

Vagrant 勉強会

2015
11/14, 12/13

Present by Koji Yamaoka

初期ファイル等

https://github.com/kozyszoo/TK_study

勉強会の目標

- 仮想マシンの構築を体験し、サーバーサイドの基礎処理を学ぶ
- アプリケーションをさらに実用的に動かすための知識を身につける

そのための 第一歩を踏み出す ことを勉強会の目標とする

注意事項

今回は、ローカルPC内に仮想的に Web サーバ を構築します。

実際に インターネット上に公開させたい場合 は
AWS, cloud nなどのホスティングサービスを
利用する必要があるのでご注意ください。

これらに関しては有料の場合が多いですが、期間限定で無料という
場合もあります(これについては後で詳しく説明します)

それでは講義の方に入っていきます

今日やること(1)



仮想サーバを構築しよう

Sinatra でサーバサイドプログラミング

iPhone アプリと連携しよう

今日やること(2)



13-inch MacBook Air



今日やること(3)



13-inch MacBook Air

目次



仮想サーバを構築しよう

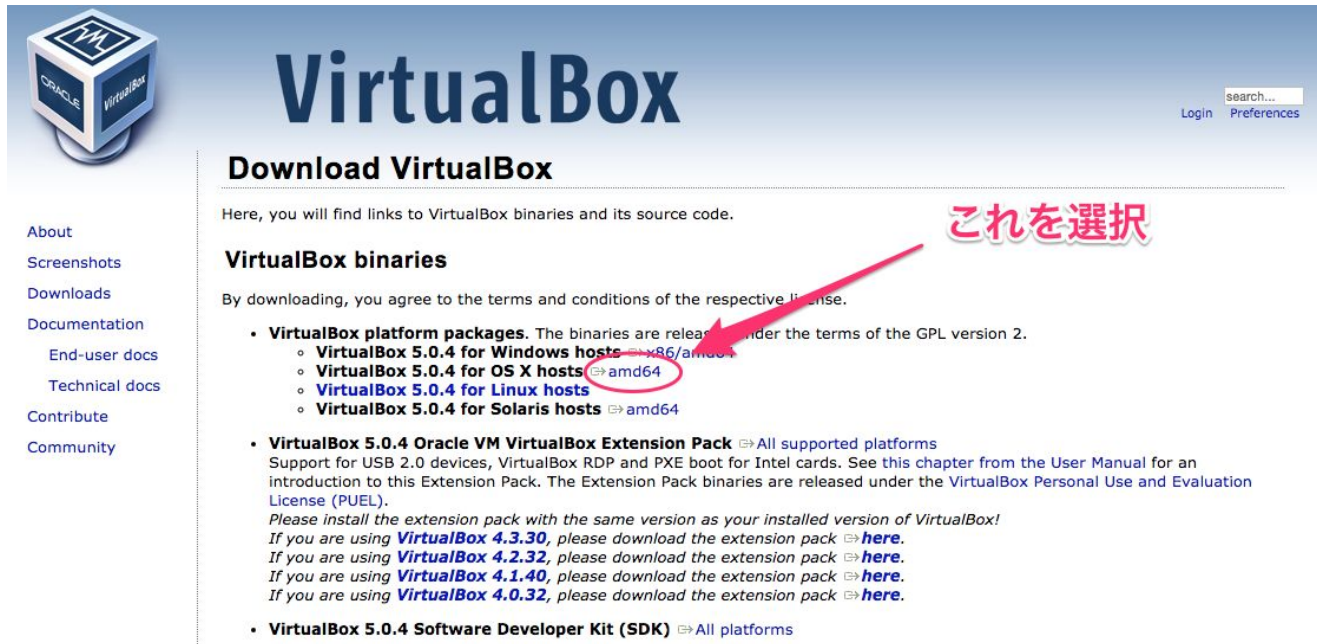
Sinatra でサーバサイドプログラミング

iPhone アプリと連携しよう

VirtualBox をインストールしよう

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

(今回は USB 経由でファイルを渡します)



VirtualBox

Download VirtualBox

Here, you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

- **VirtualBox platform packages.** The binaries are released under the terms of the GPL version 2.
 - [VirtualBox 5.0.4 for Windows hosts](#) (x86/amd64)
 - [VirtualBox 5.0.4 for OS X hosts \(amd64\)](#)
 - [VirtualBox 5.0.4 for Linux hosts](#) (amd64)
 - [VirtualBox 5.0.4 for Solaris hosts](#) (amd64)
- **VirtualBox 5.0.4 Oracle VM VirtualBox Extension Pack** (All supported platforms)
Support for USB 2.0 devices, VirtualBox RDP and PXE boot for Intel cards. See [this chapter](#) from the User Manual for an introduction to this Extension Pack. The Extension Pack binaries are released under the VirtualBox Personal Use and Evaluation License (PUEL).
Please install the extension pack with the same version as your installed version of VirtualBox!
If you are using **VirtualBox 4.3.30**, please download the extension pack [here](#).
If you are using **VirtualBox 4.2.32**, please download the extension pack [here](#).
If you are using **VirtualBox 4.1.40**, please download the extension pack [here](#).
If you are using **VirtualBox 4.0.32**, please download the extension pack [here](#).
- **VirtualBox 5.0.4 Software Developer Kit (SDK)** (All platforms)

VirtualBox とは

VirtualBox とは仮想マシン(PC内に作成できるミニPC のようなもの)を
立ち上げるための Oracle 社製のアプリケーションのことです。
今回はVirtualBox を用いて(仮想)サーバを作成します。



13-inch MacBook Air

サーバとは何か

外部から要求があった時に、その要求に対する返答を行うコンピュータ

データなどを Serve (提供) する専用のコンピュータ



仮想サーバとは何か

外部から要求があった時に、その要求に対する返答を行うコンピュータ

データなどを Serve (提供) する専用のコンピュータ

今回はそんなサーバを PC内にミニPCとして作成する、それが仮想サーバ



13-inch MacBook Air



Vagrant をインストールしよう

<https://www.vagrantup.com/downloads.html>

(今回は USB 経由でもファイルを渡します)



The screenshot shows the Vagrant website's download page. The header includes the Vagrant logo and navigation links: VMWARE INTEGRATION, DOWNLOADS, DOCUMENTATION, BLOG, and ABOUT. The left sidebar has a 'DOWNLOAD' section with links for 'Latest' and 'Old Versions'. The main content area is titled 'DOWNLOAD VAGRANT' and contains a paragraph of instructions. Below this, there are three operating system options: MAC OS X, WINDOWS, and LINUX (DEB). The MAC OS X option is highlighted with a red circle and a red arrow pointing to it, with the text 'ここを選択' (Select here) next to it. The text 'Universal (32 and 64-bit)' is underlined for each option.

VAGRANT

VMWARE INTEGRATION DOWNLOADS DOCUMENTATION BLOG ABOUT

DOWNLOAD

Latest

Old Versions

DOWNLOAD VAGRANT

Below are all available downloads for the latest version of Vagrant (1.7.4). Please download the proper package for your operating system and architecture. You can find SHA256 checksums for packages [here](#), and you can find the version changelog [here](#).

 **MAC OS X**
Universal (32 and 64-bit)

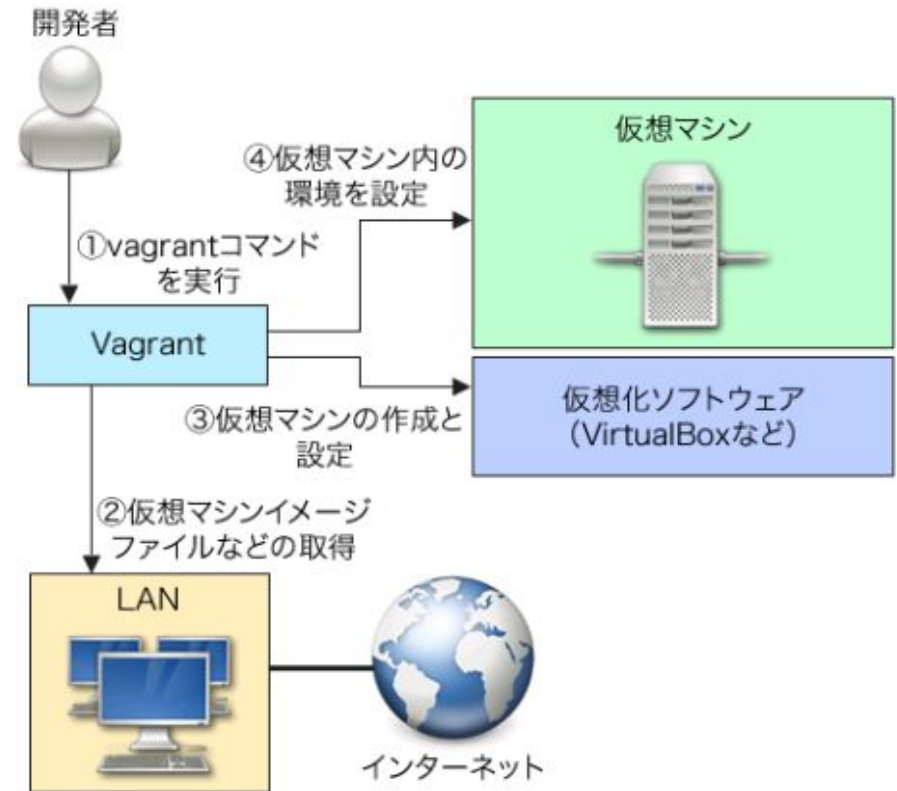
 **WINDOWS**
Universal (32 and 64-bit)

 **LINUX (DEB)**
32-bit | 64-bit

ここを選択

Vagrant とは

Vagrant とは仮想マシン(ミニPC)を立ち上げる上で、設定ファイルに記述した通りの設定で自動的に仮想マシンを立ち上げさせるアプリケーションのことである。



<https://osdn.jp/magazine/15/02/13/200000>

今回は Vagrant で CentOS を立ち上げます



13-inch MacBook Air

CentOS とは？

CentOS とは Linux に分類される
無料で使える OS（オペレーションシステム）のことです。

他の OS には有料ではあるが

- Windows 8, Windows 10
- (Mac) OSX

などが存在する。



CentOS

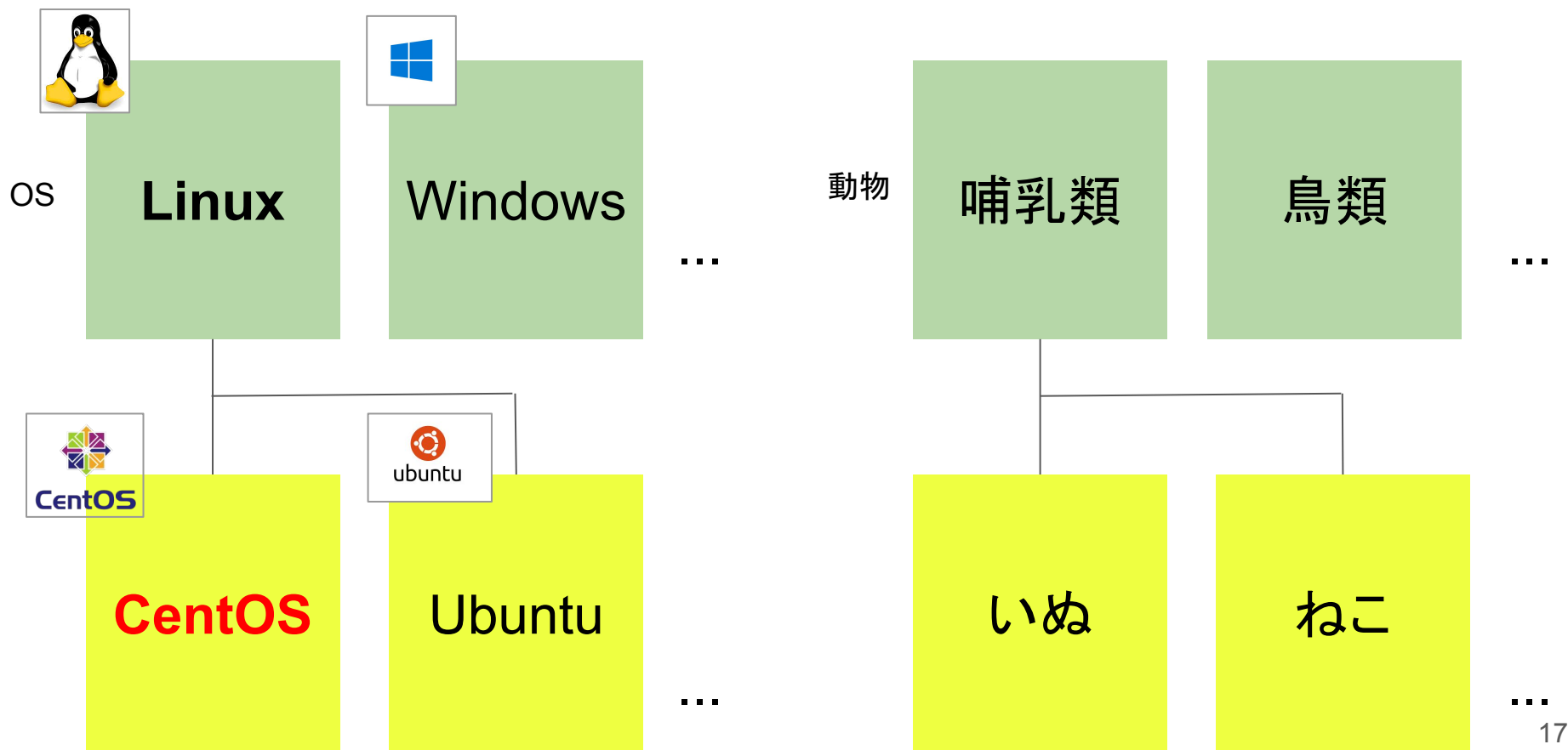
OS とはコンピュータを扱うための大きなくくりでのソフトウェアである。

この中でも CentOS は無料で扱え、古くから使われており

安定性が高く、数ある Linux の OS の中でも特に一般的な OS である。

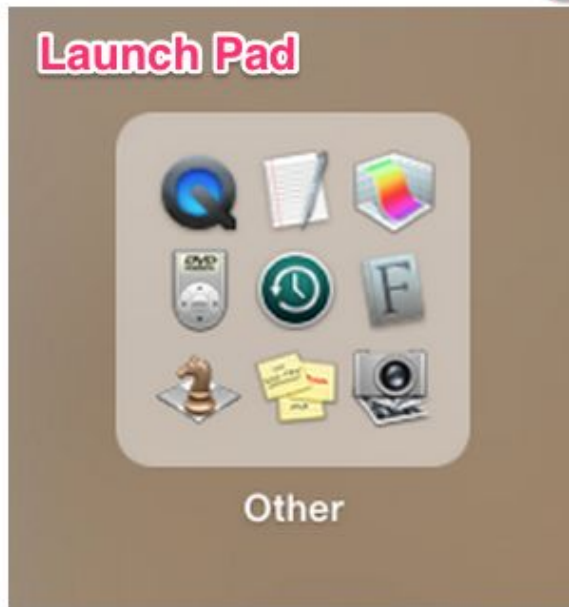
CentOS の位置づけ

Linux 系に分類される OS (オペレーションシステム)
様々な団体によって Linux 系の OS は開発・サポートされている



仮想サーバの構築を始めよう

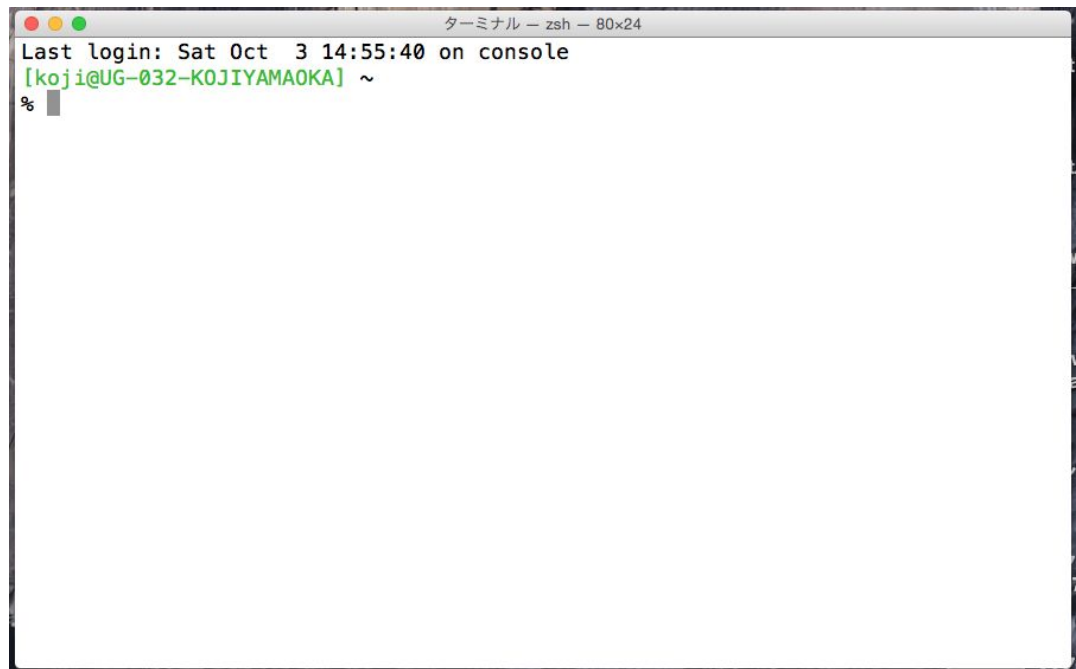
ターミナルを開こう



これを開く



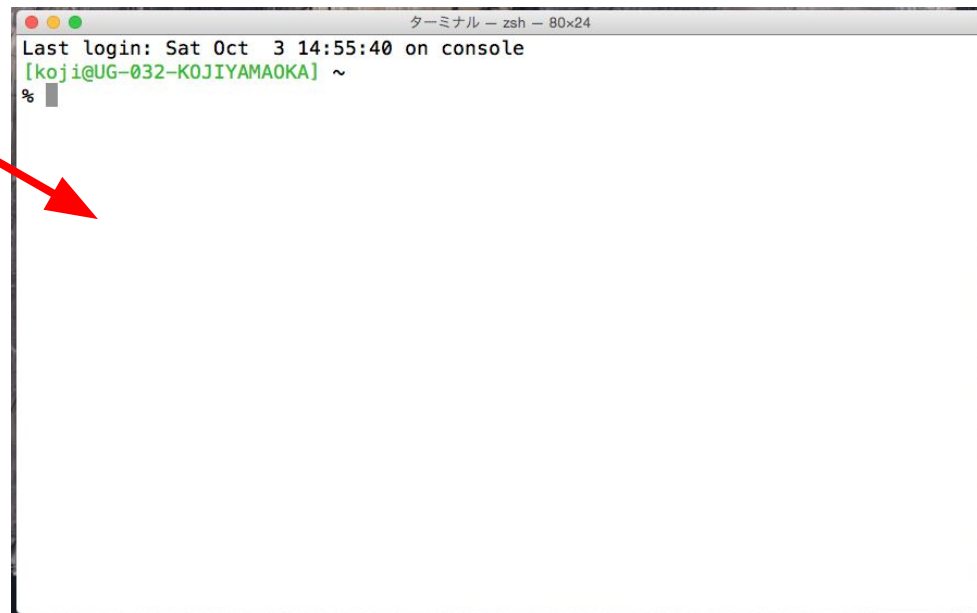
ターミナル画面は開けましたか？

A screenshot of a terminal window titled "ターミナル - zsh - 80x24". The window shows the output of a login session. The text "Last login: Sat Oct 3 14:55:40 on console" is displayed. Below it, the prompt "[kaji@UG-032-KOJIYAMAOKA] ~" is shown in green. The cursor is at the end of the prompt, indicated by a small black rectangle.

```
ターミナル - zsh - 80x24
Last login: Sat Oct 3 14:55:40 on console
[kaji@UG-032-KOJIYAMAOKA] ~
% 
```

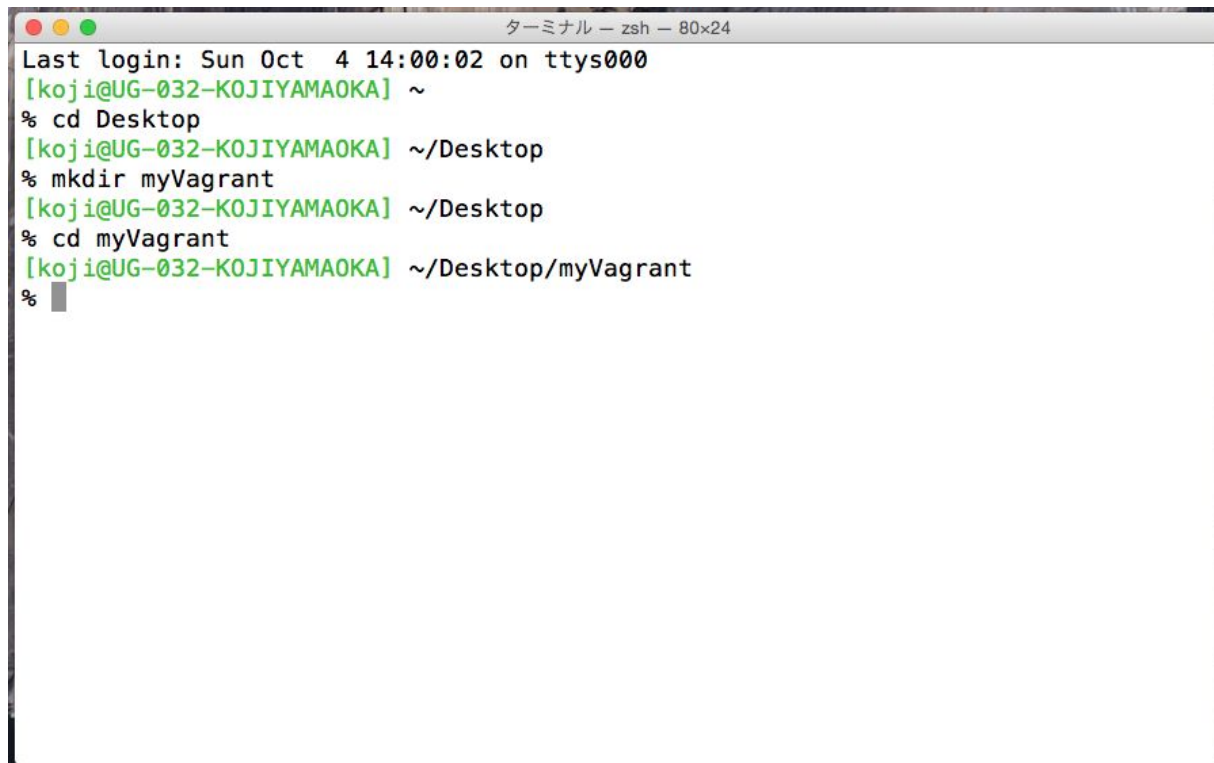
試しに入力してみよう

```
$ cd Desktop (Enter)  
$ mkdir myVagrant (Enter)  
$ cd myVagrant (Enter)
```



※ \$ は入力しないこと

こんな感じになれば OK

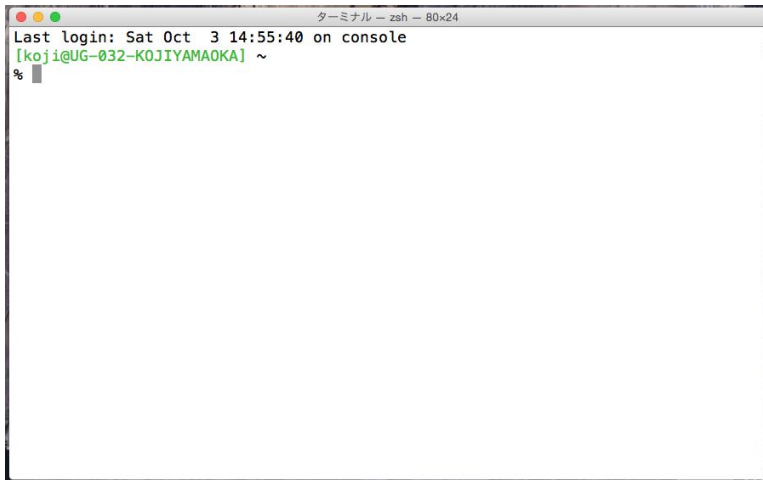
A terminal window titled "ターミナル - zsh - 80x24" showing a sequence of commands and their outputs. The user is logged in as koji on a system named UG-032-KOJIYAMA0KA. The commands executed are: 'cd Desktop', 'mkdir myVagrant', and 'cd myVagrant'. The outputs show the current directory changing from '~' to '~/Desktop' and then to '~/Desktop/myVagrant'.

```
ターミナル - zsh - 80x24
Last login: Sun Oct  4 14:00:02 on ttys000
[koji@UG-032-KOJIYAMA0KA] ~
% cd Desktop
[koji@UG-032-KOJIYAMA0KA] ~/Desktop
% mkdir myVagrant
[koji@UG-032-KOJIYAMA0KA] ~/Desktop
% cd myVagrant
[koji@UG-032-KOJIYAMA0KA] ~/Desktop/myVagrant
% █
```

ターミナル(CLI)について学ぼう

CLI

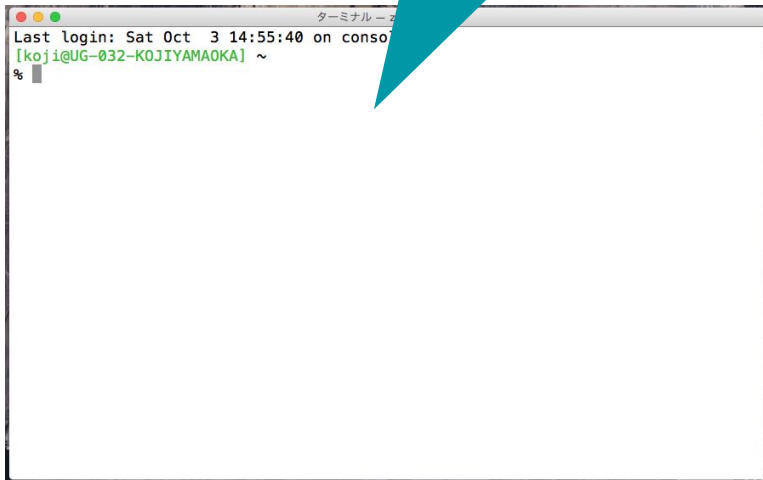
GUI



ターミナル(CLI)について学ぼう

CLI

文字のみの世界
(キーボードのみ)



GUI

画像のあるの世界
(キーボード、マウス)

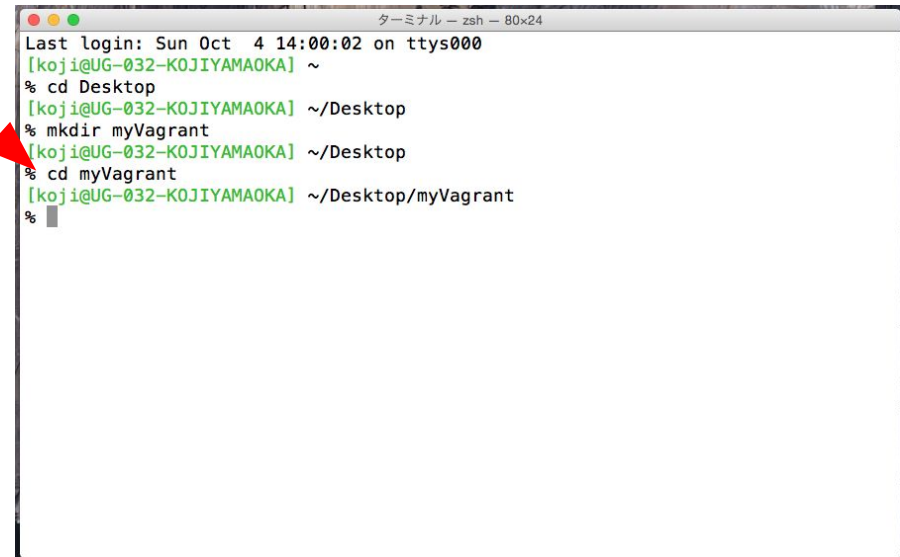


vagrant add で OS Image を加えよう

```
$ vagrant box add centos sinatra_package.box (Enter)
```

CentOS.box

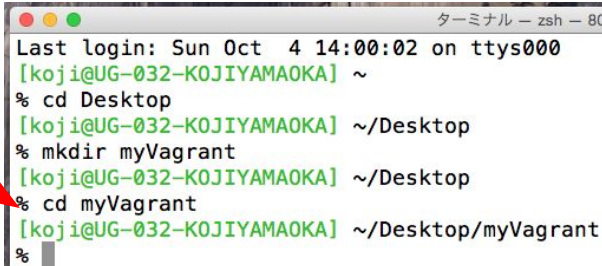
vagrant 上では box ファイルとして
設定情報を保存しておくことができる



```
ターミナル - zsh - 80x24
Last login: Sun Oct  4 14:00:02 on ttys000
[koji@UG-032-KOJIYAMAOKA] ~
% cd Desktop
[koji@UG-032-KOJIYAMAOKA] ~/Desktop
% mkdir myVagrant
[koji@UG-032-KOJIYAMAOKA] ~/Desktop
% cd myVagrant
[koji@UG-032-KOJIYAMAOKA] ~/Desktop/myVagrant
%
```

vagrant init を実行してみよう

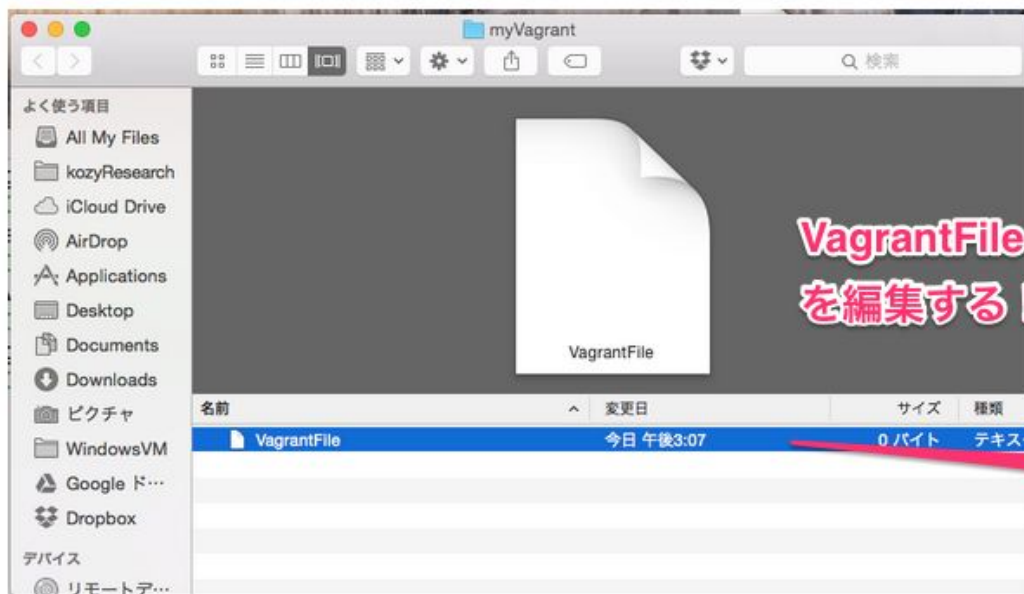
\$ vagrant init centos
(Enter)



A terminal window titled "ターミナル - zsh - 80x24" showing the execution of the command "vagrant init centos". The output shows the user's shell history: "Last login: Sun Oct 4 14:00:02 on ttys000", "[koi@UG-032-KOJIYAMA0KA] ~", "% cd Desktop", "[koi@UG-032-KOJIYAMA0KA] ~/Desktop", "% mkdir myVagrant", "[koi@UG-032-KOJIYAMA0KA] ~/Desktop", "% cd myVagrant", and "[koi@UG-032-KOJIYAMA0KA] ~/Desktop/myVagrant". A red arrow points from the text "(Enter)" in the previous block to the terminal window.

```
ターミナル - zsh - 80x24
Last login: Sun Oct 4 14:00:02 on ttys000
[koi@UG-032-KOJIYAMA0KA] ~
% cd Desktop
[koi@UG-032-KOJIYAMA0KA] ~/Desktop
% mkdir myVagrant
[koi@UG-032-KOJIYAMA0KA] ~/Desktop
% cd myVagrant
[koi@UG-032-KOJIYAMA0KA] ~/Desktop/myVagrant
% 
```

Vagrantfile ファイルを編集しよう



Vagrantfile ファイルを編集しよう

Vagrantfile 29行目

```
# config.vm.network "private_network", ip: "192.168.33.10"
```

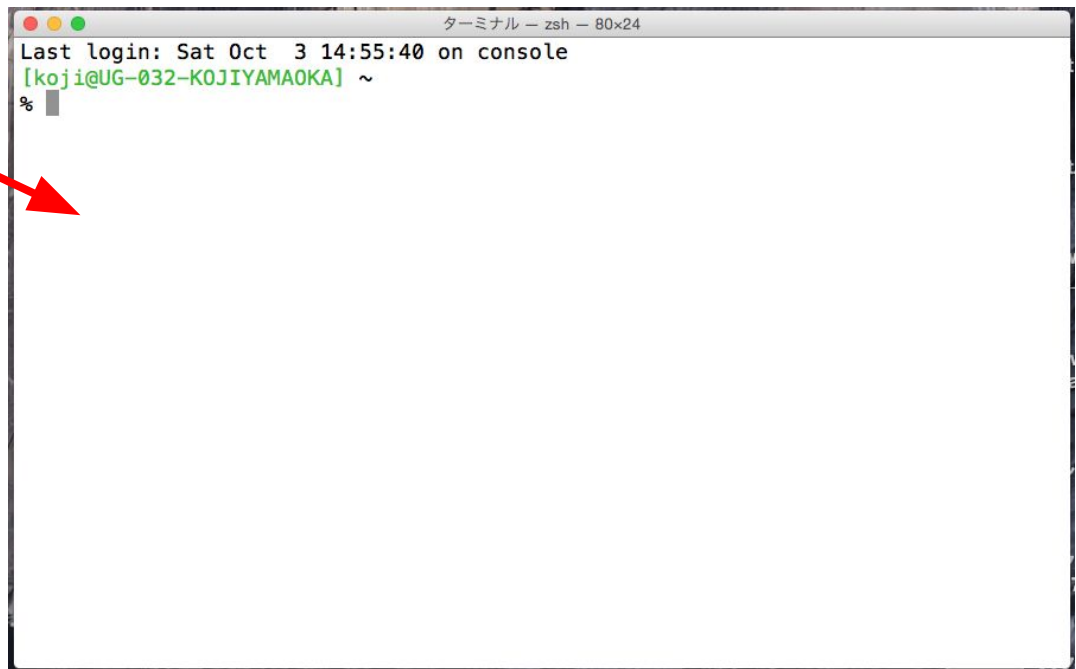


Vagrantfile 29行目

```
config.vm.network "private_network", ip: "192.168.33.10"
```

vagrant up で仮想サーバを起動してみよう

\$ vagrant up (Enter)



A terminal window titled "ターミナル - zsh - 80x24" showing the output of the `vagrant up` command. The output indicates a successful login for the user `koji` on the host `UG-032-KOJIYAMAOKA`. A red arrow points from the `vagrant up` command in the text box to the terminal window.

```
ターミナル - zsh - 80x24
Last login: Sat Oct  3 14:55:40 on console
[koji@UG-032-KOJIYAMAOKA] ~
%
```

こんな表示が出れば OK !!

```
❖ ~/Desktop/myVagrant/ vagrant up
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: Adapter 1: nat
==> default: Forwarding ports...
default: 22 => 2222 (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
default: SSH address: 127.0.0.1:2222
default: SSH username: vagrant
default: SSH auth method: private key
default: Warning: Connection timeout. Retrying...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
default: The guest additions on this VM do not match the installed version of
default: VirtualBox! In most cases this is fine, but in rare cases it can
default: prevent things such as shared folders from working properly. If you see
default: shared folder errors, please make sure the guest additions within the
default: virtual machine match the version of VirtualBox you have installed on
default: your host and reload your VM.
default:
default: Guest Additions Version: 4.2.12
default: VirtualBox Version: 4.3
==> default: Mounting shared folders...
default: /vagrant => /Users/koji/Desktop/myVagrant
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
```

これで **完了** です。**完了** なんです。

おめでとうございます、貴方用の仮想サーバが立ち上がりました！！
あっという間すぎて、何が起きたの？という感じだと思うので解説します。

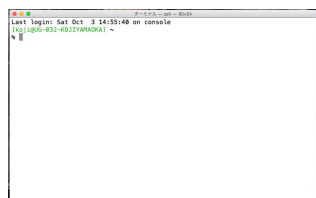
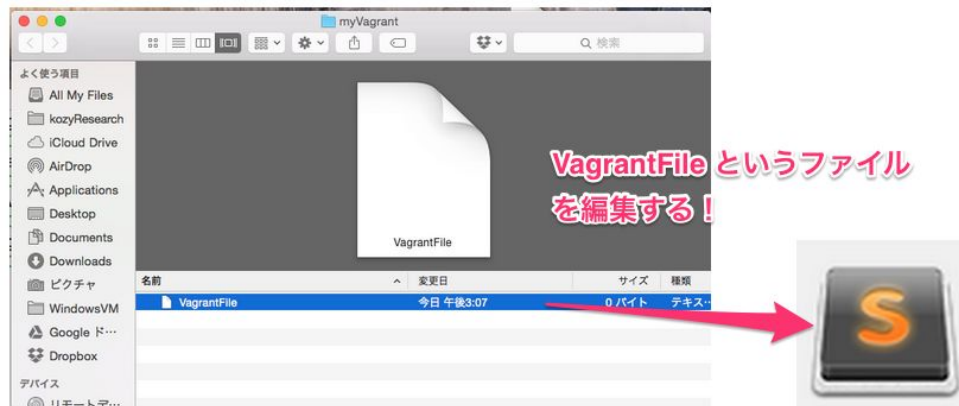
仮想サーバ起動の流れ

\$ vagrant init

- ・Vagrantfile ファイルを作成

\$ vagrant up

- ・Vagrantfile に基づいて、
仮想コンピュータを構築する



ターミナル

**Vagrant
init**

Vagrantfile
(設計書)

Vagrant up

仮想サーバ
(CentOS)

サーバ仮想化



13-inch MacBook Air

サーバ仮想化



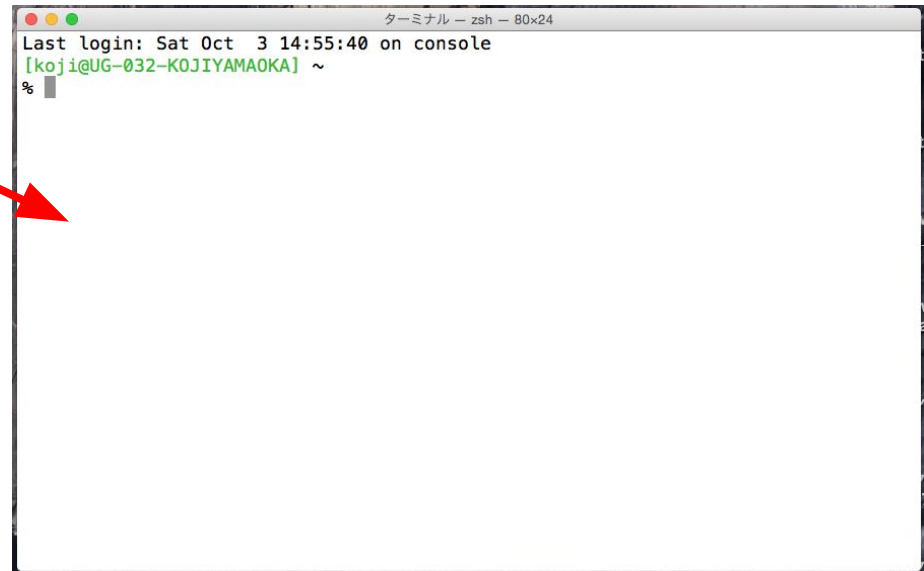
13-inch MacBook Air

サーバ仮想化 (SSHアクセス)



vagrant ssh で仮想サーバに接続しよう

\$ vagrant ssh (Enter)



A terminal window titled "ターミナル - zsh - 80x24" showing the output of the `vagrant ssh` command. The output indicates a successful login to a virtual machine named "K0JIYAMA0KA".

```
Last login: Sat Oct 3 14:55:40 on console  
[k0ji@UG-032-K0JIYAMA0KA] ~  
% 
```

vagrant ssh で仮想サーバに接続しよう

\$ vagrant ssh (Enter)

A terminal window titled "ターミナル - zsh - 80x24" showing the output of the 'vagrant ssh' command. The text in the terminal is: "Last login: Sat Oct 3 14:55:40 on console", "[kaji@UG-032-KOJIYAMA0KA] ~", and a prompt "%". A red arrow points from the "(Enter)" in the command box to the terminal window.

```
ターミナル - zsh - 80x24
Last login: Sat Oct 3 14:55:40 on console
[kaji@UG-032-KOJIYAMA0KA] ~
%
```

※ ssh (接続)とは？

安全性の高い通信形式、及びその方法。これがあると遠隔のコンピュータ(サーバ)に安全にアクセス出来て、操作することが可能となります。

目次



仮想サーバを構築しよう

Sinatra でサーバサイドプログラミング

iPhone アプリと連携しよう

Ruby に触れてみよう！

Ruby とはまつもとゆきひろ氏によって作成されたプログラミング言語。

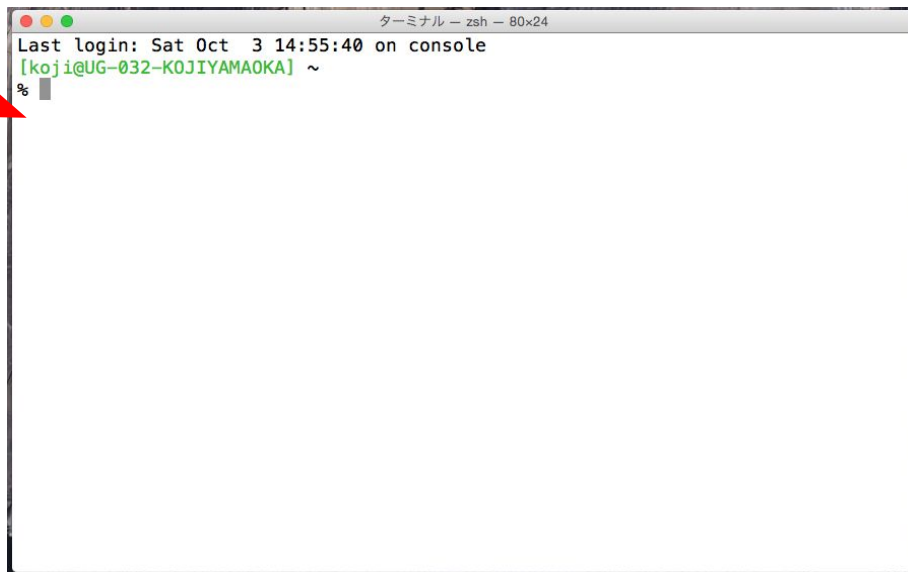
可読性(読みやすさ)を重視した構文となっており、
Ruby においては整数や文字列なども含め
データ型はすべてがオブジェクトであり純粋なオブジェクト指向言語といえる。

ここでは、Ruby を用いて Sinatra を動かす前に、
この Ruby にも少し触れてみたいと思います。



Ruby のバージョンを確認しよう

```
$ ruby -v (Enter)  
ruby 2.0.0p647 (2015-08-08  
revision 45883)
```



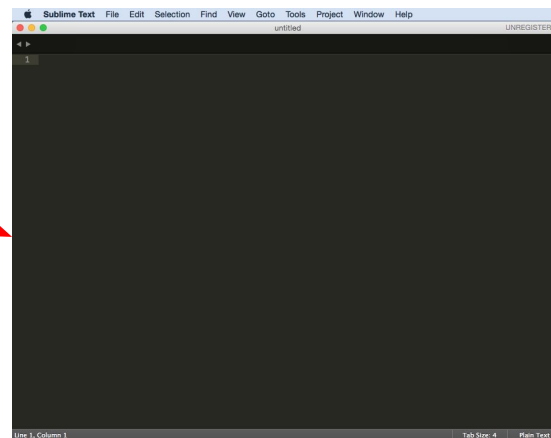
ターミナル

「ruby -v」と入力して、返答が返って来ればインストールされているので、先に進みましょう。

Ruby プログラムを動かしてみよう

ファイル名: myFirstRuby.rb

```
puts "Hello World"
```



```
ruby myFirstRuby.rb  
(Enter)
```



Sinatra で webサーバを立ち上げよう

Sinatra とは

Sinatra とは Ruby で動く
Web アプリケーションフレームワーク

Sinatra とは

Sinatra とは Ruby で動く Web アプリケーションフレームワーク

他の Web アプリケーションフレームワークとして、
Ruby on Rails があげられるが、
Ruby on Rails に比べて簡易で軽量であることがメリット




Sinatra を実行しよう

※ CentOS サーバへのアクセス

`vagrant ssh` (Enter)

※ vagrant フォルダへのアクセス

`cd /vagrant` (Enter)



A terminal window titled "ターミナル - zah - 80x24" showing a successful SSH login. The output is as follows:

```
Last login: Sat Oct 3 14:55:40 on console
[koji@UG-032-KOJIYAMA0KA] ~
%
```

A red arrow points from the `vagrant ssh` command in the text box to the terminal window.

/vagrant フォルダについて



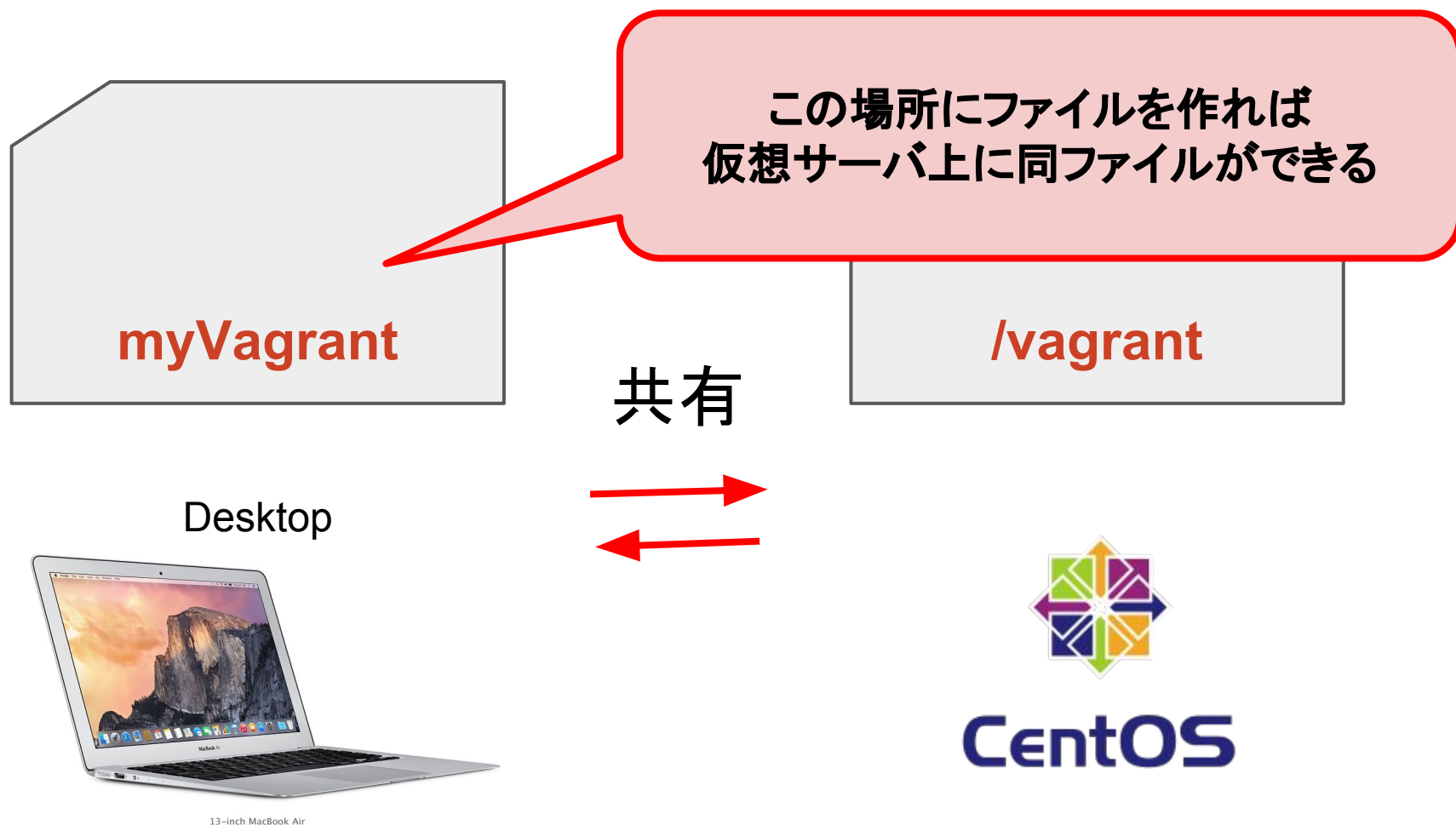
Desktop



13-inch MacBook Air



/vagrant フォルダについて



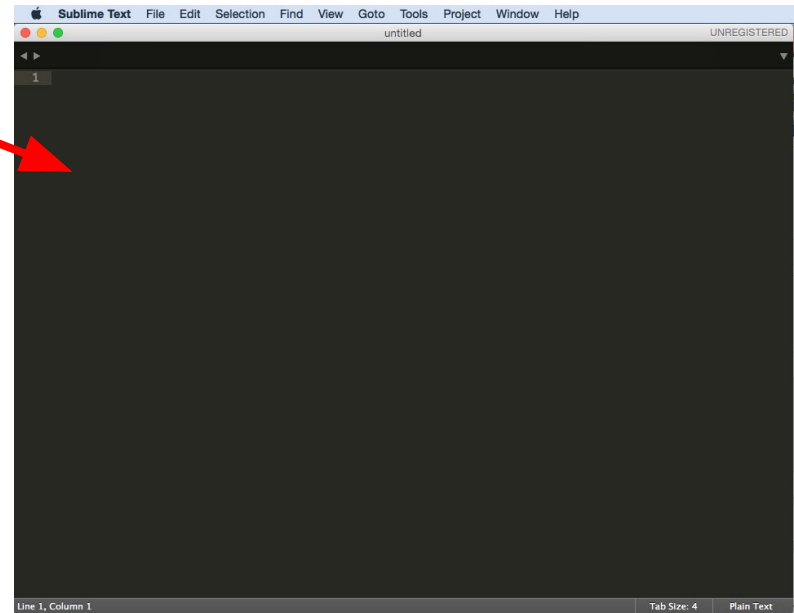
Sinatra 実行ファイルを作成しよう

ファイル名: myapp.
rb

```
require 'rubygems'
require 'sinatra'

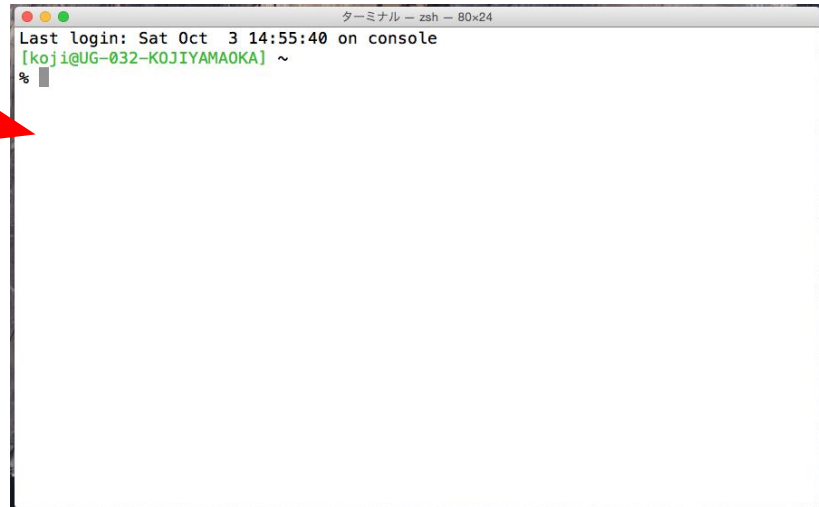
set :environment, :production
set :port, 8080

get '/' do
  'Hello world!'
end
```



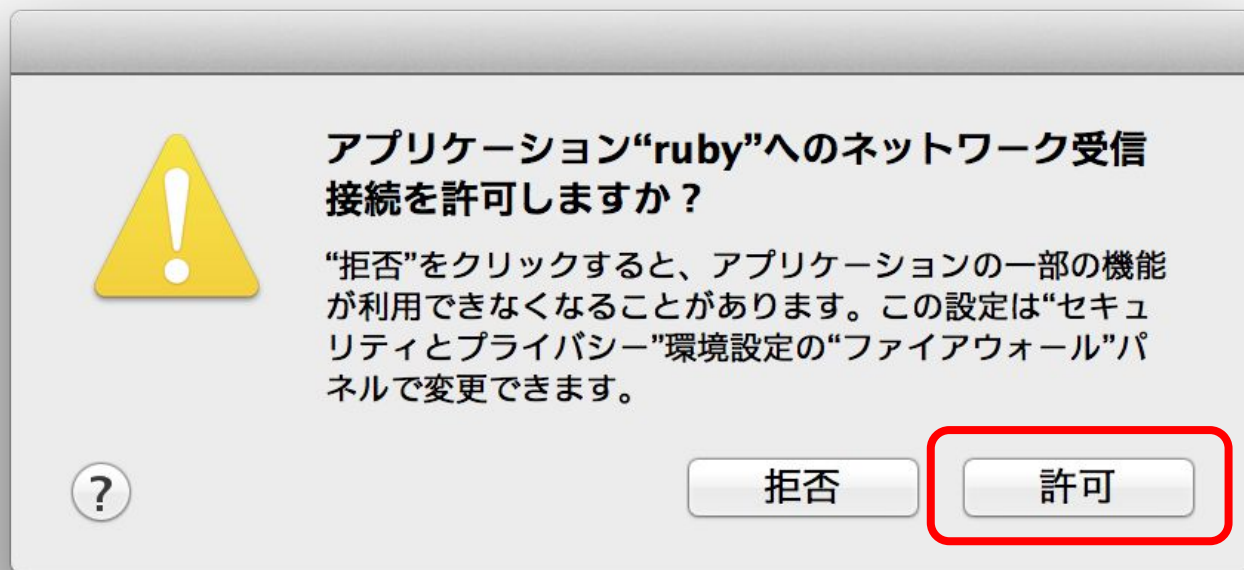
Sinatra を実行しよう

ruby myapp.rb
(Enter)

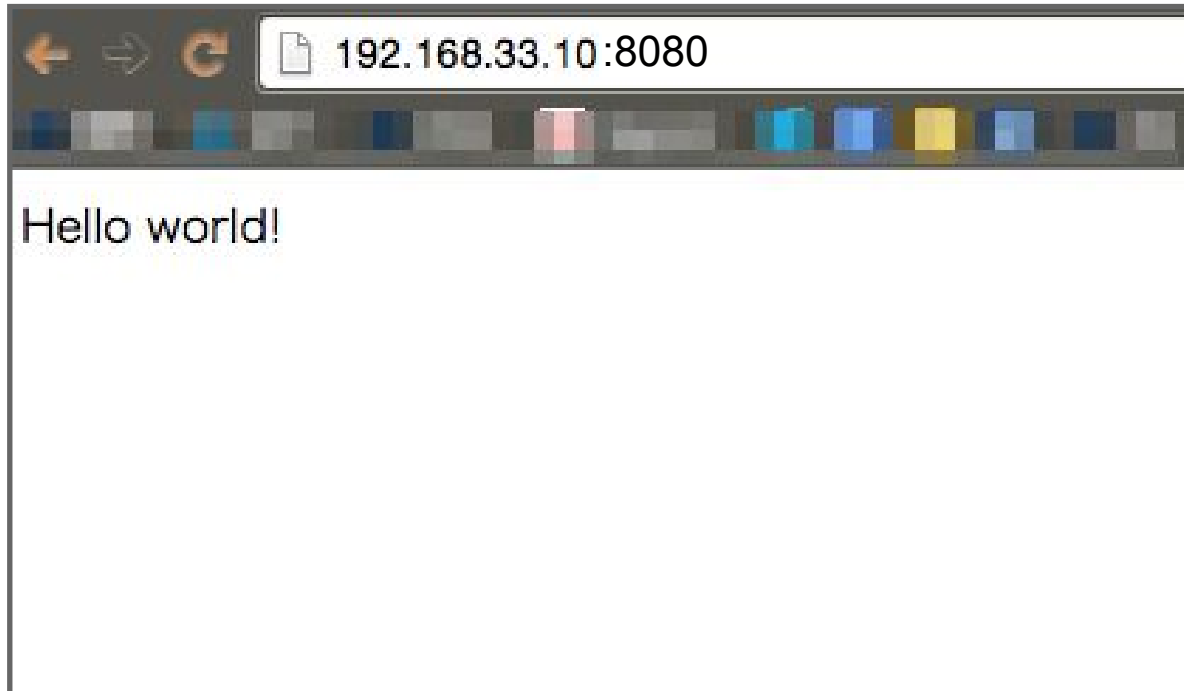


ブラウザから実行

Sinatra ファイルを実行してみよう



Sinatra ファイルを実行してみよう



ブラウザから実行

Sinatra ファイルを実行してみよう

VagrantFile

```
27 # Create a private network, which allows host-only access to the machine
28 # using a specific IP.
29 |config.vm.network "private_network", ip: "192.168.33.10"
30
```



43.p において
VagrantFile で
設定したIPアドレス

Sinatra 実行ファイルを作成しよう

ファイル名: myapp.
rb

```
require 'rubygems'  
require 'sinatra'
```

```
set :environment, :production  
set :port, 8080
```

```
get '/' do  
  'Hello world!'  
end
```

この Ruby プログラムに必要な
モジュール* の読み込み

Sinatra 実行ファイルを作成しよう

ファイル名: myapp.
rb

```
require 'rubygems'  
require 'sinatra'
```

```
set :environment, :production  
set :port, 8080
```

```
get '/' do  
  'Hello world!'  
end
```

**環境を設定
ポート番号* を 8080 に設定**

Sinatra 実行ファイルを作成しよう

ファイル名: myapp.
rb

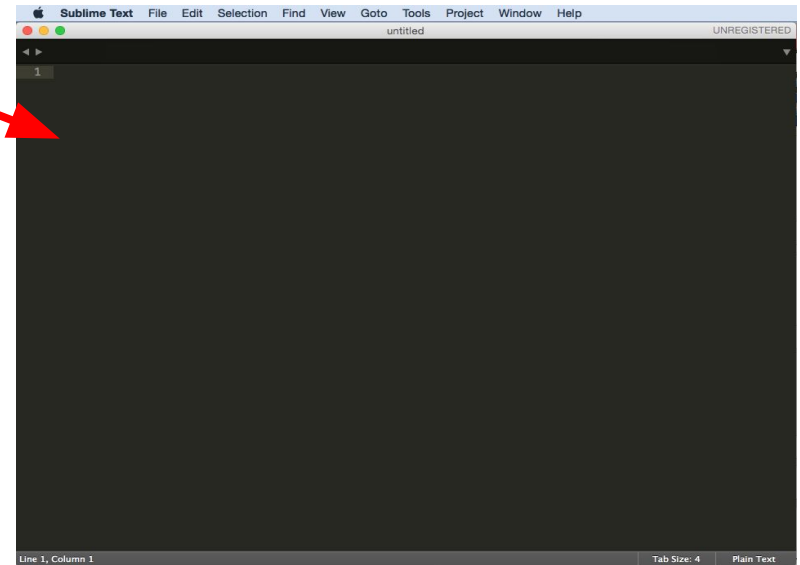
```
require 'rubygems'  
require 'sinatra'  
  
set :environment, :production  
set :port, 8080  
  
get '/' do  
  'Hello world!'  
end
```

**アクセスされたら
Hello world!**

アドレスの入力の一部を返せるようにしよう

ファイル名: myapp.rb

```
...  
get '/' do  
...  
end  
  
get "/hello/:name" do |n|  
  greeting = "#{n}"  
end  
  
__END__  
  
# テンプレート  
@@index  
<html>  
<head><style>body{font-size:64px}</style></head>  
<body>  
  <%= greeting %><br>  
</html></body>
```



Sinatra を実行しよう

ruby myapp.rb
(Enter)



A terminal window titled "ターミナル - zsh - 80x24". It shows the text "Last login: Sat Oct 3 14:55:40 on console" and the prompt "[koji@UG-032-KOJIYAMAOKA] ~" followed by a cursor. A red arrow points from the text "ruby myapp.rb (Enter)" to the terminal window.

ブラウザから

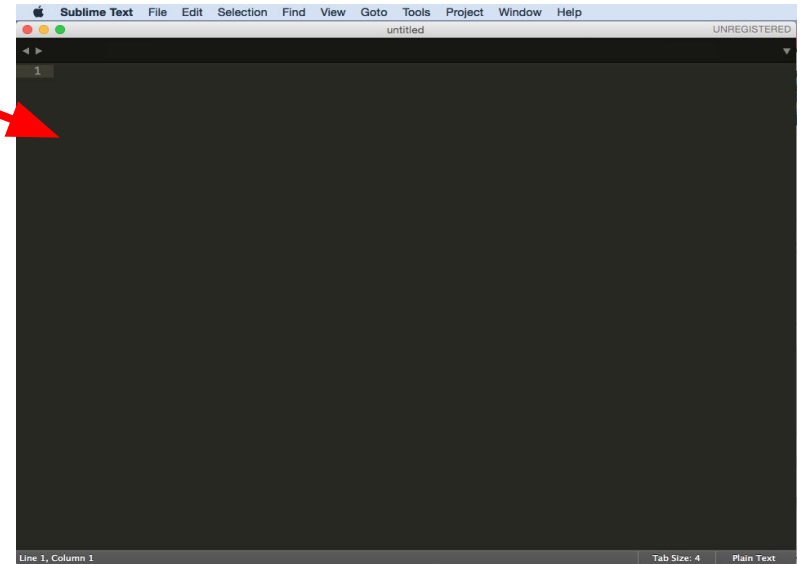
「<http://192.168.33.10:8080/hello/takashi>」にアクセス

できたら、takashi の部分を
自分の名前に変えて試してみましょう！

アドレスの入力に応じて返す内容を変えてみよう

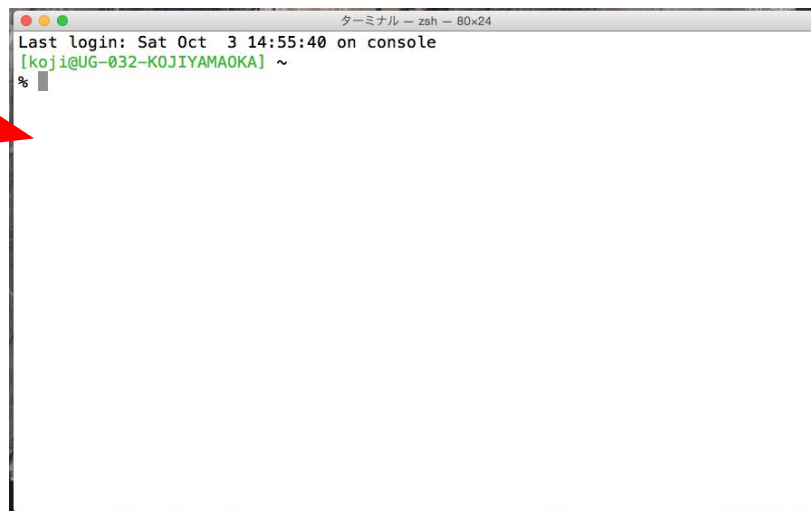
ファイル名: myapp.rb

```
...  
get "/morning" do  
  "good morning"  
end  
  
get "/noon" do  
  "hello"  
end  
  
get "/evening" do  
  "good evening"  
end  
  
__END__  
  
# テンプレート  
@@index  
<html>  
<head><style>body{font-size:64px}</style></head>  
<body><p>  
</p></html></body>
```



Sinatra を実行しよう

ruby myapp.rb
(Enter)



ブラウザから

<http://192.168.33.10:8080/morning>

<http://192.168.33.10:8080/noon>

<http://192.168.33.10:8080/evening>

にアクセス！

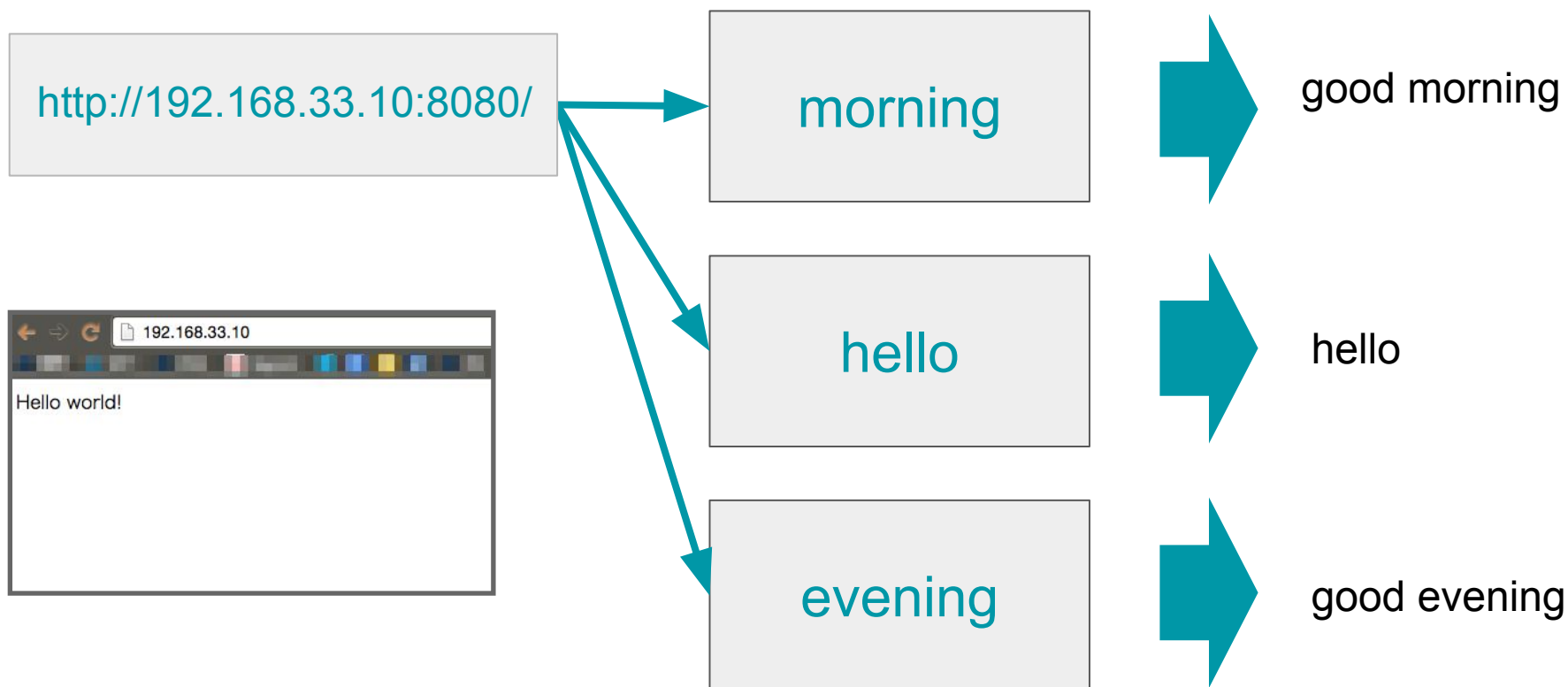
ルーティング

アドレスの入力によって出力の 場合分け を行うこと

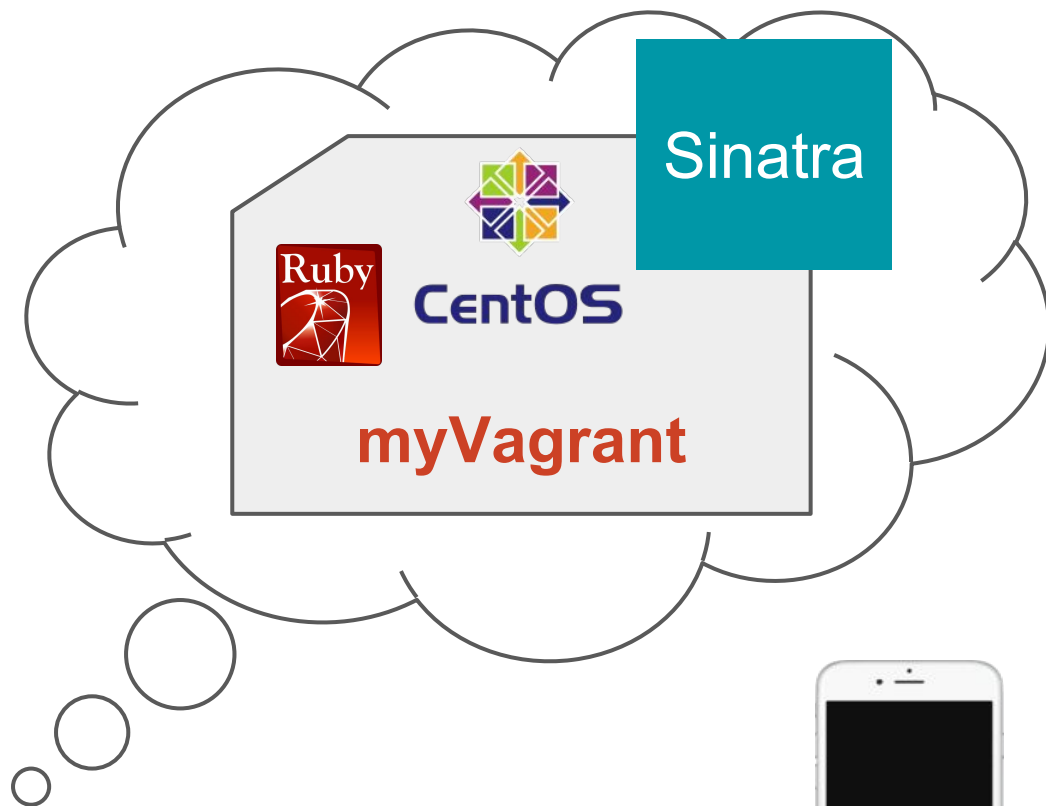
IPアドレス

パス

出力



データ連携のイメージ(1)



13-inch MacBook Air



データ連携のイメージ(2)



13-inch MacBook Air

データ連携のイメージ(3)

リクエスト



Sinatra



CentOS

myVagrant



13-inch MacBook Air



データ連携のイメージ(4)



目次



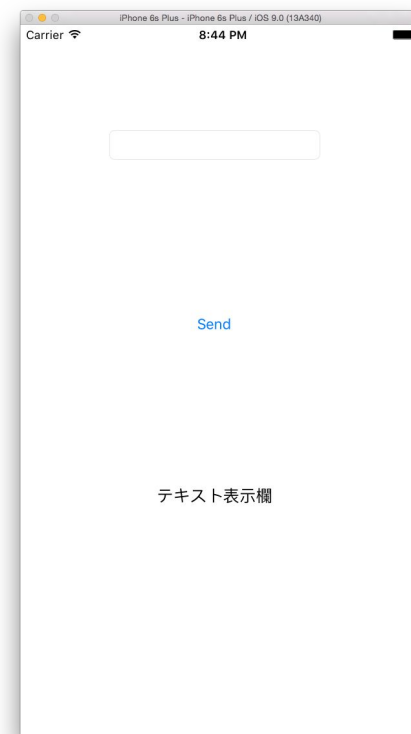
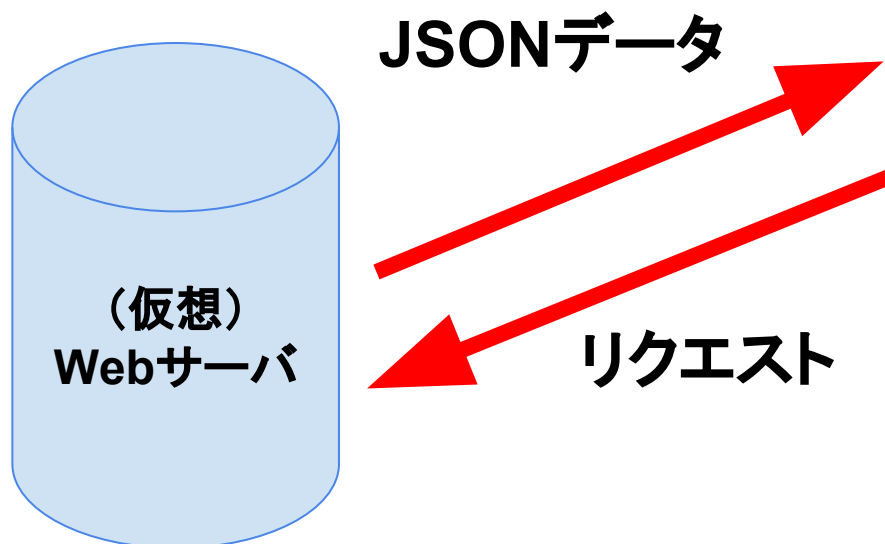
仮想サーバを構築しよう

Sinatra でサーバサイドプログラミング

iPhone アプリと連携しよう

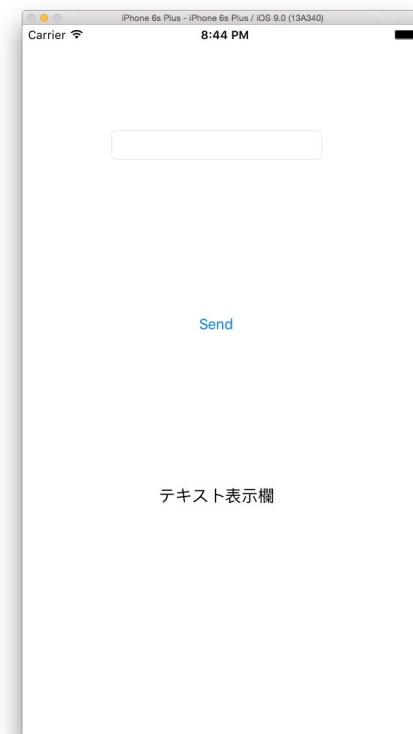
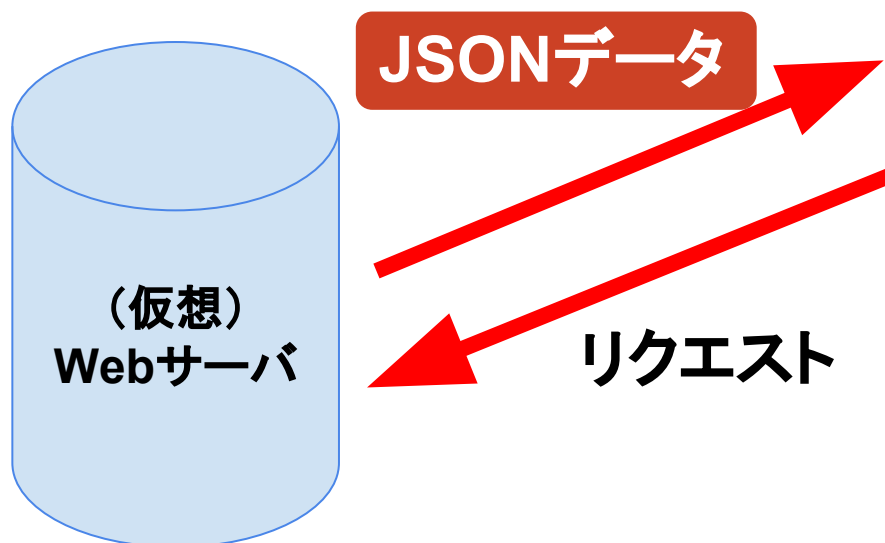
iPhone アプリと連携しよう！

次に、今回作成したこの
仮想 Web サーバからデータを取得して、
iPhone 上に表示させるという処理を
実装させてみます！



iPhone アプリと連携しよう！

次に、今回作成したこの
仮想 Web サーバからデータを取得して、
iPhone 上に表示させるという処理を
実装させてみます！



JSON とは何だ？

JSON とは JavaScript Object Notation の略で、XML などと同様のテキストベース、key-value 型のデータフォーマットのことです

右図のような形で記載されています。

```
{
  data: {
    id: 1,
    title: "myTitle",
    contents: "todayContents"
  }
}
```

JSON = key-value 型のデータ

key ... データを紐づける鍵の名前

```
{  
  data: {  
    id: 1,  
    title: "myTitle",  
    contents: "todayContents"  
  }  
}
```

value ... 鍵によって紐付けられたデータの中身

データ連携のイメージ(5)



データ連携のイメージ(6)



データ連携のイメージ(7)



データ連携のイメージ(8)



API とは

API とは ざっくりと言ってしまうと
ネットワークを超えた関数のようなものである

API とは

API とは ざっくりと言ってしまうと
ネットワークを超えた関数のようなものである

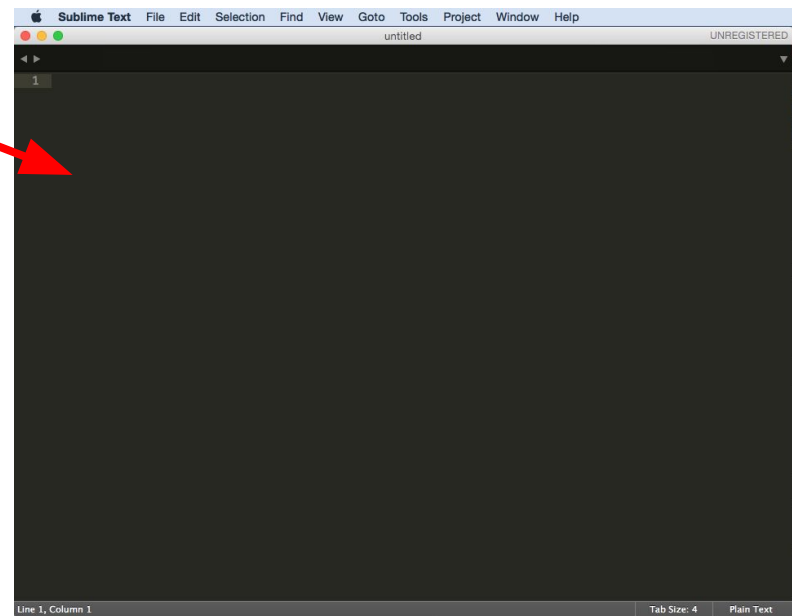
与えられたパラメータを基にそのパラメータに則した
値やオブジェクトをサーバが返答し、
クライアント(アプリ)側で扱うことができる

Sinatra から JSON を返せるようにしよう

ファイル名: show.rb

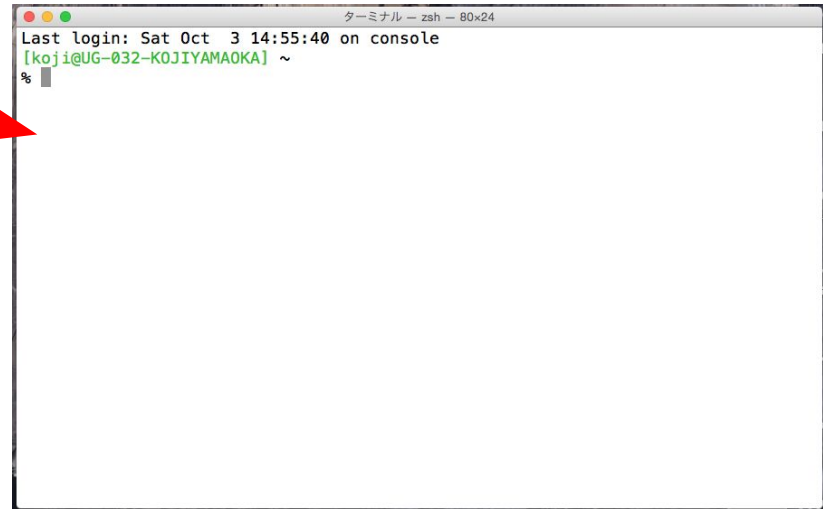
require 'json'

```
...  
get '/show' do  
  article = {  
    id: 1,  
    title: "today's dialy",  
    content: "It's a sunny day."  
  }  
  
  article.to_json  
end  
...
```



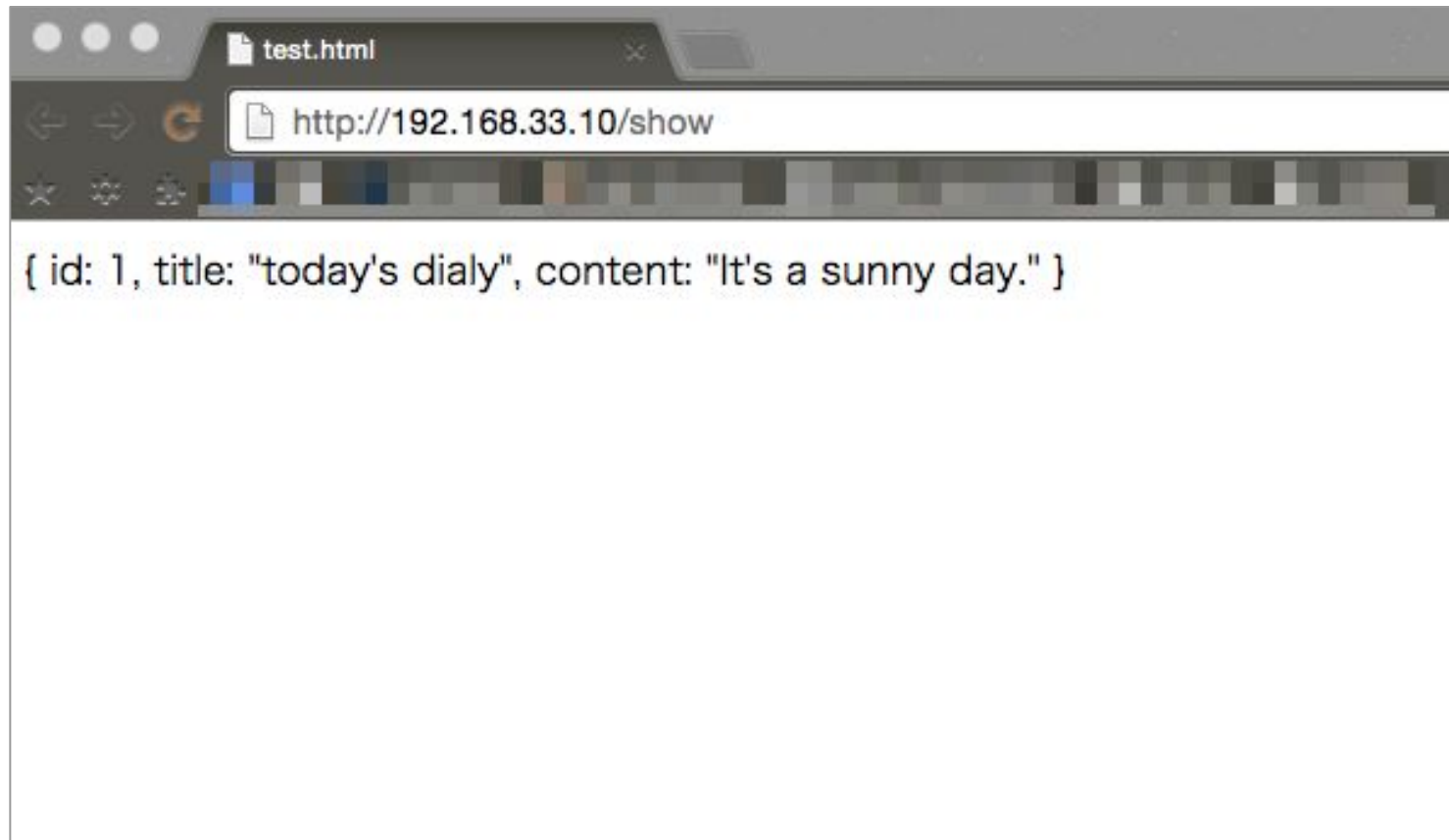
Sinatra を実行しよう

ruby show.rb (Enter)



ブラウザ(<http://192.168.33.10/show>)から実行

JSON データが返ってきた事を確認しよう



JSON = key-value 型のデータ

key ... データを紐づける鍵の名前

```
{ id: 1, title: 'today's dialy', content: 'It's a sunny day.' }
```

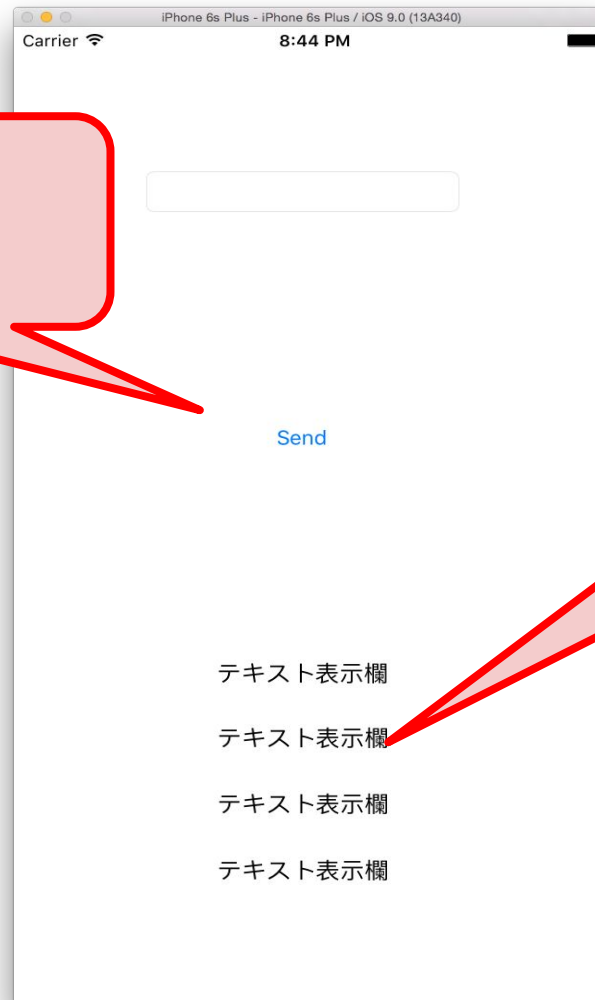
value ... 鍵によって紐付けられたデータの中身

xcode で配布ファイルを開こう！



iPhone 側の

Button



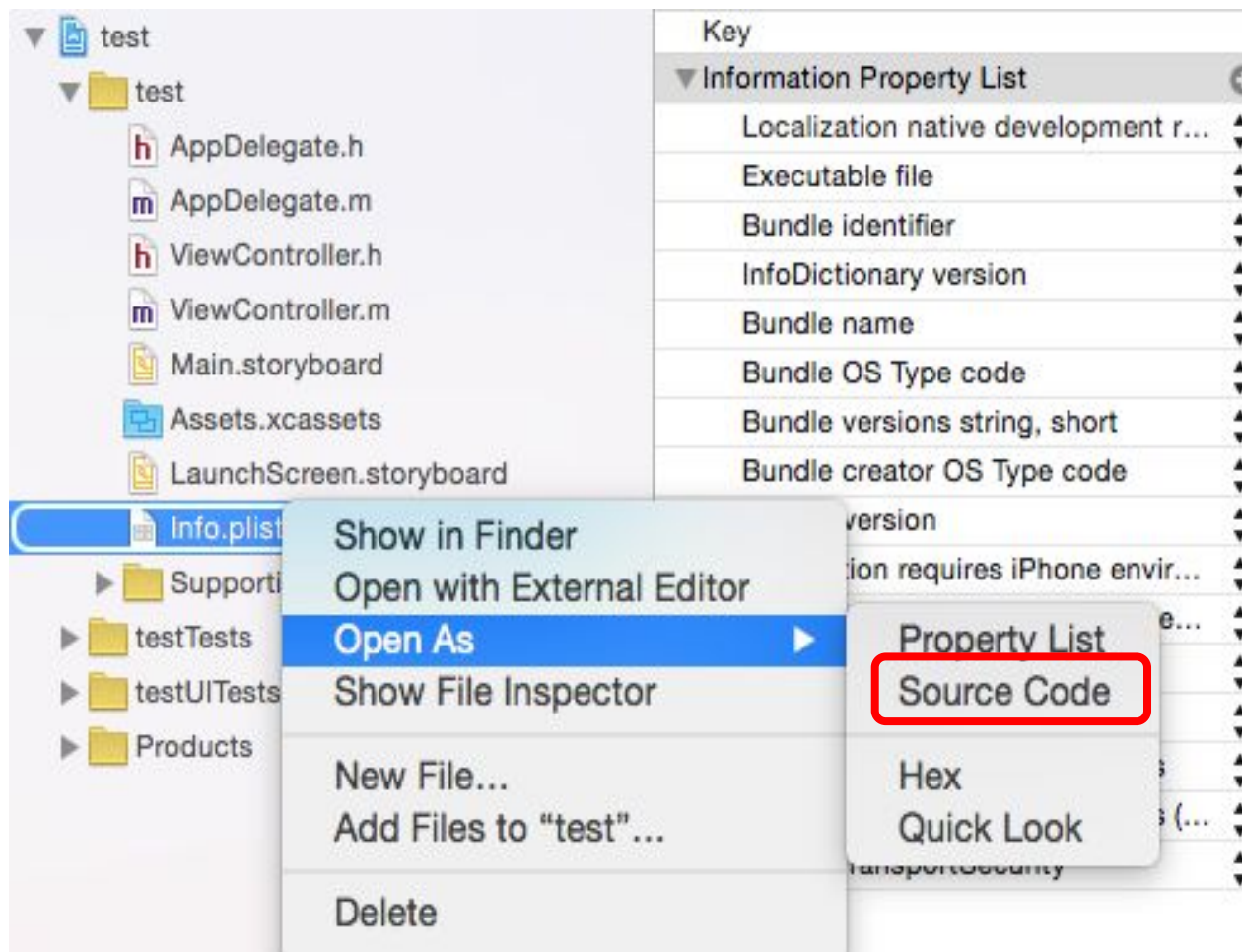
TextArea

ヘッダファイルに StoryBoard と紐づける設定をしよう

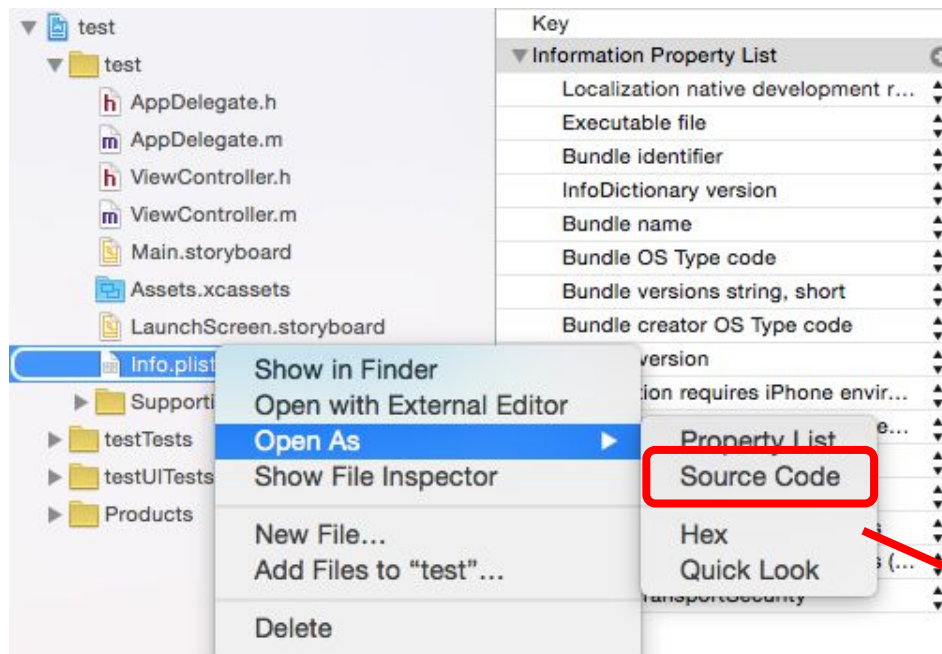
クライアント (iphone) 側 : ViewController.h

```
//  
// ViewController.m  
// test  
//  
// Created by Koji on H27/09/22.  
// Copyright © 平成27年 Koji. All rights reserved.  
//  
#import <UIKit/UIKit.h>  
// StoryBoard に紐づける  
@interface ViewController : UIViewController{  
  
    __weak IBOutlet UILabel *Label1;  
    __weak IBOutlet UILabel *Label2;  
    __weak IBOutlet UILabel *Label3;  
    __weak IBOutlet UILabel *Label4;  
  
}  
  
- (IBAction)tap:(id)sender;  
  
@end
```

HTTP 通信を許可しよう



HTTP 通信を許可しよう



最後の行に追加

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSAllowsArbitraryLoads</key>
  <true/>
</dict>
```

こんな感じに記載すれば OK

info.plist

```
        <string>UIInterfaceOrientationLandscapeLeft</string>
        <string>UIInterfaceOrientationLandscapeRight</string>
    </array>
    <key>NSAppTransportSecurity</key>
    <dict>
        <key>NSAllowsArbitraryLoads</key>
        <true/>
    </dict>
</dict>
</plist>
```

JSON データを iPhone 上に表示させよう！

ViewController.m に次のコードを加えよう ！！

@implementation ViewController

....

```
- (IBAction)tap:(id)sender{
    NSString *url = [NSString stringWithFormat:@"http://192.168.33.10:8080/show"];
    NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URLWithString:url]];

    NSData *json_data = [NSURLConnection sendSynchronousRequest:request returningResponse:nil error:nil];

    NSArray *array = [NSJSONSerialization JSONObjectWithData:json_data options:
    NSJSONReadingAllowFragments error:nil];

    NSLog(@"%@ %@ %@", [array valueForKeyPath:@"id"], [array valueForKeyPath:@"title"], [array
    valueForKeyPath:@"content"]);

    Label2.text = [NSString stringWithFormat:@"%@",[array valueForKeyPath:@"id"]];
    Label3.text = [NSString stringWithFormat:@"%@",[array valueForKeyPath:@"title"]];
    Label4.text = [NSString stringWithFormat:@"%@",[array valueForKeyPath:@"content"]];
}
```

....

@end

JSON データを iPhone 上に表示させよう！

ViewController.m

@implementation ViewController

....

- (IBAction)tap:(id)sender{

```
NSString *url = [NSString stringWithFormat:@"http://192.168.33.10:8080/show"];  
NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URLWithString:url]];
```

```
NSData *json_data = [NSURLConnection sendSynchronousRequest:request timeoutInterval:10];
```

```
NSArray *array = [NSJSONSerialization JSONObjectWithData:json_data options:0 error:nil];  
NSJSONReadingAllowFragments error:nil];
```

```
NSLog(@"%@ %@ %@", [array valueForKeyPath:@"id"],  
valueForKeyPath:@"content"]);
```

```
Label2.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"id"]];  
Label3.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"title"]];  
Label4.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"content"]];
```

```
}
```

....

@end

**API の リクエストURL
を設定**

JSON データを iPhone 上に表示させよう！

ViewController.m

@implementation ViewController

....

```
- (IBAction)tap:(id)sender{
    NSString *url = [NSString stringWithFormat:@"http://192.168.33.10:8000/...", [sender text]];
    NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URLWithString:url]];
    ...
}
```

***API レスポンス
を受け取りJSONデータへ**

```
NSData *json_data = [NSURLConnection sendSynchronousRequest:request returningResponse:nil error:nil];
```

```
NSArray *array = [NSJSONSerialization JSONObjectWithData:json_data options:
    NSJSONReadingAllowFragments error:nil];
```

```
NSLog(@"%@ %@ %@", [array valueForKeyPath:@"id"], [array valueForKeyPath:@"title"], [array
    valueForKeyPath:@"content"]);
```

```
Label2.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"id"]];
Label3.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"title"]];
Label4.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"content"]];
```

```
}
```

....

@end

JSON データを iPhone 上に表示させよう！

ViewController.m

@implementation ViewController

....

```
- (IBAction)tap:(id)sender{
    NSString *url = [NSString stringWithFormat:@"http://192.168.33.10:8080/show"];
    NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URLWithString:url]];
```

```
    NSData *json_data = [NSURLConnection sendSynchronousRequest:request fromCache:NO];
```

```
    NSArray *array = [NSJSONSerialization JSONObjectWithData:json_data options:
    NSJSONReadingAllowFragments error:nil];
```

```
    NSLog(@"%@ %@ %@", [array valueForKeyPath:@"id"], [array valueForKeyPath:@"title"], [array
    valueForKeyPath:@"content"]);
```

```
    Label2.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"id"]];
    Label3.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"title"]];
    Label4.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"content"]];
```

```
}
```

....

@end

**JSON data を
受け取り配列へ**

JSON データを iPhone 上に表示させよう！

ViewController.m

@implementation ViewController

....

```
- (IBAction)tap:(id)sender{
    NSString *url = [NSString stringWithFormat:@"http://192.168.33.10:8080/sho
    NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URL

    NSData *json_data = [NSURLConnection sendSynchronousRequest:re

    NSArray *array = [NSJSONSerialization JSONObjectWithData:json_data options:
    NSJSONReadingAllowFragments error:nil];
```

```
    NSLog(@"%@ %@ %@", [array valueForKeyPath:@"id"], [array valueForKeyPath:@"title"], [array
    valueForKeyPath:@"content"]);
```

```
    Label2.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"id"]];
    Label3.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"title"]];
    Label4.text = [NSString stringWithFormat:@"%@", [array valueForKeyPath:@"content"]];
```

```
}
```

....

@end

**配列の各要素を
コンソールへ表示**

JSON データを iPhone 上に表示させよう！

ViewController.m

@implementation ViewController

....

```
- (IBAction)tap:(id)sender{
    NSString *url = [NSString stringWithFormat:@"http://192.168.33.10:8080/"];
    NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URLWithString:url]];

    NSData *json_data = [NSURLConnection sendSynchronousRequest:request
                                     timeoutInterval:10.0];

    NSArray *array = [NSJSONSerialization JSONObjectWithData:json_data
                                                         options:0
                                                         error:nil];

    NSLog(@"%@ %@ %@", [array valueForKeyPath:@"id"], [array valueForKeyPath:@"title"], [array valueForKeyPath:@"content"]);
}
```

**JSON データの
各 key に対する value を
文字列型に変換して表示**

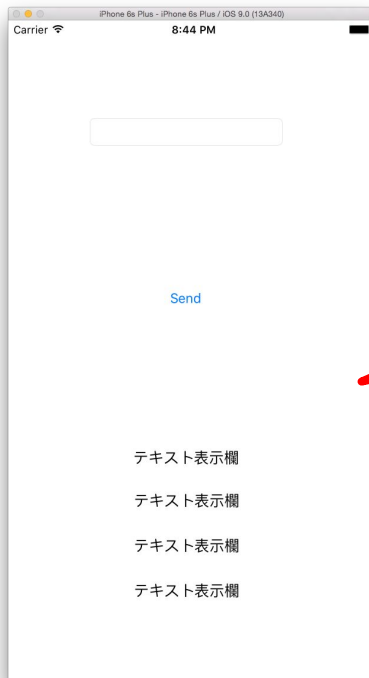
```
Label2.text = [NSString stringWithFormat:@"%@",[array valueForKeyPath:@"id"]];
Label3.text = [NSString stringWithFormat:@"%@",[array valueForKeyPath:@"title"]];
Label4.text = [NSString stringWithFormat:@"%@",[array valueForKeyPath:@"content"]];
}
```

....

@end

iPhone 側から実行してみよう！

シュミレータから実行！



ボタンを押して各key に対する
value が表示されるようになれば
ゴール！！

今回はここまで！

今回やったことの復習(概要)



今回やったことの復習(概念1)

仮想化 is ...

今回やったことの復習(概念1)

仮想化 is ≡二PC の作成

今回やったことの復習(概念2)

サーバ is ...

今回やったことの復習(概念2)

サーバ is

要求に応えるコンピュータ

今回やったことの復習(概念2)

サーバ is

データの Serve(提供) 専用のコンピュータ

おまけ

インターネット上に公開するには(1)

インターネット上に公開するには大まかに3ステップの処理が必要です

1. Web サーバ(CentOS等)をレンタル
2. 遠隔操作([SSH接続](#))する
3. サーバの設定する

※ 月数百円程度で利用することができます。

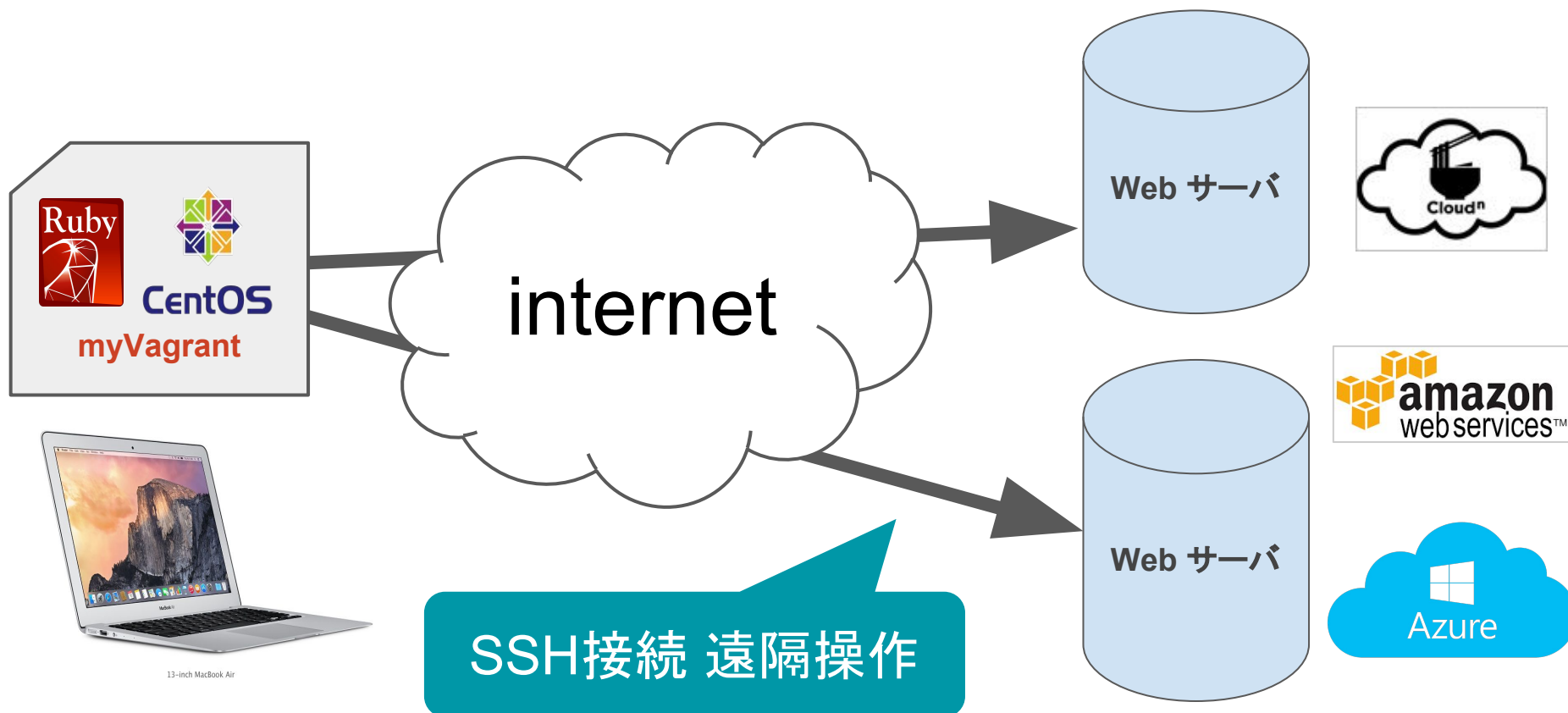
静的なWEBページはファイル転送ソフトを用いれば直ぐに配置・閲覧できますが、サーバサイドの処理を行うにはサーバを自身でカスタマイズする必要があります



<https://aws.amazon.com/jp/>

インターネット上に公開するには(2)

安全性の高い通信方法である **SSH** で Webサーバに遠隔で環境の設定、及び、自分のPC内で作成したファイルを転送することが可能となります。



参考資料(1)

Sinatra + ActiveRecord + SQLite3で、軽量なWeb-DB連携例

- <https://tamosblog.wordpress.com/2012/10/26/sinatra/>

Sinatra + ActiveRecord + MySQLで、簡単APIサーバ構築

- http://qiita.com/u1_fukui/items/88c10d4d530ec6fbaaa1

Sinatraで簡易APIサーバを作ってみた

- http://dev.classmethod.jp/server-side/ruby-on-rails/sinatra__make_api_server/

【初心者向け】RubyとSinatra、アンテナサイトの作り方

- <http://shgam.hatenadiary.jp/entry/2013/08/25/160937>

参考資料(2)

Sinatra README

- <http://www.sinatrarb.com/intro-ja.html>

Sinatraの使い方あれこれ

- <http://qiita.com/weed/items/7c83a795e37bdf52cfef>

仮想環境構築ツール「Vagrant」で開発環境を仮想マシン上に自動作成する

- <https://osdn.jp/magazine/15/02/13/200000>

VagrantでWEBアプリ実行環境構築

- <http://qiita.com/dahugani/items/a5eca25e6173331951d7>

参考資料(3)

Vagrant+VirtualBox(CentOS)でRuby開発環境を作った時のメモ

- <http://blogssom.net/post/12>

Vagrant で自分の PC に「作って、壊して、元に戻せる」サーバを作る

- <http://www.1x1.jp/blog/2013/03/vagrant.html>

Objective-CでJSON取得して表示する方法

- <http://qiita.com/draapon/items/859473840211f0ac1552>

iOSでWebAPIからJSONを取得

- <http://inter-arteq.com/ios%E3%81%A7webapi%E3%81%8B%E3%82%89json%E3%82%92%E5%8F%96%E5%BE%97/>