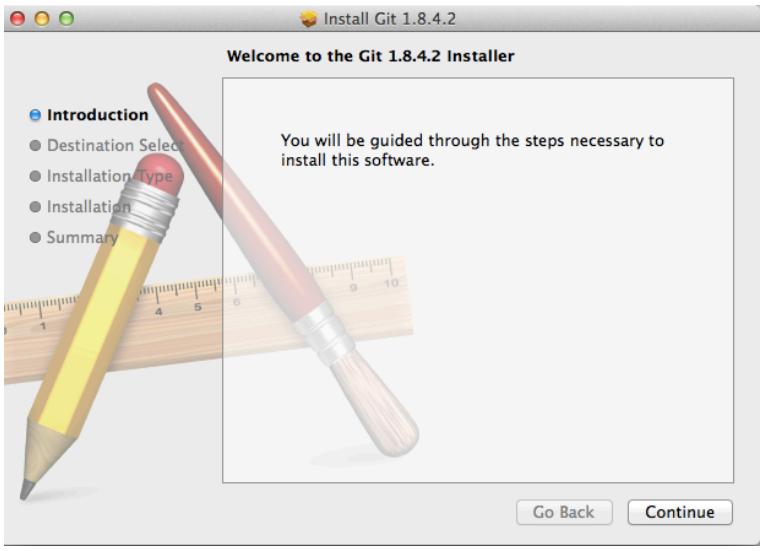

図 1.21　パッケージインストーラ起動

Git のインストールが完了したら、 ターミナルより Git の動作を確認します。

```
$ git --version
```

次のように、今インストールした Git のバージョンが表示されればインストール確認は完了です。

```
git version 1.8.3.4 (Apple Git-47)
```

1.3.2　Github へのユーザー登録

本体環境と仮想 OS 環境でコードを共有するためにはコードをクラウドに保管しておくと便利なので、Github にユーザー登録し、クラウド上にもコードも保存できるばしょを確保しておきます。

次が、Github の URL です。

```
http://github.com
```

次のような Github のランディングページが表示されます。

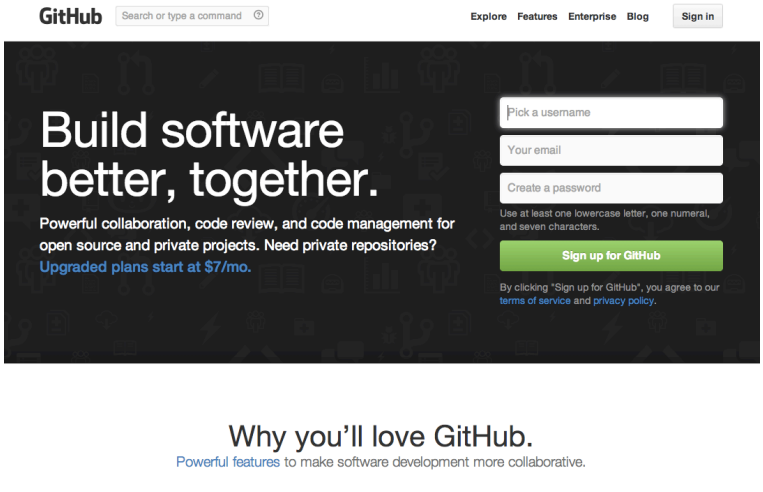

図 1.22　Github ランディングページ

右側の、入力欄に必要な情報を入力し、入力欄の右隅に緑のチェックマークが入っていることを確認したら、緑の"Sign up for Github"のボタンをクリックしてください。

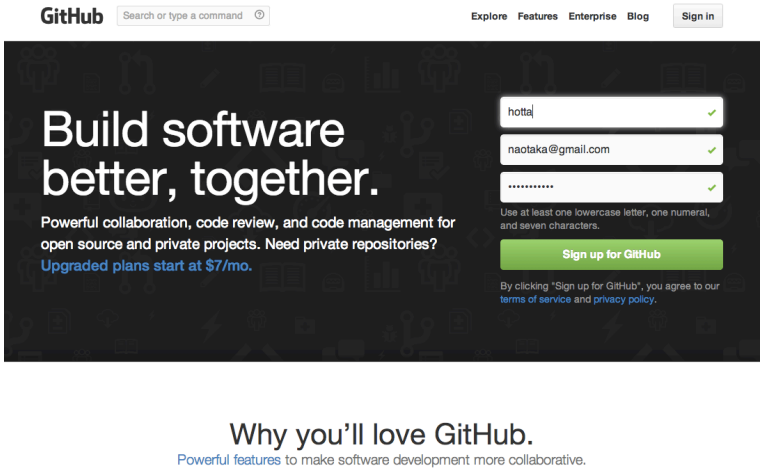

図 1.23　ユーザー情報入力状態

支払いプランを選択す画面が表示されます。"Free"(無償) のプランを選択してくしていることを確認し、緑の"Fnish sign up"ボタンをクリックします。

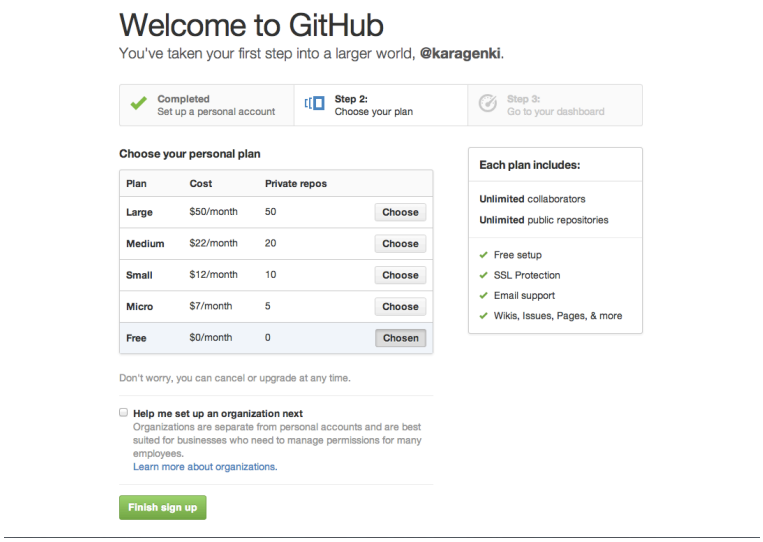

図 1.24　Github プラン選択ページ

次のような Github ダッシュボードが表示されれば仮登録は完了です。

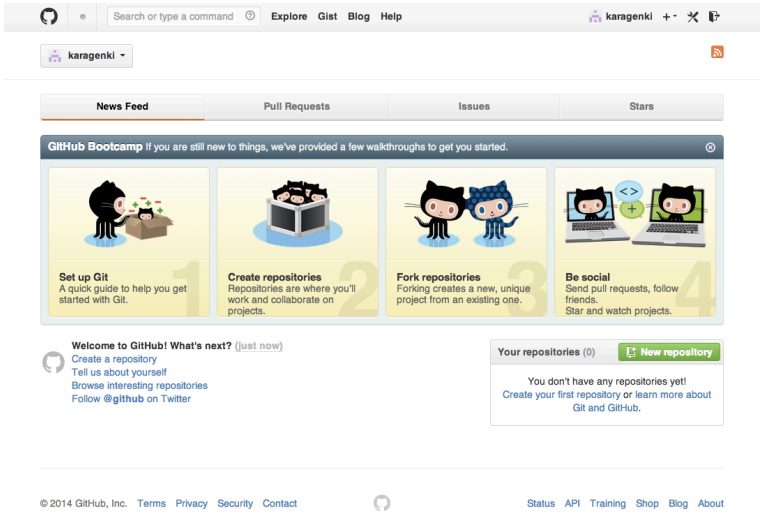

図 1.25　Github ダッシュボード

登録の際に入力したメールアドレスに、サービスをアクティベーションする内容のメールが届きます。そのメールに記載されている URL をクリックすると、アクティベーションが完了し、仮登録の状態から本登録の状態に移行します。

1.3.3　開発していくソースのレポジトリ作成

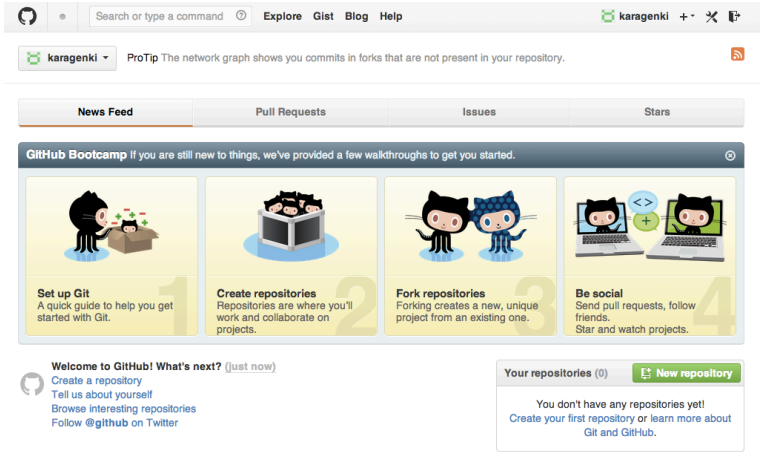

図 1.26　myapp 完成ページ

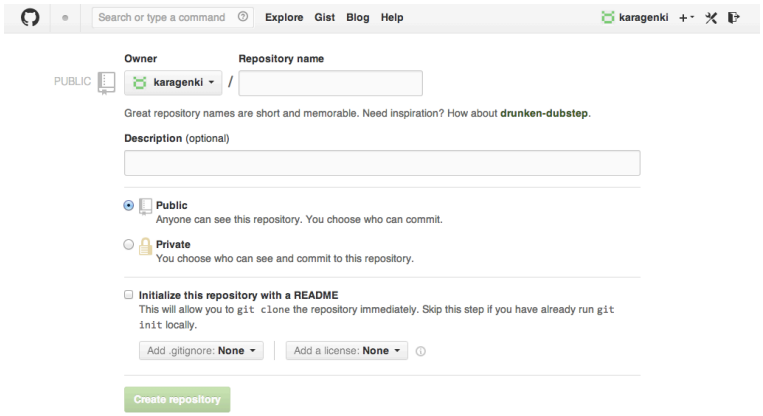

図 1.27　myapp 完成ページ

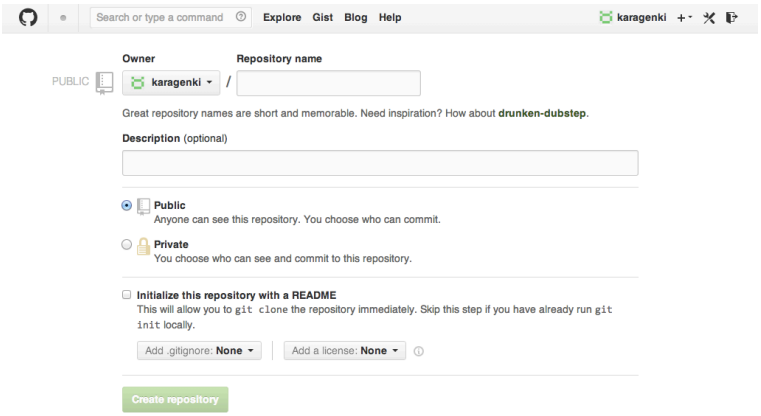

図 1.28　myapp 完成ページ

図 1.29　myapp 完成ページ

1.3.4　開発コードの最初にコミットまで

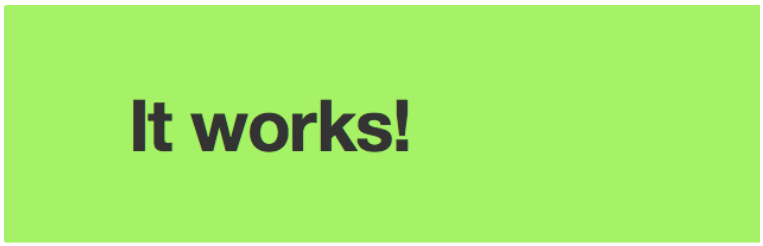

図 1.30　myapp 完成ページ

第 2 章

デモアプリケーション

Go 言語と Revel web framework が、実行できる環境を構築していきます。

図: 想い

2.1 VirtualBox と ubuntu のインストール

読者の皆さんの作業内容を統一するために今回は、virtualbox に Linux OS をインストールし、その上で作業を進めることにします。

2.1.1 VitualBox の準備

VirtualBox は、x86 仮想化ソフトウェア・パッケージの一つで、米国オラクル社によって開発がすすめられています。サポートされているホスト OS は Linux、Mac OSX、Windows、Solaris です。ゲスト OS としては、FreeBSD、Linux、OpenBSD、OS/2 Warp、Windows、Mac OS X Server、Solaris など x86/x64 アーキテクチャの OS であれば基本的には起動できます。GPL ver.2 で公開されている FOSS なので、無料で使用することが出来ます。

この本では、このソフトを利用し学習ベースとなる LinuxOS を起動することにします。VirtualBox は、以下の URL のページからダウンロードできます。目的のコンピューターの OS に適応した VirtualBox のパッケージをダウンロードしインストールしていください。

VirtualBox のダウンロードページ:

https://www.virtualbox.org/wiki/Downloads

2.1.2 ubuntu のインストール

今回は、VirtualBox 上の OS に ssh で接続し作業することにします。従って、ubuntu 13.10 64bit server 版をインストールすることにします。

インストールの詳細に関しては、○○○○○○○○○○○○○○○○○○してください。

ubuntu 13.10 server 64bit 版 ISO イメージのダウンロードページ:

http://www.ubuntu.com/download/server

2.2 Go 言語のインストール

VirtualBox 上に駆動している ubuntu の仮想マシンへ ssh を使ってアクセスします。

最初の段落です。この行も同じ段落です。

次の段落です。

２行以上以上空いていても１行空いているのと同様に処理します。

2.3 Revel web framework インストール

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて

見出しの先頭に「[column]」と書くと、そこはコラムになります。

2.4 実行に必要な環境変数の設定

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- 1 つ目
- 2 つ目
- 3 つ目

第 3 章

静的ページの作成

書き方のサンプルです。上書きするなりして消して下さい。

3.1　本文の書き方

最初の段落です。この行も同じ段落です。
次の段落です。
２行以上以上空いていても１行空いているのと同様に処理します。

3.1.1　見出し

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて
見出しの先頭に「[column]」と書くと、そこはコラムになります。

3.2　箇条書き

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- １つ目
- ２つ目
- ３つ目

番号つきの箇条書きには、「1.」「2.」などと書きます。数字の値は実際には無視され、連番が振られます。

1. １つ目
2. ２つ目

3. 3 つ目

3.3 ソースコードなどのリスト

リストには「//list」ブロックや「//emlist」ブロックを使います。

リスト 3.1: リストのサンプル

```
int main(int argc, char **argv) {
  puts("OK");
  return 0;
}
```

文中にリストを書くには「//emlist」になります。

```
def main
  puts "ok"
end
```

3.4 画像

画像は「//image」ブロックを使います。

```
--[[path =  (not exist)]]--
```

画像サンプル

より詳しくは、https://github.com/kmuto/review/blob/master/doc/format.rdoc を御覧ください。

第 4 章

Revel Framework で必要な Go 言語超基礎

書き方のサンプルです。上書きするなりして消して下さい。

4.1　本文の書き方

最初の段落です。この行も同じ段落です。

次の段落です。

２行以上以上空いていても１行空いているのと同様に処理します。

4.1.1　見出し

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて

見出しの先頭に「[column]」と書くと、そこはコラムになります。

4.2　箇条書き

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- １つ目
- ２つ目
- ３つ目

番号つきの箇条書きには、「1.」「2.」などと書きます。数字の値は実際には無視され、連番が振られます。

1. 1 つ目
2. 2 つ目
3. 3 つ目

4.3　ソースコードなどのリスト

リストには「//list」ブロックや「//emlist」ブロックを使います。

リスト 4.1: リストのサンプル

```
int main(int argc, char **argv) {
  puts("OK");
  return 0;
}
```

文中にリストを書くには「//emlist」になります。

```
def main
  puts "ok"
end
```

4.4　画像

画像は「//image」ブロックを使います。

```
--[[path =  (not exist)]]--
```

画像サンプル

より詳しくは、https://github.com/kmuto/review/blob/master/doc/format.rdoc を御覧ください。

第 5 章

レイアウトを作成する

書き方のサンプルです。上書きするなりして消して下さい。

5.1　本文の書き方

最初の段落です。この行も同じ段落です。
次の段落です。
２行以上以上空いていても１行空いているのと同様に処理します。

5.1.1　見出し

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて
見出しの先頭に「[column]」と書くと、そこはコラムになります。

5.2　箇条書き

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- １つ目
- ２つ目
- ３つ目

番号つきの箇条書きには、「1.」「2.」などと書きます。数字の値は実際には無視され、連番が振られます。

1. １つ目
2. ２つ目

3. 3 つ目

5.3 ソースコードなどのリスト

リストには「//list」ブロックや「//emlist」ブロックを使います。

リスト 5.1: リストのサンプル

```
int main(int argc, char **argv) {
  puts("OK");
  return 0;
}
```

文中にリストを書くには「//emlist」になります。

```
def main
  puts "ok"
end
```

5.4 画像

画像は「//image」ブロックを使います。

```
--[[path =  (not exist)]]--
```

画像サンプル

より詳しくは、https://github.com/kmuto/review/blob/master/doc/format.rdoc を御覧ください。

第 6 章

ユーザーのモデルを作成する

書き方のサンプルです。上書きするなりして消して下さい。

6.1　本文の書き方

最初の段落です。この行も同じ段落です。
次の段落です。
２行以上以上空いていても１行空いているのと同様に処理します。

6.1.1　見出し

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて
見出しの先頭に「[column]」と書くと、そこはコラムになります。

6.2　箇条書き

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- １つ目
- ２つ目
- ３つ目

番号つきの箇条書きには、「1.」「2.」などと書きます。数字の値は実際には無視され、連番が振られます。

1. １つ目
2. ２つ目

3. 3 つ目

6.3 ソースコードなどのリスト

リストには「//list」ブロックや「//emlist」ブロックを使います。

リスト 6.1: リストのサンプル

```
int main(int argc, char **argv) {
  puts("OK");
  return 0;
}
```

文中にリストを書くには「//emlist」になります。

```
def main
  puts "ok"
end
```

6.4 画像

画像は「//image」ブロックを使います。

```
--[[path =  (not exist)]]--
```

画像サンプル

より詳しくは、https://github.com/kmuto/review/blob/master/doc/format.rdoc を御覧ください。

第 7 章

ユーザー登録

書き方のサンプルです。上書きするなりして消して下さい。

7.1　本文の書き方

最初の段落です。この行も同じ段落です。
次の段落です。
２行以上以上空いていても１行空いているのと同様に処理します。

7.1.1　見出し

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて
見出しの先頭に「[column]」と書くと、そこはコラムになります。

7.2　箇条書き

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- １つ目
- ２つ目
- ３つ目

番号つきの箇条書きには、「1.」「2.」などと書きます。数字の値は実際には無視され、連番が振られます。

1. １つ目
2. ２つ目

3. 3 つ目

7.3 ソースコードなどのリスト

リストには「//list」ブロックや「//emlist」ブロックを使います。

リスト 7.1: リストのサンプル

```
int main(int argc, char **argv) {
  puts("OK");
  return 0;
}
```

文中にリストを書くには「//emlist」になります。

```
def main
  puts "ok"
end
```

7.4 画像

画像は「//image」ブロックを使います。

```
--[[path =  (not exist)]]--
```

画像サンプル

より詳しくは、https://github.com/kmuto/review/blob/master/doc/format.rdoc を御覧ください。

第 8 章

サインイン、サインアウト

書き方のサンプルです。上書きするなりして消して下さい。

8.1　本文の書き方

最初の段落です。この行も同じ段落です。
次の段落です。
２行以上以上空いていても１行空いているのと同様に処理します。

8.1.1　見出し

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて
見出しの先頭に「[column]」と書くと、そこはコラムになります。

8.2　箇条書き

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- １つ目
- ２つ目
- ３つ目

番号つきの箇条書きには、「1.」「2.」などと書きます。数字の値は実際には無視され、連番が振られます。

1. １つ目
2. ２つ目

3. 3 つ目

8.3 ソースコードなどのリスト

リストには「//list」ブロックや「//emlist」ブロックを使います。

リスト 8.1: リストのサンプル

```
int main(int argc, char **argv) {
  puts("OK");
  return 0;
}
```

文中にリストを書くには「//emlist」になります。

```
def main
  puts "ok"
end
```

8.4 画像

画像は「//image」ブロックを使います。

```
--[[path =  (not exist)]]--
```

画像サンプル

より詳しくは、https://github.com/kmuto/review/blob/master/doc/format.rdoc を御覧ください。

第 9 章

ユーザーの更新・表示・削除

書き方のサンプルです。上書きするなりして消して下さい。

9.1　本文の書き方

最初の段落です。この行も同じ段落です。
次の段落です。
２行以上以上空いていても１行空いているのと同様に処理します。

9.1.1　見出し

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて
見出しの先頭に「[column]」と書くと、そこはコラムになります。

9.2　箇条書き

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- １つ目
- ２つ目
- ３つ目

番号つきの箇条書きには、「1.」「2.」などと書きます。数字の値は実際には無視され、連番が振られます。

1. １つ目
2. ２つ目

3. 3 つ目

9.3 ソースコードなどのリスト

リストには「//list」ブロックや「//emlist」ブロックを使います。

リスト 9.1: リストのサンプル

```
int main(int argc, char **argv) {
  puts("OK");
  return 0;
}
```

文中にリストを書くには「//emlist」になります。

```
def main
  puts "ok"
end
```

9.4 画像

画像は「//image」ブロックを使います。

```
--[[path =  (not exist)]]--
```

画像サンプル

より詳しくは、https://github.com/kmuto/review/blob/master/doc/format.rdoc を御覧ください。

第 10 章

ユーザーのマイクロポスト

書き方のサンプルです。上書きするなりして消して下さい。

10.1　本文の書き方

最初の段落です。この行も同じ段落です。

次の段落です。

２行以上以上空いていても１行空いているのと同様に処理します。

10.1.1　見出し

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて

見出しの先頭に「[column]」と書くと、そこはコラムになります。

10.2　箇条書き

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- １つ目
- ２つ目
- ３つ目

番号つきの箇条書きには、「1.」「2.」などと書きます。数字の値は実際には無視され、連番が振られます。

1. １つ目
2. ２つ目

3. 3 つ目

10.3 ソースコードなどのリスト

リストには「//list」ブロックや「//emlist」ブロックを使います。

リスト 10.1: リストのサンプル

```
int main(int argc, char **argv) {
  puts("OK");
  return 0;
}
```

文中にリストを書くには「//emlist」になります。

```
def main
  puts "ok"
end
```

10.4 画像

画像は「//image」ブロックを使います。

```
--[[path =  (not exist)]]--
```

画像サンプル

より詳しくは、https://github.com/kmuto/review/blob/master/doc/format.rdoc を御覧ください。

第 11 章

ユーザーをフォローする

書き方のサンプルです。上書きするなりして消して下さい。

11.1　本文の書き方

最初の段落です。この行も同じ段落です。
次の段落です。
２行以上以上空いていても１行空いているのと同様に処理します。

11.1.1　見出し

「=」「==」「===」の後に一文字空白をあけると見出しになります。

■コラム: コラムについて
見出しの先頭に「[column]」と書くと、そこはコラムになります。

11.2　箇条書き

番号のない箇条書きは「*」を使います。前後に空白を入れて下さい。

- １つ目
- ２つ目
- ３つ目

番号つきの箇条書きには、「1.」「2.」などと書きます。数字の値は実際には無視され、連番が振られます。

1. １つ目
2. ２つ目

3. 3 つ目

11.3 ソースコードなどのリスト

リストには「//list」ブロックや「//emlist」ブロックを使います。

リスト 11.1: リストのサンプル

```
int main(int argc, char **argv) {
  puts("OK");
  return 0;
}
```

文中にリストを書くには「//emlist」になります。

```
def main
  puts "ok"
end
```

11.4 画像

画像は「//image」ブロックを使います。

```
--[[path =  (not exist)]]--
```

画像サンプル

より詳しくは、https://github.com/kmuto/review/blob/master/doc/format.rdoc を御覧ください。

第 12 章

Test, CI, CD

サンプルその 2
基本はサンプルと同様です。
- 2asdfa - asdfasdfa - asdfasdfs

付録 A

asdfasdfas

A.1　hosts　ファイルサンプル

リスト 1.2: bashrc のサンプル

```
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
127.0.1.1 vagrant-ubuntu-saucy-64
91.189.92.200 archive.ubuntu.com
91.189.92.200 security.ubuntu.com
192.30.252.131 github.com
74.125.235.129 code.google.com
```

A.2　.bashrc ファイルサンプル

リスト 1.2: bashrc のサンプル

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
```

```
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# If set, the pattern "**" used in a pathname expansion context will
# match all files and zero or more directories and subdirectories.
#shopt -s globstar

# make less more friendly for non-text input files, see lesspipe(1)
[ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"

# set variable identifying the chroot you work in (used in the prompt below)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
    xterm-color) color_prompt=yes;;
esac

# uncomment for a colored prompt, if the terminal has the capability; turned
# off by default to not distract the user: the focus in a terminal window
# should be on the output of commands, not on the prompt
#force_color_prompt=yes

if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
  # We have color support; assume it's compliant with Ecma-48
  # (ISO/IEC-6429). (Lack of such support is extremely rare, and such
  # a case would tend to support setf rather than setaf.)
  color_prompt=yes
    else
  color_prompt=
    fi
fi

if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
```

```
    ;;
*)
    ;;
esac

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# golang env
export GOPATH=/home/vagrant/gocode
export PATH=$PATH:/home/vagrant/gocode/bin
```

付録 B

poiupoiuiopu

B.1 lkjlkjlkjklajfkl;sj

B.2 kljasdfkljasweoriwpoierw

■コラム:
初回の起動時のみ下記のような ERROR メッセージが表示されます。

```
ERROR 2014/03/03 07:32:04 build.go:84: src/revelFramework4Go/sampleBlogSite/app/contr
  /usr/lib/go/src/pkg/code.google.com/p/go.crypto/bcrypt (from $GOROOT)
  /home/vagrant/gocode/src/code.google.com/p/go.crypto/bcrypt (from $GOPATH)
```

このエラーメッセージは、パスワードの暗号化のために指定した bcrypt のバイナリーが gocode/pkg ディレクトリ以下に存在していないために、表示されています。

```
.
└── linux_amd64
    ├── code.google.com
    │   └── p
    │       └── go.net
```

revel を実行した後に先のディレクトリを確認すると、"go.crypto"が生成されていることが確認できます。次回以降は ERROR メッセージが表示されること無く Revel web framework を起動することが出来るようになります。

```
└──── linux_amd64
    ├──── code.google.com
    │  └──── p
    │      ├──── go.crypto
    │      └──── go.net
```

Revel Web Framework 入門

2014 年 05 月 03 日 v1.0.0 版発行

著　者　堀田直孝
編集者　堀田直孝
発行所　Hungry Foolish Gray Brain.