CRYSTAL

ROB

@DATACHOMP

DATACHOMP.COM (DEAD WEBSITE)

DEVELOPMENT OPERATORS

DEVIL OPERAS

DEVOPS

OPS

FORMAT:

STEP 0: BACKGROUNDY META-STUFF

STEP 1: LOOK AT AWESOME STUFF

STEP 2: REALITY

TECH RADARS FOREVER

- I deal with lots of languages
- I deal with postgres
- postgres people I like were excited about crystal

MARKETING

- Have a syntax similar to Ruby (but compatibility with it is not a goal)
- Statically type-checked but without having to specify the type of variables or method arguments.
- Be able to call C code by writing bindings to it in Crystal.
- Have compile-time evaluation and generation of code, to avoid boilerplate code.
- Compile to efficient native code.

NON-MARKETING VERSION

- It is fast and light on memory
  - high level lang, low level perf
- Easy to build off existing C work (jeremy evans mode)
  - First versions of pg driver were build straight off libpq
- Static Types (type inference / type union / etc keeps ruby feel)
- No meta programming (or is there?)
- Compile to efficient native code instead of chilling on an interpreter
- I don’t have to learn new syntax … or a new way of building

BREAK IT DOWN INTO HUMAN

Ghostbusters comes out tomorrow…. won’t have time to learn Rust

No Mans Sky ships soon….won’t have time to learn Go

Skyrim is being ported to VR… won’t have time to learn C

Average lifespan of a human is around 90 years…. won't have time to learn clojure

Just kidding, it is 71 years

THE REAL

- Easy
- Speed
- Deployments / Portable
- Smart Typing ?

KILL YOUR DEPENDENCIES

Super easy in Crystal

nothing exists!

SIDEKIQ

Tried Javascript - nope. Tried golang - nope

Sidekiq for Crystal, 3x faster, 3x smaller!

Psst, just between you and me, I have a very early version of Sidekiq for Crystal executing over 20,000 jobs/sec. MRI, with a similar setup, does 4500 jobs/sec.

Ported to Crystal in 3 days

WHAT ABOUT WEBSITES?

```
require "http/server"

server = HTTP::Server.new(8080) do |context|
  context.response.content_type = "text/plain"
  context.response.print "Hello Ruby World!”
end

puts "Listening on http://0.0.0.0:8080"
server.listen
```

SPEED

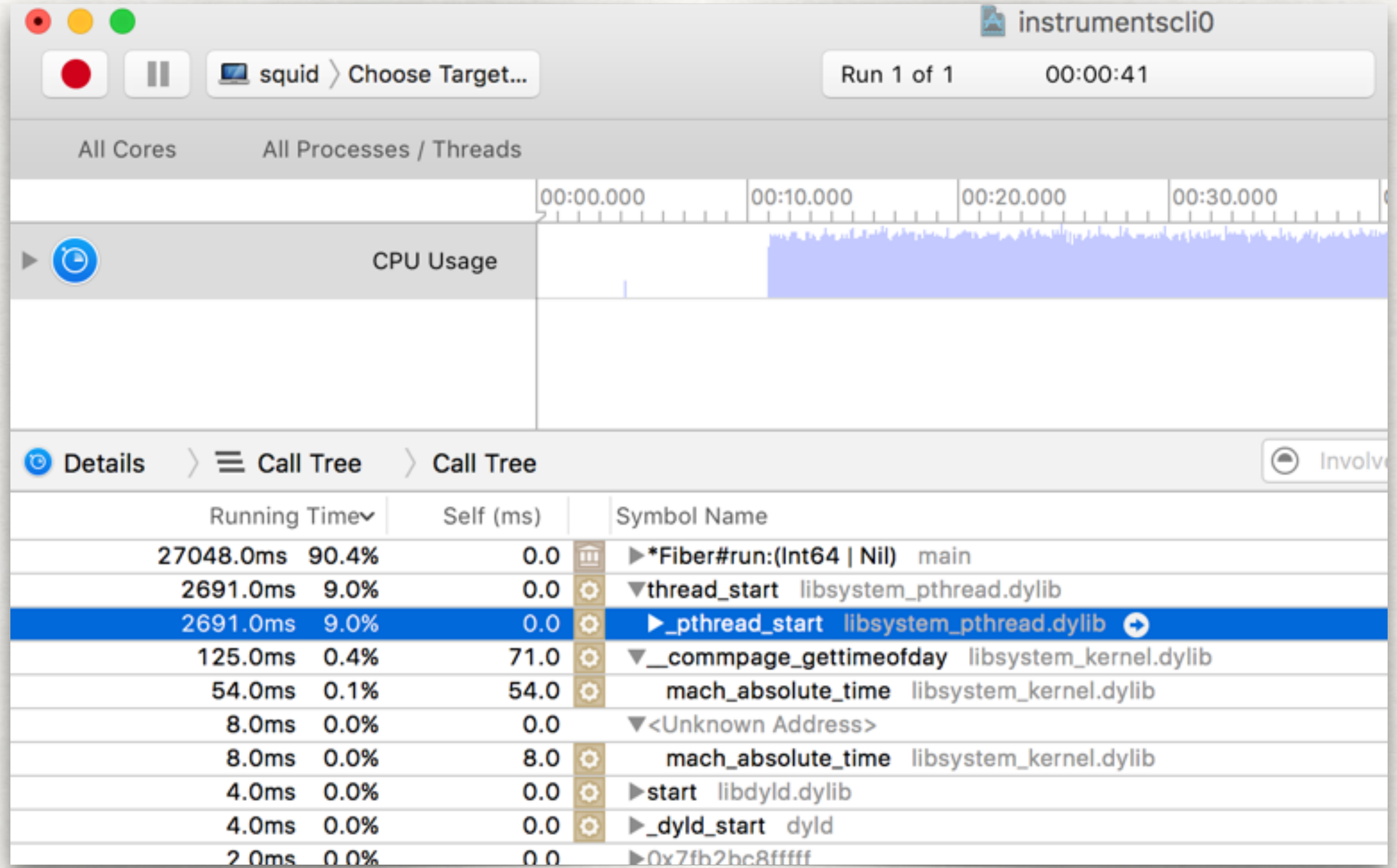
```
# ruby    2.3.1
# sinatra 1.4.7

require 'sinatra'

get "/" do
    puts "Hello Sinatra"
end
```

```
ROB@SQUID ~ :( $WRK -D 1M HTTP://LOCALHOST:4567
RUNNING 1M TEST @ HTTP://LOCALHOST:4567
  2 THREADS AND 10 CONNECTIONS
  THREAD STATS   AVG      STDEV     MAX   +/- STDEV
    LATENCY     3.37MS    3.12MS 170.04MS   96.55%
    REQ/SEC     1.58K   156.99     1.81K    87.83%
  188645 REQUESTS IN 1.00M, 30.94MB READ
REQUESTS/SEC:   3142.60
TRANSFER/SEC:    527.86KB
```

```
# crystal   0.18.7
# kemal     0.14.1

require "kemal"

get "/" do
    puts "Hello Kemal"
end

Kemal.run
```

```
ROB@SQUID ~ :) $WRK -D 1M HTTP://LOCALHOST:3000
RUNNING 1M TEST @ HTTP://LOCALHOST:3000
  2 THREADS AND 10 CONNECTIONS
  THREAD STATS   AVG      STDEV     MAX   +/- STDEV
    LATENCY     1.30MS    3.94MS  70.02MS   95.69%
    REQ/SEC    12.09K     2.29K   15.94K    74.00%
  1444567 REQUESTS IN 1.00M, 148.79MB READ
REQUESTS/SEC:   24055.45
TRANSFER/SEC:      2.48MB
```

INSTRUMENT

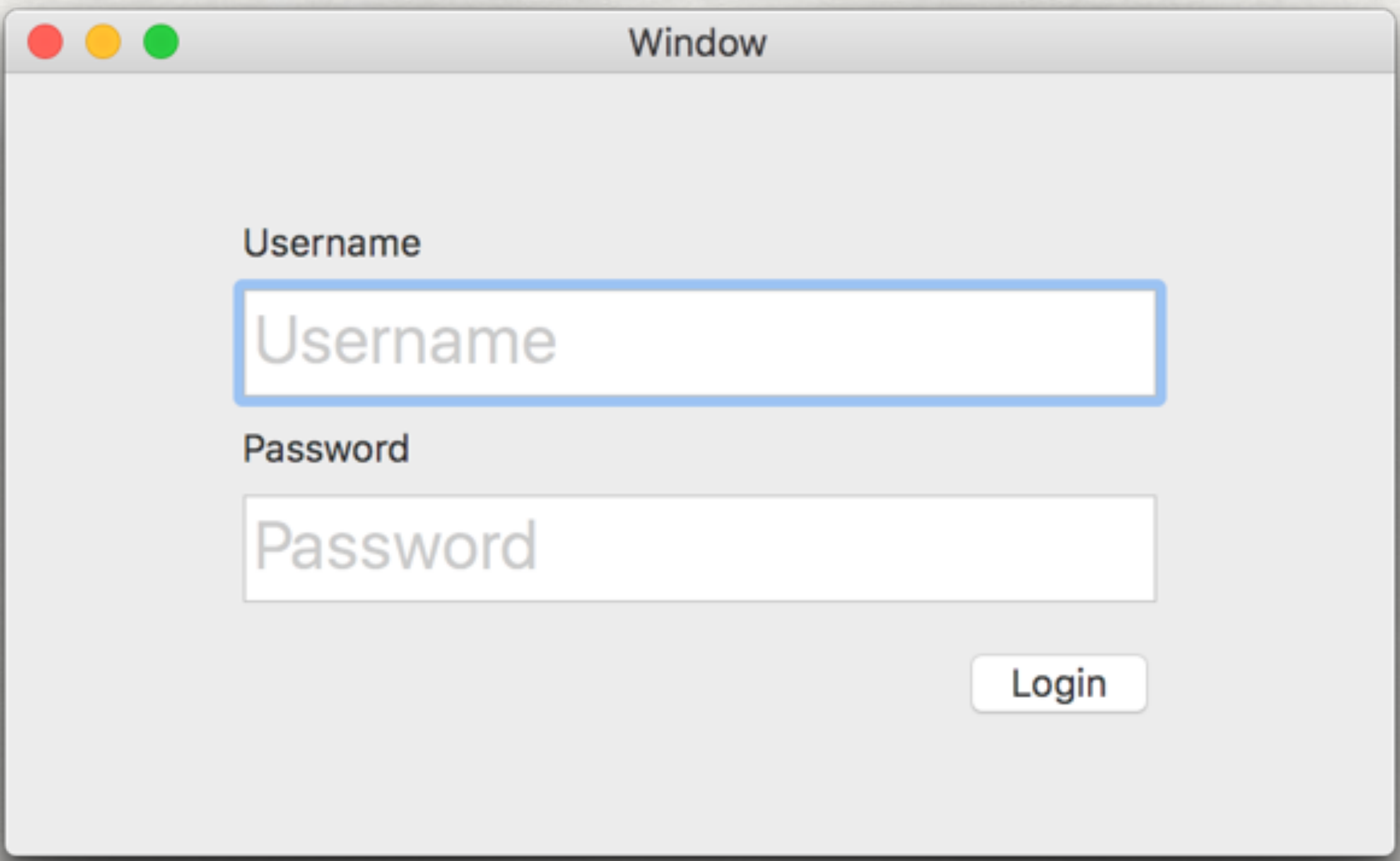


Running Time		Self (ms)	Symbol Name
27048.0ms	90.4%	0.0	▶*Fiber#run:(Int64 | Nil) main
2691.0ms	9.0%	0.0	▼thread_start libsystem_pthread.dylib
2691.0ms	9.0%	0.0	▶_pthread_start libsystem_pthread.dylib
125.0ms	0.4%	71.0	▼__commpage_gettimeofday libsystem_kernel.dylib
54.0ms	0.1%	54.0	mach_absolute_time libsystem_kernel.dylib
8.0ms	0.0%	0.0	▼<Unknown Address>
8.0ms	0.0%	8.0	mach_absolute_time libsystem_kernel.dylib
4.0ms	0.0%	0.0	▶start libdyld.dylib
4.0ms	0.0%	0.0	▶_dyld_start dyld
2.0ms	0.0%	0.0	▶0x7fb2bc8fffff

@DATACHOMP

WHAT ABOUT APP THINGS?

https://github.com/hoopcr/hoop



WHAT ABOUT APP THINGS?

```
require "src/hoop"

include Hoop

NSAutoreleasePool.new
NSApp.activation_policy = LibAppKit::NSApplicationActivationPolicy::Regular
appName = "Hello, World !".to_objc

$window = NSWindow.new(NSRect.new(0, 0, 700, 700).to_objc,
LibAppKit::NSWindowMask::Titled, LibAppKit::NSBackingStoreType::Buffered, false)
$window.set_background_color = NSColor.white_color.to_objc
$window.cascade_top_left_from_point NSPoint.new(20, 20).to_objc
$window.title = appName
$window.make_key_and_order_front nil.to_objc

class WebViewDelegate < NSObject
  export_class
  add_delegate "WKNavigationDelegate"
end
```

IN THE WILD

```
#!/usr/bin/env ruby

require “faker"
require “pg”

conn = PG.connect( dbname: ‘sf_customers_list’ )
conn.exec("insert into customers(title, email, cc_number, cc_expiration_date, cc_type, created_at,
updated_at)

VALUES('#{Faker::Name.name}','#{Faker::Internet.email}','#{Faker::Business.credit_card_number}',
        '#{Faker::Business.credit_card_expiry_date}','#{Faker::Business.credit_card_type}', now(), now())")
```

IN THE WILD

```
require "faker"
require "pg"
require "yaml"
require "http/client"

CONFIG_FILE = File.join(Dir.current, "config.yml")

CONFIG = YAML.parse(File.read(CONFIG_FILE))
PG_URL = CONFIG["db"].as_s
DB     = PG.connect PG_URL

DB.exec("insert into customers(title, email, cc_number, cc_expiration_date, cc_type, created_at,
updated_at)
        VALUES($1::text, $2::text, $3::text, $4::date, $5::text, now(), now())",
 [Faker::Name.name, Faker::Internet.email, Faker::Business.credit_card_number,
Faker::Business.credit_card_expiry_date, Faker::Business.credit_card_type])
```

IN THE WILD

```
#!/usr/bin/env ruby

require 'open-uri'
require 'json'
require 'pg'


geo_source = 'http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_week.geojson'.freeze
data = JSON.load(open(geo_source))
# puts data
conn = PG.connect( dbname: 'funquake' )
if data['metadata']['status'] == 200
  data['features'].each do |quake|

    title = quake['properties']['title'].gsub("'","")
    mags = quake['properties']['mag'] ||= 0.0

     conn.exec("INSERT INTO quakes(usgs_id, title, magnitude, detail_url, location)
       VALUES('#{quake['id']}','#{title}','#{mags}',
       '#{quake['properties']['detail']}', '(#{quake['geometry']['coordinates'][0]}, #{quake['geometry']['coordinates'][1]})' )
         ON CONFLICT (usgs_id) DO NOTHING;")
  end
end
```

IN THE WILD

```
require "http/client"
require "json"
require "pg"
require "yaml"

CONFIG_FILE = File.join(Dir.current, "config.yml")

CONFIG = YAML.parse(File.read(CONFIG_FILE))
PG_URL = CONFIG["db"].as_s
DB     = PG.connect PG_URL

geo_source = "http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_week.geojson"
  response = HTTP::Client.get geo_source
  puts response.status_code
  data = JSON.parse(response.body)

if data["metadata"]["status"] == 200
  data["features"].each do |quake|

    title = quake["properties"]["title"].to_s.gsub("'") { "" }
    mags = quake["properties"]["mag"].to_s.to_f64

     DB.exec("INSERT INTO quakes(usgs_id, title, magnitude, detail_url, location)
       VALUES($1::text, $2::text, $3::numeric, $4::text, POINT($5::numeric, $6::numeric))
       ON CONFLICT (usgs_id) DO NOTHING;",
       [quake["id"], title, mags, quake["properties"]["detail"],
          quake["geometry"]["coordinates"][0], quake["geometry"]["coordinates"][1]])

  end
end
```

THE BAD!!!

Very Very Alpha (expected to launch as soon as other fads die)

DocumentatLOL!!!!!!!

Only works on modern development platforms

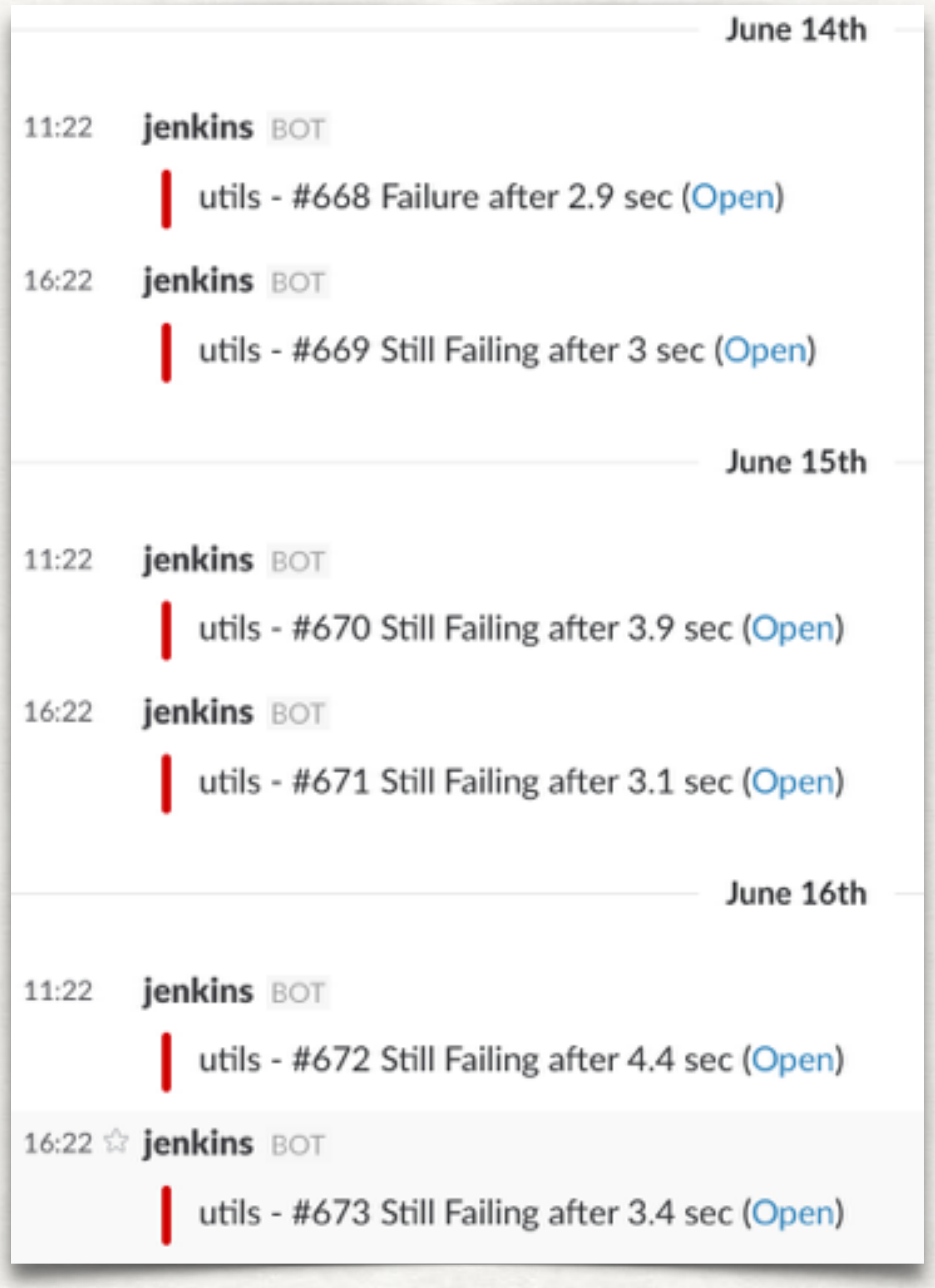
Basic smtp lib? oh, just http

builtin spec lib instead of minitest

THE BAD!!!

Program exited because of
a segmentation fault (11)

Error in macro 'macro_61405376'



June 14th

11:22 jenkins BOT

utils - #668 Failure after 2.9 sec (Open)

16:22 jenkins BOT

utils - #669 Still Failing after 3 sec (Open)

June 15th

11:22 jenkins BOT

utils - #670 Still Failing after 3.9 sec (Open)

16:22 jenkins BOT

utils - #671 Still Failing after 3.1 sec (Open)

June 16th

11:22 jenkins BOT

utils - #672 Still Failing after 4.4 sec (Open)

16:22 jenkins BOT

utils - #673 Still Failing after 3.4 sec (Open)

LINKS

- Crystal for Rubyists - http://www.crystalforrubyists.com/
- Crystal Shards - http://crystalshards.herokuapp.com/
- The Docs - https://crystal-lang.org/docs/index.html
- Playground - https://play.crystal-lang.org/