

HELIX



Stanko Krtalic Rusendic

 github.com/stankec

 [@monorkin](https://twitter.com/monorkin)

 hi@stanko.io

 stanko.io

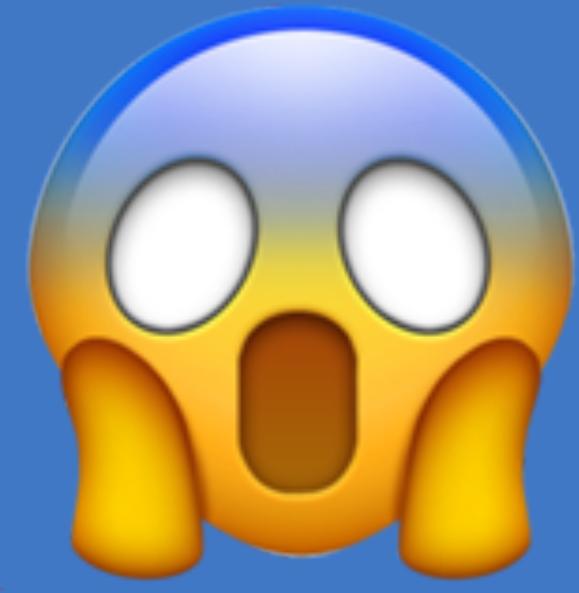


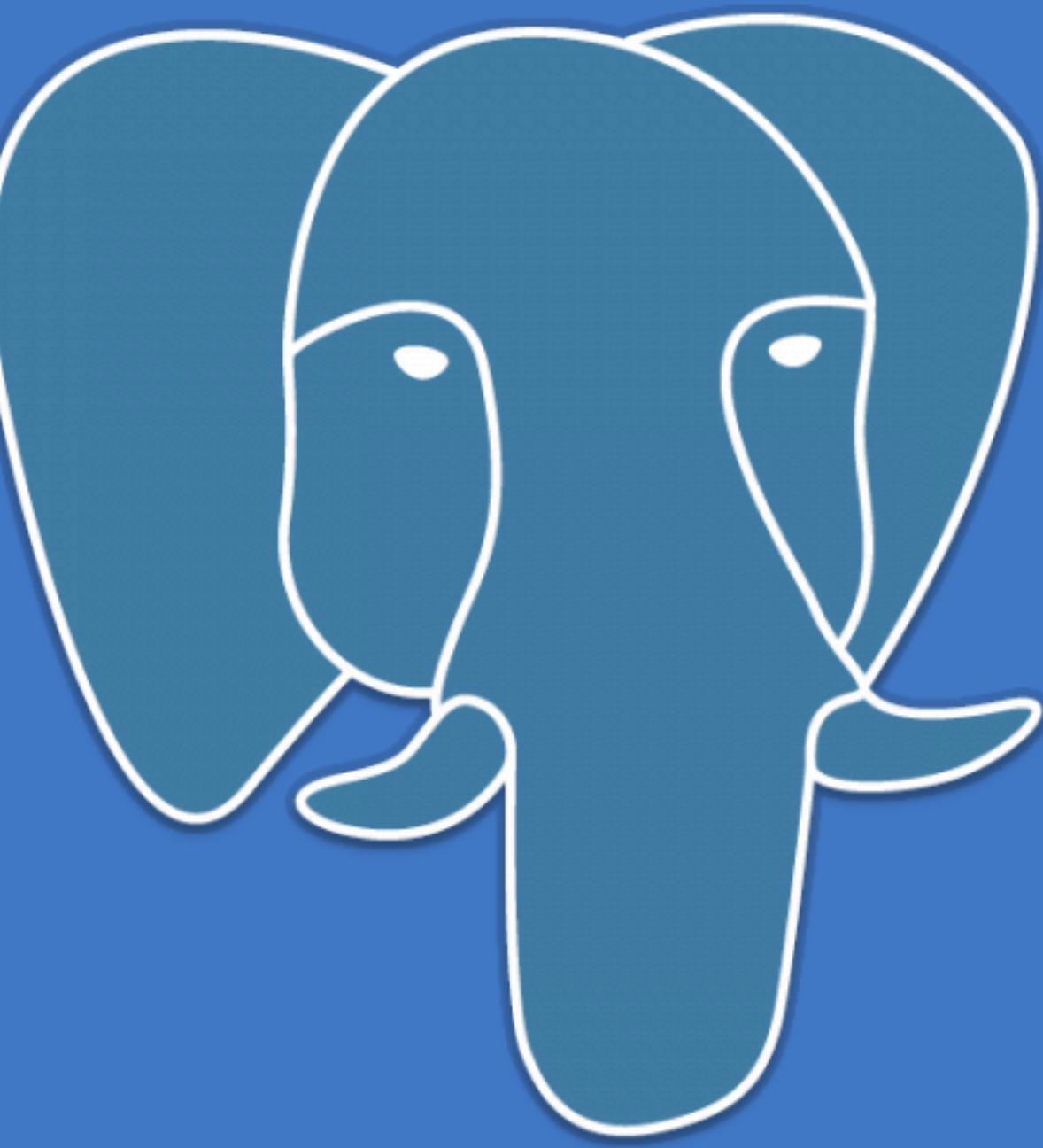
EXTENSIONS





But why?







(That's Nokogiri's logo)



<8-ruby_and_rust_with_helix/
└ demo/01-c_extensions/
 demo.c
 extconf.rb
 hello_world.rb
 run.sh*
 helix.key

```
1 #include "ruby.h"
2
3 VALUE Demo = Qnil;
4 void Init_demo();
5 VALUE method_hello_world(int argc, VALUE *argv, VALUE self);
6
7 void Init_demo() {
8     Demo = rb_define_module("Demo");
9     rb_define_method(Demo, "hello_world", method_hello_world, -1);
10 }
11
12 VALUE method_hello_world(int argc, VALUE *argv, VALUE self) {
13     if (argc == 0) {
14         printf("Hello World!\n");
15         return Qnil;
16     }
17
18     if (argc >= 1) {
19         VALUE name = argv[0];
20         printf("Hello %s\n", StringValueCStr(name));
21         return Qnil;
22     }
23
24     return Qnil;
25 }
```

Stanko ➤ 1 ➤ sumirra ➤ 2 ➤ nvim ➤ ⏴ ⚡ 94% ⏴ 2017/11/26 ⏴ 23:34

```
<and_rust_with_helix/
└ demo/
  └ 02-c_extensions-
    └ demo.c
      extconf.rb
      hello_world.rb
      run.sh*
      helix.key
```

```
1 #include "ruby.h"
2
3 VALUE Demo = Qnil;
4 void Init_demo();
5 VALUE method_halt_and_catch_fire(VALUE self);
6
7 void Init_demo() {
8 | \Demo = rb_define_module("Demo");
9 | \rb_define_method(Demo, "halt_and_catch_fire", method_halt_and_catch_fire, 0);
10 }
11
12 VALUE method_halt_and_catch_fire(VALUE self) {
13   printf("*\u262f\ufe0f\ufe0f\ufe0f*\u262f\ufe0f\n");
14   char *s="hello world";
15   *s='H';
16   return Qnil;
17 }
```

```
1 # frozen_string_literal: true
2
3 require_relative 'demo'
4 include Demo
5
6 begin
7   Demo.halt_and_catch_fire
8 rescue
9   puts 'All is good'
10 ensure
11   puts "The VM didn't crash this time"
12 end
```

```
~
```

The background features a large, semi-transparent watermark of the Rust logo. The logo consists of two overlapping circles: a larger light blue circle on the left and a smaller dark blue circle on the right. The word "RUST" is written in white, sans-serif capital letters across the center of the circles.

RUST

@stanko
hi@stanko.io
Leave feedback on:
<https://goo.gl/EXd>





Why do people use for Rust
when writing FFI?

SAFETY

A program written in Rust
can't crash unexpectedly

Memory safety

(Data) Race condition safe

Performance

Results / Options

Ownership

Borrowing

Lifetimes

```
1 pub enum Option<T> {
2     None,
3     Some(T),
4 }
5
6 pub enum Result<T, E> {
7     Ok(T),
8     Err(E),
9 }
```

```
1 fn main() {
2     let name = match get_name() {
3         Some(name) => name,
4         _ => return println!("ERROR: No name given!")
5     };
6
7     let message = match build_message(name) {
8         Ok(message) => message,
9         Err(error) => return println!("ERROR: {}", error)
10    };
11
12    println!("{} - {}", name, message)
13 }
14
15 fn get_name() → Option<String> {
16 }
17
18 fn build_message() → Result<String, String> {
19 }
```

```
1 fn main() {  
2     let name = match get_name() {  
3         Some(name) => name,  
4         _ => return println!("ERROR: No name given!")  
5     };  
6  
7     let message = match build_message(name) {  
8         Ok(message) => message,  
9         Err(error) => return println!("ERROR: {}", error)  
10    };  
11  
12    println!("{} - {}", name, message);  
13 }  
14  
15 fn get_name() → Option<String> {  
16 }  
17  
18 fn build_message() → Result<String, String> {  
19 }
```

This method can return a string or nothing





Check what the method returned

```
1 fn main() {  
2     let name = match get_name() {  
3         Some(name) => name,  
4         _ => return println!("ERROR: No name given!")  
5     };  
6  
7     let message = match build_message(name) {  
8         Ok(message) => message,  
9         Err(error) => return println!("ERROR: {}", error)  
10    };  
11  
12    println!("{} - {}", name, message)  
13 }  
14  
15 fn get_name() → Option<String> {  
16 }  
17  
18 fn build_message() → Result<String, String> {  
19 }
```

```
1 fn main() {
2     let name = match get_name() {
3         Some(name) => name,
4         _ => return println!("ERROR: No name given!")
5     };
6
7     let message = match build_message(name) {
8         Ok(message) => message,
9         Err(error) => return println!("ERROR: {}", error)
10    };
11
12    println!("{} - {}", name, message)
13 }
14
15 fn get_name() → Option<String> {
16 }
17
18 fn build_message() → Result<String, String> {
19 }
```

If a string was returned, unwrap it

```
1 fn main() {  
2     let name = match get_name() {  
3         Some(name) => name,  
4         _ => return println!("ERROR: No name given!")  
5     };  
6  
7     let message = match build_message(name) {  
8         Ok(message) => message,  
9         Err(error) => return println!("ERROR: {}", error)  
10    };  
11  
12    println!("{} - {}", name, message)  
13 }  
14  
15 fn get_name() → Option<String> {  
16 }  
17  
18 fn build_message() → Result<String, String> {  
19 }
```



If a string wasn't returned, Error

```
1 fn main() {
2     let name = match get_name() {
3         Some(name) => name,
4         _ => return println!("ERROR: No name given!")
5     };
6
7     let message = match build_message(name) {
8         Ok(message) => message,
9         Err(error) => return println!("ERROR: {}", error)
10    };
11
12    println!("{} - {}", name, message)
13 }
14
15 fn get_name() → Option<String> {
16 }
17
18 fn build_message() → Result<String, String> {
19 }
```

```
1 fn main() {  
2     let name = match get_name() {  
3         Some(name) => name,  
4         _ => return println!("ERROR: No name given!")  
5     };  
6  
7     let message = match build_message(name) {  
8         Ok(message) => message,  
9         Err(error) => return println!("ERROR: {}", error)  
10    };  
11  
12    println!("{} - {}", name, message)  
13 }  
14  
15 fn get_name() -> Option<String> {  
16 }  
17  
18 fn build_message() -> Result<String, String> {  
19 }
```

Results have two types



```
1 fn main() {  
2     let name = match get_name() {  
3         Some(name) => name,  
4         _ => return println!("ERROR: No name given!")  
5     };  
6  
7     let message = match build_message(name) {  
8         Ok(message) => message,  
9         Err(error) => return println!("ERROR: {}", error)  
10    };  
11  
12    println!("{} - {}", name, message)  
13 }  
14  
15 fn get_name() -> Option<String> {  
16 }  
17  
18 fn build_message() -> Result<String, String> {  
19 }
```



Checks if the result was Ok or an Error

```
1 fn main() {
2     let name = match get_name() {
3         Some(name) => name,
4         _ => return println!("ERROR: No name given!")
5     };
6
7     let message = match build_message(name) {
8         Ok(message) => message,
9         Err(error) => return println!("ERROR: {}", error)
10    };
11
12    println!("{} - {}", name, message)
13 }
14
15 fn get_name() → Option<String> {
16 }
17
18 fn build_message() → Result<String, String> {
19 }
```



HELIX

```
cargo new demo
```

```
demo_  
├── Cargo.lock  
├── Cargo.toml  
└── src  
    └── lib.rs
```

1 directory, 3 files

```
1 [package]
2 name = "demo"
3 version = "0.1.0"
4 authors = ["Stanko Krtalić <stanko.krtalic@gmail.com>"]
5
6 [lib]
7 crate-type = ["cdylib"]
8
9 [dependencies]
10 helix = "0.7.2"
```



```
1 [package]
2 name = "demo"
3 version = "0.1.0"
4 authors = ["Stanko Krtalić <stanko.krtalic@gmail.com>"]
5
6 [lib]
7 crate-type = ["cdylib"]
8
9 [dependencies]
10 helix = "0.7.2"
```



```
1 #[macro_use]
2 extern crate helix;
3
4 ruby! {
5     class Demo {
6         def hello_world(name: Option<String>) {
7             let name = match name {
8                 Some(string) => string,
9                 _ => "World".to_string()
10            };
11
12            println!("Hello {}", name);
13        }
14    }
15 }
```

```
1 #[macro_use]
2 extern crate helix;
3
4 ruby! {
5     class Demo {
6         def hello_world(name: Option<String>) {
7             let name = match name {
8                 Some(string) => string,
9                 _ => "World".to_string()
10            };
11
12            println!("Hello {}", name);
13        }
14    }
15 }
```



```
1 #[macro_use]
2 extern crate helix;
3
4 ruby! {
5     class Demo {
6         def hello_world(name: Option<String>) {
7             let name = match name {
8                 Some(string) => string,
9                 _ => "World".to_string()
10            };
11
12            println!("Hello {}", name);
13        }
14    }
15 }
```

```
1 #[macro_use]
2 extern crate helix;
3
4 ruby! {
5     class Demo {
6         def hello_world(name: Option<String>) {
7             let name = match name {
8                 Some(string) => string,
9                 _ => "World".to_string()
10            };
11
12            println!("Hello {}", name);
13        }
14    }
15 }
```

```
1 #[macro_use]
2 extern crate helix;
3
4 ruby! {
5     class Demo {
6         def hello_world(name: Option<String>) {
7             let name = match name {
8                 Some(string) => string,
9                 _ => "World".to_string()
10            };
11
12            println!("Hello {}", name);
13        }
14    }
15 }
```

```
1 #[macro_use]
2 extern crate helix;
3
4 ruby! {
5     class Demo {
6         def hello_world(name: Option<String>) {
7             let name = match name {
8                 Some(string) => string,
9                 _ => "World".to_string()
10            };
11            println!("Hello {}", name);
12        }
13    }
14 }
15 }
```



```
1 #[macro_use]
2 extern crate helix;
3
4 ruby! {
5     class Demo {
6         def hello_world(name: Option<String>) {
7             let name = match name {
8                 Some(string) => string,
9                 _ => "World".to_string()
10            };
11
12            println!("Hello {}", name);
13        }
14    }
15 }
```



```
demo_  
└── Cargo.lock  
└── Cargo.toml  
└── Gemfile  
└── demo.gemspec  
└── src  
    └── lib.rs
```

1 directory, 5 files

```
demo_  
└── Cargo.lock  
Cargo.toml  
Gemfile  
demo.gemspec  
src  
└── lib.rs
```

1 directory, 5 files

```
1 Gem::Specification.new do |s|
2   s.name = 'demo'
3   s.version = '1.0.0'
4   s.authors = ['Stanko Krtalic Rusendic <stanko.krtalic@gmail.com>']
5   s.summary = "A Helix demo project"
6   s.files = Dir['{lib/**/*,[A-Z]*}']
7
8   s.platform = Gem::Platform::RUBY
9   s.require_path = 'lib'
10
11  s.add_dependency 'helix_runtime', '~> 0.7.2'
12 end
```

```
1 Gem::Specification.new do |s|
2   s.name = 'demo'
3   s.version = '1.0.0'
4   s.authors = ['Stanko Krtalic Rusendic <stanko.krtalic@gmail.com>']
5   s.summary = "A Helix demo project"
6   s.files = Dir['{lib/**/*,[A-Z]*}']
7
8   s.platform = Gem::Platform::RUBY
9   s.require_path = 'lib'
10
11  s.add_dependency 'helix_runtime', '~> 0.7.2'
12 end
```

```
1 Gem::Specification.new do |s|
2   s.name = 'demo'
3   s.version = '1.0.0'
4   s.authors = ['Stanko Krtalic Rusendic <stanko.krtalic@gmail.com>']
5   s.summary = "A Helix demo project"
6   s.files = Dir['{lib/**/*,[A-Z]*}']
7
8   s.platform = Gem::Platform::RUBY
9   s.require_path = 'lib'
10
11  s.add_dependency 'helix_runtime', '~> 0.7.2'
12 end
```



rake build

```
3 # Optional and KV arguments are not yet supported
4 Demo.hello_world(nil)
5 Demo.hello_world('Ruby Zagreb!')
6
7 begin
8   Demo.hello_world(Object.new)
9 rescue TypeError
10  puts 'Rescued from TypeError'
11 end
```

```
" Press ? for help
.. (up a dir)
<d_rust_with_helix/
└ [x]demo/
  └ 01-c_extension>
  └ 02-c_extension>
  └ 03-rust_exampl>
└ [x]04-helix/
  └ [x]crates/de>
    └ lib/
      └ src/
        lib.rs
  └ [x]target/
    Cargo.lock
    Cargo.toml
    demo.gemsp>
    Gemfile
    Gemfile.lo>
    Rakefile
    tags
```

```
1 #[macro_use]
2 extern crate helix;
3
4 ruby! {
5     class Demo {
6         def hello_world(name: Option<String>) {
7             let name = match name {
8                 Some(string) => string,
9                 _ => "World".to_string()
10            };
11
12            println!("Hello {}", name);
13        }
14    }
15 }
```

```
rails generate helix:crate demo
```



WHEN TO USE IT



WORDCOUNT

WordCount uses the [word_count](#) crate, which defines a very simple class that implements a single class method: `search`. This method takes a path to a file and a search string as input, performing a count of search string in the source file.

Select a Dataset:

Shakespeare
 Upload: [Browse...](#) No file selected.

Search Text:

[Count](#)

Source

- Crate: [Class / Tests](#)
- Controller
- View

©2017 [Tilde, Inc.](#)





INLINE CSS

InlineCSS uses the `inline_css` crate, which uses Servo's `cssparser` and `html5ever` crates to inline a CSS file into an HTML file.

HTML

```
<html id="facebook" class="" lang="en"><head><meta charset="utf-8"><meta name="referrer" content="origin-when-crossorigin" id="meta_referrer"><script>window._cstart=new Date();</script><script>function envFlush(a){function b(c){for(var d in a)c[d]=a[d]}if(window.requireLazy)window.requireLazy(["Env"],b);else{window.Env=window.Env||{};b(window.Env)}}envFlush({"ajaxpipe_token":"AXjzeWeAHBiMjj1k","timeslice_heartbeat_config":{"pollIntervalMs":33,"idleGapThresholdMs":60,"ignoredTimesliceNames":{"requestAnimationFrame":true,"Event listenHandler mousemove":true,"Event listenHandler mouseover":true,"Event listenHandler mouseout":true,"Event listenHandler scroll":true,"enableOnRequire":false,"shouldLogCounters":true,"+timeslice_categories":[]}});</script></head><body></body></html>
```

css

```
.fbEmuTracking{position: absolute; visibility: hidden}
.sp_fWm8gQH0FqP_2x.sx_0c5538{background-position: 0
-376px}.sp_fWm8gQH0FqP_2x.sx_41c7c8{background-position: -17px
-376px}.sp_fWm8gQH0FqP_2x.sx_358d9e{background-position: -34px
-376px}.sp_fWm8gQH0FqP_2x.sx_83afbe{background-position: 0
-393px}.sp_fWm8gQH0FqP_2x.sx_4dc3ee{background-position: -17px
-393px}.sp_fWm8gQH0FqP_2x.sx_85fa58{width: 20px; height: 20px; background-position: 0
-321px}.sp_fWm8gQH0FqP_2x.sx_44115a{background-position: -34px
-393px}.sp_fWm8gQH0FqP_2x.sx_80a83f{width: 20px; height: 20px; background-position: -21px
-321px}.sp_fWm8gQH0FqP_2x.sx_f2d04e{background-position: 0
```

No KV or optional arguments

Can't write Modules

No cross-compilation

Non-standard compiler*

Safety and speed

CANONICAL



Wire



SENTRY

redhat.

mozilla

coursera

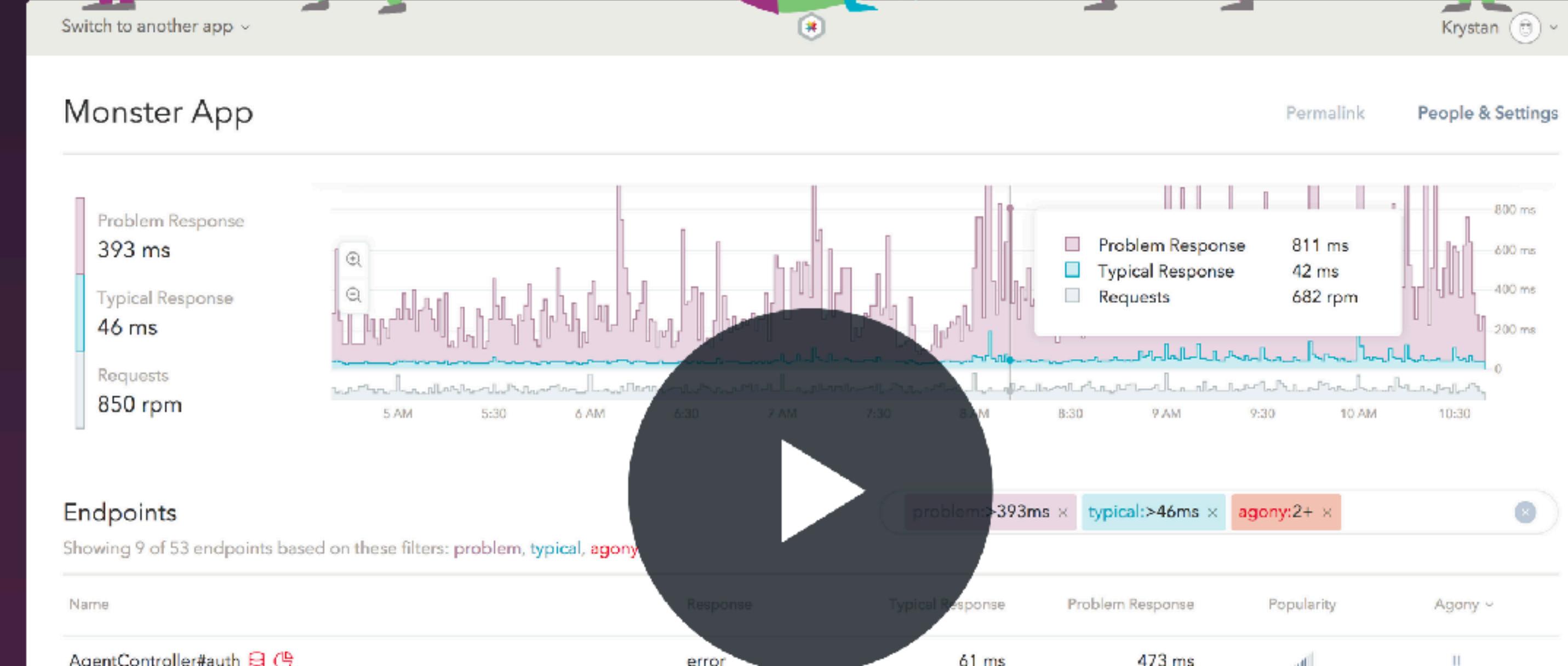
Braintree

DO THE ANSWER DANCE.

Don't struggle to learn why your app is slow.

Get answers with Skylight.

[START YOUR FREE MONTH](#)



The screenshot shows the Skylight application interface. At the top, there's a navigation bar with links for Features, Pricing, and Support, followed by the Skylight logo and Log In/Start Your Free Month buttons. The main heading "DO THE ANSWER DANCE." is displayed prominently in large white letters against a dark background with star-like sparkles.

Below the heading, two text blocks encourage users to avoid struggling with app performance and instead get answers with Skylight, accompanied by a "START YOUR FREE MONTH" call-to-action button.

The central part of the interface features a "Monster App" dashboard. It includes a "Problem Response" section showing 393 ms, a "Typical Response" section showing 46 ms, and a "Requests" section showing 850 rpm. A large, semi-transparent circular overlay with a play button icon is positioned over the dashboard area.

The dashboard also displays a timeline chart showing request activity from 5 AM to 10:30 AM, with three colored bars representing Problem Response (pink), Typical Response (blue), and Requests (light blue). Below the chart, a search bar contains filters: problem:393ms, typical:>46ms, and agony:2+.

At the bottom, there's a table titled "Endpoints" showing 9 of 53 endpoints based on the specified filters. The table has columns for Name, Response, Typical Response, Problem Response, Popularity, and Agony.

QUESTIONS



github.com/stankec



[@monorkin](https://twitter.com/@monorkin)



hi@stanko.io



EXTRAS

```
1
2 fn sign_in(username: String, password: String) → Result<User, String> {
3     if username == "user".to_string() && password == "password".to_string() {
4         return Ok(
5             User::new(username, password)
6         );
7     }
8     return Err("No such user found");
9 }
10 }
```

Username and password are deallocated

```
1 fn main {
2     let name = "petar".to_string();
3     let amne = shuffle(name); // ← The function shuffle became the owner of
4                                //           the name variable
5
6     println!("Original: {}", name); // ← Errors during compilation
7                                //   We tried to use a variable we don't own
8     println!("Shuffled: {}", amne);
9 }
```