Introduction to Ruby on Rails

Software Engineering II
WS 2019/20

Enterprise Platform and Integration Concepts

Rails by *Erik* (CC BY-NC 2.0) https://www.flickr.com/photos/calaveth/3904320343

Introduction to Ruby on Rails

1. **Ruby & Ruby on Rails**
   - What is Ruby on Rails?
   - A few words about Ruby
   - Rails' core components
   - RESTful architecture
2. Your first Rails application
3. Your introductory Rails exercise

Ruby

Following based on https://speakerdeck.com/y310/ruby-on-rails-introduction

Ruby

Some basic facts

http://www.ruby-lang.org/

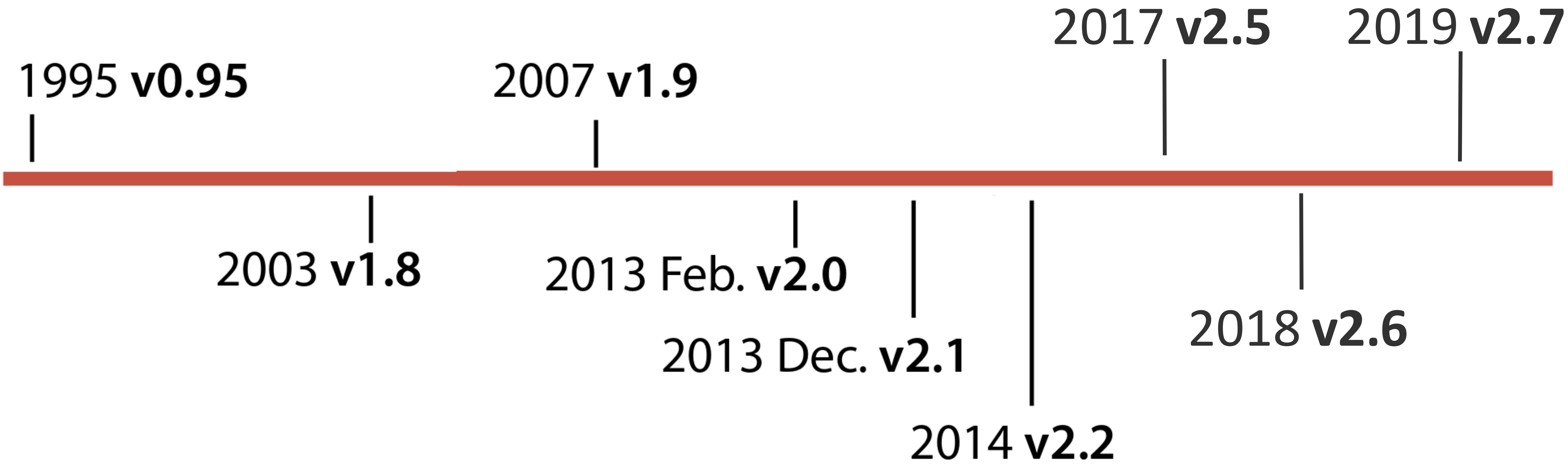
- Interpreted, high-level, general-purpose language
- Supports multiple programming paradigms, incl.
  - □ procedural
  - □ object-oriented
  - □ functional
- Influenced by Perl, Smalltalk, Eiffel, and Lisp
- Open-source, mature software
- Additionally different VMs available, e.g. JRuby

Yukihiro "Matz" Matsumoto with R. Stallman

Image by Rubén Rodríguez (CC BY 3.0) - http://commons.wikimedia.org/wiki/File:Matz.jpg

History of Ruby

More than 20 years of active development



1995 **v0.95**

2003 **v1.8**

2007 **v1.9**

2013 Feb. **v2.0**

2013 Dec. **v2.1**

2014 **v2.2**

2017 **v2.5**

2018 **v2.6**

2019 **v2.7**

Basic Ruby Syntax

Object-orientation and procedural programming

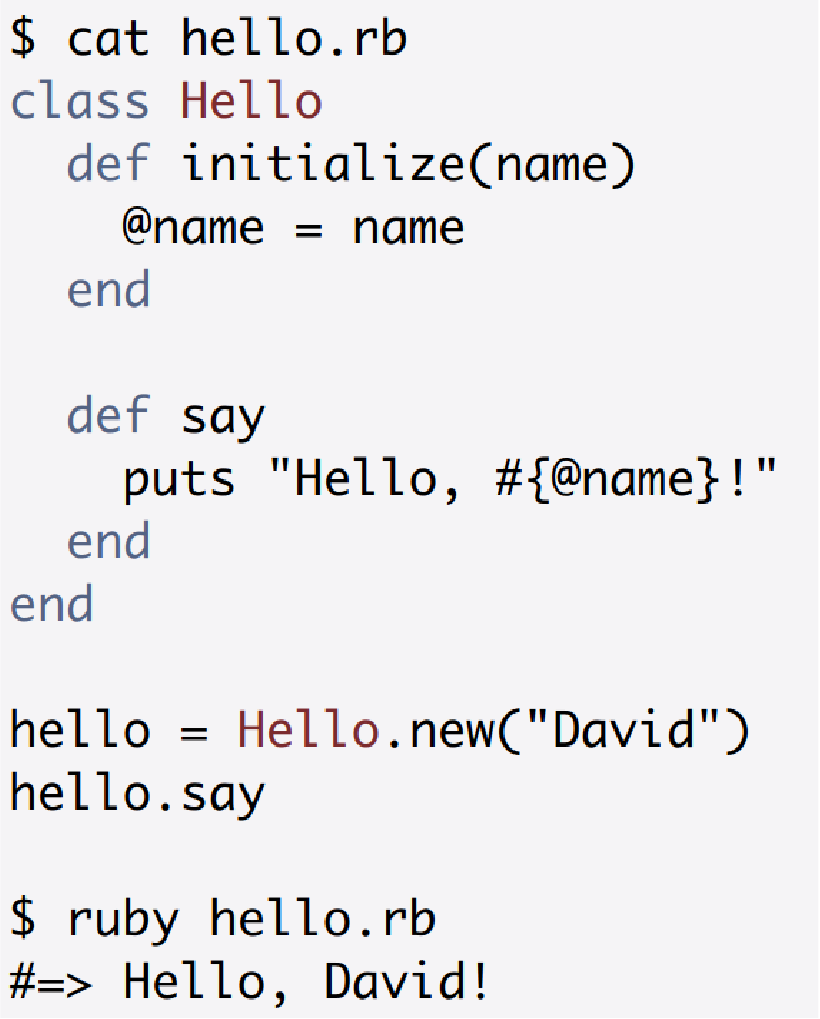
class declaration →

instance variable →

method declaration →

Instantiation →

Method call →

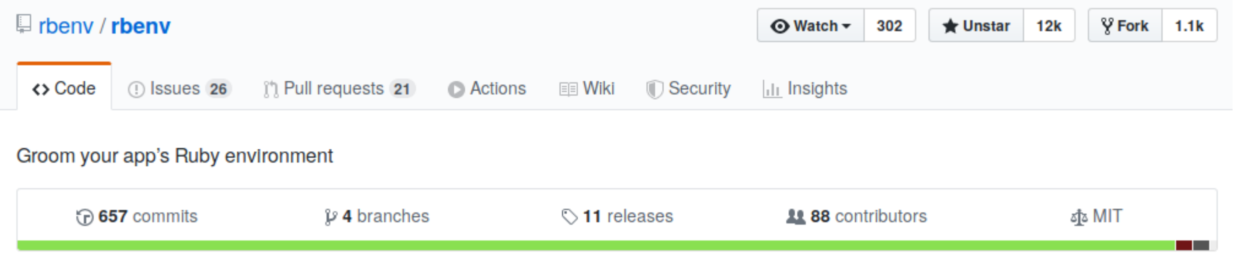


```
$ cat hello.rb
class Hello
  def initialize(name)
    @name = name
  end

  def say
    puts "Hello, #{@name}!"
  end
end

hello = Hello.new("David")
hello.say

$ ruby hello.rb
#=> Hello, David!
```

Important Ruby Tools

Tools needed for every day development

- A Ruby version manager - e.g. **rbenv**
- A package manager - **RubyGems**
- A dependency management and provisioning tool - **bundler**



rbenv / **rbenv**

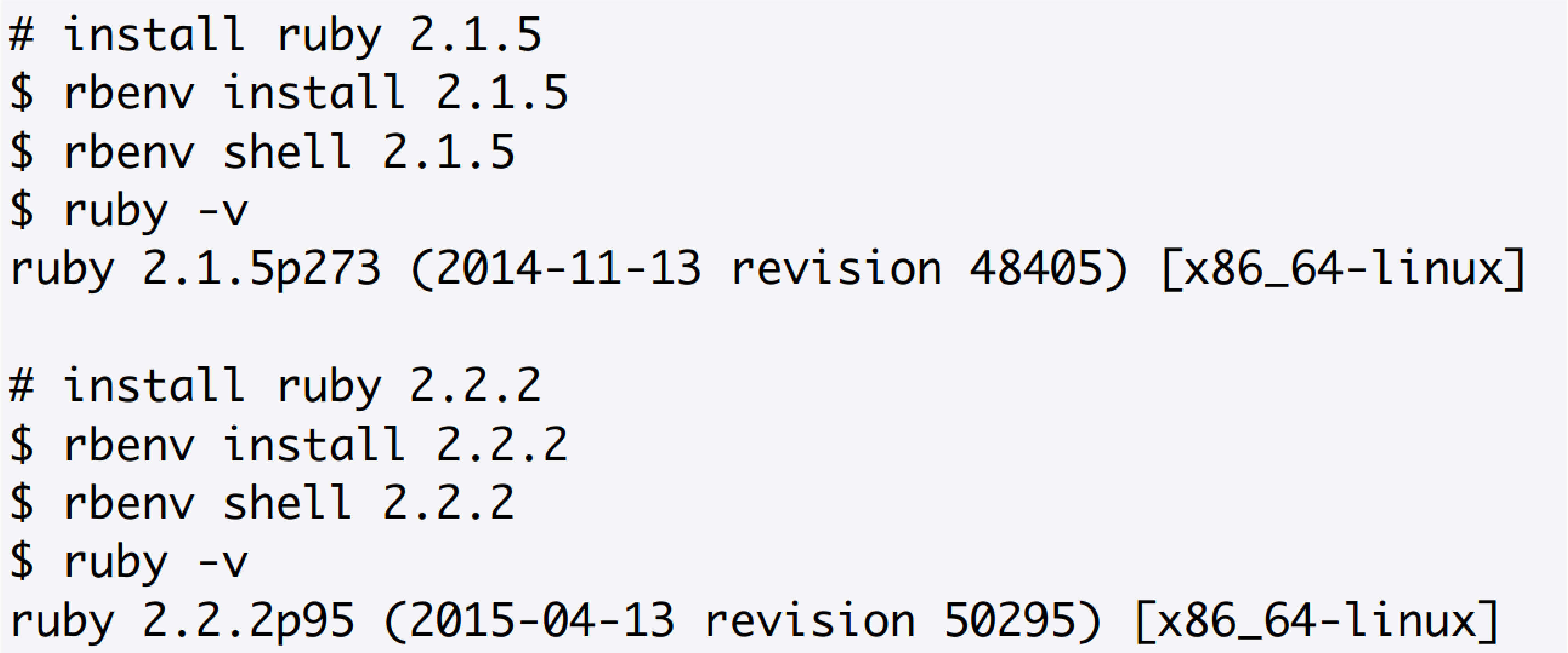
Groom your app's Ruby environment

657 commits · **4** branches · **11** releases · **88** contributors · MIT

Find, install, and publish RubyGems.

Version Management

You want to run a specific version of Ruby, here using **rbenv**, alternative **RVM**

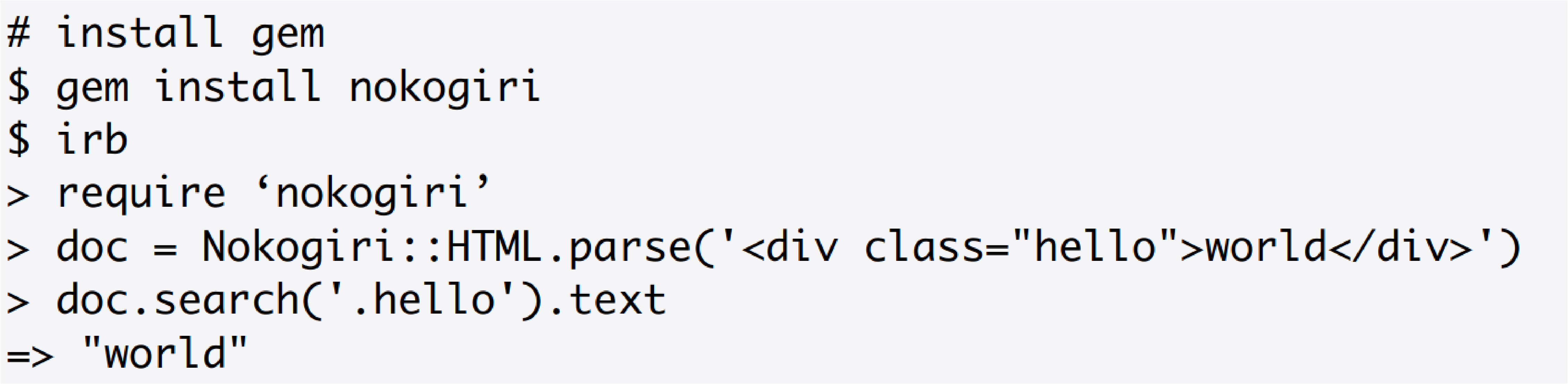


```
# install ruby 2.1.5
$ rbenv install 2.1.5
$ rbenv shell 2.1.5
$ ruby -v
ruby 2.1.5p273 (2014-11-13 revision 48405) [x86_64-linux]

# install ruby 2.2.2
$ rbenv install 2.2.2
$ rbenv shell 2.2.2
$ ruby -v
ruby 2.2.2p95 (2015-04-13 revision 50295) [x86_64-linux]
```

Package Management

RubyGems: standard package manager for Ruby

- Anyone can make a "gem" and publish to https://rubygems.org
- pip for Python, npm for Node.js, etc.
- *irb* = interactive Ruby, Ruby's REPL (read-eval-print-loop)



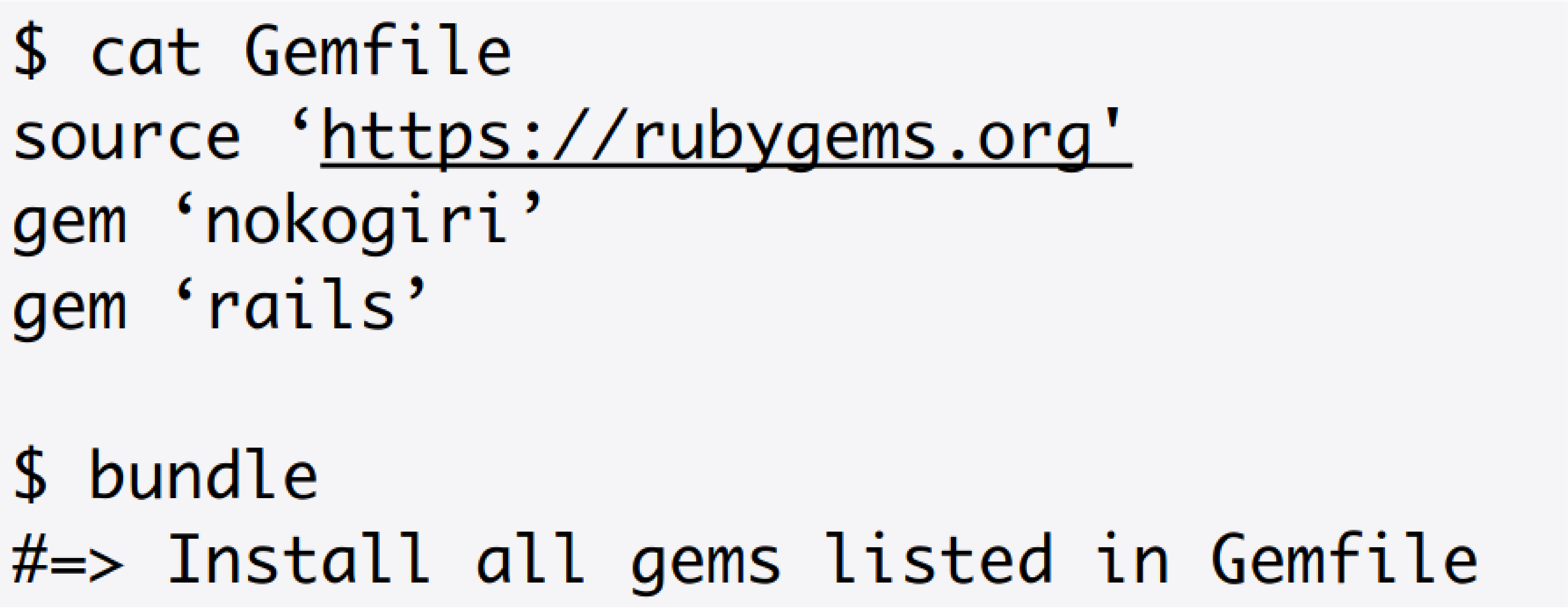
```
# install gem
$ gem install nokogiri
$ irb
> require ‘nokogiri’
> doc = Nokogiri::HTML.parse('<div class="hello">world</div>')
> doc.search('.hello').text
=> "world"
```

Image by Rubén Rodríguez (CC BY 3.0) - http://commons.wikimedia.org/wiki/File:Matz.jpg

Dependency Management

An application depends on many gems. Install them all with **bundler**

- Persist list of gems that the application depends on
- composer for PHP, CocoaPods for iOS, etc. *(what is the Python equivalent?)*
- Provision the same (version of) dependencies on every machine, using **Gemfile**



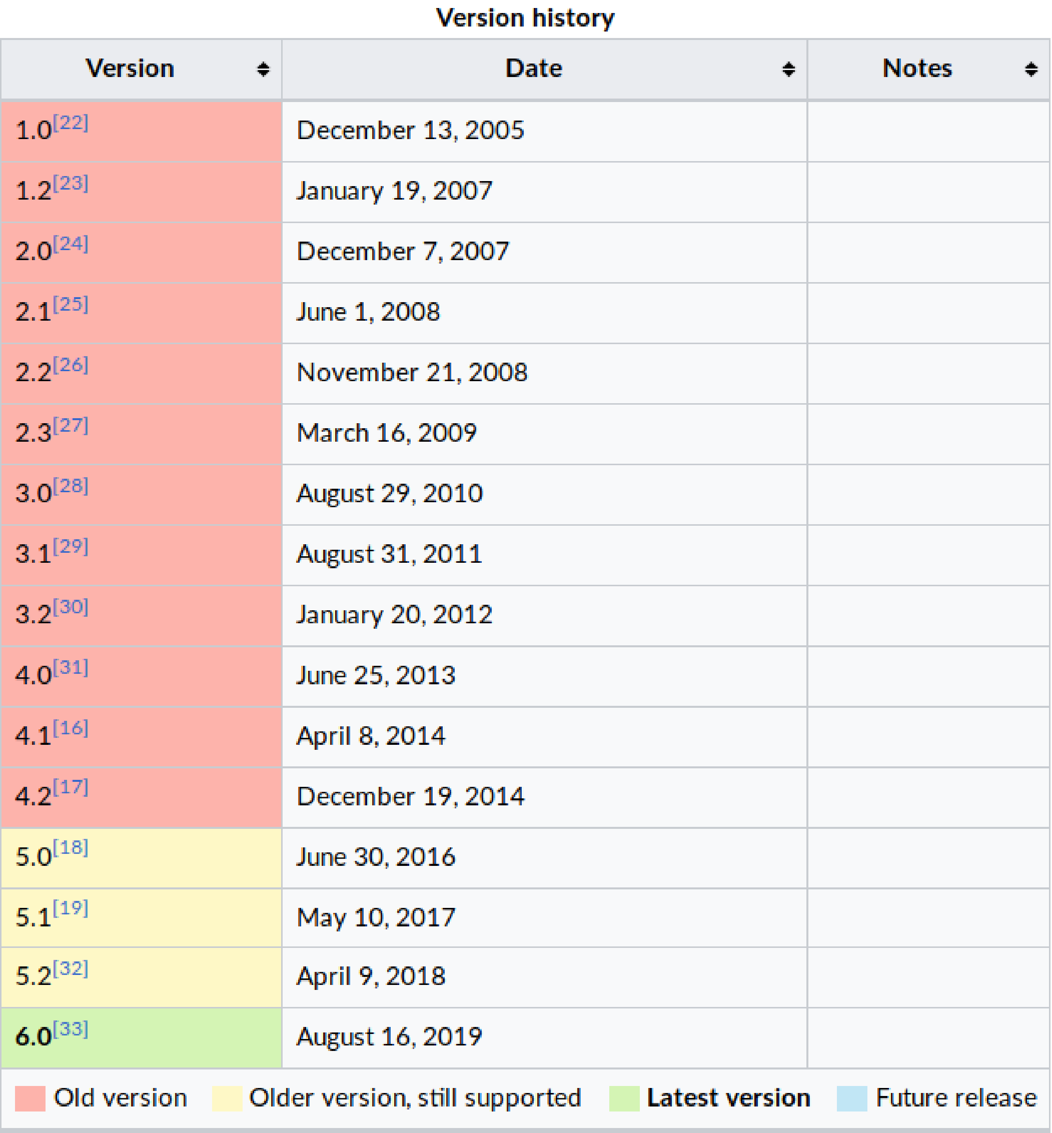
```
$ cat Gemfile
source ‘https://rubygems.org'
gem ‘nokogiri’
gem ‘rails’

$ bundle
#=> Install all gems listed in Gemfile
```

RAILS

History of Ruby on Rails

Web application development framework written in Ruby

- http://rubyonrails.org/



Version history

Version	Date	Notes
1.0[22]	December 13, 2005	
1.2[23]	January 19, 2007	
2.0[24]	December 7, 2007	
2.1[25]	June 1, 2008	
2.2[26]	November 21, 2008	
2.3[27]	March 16, 2009	
3.0[28]	August 29, 2010	
3.1[29]	August 31, 2011	
3.2[30]	January 20, 2012	
4.0[31]	June 25, 2013	
4.1[16]	April 8, 2014	
4.2[17]	December 19, 2014	
5.0[18]	June 30, 2016	
5.1[19]	May 10, 2017	
5.2[32]	April 9, 2018	
6.0[33]	August 16, 2019	

🟥 Old version 🟨 Older version, still supported 🟩 **Latest version** 🟦 Future release

https://en.wikipedia.org/wiki/Ruby_on_Rails#History

Ruby on Rails

Philosophy

- "Don't repeat yourself" – DRY
- Convention over Configuration
  - **There is "The Rails Way"**
- RESTful architecture
- Everything in its place
- No prebuilt backend, generators for creating needed code

Users

- Twitter (early days), Github, SlideShare, Shopify, Groupon, AirBnB, **openHPI**

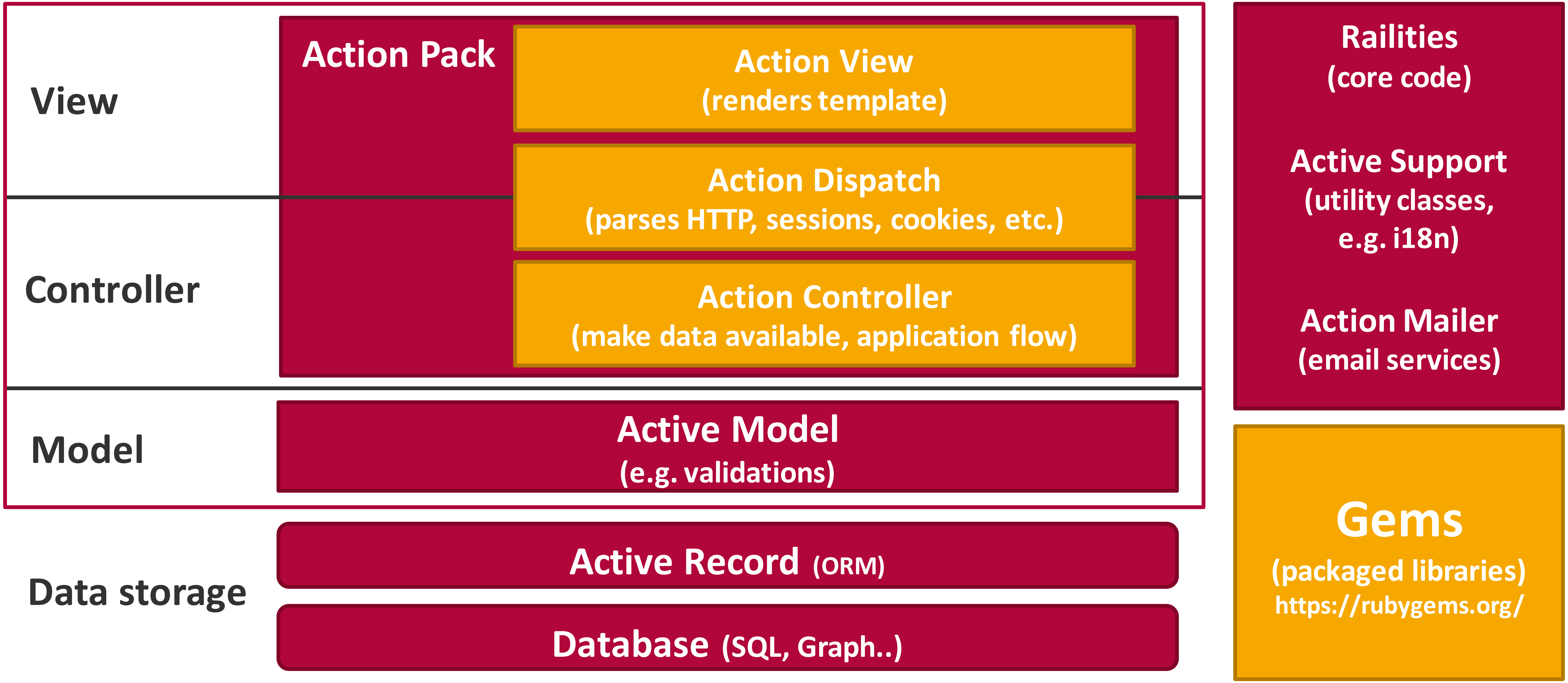
Convention Over Configuration

- The basics of a web app are always similar: there are **conventions**
  - □ Database layout, e.g. tables for model instances
  - □ A controller that manages model instances
  - □ Views that provide create, read, update delete (CRUD) functionality
  - □ Setting these up manually is time consuming and adds little value

- Ruby on Rails has **hardcoded** these conventions
  - □ E.g. AuthorsController searches '`new`' view in '`app/views/authors/new.html.erb`'
  - □ If you want to deviate, this must be configured
- Running '`rails new app_name`' or the generators produce files that adhere to these conventions

https://rubyonrails.org/doctrine/#convention-over-configuration

Rails Core Components



View

Controller

Model

Data storage

Action Pack

Action View
(renders template)

Action Dispatch
(parses HTTP, sessions, cookies, etc.)

Action Controller
(make data available, application flow)

Active Model
(e.g. validations)

Active Record (ORM)

Database (SQL, Graph..)

Railities
(core code)

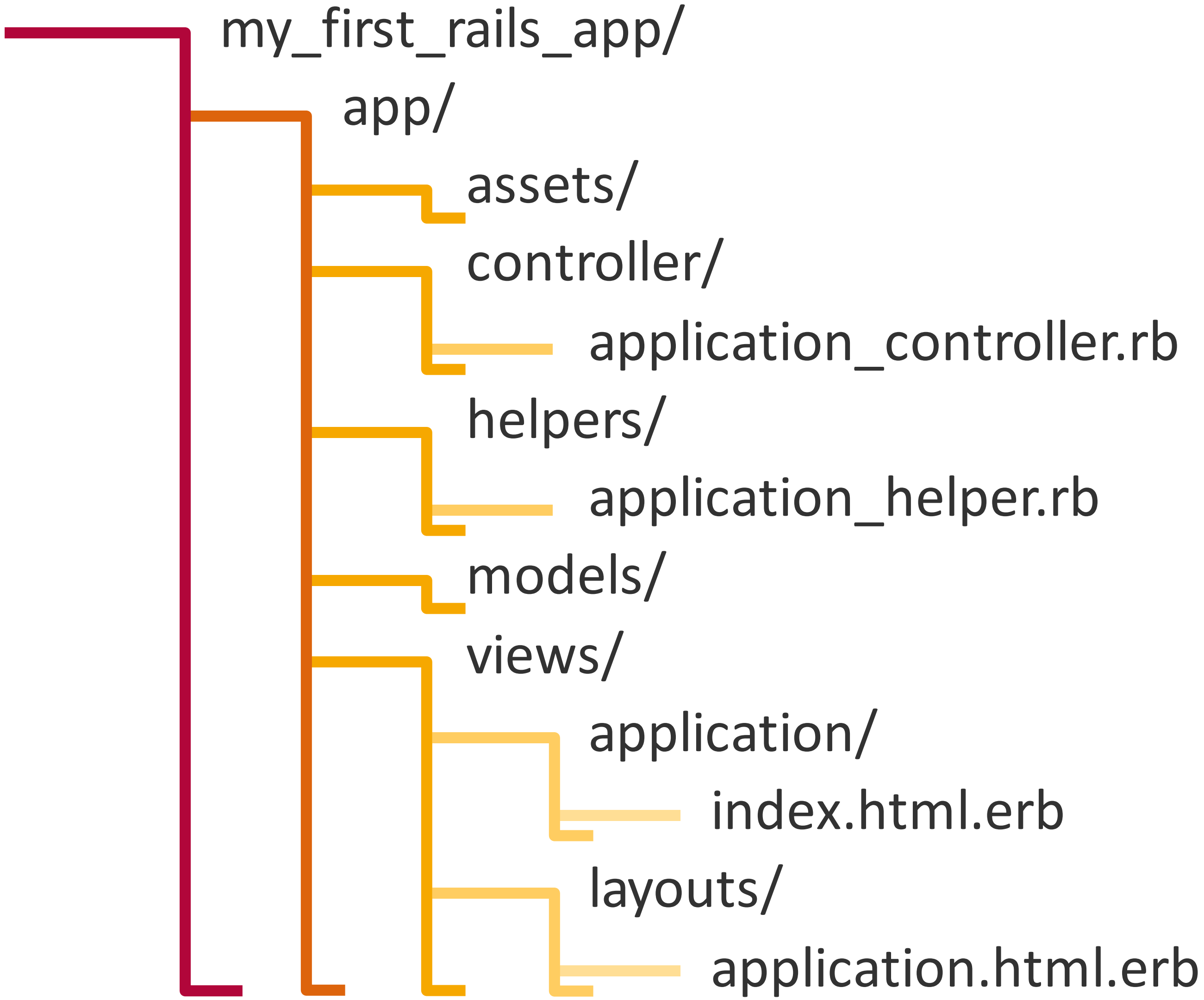
Active Support
(utility classes, e.g. i18n)

Action Mailer
(email services)

Gems
(packaged libraries)
https://rubygems.org/

http://adrianmejia.com/blog/2011/08/11/ruby-on-rails-architectural-design/

Rails Application Layout



```
my_first_rails_app/
    app/
        assets/
        controller/
            application_controller.rb
        helpers/
            application_helper.rb
        models/
        views/
            application/
                index.html.erb
            layouts/
                application.html.erb
```

RESTful Architecture

- **Re**presentational **S**tate **T**ransfer (REST) is a software architecture style

- **Client-server architecture**
  - □ Separation of concerns: user interface and data storage
- **Stateless Interactions**
  - □ No client context being stored on the server between requests
  - □ Session state is held in the client, can be transferred db for a period
- **Cacheability**
  - □ Responses must, implicitly or explicitly, define themselves as cacheable
- **Layered system**
  - □ Client does not know whether it is connected to the end server, or an intermediary

RESTful Architecture

- **Re**presentational **S**tate **T**ransfer (REST) is a software architecture style

- **Uniform Interface**
  - Requests specify the resource they want to access.
  - Resources are conceptually separate from return values, e.g. JSON, HTML
  - A returned resource (incl. metadata) is sufficient to modify or delete the resource
  - Messages include information on how to process them, e.g. media type
  - From an initial URI a client can use server-provided links to discover needed info
  - No need for hard-coded information on application structure

- Largest RESTful implementation: HTTP in World Wide Web

RESTful Architecture – HTTP verbs

- REST supports all 4 HTTP 1.1 verbs: GET, PUT, POST, DELETE
- Differentiation of collections and individual elements

Resource	GET	PUT	POST	DELETE
Single element http://localhost:3000/authors/1	Retrieve	Update or create	Create	Delete
Collection http://localhost:3000/authors	List	Replace	Create	Delete

Examples of Routes

GET /	# invoke “home” controller
GET /authors	# retrieve a list of all authors
GET /authors/new	# get the form to enter a new author
POST /authors	# create a new author
GET /authors/1	# show details of the first author
GET /authors/1/edit	# get the form to edit the first author
PUT /authors/1	# update the first author
DELETE /authors/1	# delete the first author

Introduction to Ruby on Rails

1. Ruby & Ruby on Rails
2. **Rails application**
3. Introductory Rails exercise

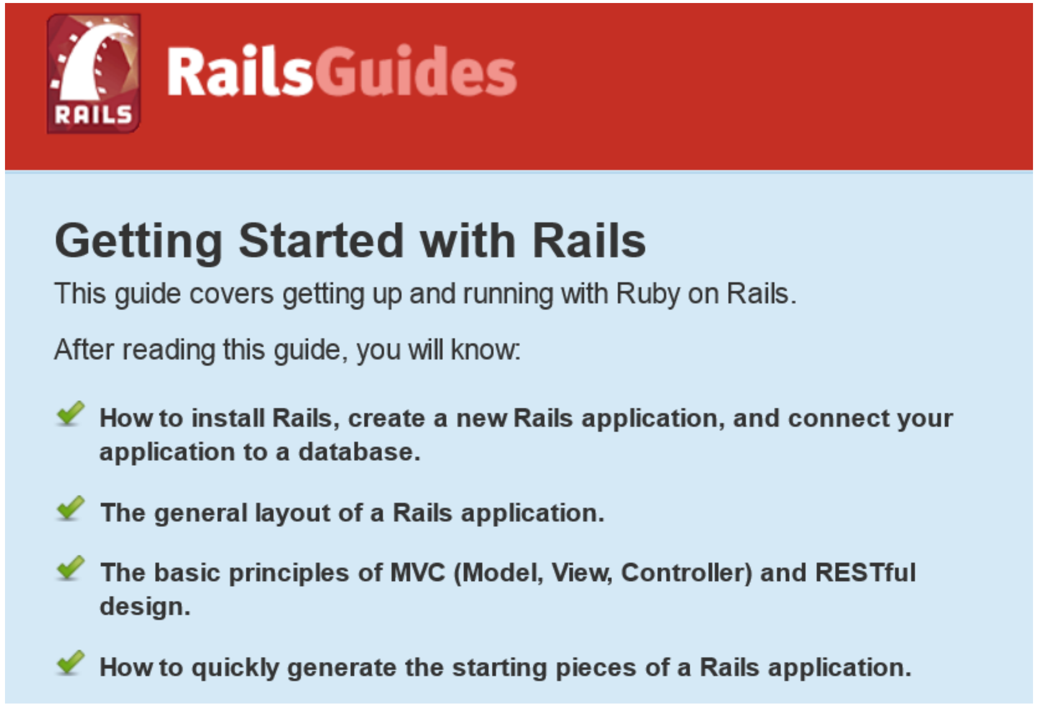
How to Start?

- **Option 1:** You use *Mac* or *Linux*
  - □ Install and use Ruby on Rails on your OS
  - □ Ruby version manager (e.g. rbenv, RVM)
  - □ http://guides.rubyonrails.org/getting_started.html#installing-rails
- **Option 2:** You have *Windows*
  - □ Install Windows Subsystem for Linux (WSL)
  - □ Linux subsystem in Windows! Install Ubuntu, then follow the normal Linux setup
  - □ Use Windows tools & editors
- **Option 3:** You want *separation*
  - □ Use some sort of virtual machine
  - □ E.g. using VirtualBox VirtualBox (https://www.virtualbox.org/) as the backend and Vagrant (https://www.vagrantup.com/) for provisioning
  - □ Use your own tools & editors, run the project in a headless VM

Comprehensive RoR tutorial

Recommended to work through / read this hands-on tutorial. Seriously.

http://guides.rubyonrails.org/getting_started.html



RailsGuides

Getting Started with Rails

This guide covers getting up and running with Ruby on Rails.

After reading this guide, you will know:

- ✓ **How to install Rails, create a new Rails application, and connect your application to a database.**
- ✓ **The general layout of a Rails application.**
- ✓ **The basic principles of MVC (Model, View, Controller) and RESTful design.**
- ✓ **How to quickly generate the starting pieces of a Rails application.**

Tip:

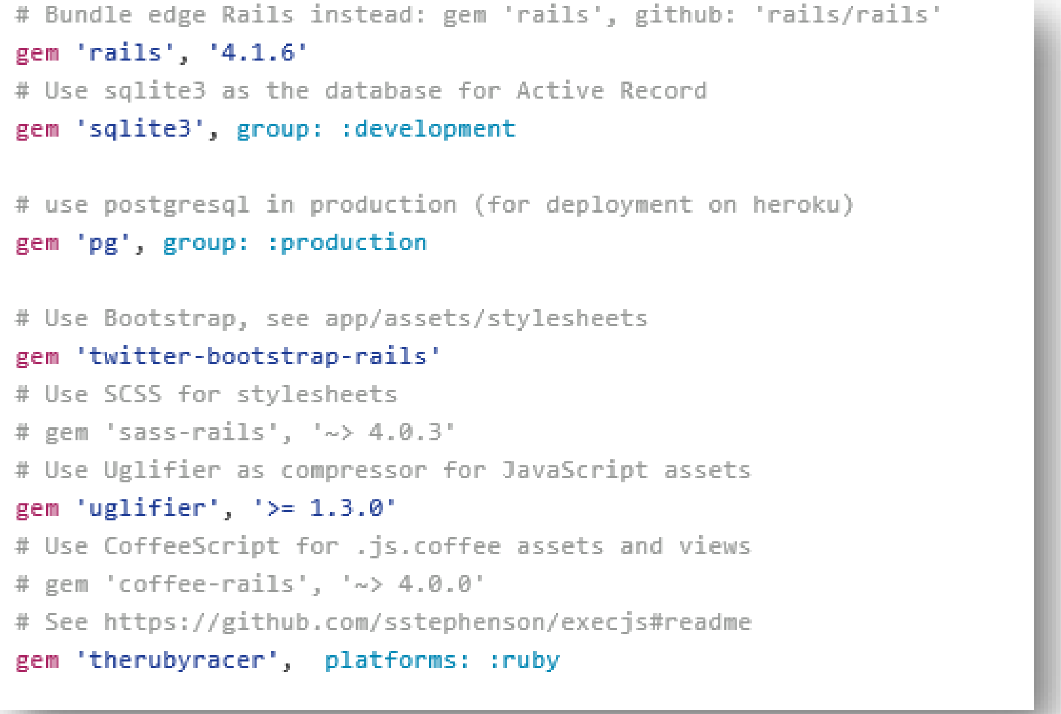
Before you start coding, make sure, the correct versions are installed.

```
$ ruby --version
$ rails --version
```

The following slides give a general overview

Bundler – Ruby package manager

- Ruby libraries :"gems"
- **Large ecosystem**
  - □ Most likely there is a gem for your need!
- `Gemfile` holds a list of required gems
  - □ Specify versions, e.g. gem 'rails' >= '4.1.6’
  - □ Alt. sources, e.g. :github => ”tkowark/sawyer”
- `Gemfile.lock` contains resolved dependencies
  - □ Should be under version control



```ruby
# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '4.1.6'
# Use sqlite3 as the database for Active Record
gem 'sqlite3', group: :development

# use postgresql in production (for deployment on heroku)
gem 'pg', group: :production

# Use Bootstrap, see app/assets/stylesheets
gem 'twitter-bootstrap-rails'
# Use SCSS for stylesheets
# gem 'sass-rails', '~> 4.0.3'
# Use Uglifier as compressor for JavaScript assets
gem 'uglifier', '>= 1.3.0'
# Use CoffeeScript for .js.coffee assets and views
# gem 'coffee-rails', '~> 4.0.0'
# See https://github.com/sstephenson/execjs#readme
gem 'therubyracer',  platforms: :ruby
```

Manually install a gem (Ruby package)

```
$ gem install
```

Install all gems listed as depencies in Gemfile

```
$ bundle install
```

`'--without'`, e.g. `--without production`, allows ignoring a section of the Gemfile

Info:

`Gemfile.lock` contains all the actually installed versions of gems.

rails – Main executable

Start interactive shell to test out ideas

```
$ rails console
```

Start new rails application

```
$ rails new
```

Generate boilerplate for models, controllers & views (vs ready-made backends of e.g. Django)

```
$ rails generate
```

Start the development server

```
$ rails server
```

Start a direct database shell

```
$ rails dbconsole
```

- Example: generate model, controller and view without controller specs

```
$ rails g scaffold author last_name:string
homepage:string --controller-specs false
```

http://guides.rubyonrails.org/command_line.html

rails – The cmd Utility

List all available commands

```
$ rails --help
```

List all routes, i.e. application URIs and controller actions

```
$ rails routes # Rails 5
```

Setup the database and run all migrations

```
$ rails db:setup db:migrate
```

Replace database with db layout from db/schema.rb
Do not run migrations.

```
$ rails db:schema:load
```

Run Rspec (testing framework for RoR) tests

```
$ bundle exec rspec
```

or

```
$ rspec
```

Tip:
Rails 5 changed some responsibilities.
Tasks are part of the 'rails' command now (instead of 'rake')

Info:
Running `schema:load` is advisable when setting up a completely new project. It is not intended to work around bad migrations.

Git – Distributed VCS

- **Install Git:**
  - sudo apt-get install git
  - http://git-scm.com/ (Installers for all systems)

- **Setting up user name and email:**
  - Mandatory to commit changes
  - Use your github credentials!

```
$ git config --global user.email “vorname.nachname@student.hpi.de”
$ git config --global user.name “Max Mustermann”
```

- Alternative: setting parameters only for one project:

```
$ cd /path/to/your/project
$ git config user.email “vorname.nachname@student.hpi.de”
$ git config user.name “Max Mustermann”
```

Quick Git Workflow

Checkout remote repository to local copy

```
$ git clone https://github.com/hpi-swt2/sport-portal
```

Change main layout template app/views/layouts/application.html.erb

Stage changes (add files from working copy to repository index)

```
$ git add app/views/layouts
```

List changes to be committed

```
$ git status
```

Commit with commit messages. Reference Github issue #25

```
$ git commit –m "Fixed issue #25"
```

Fetch and merge changes from remote repository

```
$ git pull
```

Publish local commits

```
$ git push
```

Note:
Difference in cloning
HTTPS (typing credentials)
and
SSH (submitting pubkey to GitHub)

Introduction to Ruby on Rails

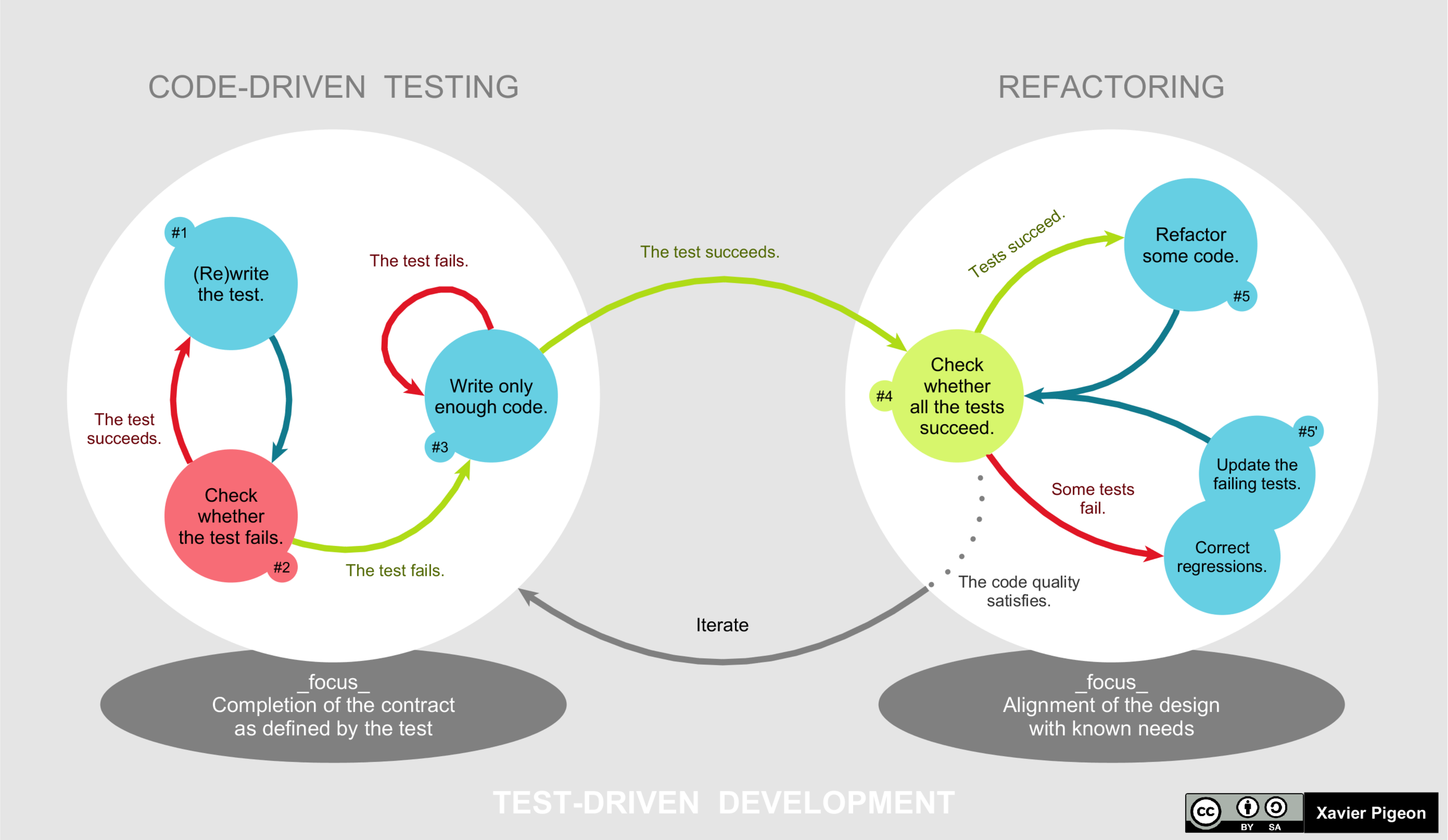
1. Ruby & Ruby on Rails
2. Rails application
3. **Introductory Rails exercise**
4. Additional Literature

Introductory Exercise

- **Goals**
  - Get familiar with Ruby (on Rails)
  - Create necessary accounts for the project
  - Setup your machine, editors and tools
  - Apply **Test-First Development** *(why?)*
- **Notes**
  - POs are not required to complete this task
  - POs will have customer meetings

Check your favorite editor for plug-ins / extensions that support Ruby on Rails.

Better tools == faster & more fun



https://upload.wikimedia.org/wikipedia/commons/0/0b/TDD_Global_Lifecycle.png

Acceptance Test-Driven Dev.

- **Acceptance Test**
  - Test to determine if requirements of specification are met
  - May involve black-box testing performed on a system
  - Enable user, customer or authorized entity to determine whether to accept the system
- **Test-Driven Development (TDD)**
  - Developer's tool to help create code that correctly performs operations
- **Acceptance Test-Driven Development (ATDD)**
  - Communication tool between the customer, developer, and tester to ensure that the requirements are well-defined.
  - TDD requires test automation. ATDD does not, although it helps
  - ATDD tests should be readable by the customer

Capybara Test Framework

- Simulate how a real user would interact with a web application
- Well suited for writing **acceptance & integration tests** for web applications
- Provides DSL for "surfing web pages"
  - e.g. `visit`, `fill_in`, `click_button`
- Integrates with RSpec
- Supports different "drivers", some support Javascript evaluation
  - Webkit browser engine
  - Selenium
    - Opens an actual browser window and performs actions within it

Integration & Acceptance Tests

```
require 'capybara/rspec'

describe "the signin process", :type => :feature do
  before :each do
    User.create!(:email => 'user@example.com', :password => 'password')
  end

  it "signs me in" do
    visit new_session_path
    within("#session") do
      fill_in 'Email', :with => 'user@example.com'
      fill_in 'Password', :with => 'password'
    end
    click_button 'Sign in'
    expect(page).to have_css('div#success')
  end
end
```

- *What are some issues with this test?*

Tip:

Capybara includes aliases for RSpec syntax:

`feature` instead of **`describe …, :type => :feature`**,

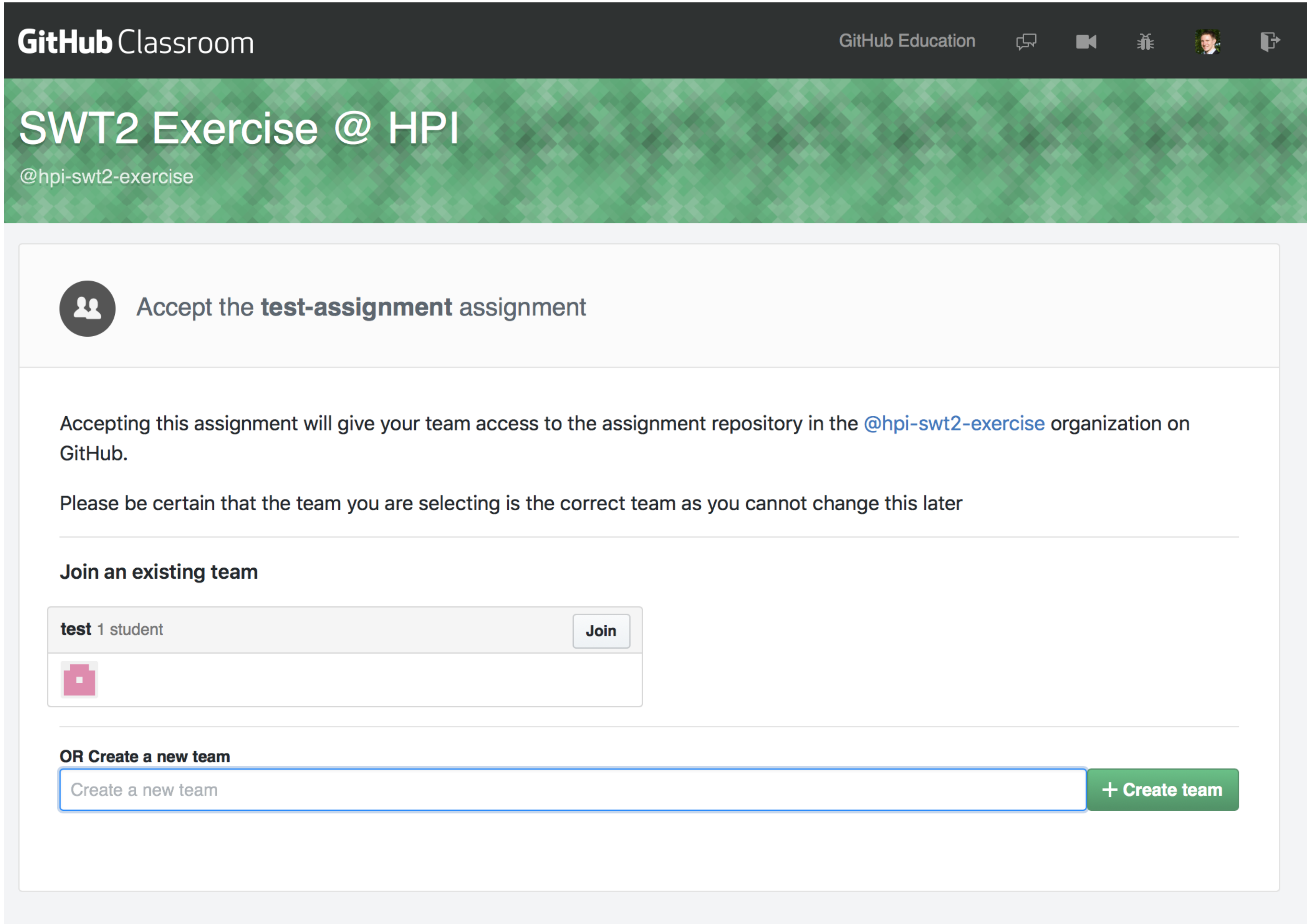
`scenario` instead of **`it`**,

`background` instead of **`before`**,

`given/given!` instead of **`let/let!`**

- https://github.com/jnicklas/capybara

Exercise – Rails Project



SWT2 Exercise @ HPI

@hpi-swt2-exercise

Accept the **test-assignment** assignment

Accepting this assignment will give your team access to the assignment repository in the @hpi-swt2-exercise organization on GitHub.

Please be certain that the team you are selecting is the correct team as you cannot change this later

Join an existing team

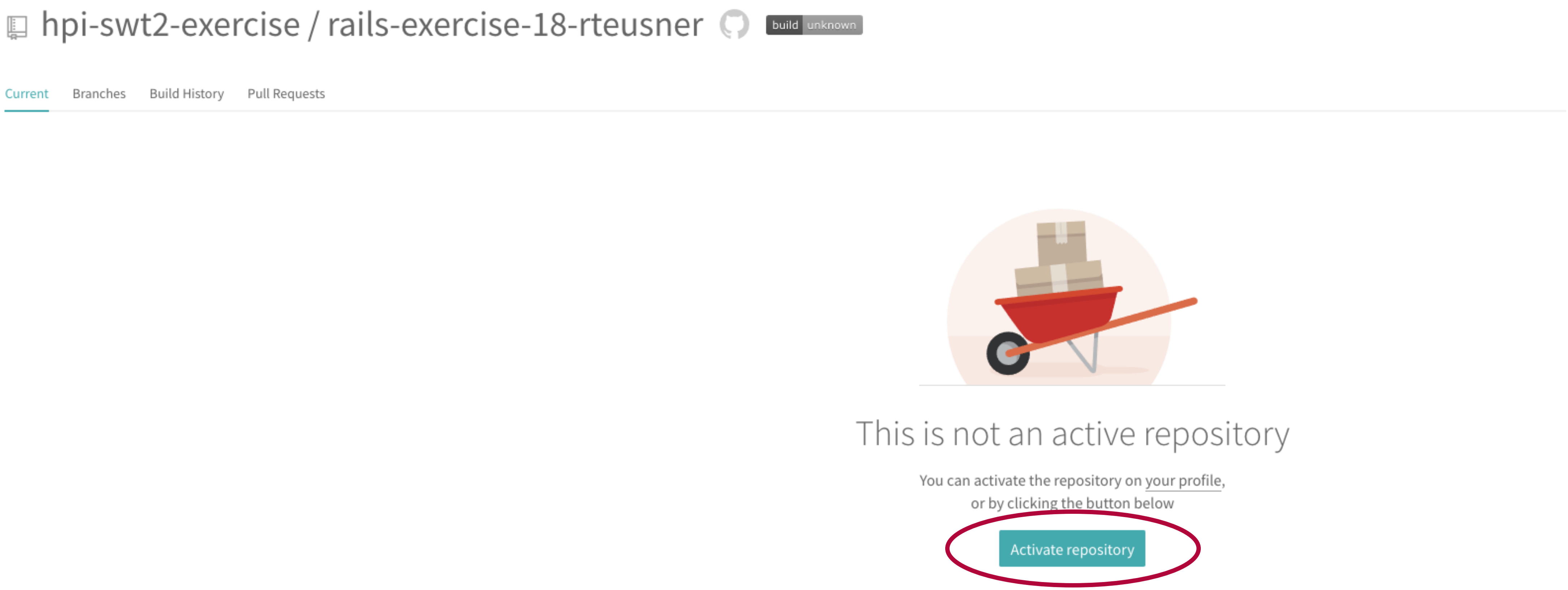
test 1 student **Join**

OR Create a new team

Create a new team **+ Create team**

Exercise – Travis CI

- **Travis CI**
  - □ Continuous Integration Server
  - □ *(Experience? What does that integrate? Why is it needed?)*

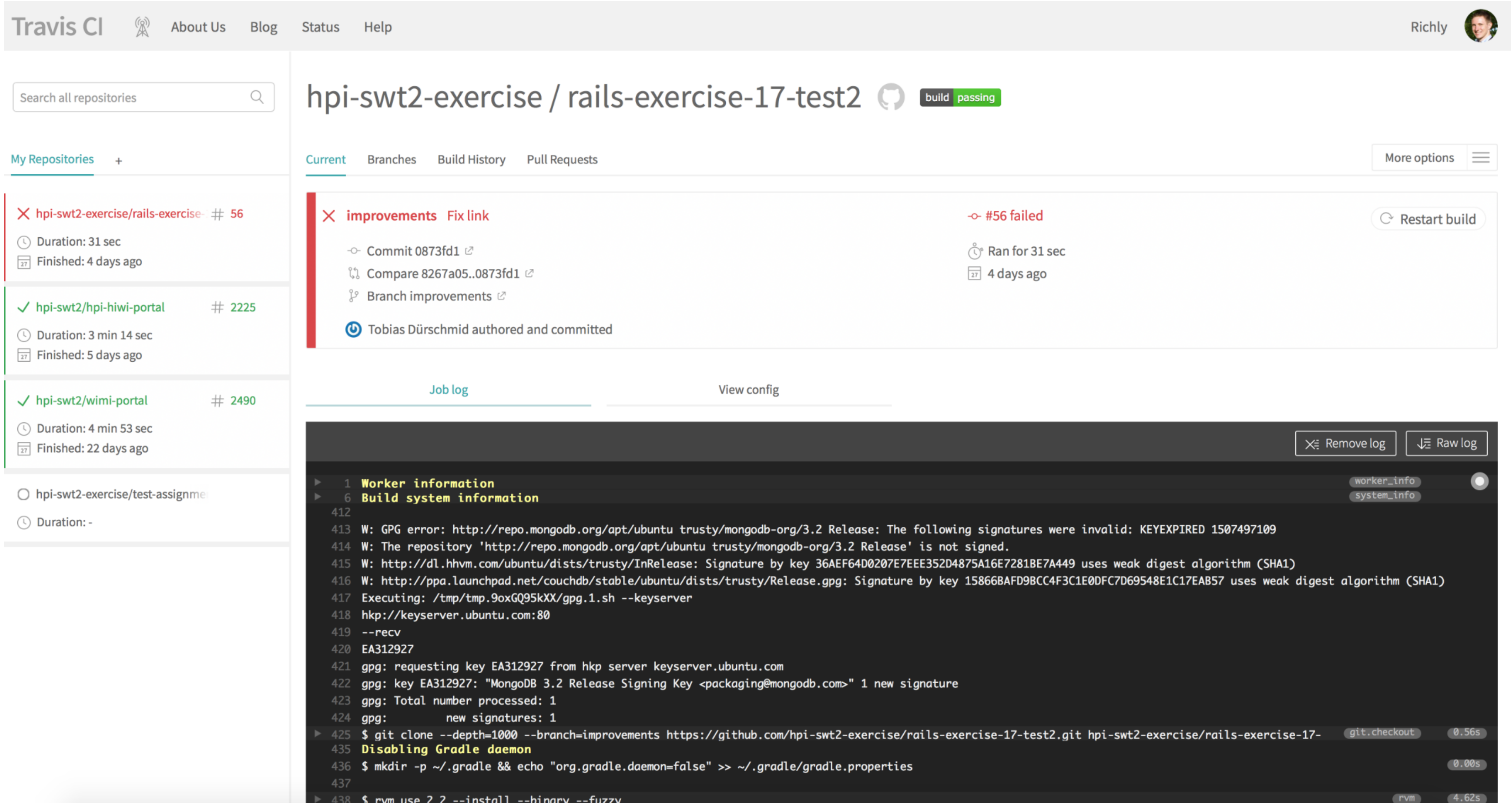


hpi-swt2-exercise / rails-exercise-18-rteusner

This is not an active repository

You can activate the repository on your profile,
or by clicking the button below

Exercise – Travis CI



hpi-swt2-exercise / rails-exercise-17-test2 build passing

✗ hpi-swt2-exercise/rails-exercise- # 56
Duration: 31 sec
Finished: 4 days ago

✓ hpi-swt2/hpi-hiwi-portal # 2225
Duration: 3 min 14 sec
Finished: 5 days ago

✓ hpi-swt2/wimi-portal # 2490
Duration: 4 min 53 sec
Finished: 22 days ago

○ hpi-swt2-exercise/test-assignme
Duration: -

✗ improvements Fix link
Commit 0873fd1
Compare 8267a05..0873fd1
Branch improvements
Tobias Dürschmid authored and committed

#56 failed
Ran for 31 sec
4 days ago

```
Worker information
Build system information

W: GPG error: http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 Release: The following signatures were invalid: KEYEXPIRED 1507497109
W: The repository 'http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 Release' is not signed.
W: http://dl.hhvm.com/ubuntu/dists/trusty/InRelease: Signature by key 36AEF64D0207E7EEE352D4875A16E7281BE7A449 uses weak digest algorithm (SHA1)
W: http://ppa.launchpad.net/couchdb/stable/ubuntu/dists/trusty/Release.gpg: Signature by key 15866BAFD9BCC4F3C1E0DFC7D69548E1C17EAB57 uses weak digest algorithm (SHA1)
Executing: /tmp/tmp.9oxGQ95kXX/gpg.1.sh --keyserver
hkp://keyserver.ubuntu.com:80
--recv
EA312927
gpg: requesting key EA312927 from hkp server keyserver.ubuntu.com
gpg: key EA312927: "MongoDB 3.2 Release Signing Key <packaging@mongodb.com>" 1 new signature
gpg: Total number processed: 1
gpg:         new signatures: 1
$ git clone --depth=1000 --branch=improvements https://github.com/hpi-swt2-exercise/rails-exercise-17-test2.git hpi-swt2-exercise/rails-exercise-17-
Disabling Gradle daemon
$ mkdir -p ~/.gradle && echo "org.gradle.daemon=false" >> ~/.gradle/gradle.properties

$ rvm use 2.2 --install --binary --fuzzy
```

Next Steps

- **Get to know GitHub**
  - Create a GitHub account and get familiar with it, all the cool projects are there!
- **Familiarize yourselves with Ruby on Rails and web dev**
  - Setup your machine and tools
  - Read the Ruby on Rails guide (https://guides.rubyonrails.org/getting_started.html)
  - Understand the core Ruby language constructs (coming from other languages) (https://www.ruby-lang.org/en/documentation/ruby-from-other-languages/)
- **Start the exercise once released**
  - Visit GitHub classroom link (on the SWTII course website)
  - Accept the assignment
  - Follow the instructions in the README file

Introduction to Ruby on Rails

1. Ruby & Ruby on Rails
   - Ruby on Rails
   - Convention over configuration
   - Rails' core components
   - RESTful architecture
2. Rails application
   - Folder structure
   - Databases
   - `rails`, `git`
3. Introductory Rails exercise
   - TDD & ATDD
   - Travis CI
4. **Small demo?**

Introduction to Ruby on Rails

Software Engineering II
WS 2019/20

Enterprise Platform and Integration Concepts

Rails by *Erik* (CC BY-NC 2.0) https://www.flickr.com/photos/calaveth/3904320343