Ruby on Rails

Sustainable productivity for web-application development

Ostrya Labs

Clean Code Matters

Agenda – Day 1

- Theory -
  - The Language
  - The Platform
  - Rich UI
  - Eco System
  - Testing
  - Deployment

Agenda – Day 1

- Lab -
    - ✗ Install Rails
    - ✗ Install Ruby
    - ✗ Rails Tutorial – Chapter 1
    - http://www.codecademy.com/tracks/javascript
    - http://www.codecademy.com/tracks/web

The Language

Static Vs Dynamic

- Clarity at Compile Time
- Strong Type checking
- No unexpected type issues at runtime
- C, C++, Java

- No type check at Compilation
- Evaluation with real time data at run time
- Runtime surprises possible
- Ruby, Python, PHP

Compiled Vs Interpreted

Compiled -

- Original program translated to native machine instructions and is ready to execute.
- Executable is a separate entity
- Eg: C, C++, Java

Interpreted -

- Translation to machine instructions at the time of execution
- Executable is not a separate entity
- Eg: Python, Ruby

Ruby

- Ruby is a Dynamic, Object Oriented General purpose programming language
- Developed in mid 1990s
- Yukihiro "Matz" Matsumoto
- Current Version – 2.1.0 – Dec 2013
- Multiple installations available – MRI/JRuby/MacRuby

Ruby Flavors

- Matz Ruby Interpreter[MRI]
- Ruby Enterprise Edition[REE]
- JRuby – On JVM
- Rubinius - C++
- MacRuby – For Mac Users
- IronRuby - .Net
- MagLev - Smalltalk

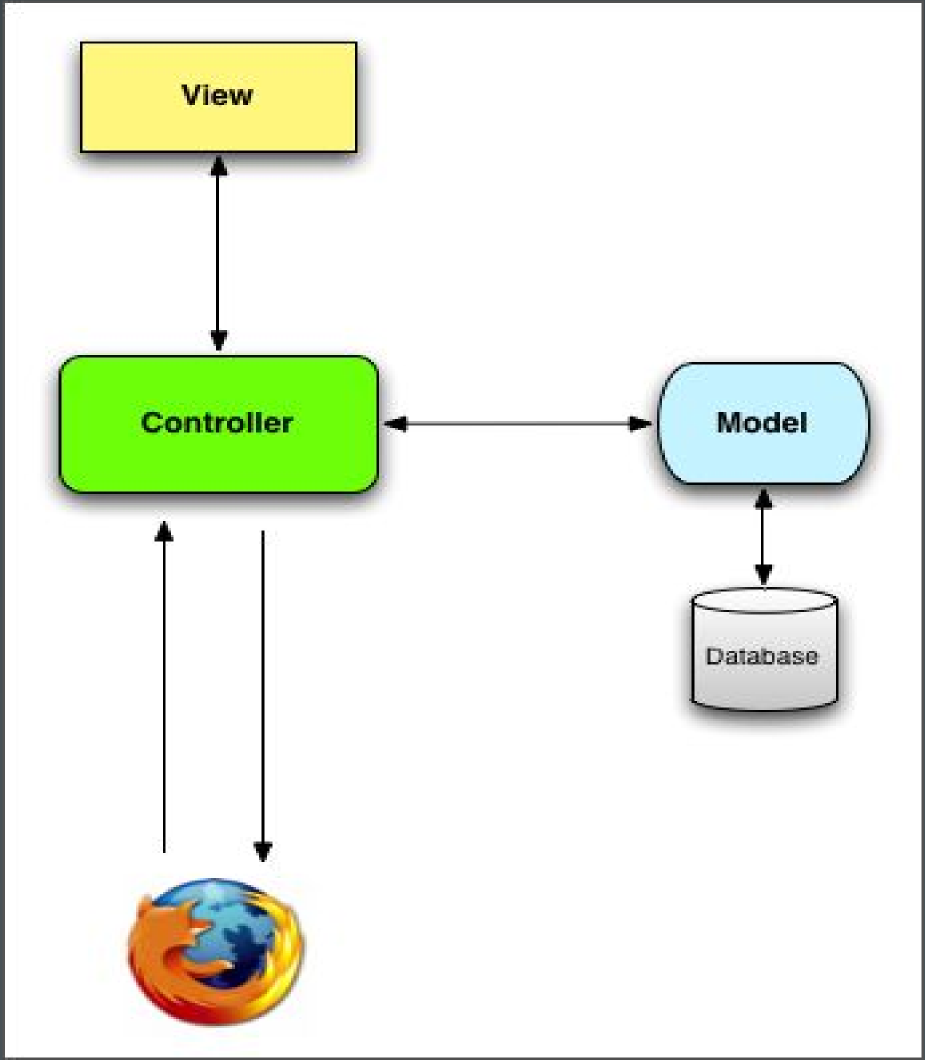
RubyGems

- Gem is a packaged Ruby Library
- RubyGems is the standard package manager for the language
- Equivalent to JARs in Java Language
- Rails is just a Ruby Gem
- Gemfile – A place to mention all the gems in the application

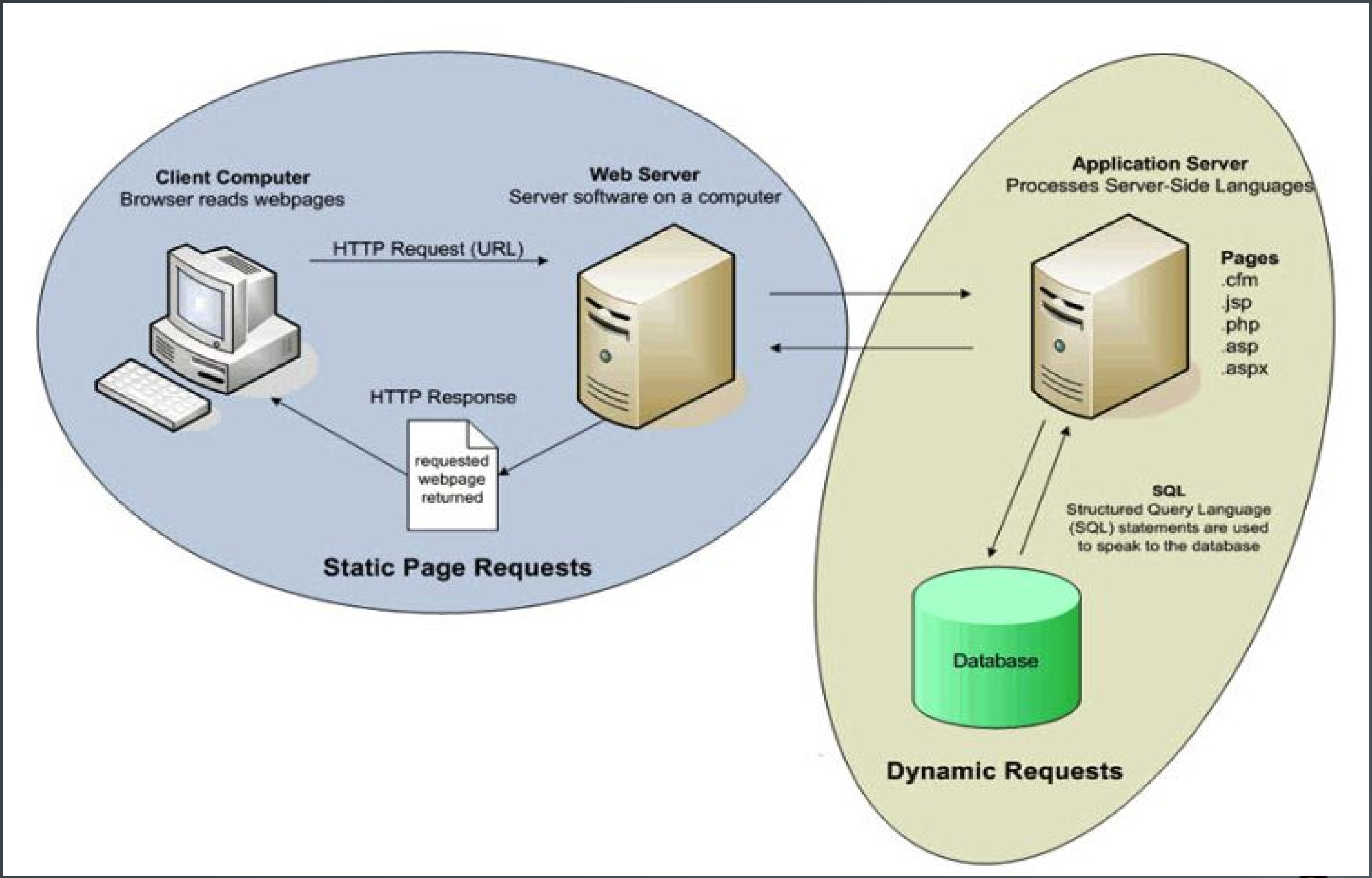
The Platform

Architecture

- MVC
  - Model
  - View
  - Controller



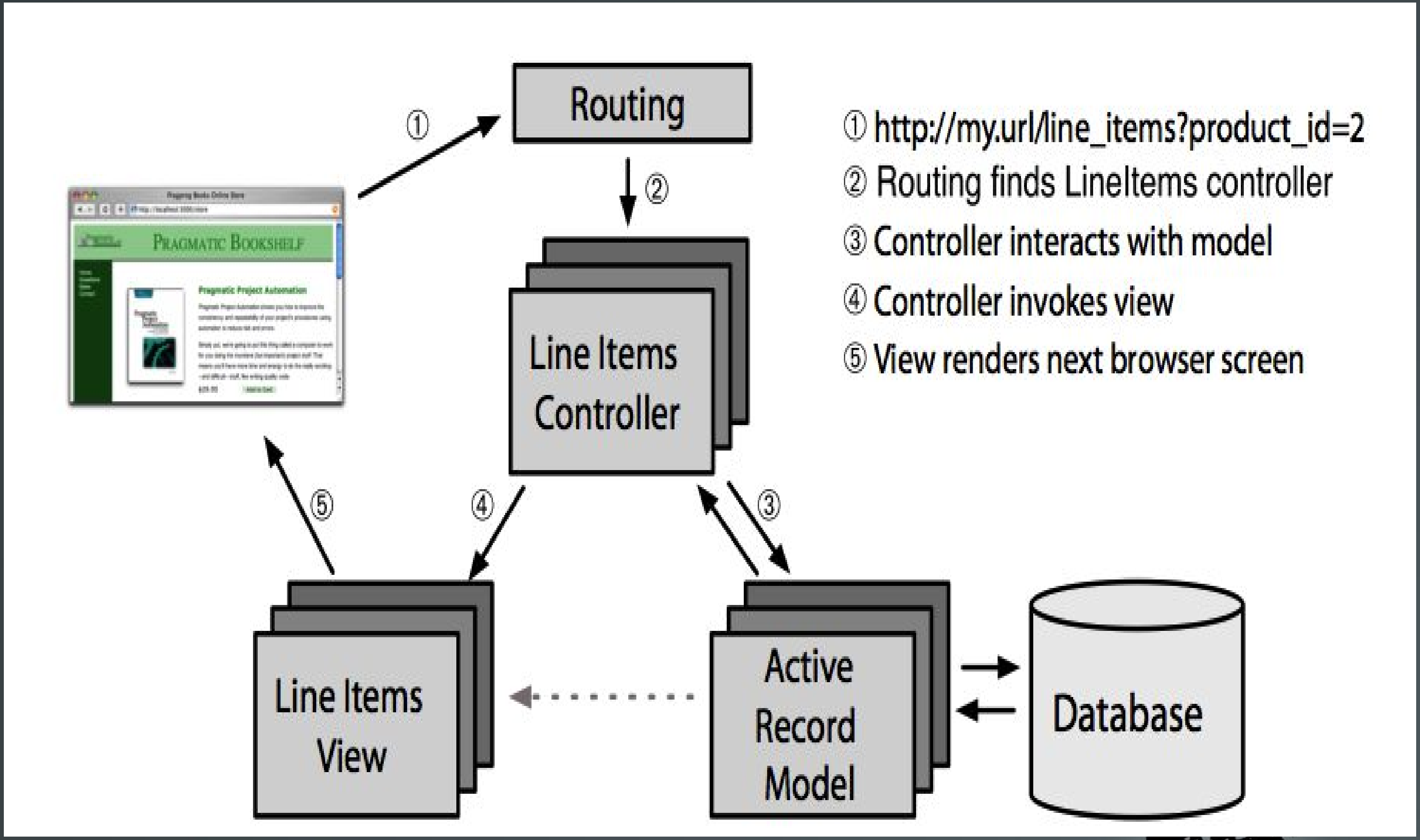
Static Vs Dynamic Requests



Ruby on Rails

- Rails is an Open source Web Application development framework
- It's an MVC framework
- Written in Ruby language
- Developed by David Heinemeier Hansson
- Released in 2006
- Current Release – 4.0 [2013]

Rails Architecture


Routing
Line Items Controller
Line Items View
Active Record Model
Database
PRAGMATIC BOOKSHELF
① http://my.url/line_items?product_id=2
② Routing finds LineItems controller
③ Controller interacts with model
④ Controller invokes view
⑤ View renders next browser screen

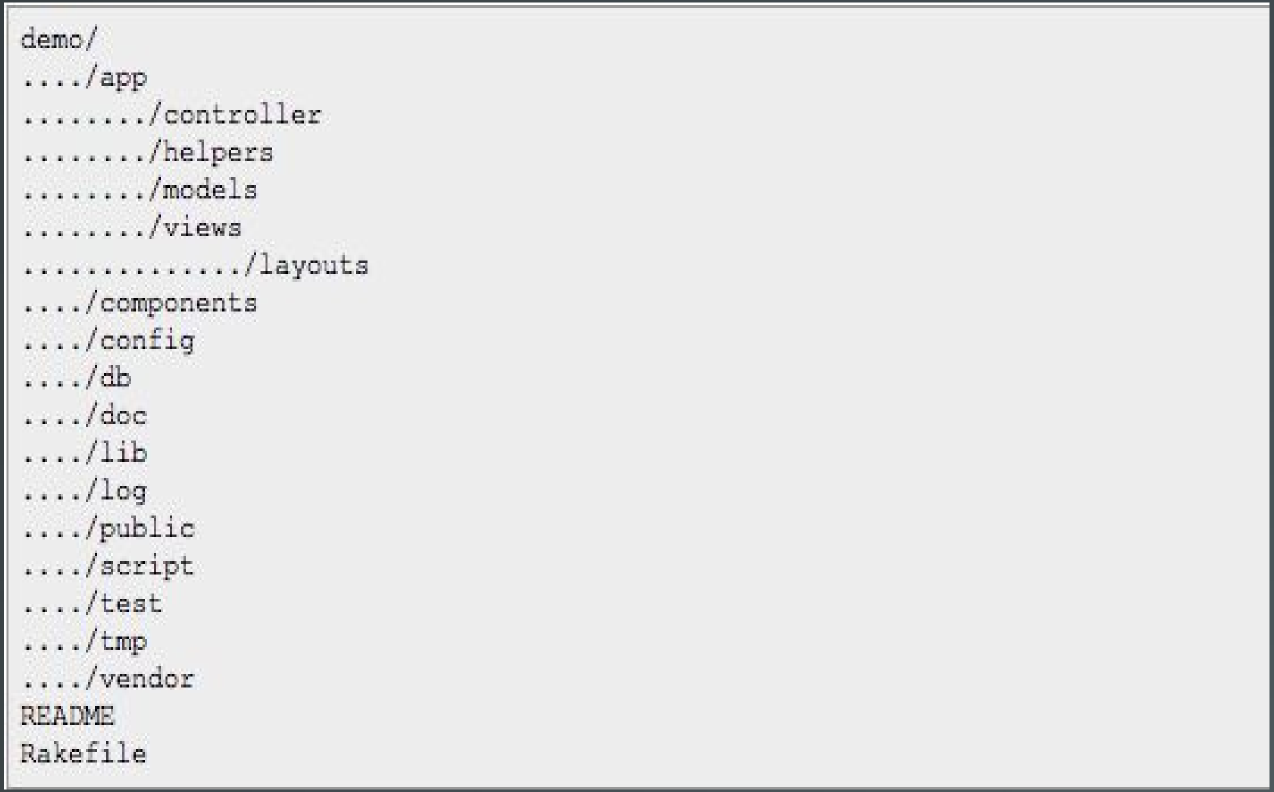
Rails Philosophy

- DRY – Don't Repeat Yourself
  - Writing same code again and again is a bad thing
- Convention over Configuration
  - Follow the conventions and there is no need to spend time in configuration

Rails Application Structure

- Rails is an opiniated software
- Starting a new application creates a directory structure by default.
- Every component of the Application has a predefined place holder
- All Applications have standard directory structure
- Establishing convention over configuration

Rails Application Structure



```
demo/
..../app
......../controller
......../helpers
......../models
......../views
............../layouts
..../components
..../config
..../db
..../doc
..../lib
..../log
..../public
..../script
..../test
..../tmp
..../vendor
README
Rakefile
```

Rich UI

JavaScript

- A programming language which runs in the browser
- Makes web pages interactive
- Eg. if you have to update cart in a page when user shops

Cascading StyleSheets

- Stylesheet language used for describing the look and feel of web pages
- Can be used with any markup language
- Separation of document cotent from document presentation
- Have a 'property:value' format
- Eg: font, color etc

Asset Pipeline

- Framework to concatenate and compress JavaScript and CSS assets
- CoffeeScript
- SASS
- ERB – Embedded Ruby

jQuery

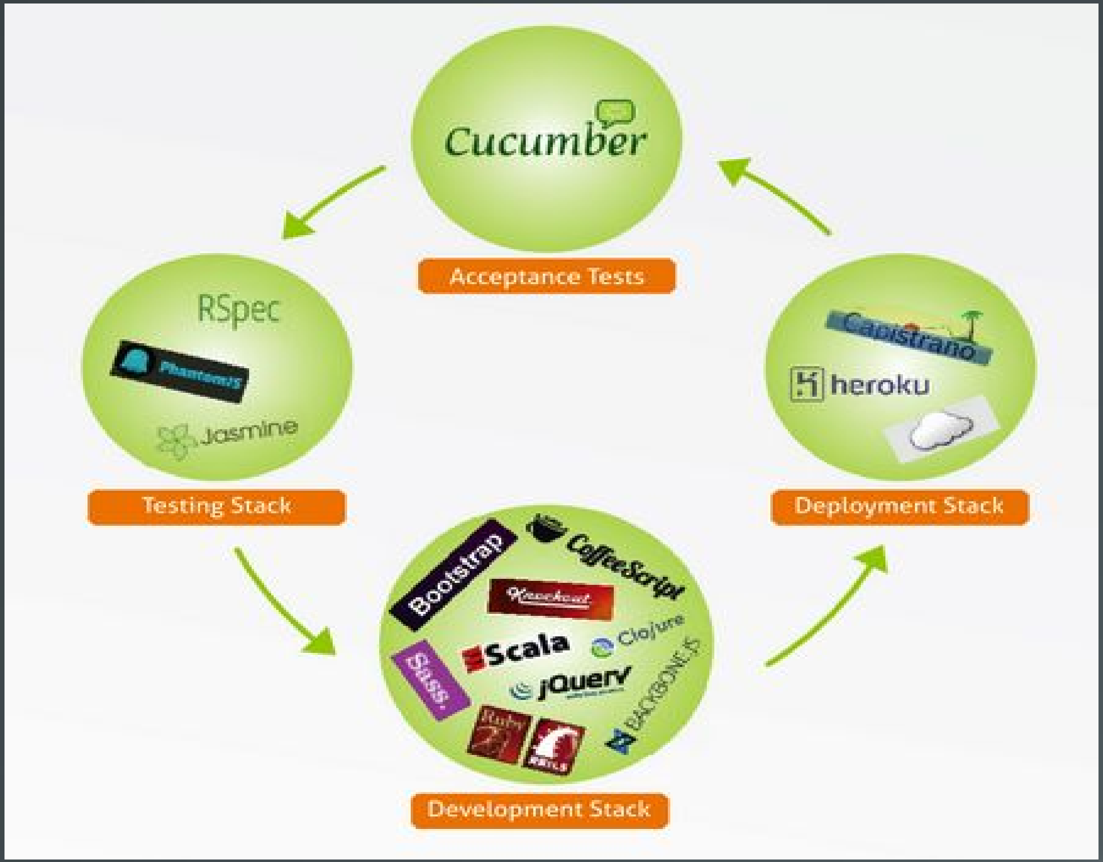
- Fast, small, and feature-rich JavaScript library
- Makes client side scripting easier and faster
- Highly versatile and extensible
- Frameworks
    - KnockoutJS
    - AngularJS

The Eco System

Developing a Rails App

- Editor
- UX Design
- Databases
- Testing
- Version Control
- Deployment

Components of a Rails App



Editors

- EMacs
- Vim
- Sublime Text
- RubyMine
- TextMate
- Komodo Edit
- NetBeans
- RadRails

Twitter Bootstrap

- Framework which gives out of the box CSS and JavaScript for building rich and attractive UI
- Current version is 3.0
- Sass[Sazzy CSS] is the default for Rails
- gem 'bootstrap-sass'

Databases

- Rails supports a suite of databases
- Default implementation covers support for MySQL, PostgreSQL, SQLite
- Active Record – Objects carry both the persistent data and the behaviour which operates on the data

Version Control - Git

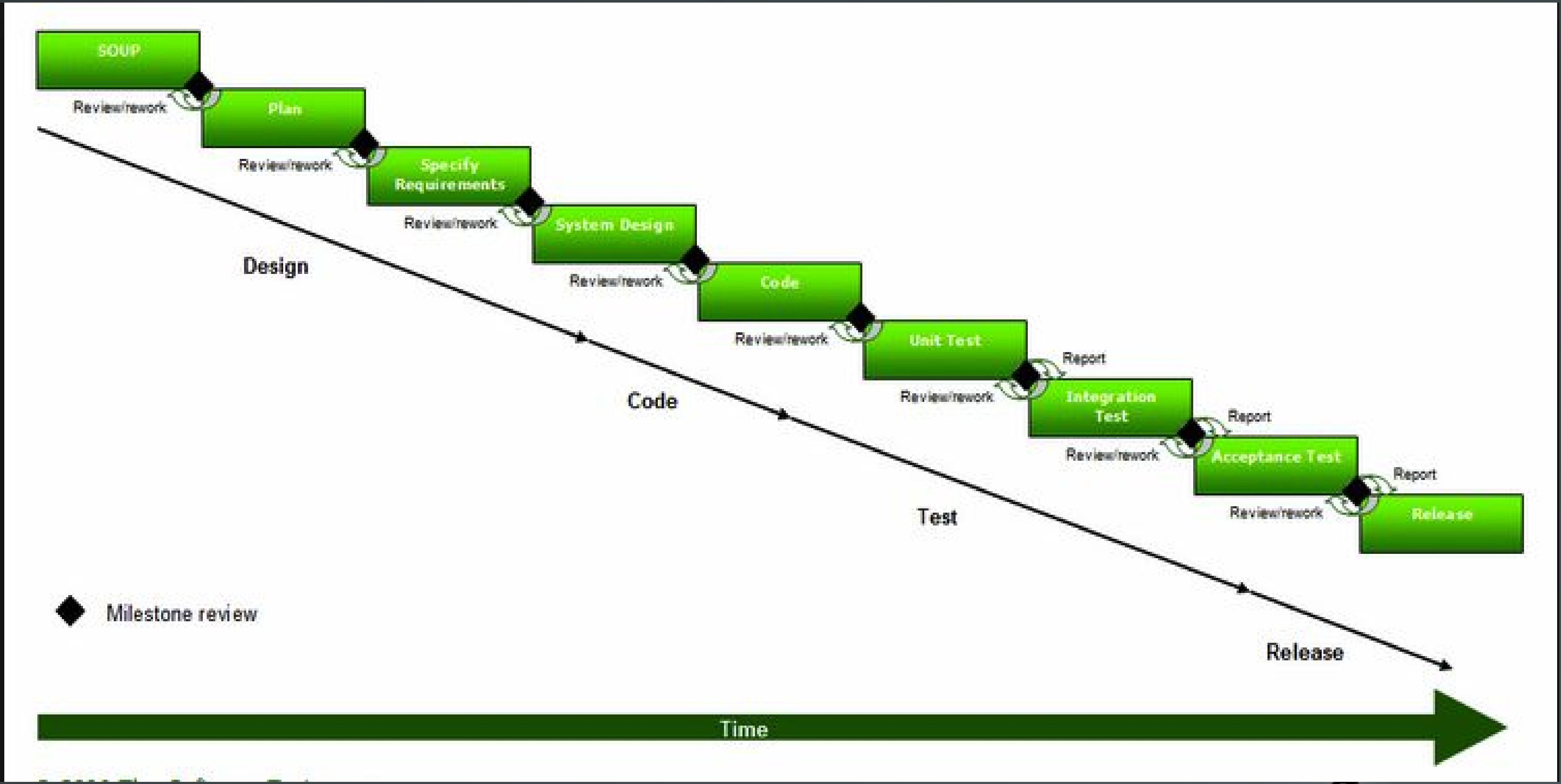
- Distributed version control system
- Creates repositories, adds files to it, create branches, merge branches and undoes changes
- Most popular version control system in RoR community
- Github is the remote repository where the source code can be backed up
- Need to have a Github account

Testing

Testing Approaches

- Traditional Waterfall Model –
  - Post Development
- Test Diven Development [TDD]
  - Before Development
- Behaviour Driven Development [BDD]
  - Acceptance Testing

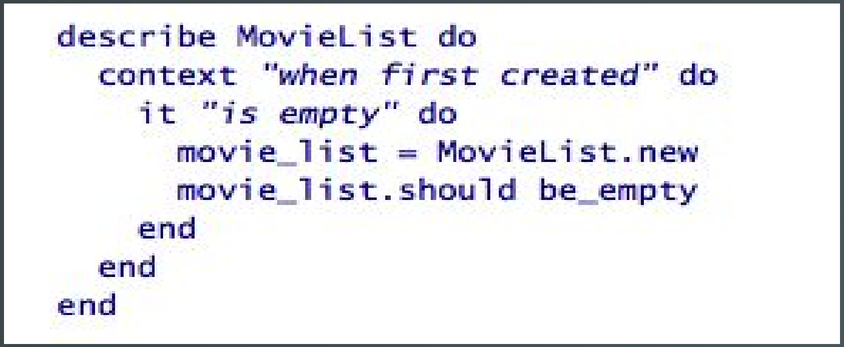
Traditional Waterfall Model



TDD - RSpec

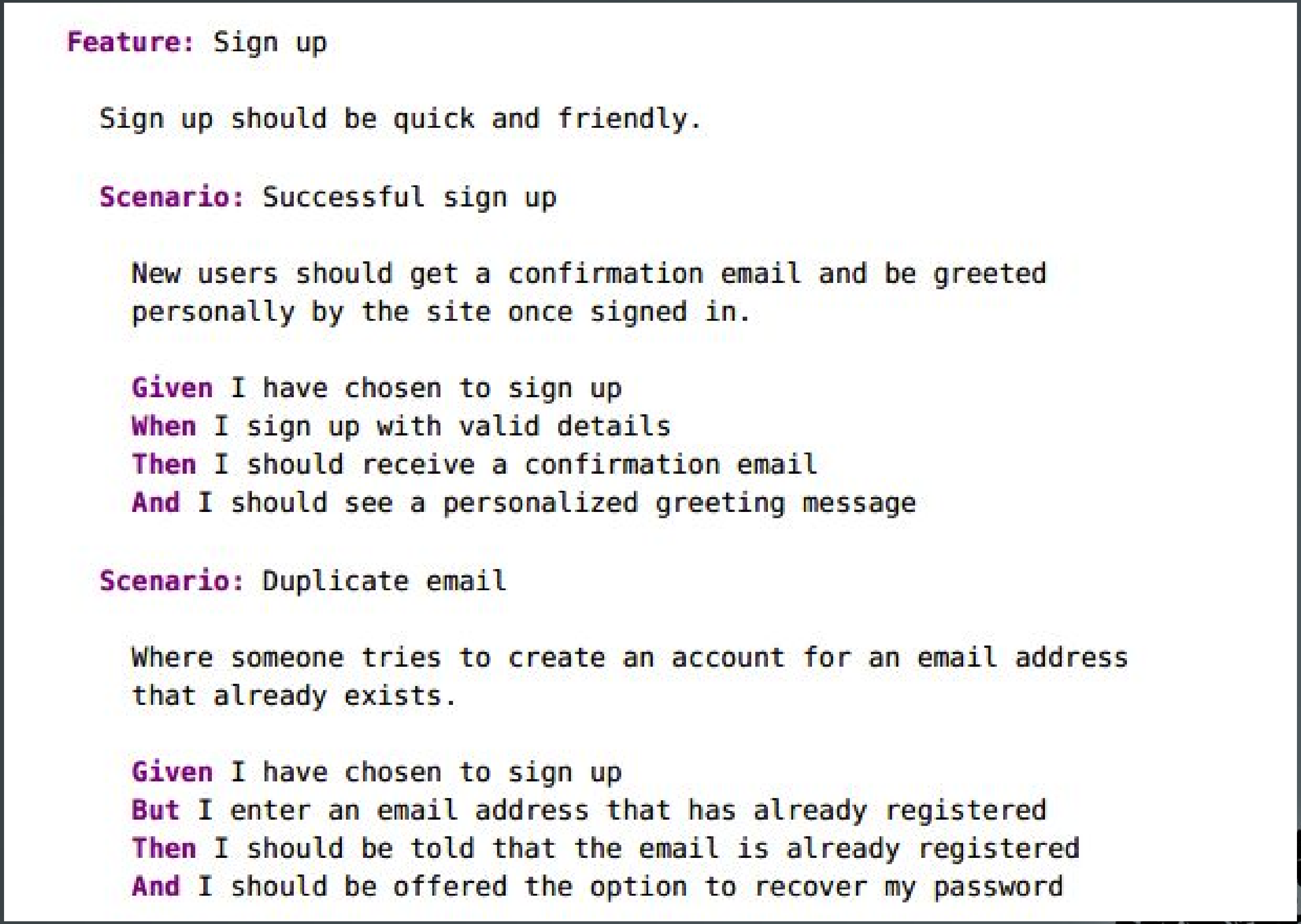
Red / Green / Refactor

- Write a Test Case and Run it
  - Watch it Fail
- Write the Code and Run Test case again
  - Watch it Pass
- Refactor Code – For Better code



```
describe MovieList do
  context "when first created" do
    it "is empty" do
      movie_list = MovieList.new
      movie_list.should be_empty
    end
  end
end
```

BDD – Scenarios



```
Feature: Sign up

  Sign up should be quick and friendly.

  Scenario: Successful sign up

    New users should get a confirmation email and be greeted
    personally by the site once signed in.

    Given I have chosen to sign up
    When I sign up with valid details
    Then I should receive a confirmation email
    And I should see a personalized greeting message

  Scenario: Duplicate email

    Where someone tries to create an account for an email address
    that already exists.

    Given I have chosen to sign up
    But I enter an email address that has already registered
    Then I should be told that the email is already registered
    And I should be offered the option to recover my password
```

BDD – Using Cucumber

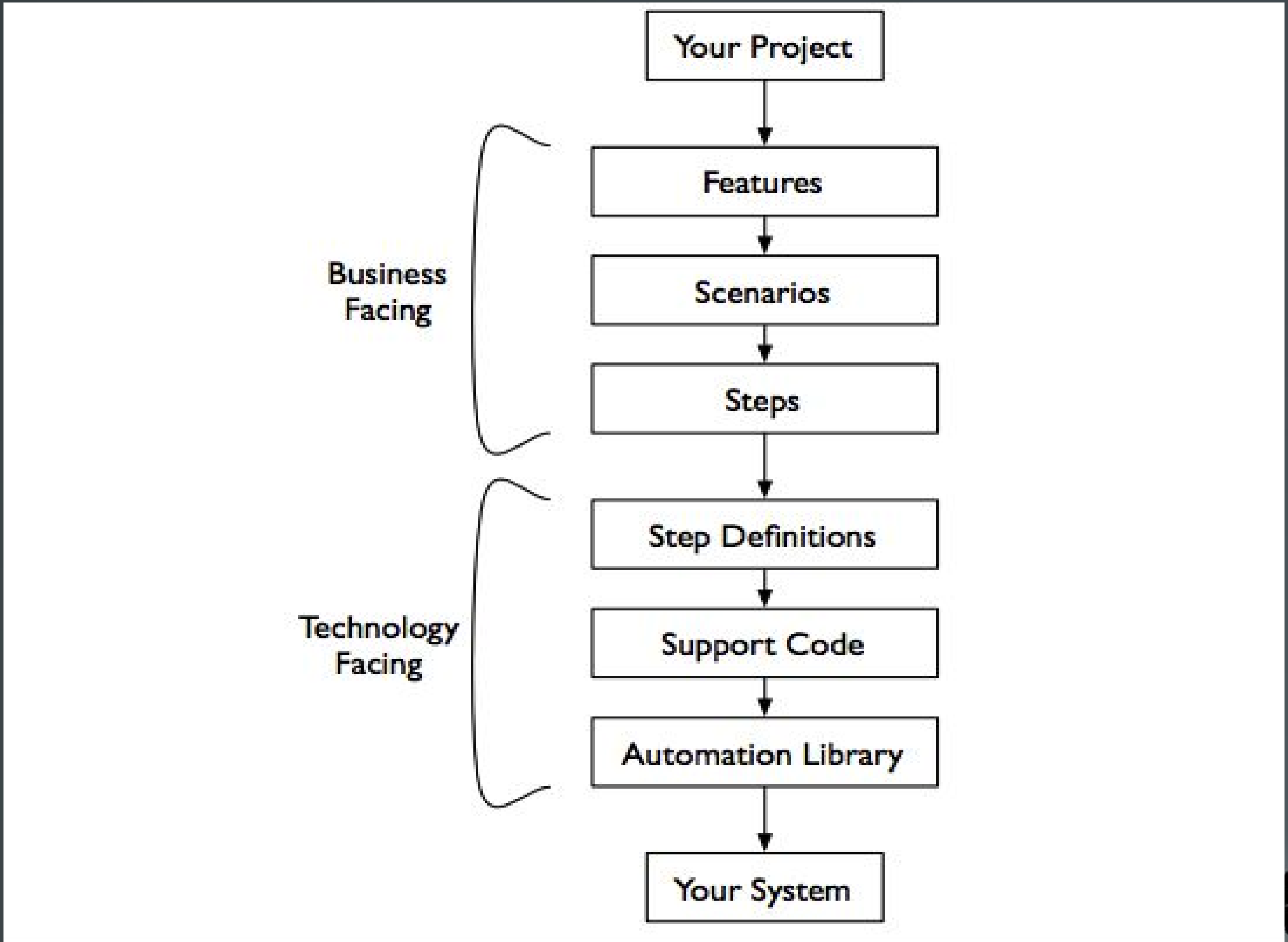
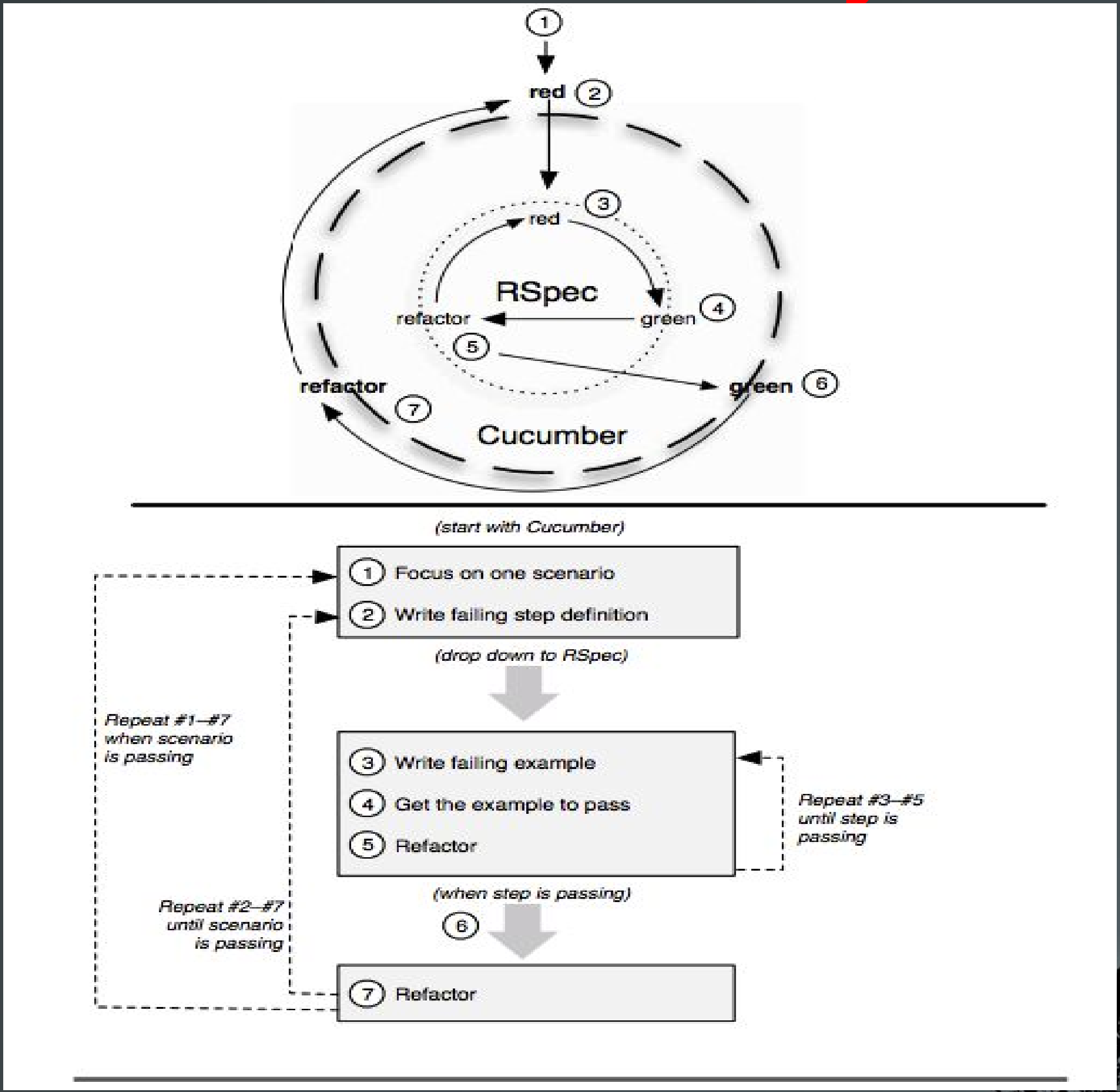


Figure 1—Cucumber testing stack

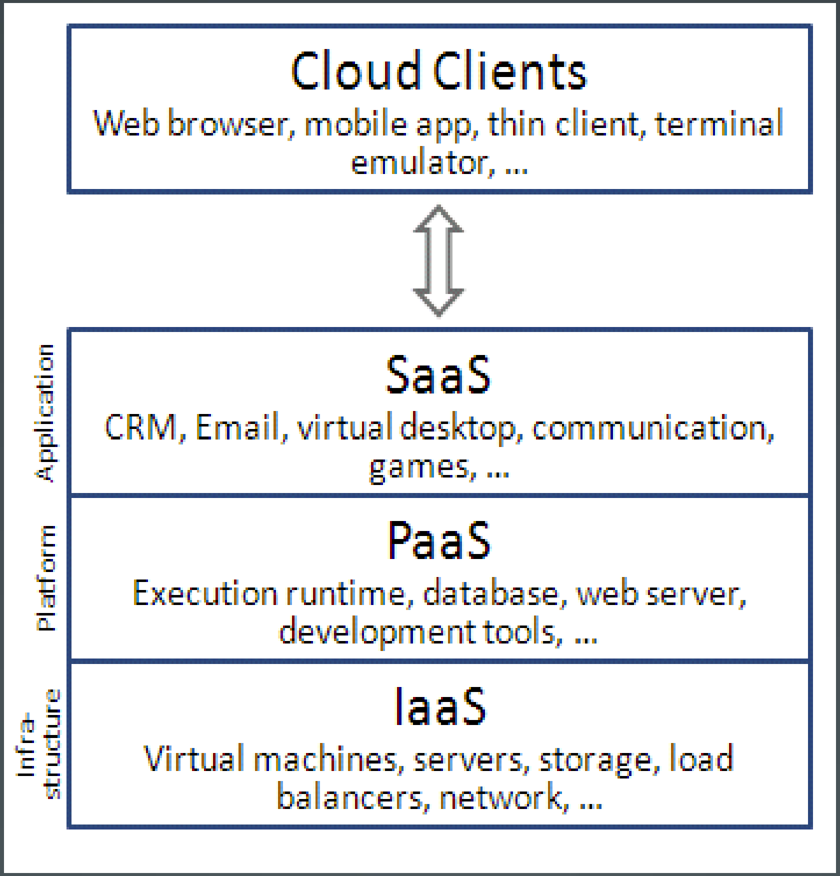
BDD – TDD Cycle



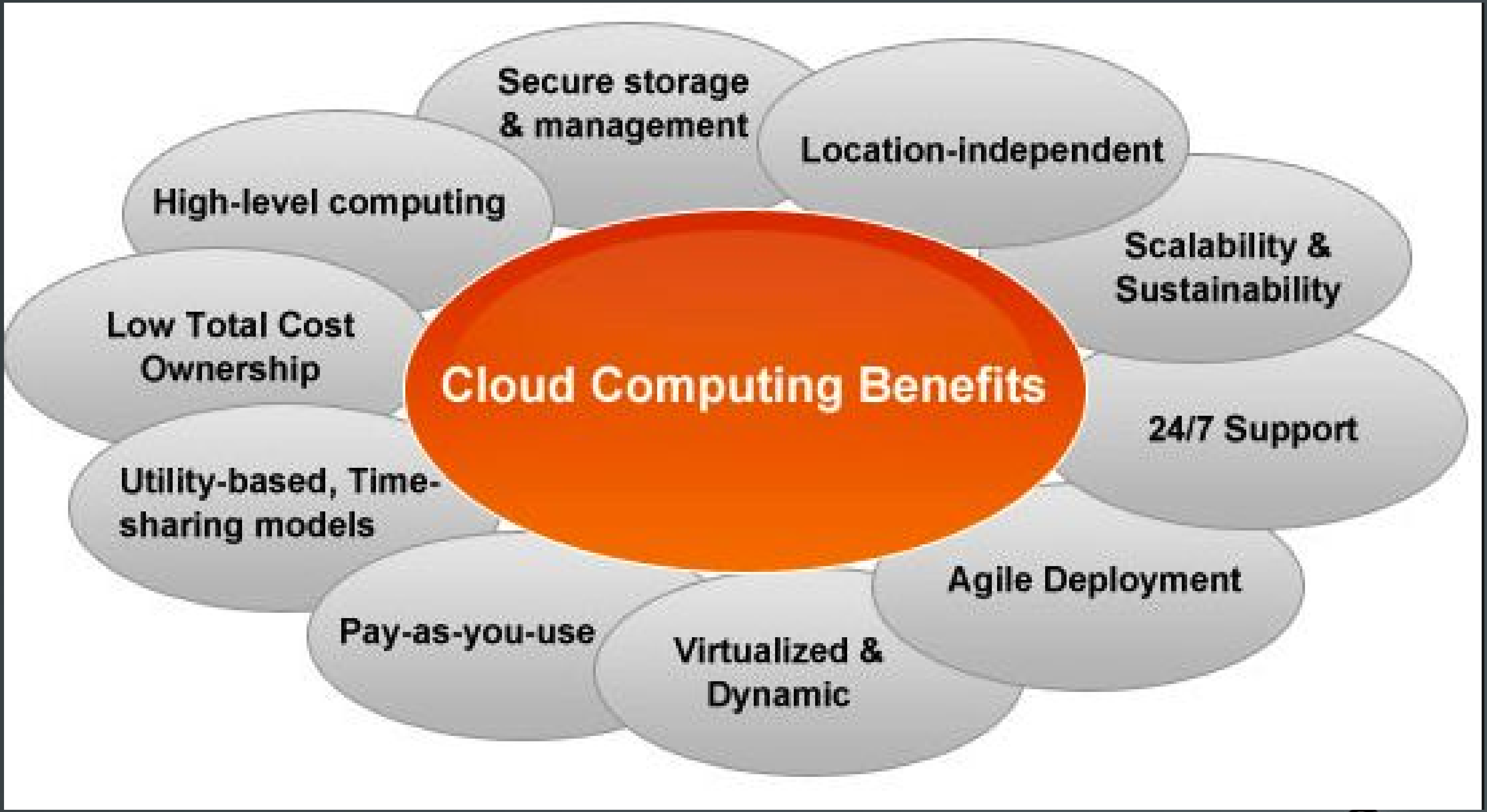
Deployment

Cloud Computing

Computing services offered by a third party, available for use when needed, that can be scaled dynamically in response to changing needs.


Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
Application
SaaS
CRM, Email, virtual desktop, communication, games, ...
Platform
PaaS
Execution runtime, database, web server, development tools, ...
Infra-structure
IaaS
Virtual machines, servers, storage, load balancers, network, ...

Cloud Benefits



Deployment - Heroku

- Cloud – Platform as a Service
- Acquired by Salesforce.com
- Initially supported only Ruby – now supports a range of languages
- Heroku lets you deploy, run and manage applications
- ”Heroku” Toolbelt

Some extras....

DRY KISS YAGNI

- DRY
  - Do not Repeat Yourself
  - Avoid code duplication
  - If code is duplicated try to abstract the code into a method or new class
- KISS
  - Keep It Simple & Stupid
  - Make your designs simple & easy to understand
- YAGNI
  - You Are not Going to Need It
  - Do not over-enigeer
  - Solve the problem in hand, don't assume and do things for future

Make the Code Talk

- Bad
    - var x, y, z
    - def do_it(x)
      end
    - No one understands what you are trying to do
- Good
    - var deposit, customer, interest
    - def calculate_and_credit_interest(deposit)
      end
    - Variable, Method & Class name should convey the meaning and intent of what you are trying to do

Make the Code Talk

- Write short methods
  - Methods more than 10-20 lines are difficult to understand in long run
  - Practice writing short methods
  - If method is getting bigger split into smaller methods
- Write short classes
  - Good thumb rule is having classes 60-70 lines
  - If class is getting longer see possibility of creating new class

SOLID Principles

- Single responsibility principle
  - A class should have only a single responsibility
- Open/closed principle
  - software entities … should be open for extension, but closed for modification
- Liskov substitution principle
  - objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program
- Interface segregation principle
  - many client-specific interfaces are better than one general-purpose interface
- Dependency inversion principle
  - one should Depend upon Abstractions. Do not depend upon concretions

REpresentational State Transfer

- Web Services are viewed as resources
- Can be uniquely identified by their URLs
- Explicit use of HTTP methods to denote the invocation of different operations
- highly reusable across platforms since they rely on basic HTTP protocol
- being preferred for integration with backend enterprise services

REpresentational State Transfer

CRUD Principle

- POST - Create a resource
- GET - Retrieve a resource
- PUT – Update a resource
- DELETE - Delete a resource

Thank You!