

JRuby (for Rubyists)

Tyler Jennings
Principal Consultant

tyler@obtiva.com
[@tyler_jennings](https://twitter.com/tyler_jennings)



You Should:



AGILITY APPLIED.
SOFTWARE DELIVERED.

- Have JRuby installed
- Have Java 1.6
- Run some examples

History

Bootstrapping



AGILITY APPLIED.
SOFTWARE DELIVERED.



(j)irb

```
> 1 + 1  
=> 2  
> ary = []  
=> []  
> ary << [1,2,3]  
=> [[1, 2, 3]]  
> ary.flatten  
=> [1, 2, 3]  
>
```

Using Java (Classes)

Importing

```
1 require 'java'  
2  
3 import javax.swing.JFrame  
4  
5 frame = JFrame.new("Test")  
6 frame.default_close_operation =  
7   JFrame::EXIT_ON_CLOSE  
8 frame.setSize(450,450)  
9 frame.visible = true  
10
```

importing_classes.rb

By Package



AGILITY APPLIED.
SOFTWARE DELIVERED.

```
3 class TestUI
4   include_package "javax.swing"
5   include_package "java.awt"
6
7   def initialize
8     frame = JFrame.new("Test")
9     frame.default_close_operation =
10       JFrame::EXIT_ON_CLOSE
11     frame.setSize(450,450)
12
13   box = JTextField.new
14   box.maximum_size = Dimension.new(30,30)
15   frame.add box
16
17   frame.visible = true
18 end
19 end
20 TestUI.new
```

importing_by_package.rb

Subclassing



AGILITY APPLIED.
SOFTWARE DELIVERED.

```
1 require 'java'  
2 class RubyMap < java.util.HashMap  
3 end  
4  
5 p RubyMap.new.methods.grep(/key/).sort
```

subclassing.rb

Interfaces

```
7 #Implement an interface (optional)
8 include ActionListener
9
10 def initialize
11   frame = JFrame.new("Calculator")
12   frame.default_close_operation = JFrame::EXIT_ON_CLOSE
13   frame.setSize(450,450)
14   button = JButton.new("Press ME")
15
16   button.add_action_listener self
17   button.add_action_listener do |event|
18     puts "BLOCK: " + event.source.label
19   end
20
21   frame.add button
22   frame.visible = true
23 end
24
25 def actionPerformed(event)
26   puts "METHOD: " + event.source.label
27 end
```

interfaces.rb

Extensions



AGILITY APPLIED.
SOFTWARE DELIVERED.

```
3 JHashMap = java.util.HashMap
4
5 class JHashMap
6   def +(other)
7     map = JHashMap.new
8     map.put_all self
9     map.put_all other
10    map
11  end
12  def print
13    to_a.join(">")
14  end
15 end
16
17 one_two = JHashMap.new(:one => 1) +
18           JHashMap.new(:two => 2)
19
20 puts one_two.print
```

extensions.rb

Methods

```
1 require 'java'  
2 import javax.swing.JFrame  
3  
4 frame = JFrame.new("Test")  
5 #Java version  
6 frame.setSize(450,450)  
7 #Jruby's  
8 frame.set_size(450, 450)  
9  
10 #Java setter  
11 frame.setVisible true  
12 #JRuby's version  
13 frame.visible = true
```

methods.rb

Grid layout

```
3 class ButtonLayoutUI
4   include_package 'javax.swing'
5   include_package 'java.awt'
6
7   def initialize
8     frame = JFrame.new("Calculator")
9     frame.default_close_operation = JFrame::EXIT_ON_CLOSE
10    frame.setSize(100,100)
11    digits = JPanel.new
12    digits.layout = GridLayout.new 3, 3
13
14    (1..9).each do |n|
15      digits.add JButton.new n.to_s
16    end
17    frame.add digits
18    frame.visible = true
19  end
20 end
21 ButtonLayoutUI.new
```

layout.rb

Exercise

- Build a Calculator UI
- Using calculator.rb

```
9 require 'java'  
10 require 'calculator'  
11  
12 class CalculatorUI  
13   def initialize  
14     @calculator = Calculator.new  
15   end  
16 end  
17  
18 CalculatorUI.new
```

calculator_ui_exercise.rb

```
1 require 'java'  
2  
3 # Usage:  
4 #  
5 # calculator = Calculator.new  
6 #  
7 # calculator.evaluate("1")  
8 # calculator.evaluate("+")  
9 # calculator.evaluate("1")  
10 # calculator.evaluate("=")  
11 # 2 == calculator.current_value  
12 #  
13 class Calculator  
14   OPERATORS = ["+", "-", "/", "*"]  
15   FUNCTIONS = ["=", "CE"]  
16   OPERATIONS = OPERATORS + FUNCTIONS
```

calculator.rb



BREAK TIME



AGILITY APPLIED.
SOFTWARE DELIVERED.



Deploying (Rails)



Warbler

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

warble - T

```
warble compiled           # Feature: precompile all Ruby files
warble config              # Generate a configuration file to customize your war
warble executable          # Feature: make an executable archive
warble gemjar               # Feature: package gem repository inside a jar
warble pluginize           # Install Warbler tasks in your Rails application
warble version              # Display version of Warbler
warble war                  # Create the project .war file
warble war:clean            # Remove the .war file
warble war:debug             # Dump diagnostic information
```



JRuby Rack



AGILITY APPLIED.
SOFTWARE DELIVERED.

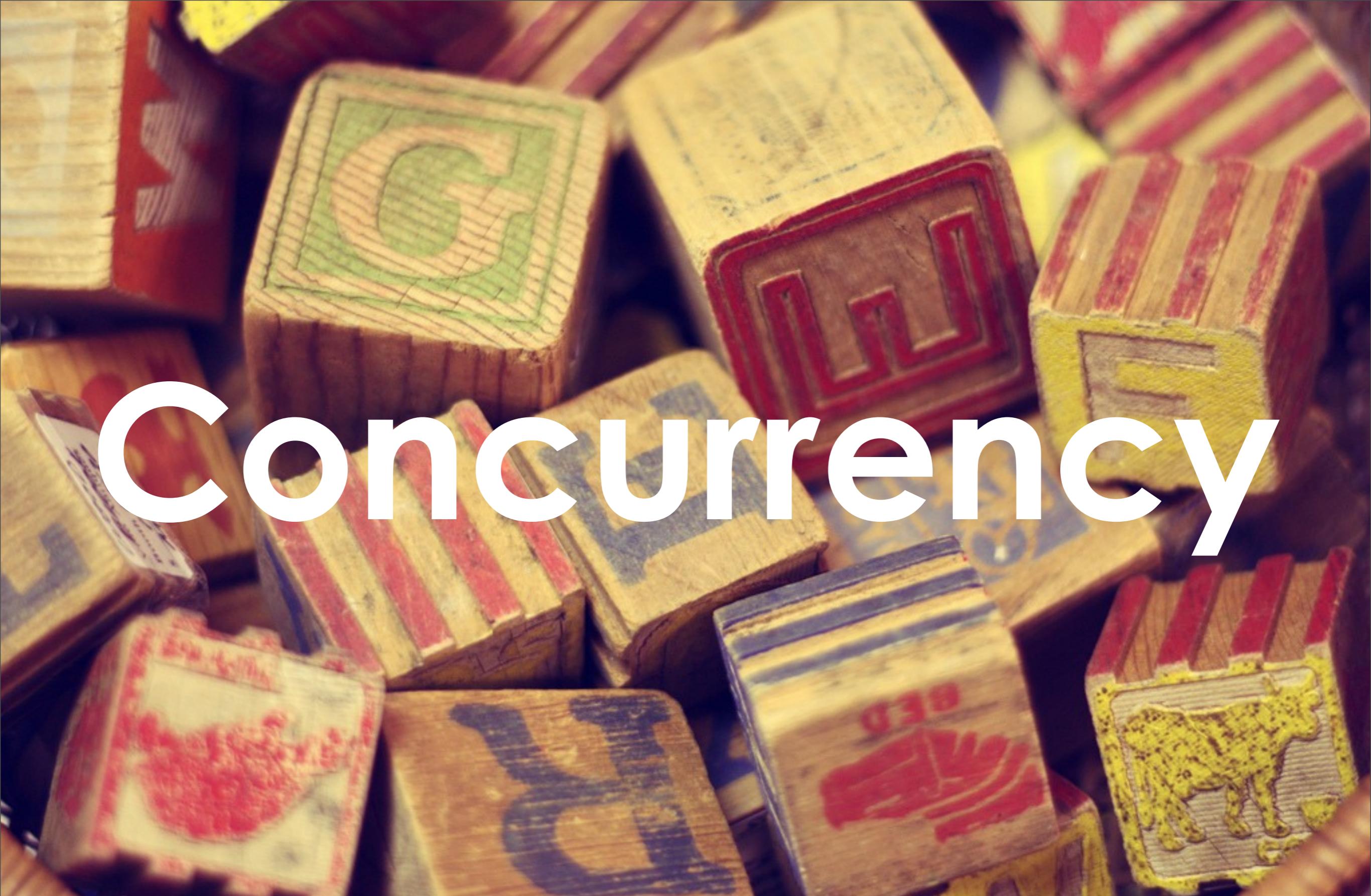


Excercise



AGILITY APPLIED.
SOFTWARE DELIVERED.

Concurrency





In Rails



AGILITY APPLIED.
SOFTWARE DELIVERED.



```
38 # config.action_mailer.raise_delivery_errors = true  
39  
40 # Enable threaded mode  
41 config.threadsafe!  
42  
43 # Enable locale fallbacks for I18n (default: true)
```

config/environments/production.rb



3 Ways

1: Immutable

2: Mutex





```
1 require 'thread'
2 semaphore = Mutex.new
3
4 a = Thread.new {
5   semaphore.synchronize {
6     # access shared resource
7   }
8 }
9
10 b = Thread.new {
11   semaphore.synchronize {
12     # access shared resource
13   }
14 }
```

threads/mutex.rb

3: Threadlocal



```
1 a = Thread.new { Thread.current["name"] = "A"; Thread.stop }
2 b = Thread.new { Thread.current[:name] = "B"; Thread.stop }
3 c = Thread.new { Thread.current["name"] = "C"; Thread.stop }
4
5 Thread.list.each {|x| puts "#{x.inspect}: #{x[:name]}"} 
```

threads/threadlocal.rb



Exercise

```
2 require 'thread'  
3  
4 output, threads = [], []  
5 string = "Don't muddle me up!!\n"  
6  
7 100.times do |t|  
8   threads << Thread.start do  
9     100.times do  
10       string.each_char do |c|  
11         output << c  
12       end  
13     end  
14   end  
15 end  
16  
17 threads.each { |t| t.join}  
18 puts output.join.split("\n").reject{|s| s == string.strip}  
19
```

threads/concurrency_exercise.rb

A photograph of a lush, green forest. In the foreground, a stream flows over mossy rocks, its motion blurred by a long exposure. The background is filled with tall trees and dense foliage, with sunlight filtering through the canopy.

Tuning JRuby



AGILITY APPLIED.
SOFTWARE DELIVERED.

A photograph of a lush, green forest. A river or stream flows through the center of the frame, its water appearing smooth and light blue due to a long exposure effect. Large, moss-covered rocks are scattered along the water's path. The background is filled with tall trees and dense foliage, with sunlight filtering through the canopy in bright, dappled rays.

JIT

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.



optimization



AGILITY APPLIED.
SOFTWARE DELIVERED.



--help



AGILITY APPLIED.
SOFTWARE DELIVERED.

- Pass JVM options
 - -J[java options]
- Ruby compatibility
 - --1.9, --1.8
- JVM mode
 - --server, --client

- 
- A photograph of a lush, green forest. A stream flows through the center of the frame, its water appearing smooth and light blue due to a long exposure effect. The banks of the stream are covered in thick green moss. Large, gnarled trees with many branches hang over the scene from the top and sides. The overall atmosphere is mysterious and serene.
- Eval
 - -e “puts ‘hello’”



--properties

- 
- Specify them:
 - `-J-D<prop>=<value>`

- JIT compilation
- `jruby.compile.mode`
- JIT threshold
- `jruby.jit.threshold`
- JIT max methods
- `jruby.jit.max`

- 
- A photograph of a lush, green forest. In the foreground, a stream flows over mossy rocks, its motion blurred by a long exposure. The background is filled with tall trees and thick foliage, creating a sense of depth and mystery. The overall atmosphere is cool and serene.
- Stand up classes
 - jruby.reify.classes

A photograph of a lush, green forest. In the foreground, a stream flows over mossy rocks, its motion blurred by a long exposure. The background is filled with tall trees and thick foliage, creating a sense of depth and mystery. The overall atmosphere is cool and serene.

Java Props



AGILITY APPLIED.
SOFTWARE DELIVERED.

- 
- The background of the slide is a photograph of a lush, green forest. A small stream flows through the center of the frame, its water appearing blurred due to motion, suggesting a slow shutter speed. The forest is filled with tall trees, their branches reaching out over the stream. The lighting is soft and diffused, creating a hazy atmosphere. In the foreground, there are large, moss-covered rocks and some low-lying plants.
- Specify them:
 - `-J<prop>=<value>`
 - `-J-X<prop>=<value>`

- Min heap
- `-Xms<size>` (min)
- Max heap
- `-Xmx<size>` (max)
- Max Permgen
- `-XX:MaxPermSize`



Exercise

optimization/start_rails.sh

optimization/benchmark_rails.rb



Profiling



AGILITY APPLIED.
SOFTWARE DELIVERED.

A photograph of a lush, green forest. A river or stream flows through the center of the frame, its water appearing smooth and light blue due to a long exposure effect. The banks of the stream are covered in large, mossy rocks. The background is filled with tall trees and dense foliage, with sunlight filtering through the canopy in bright, dappled rays.

Native



AGILITY APPLIED.
SOFTWARE DELIVERED.

- 
- Profile
 - --profile
 - --profile.flat
 - --profile.graph

Exercise

`profiling/start_rails.sh`

`profiling/benchmark_rails.rb`

Stickshift

source: <https://github.com/nicksieger/stickshift>

```
1 task :sleep => :snooze do
2   sleep 1
3 end
4 task :snooze do
5   sleep 2
6 end
7 task :default => :sleep
8
9 require 'stickshift'
10 ::Rake::Application.instrument :top_level
11 ::Rake::Application.instrument [], :with_args => 0
12 ::Rake::Task.instrument :invoke, :execute, :inspect_self => true
```



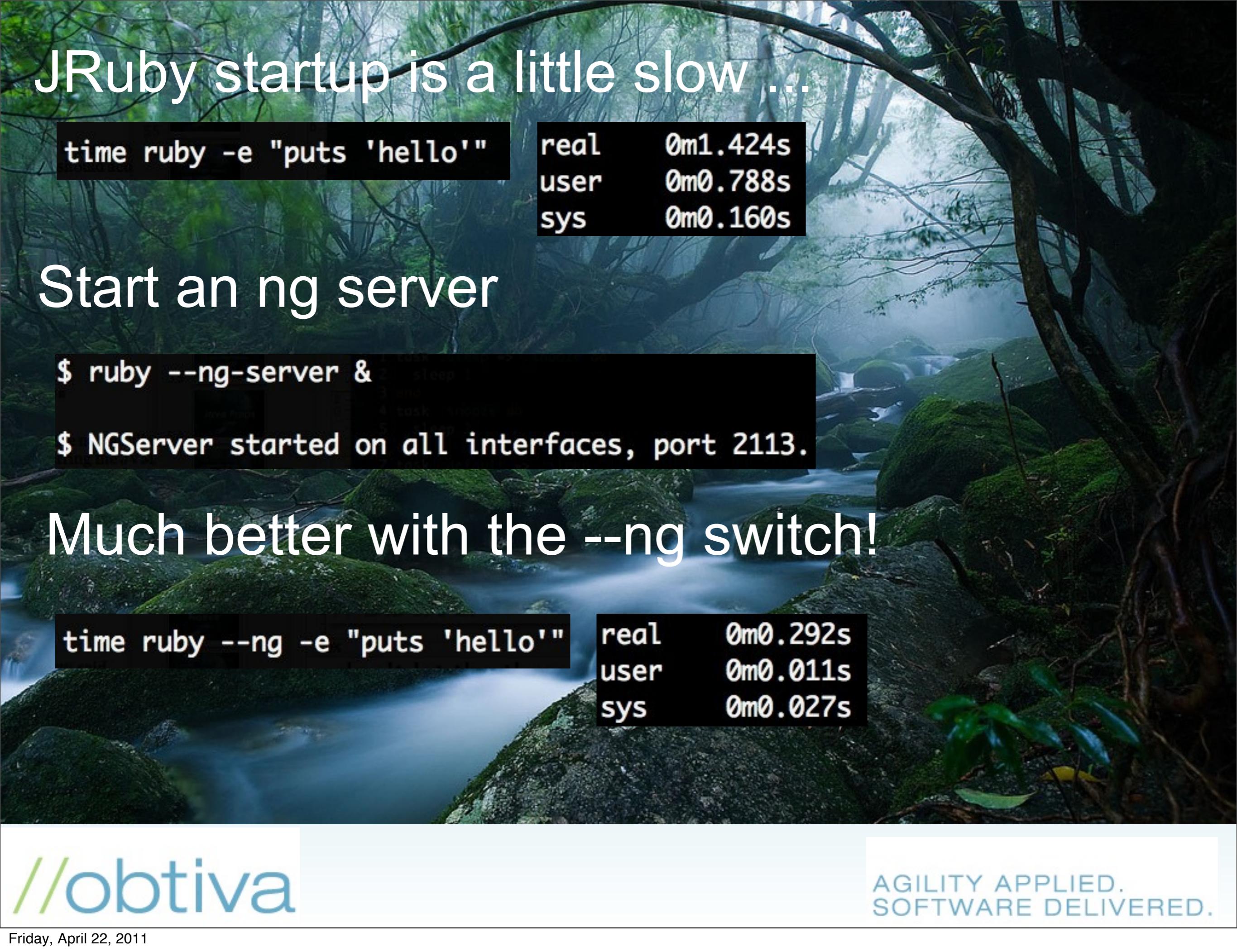
```
14 0ms > Rake::Application#top_level < 3001ms
15 0ms > Rake::Application#[]("default") < 0ms
16 0ms > Rake::Task#invoke{<Rake::Task default => [sleep]>} < 3001ms
17 0ms > Rake::Application#[]("sleep") < 0ms
18 0ms > Rake::Application#[]("snooze") < 0ms
19 2000ms > Rake::Task#execute{<Rake::Task snooze => []>} < 2000ms
20 1000ms > Rake::Task#execute{<Rake::Task sleep => [snooze]>} < 1000ms
21 [] 0ms > Rake::Task#execute{<Rake::Task default => [sleep]>} < 0ms
22
```



Nailgun



AGILITY APPLIED.
SOFTWARE DELIVERED.



JRuby startup is a little slow ...

```
time ruby -e "puts 'hello'"
```

real	0m1.424s
user	0m0.788s
sys	0m0.160s

Start an ng server

```
$ ruby --ng-server &
```

```
$ NGServer started on all interfaces, port 2113.
```

Much better with the --ng switch!

```
time ruby --ng -e "puts 'hello'"
```

real	0m0.292s
user	0m0.011s
sys	0m0.027s



Thank You!



AGILITY APPLIED.
SOFTWARE DELIVERED.