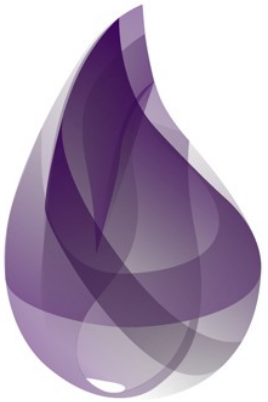
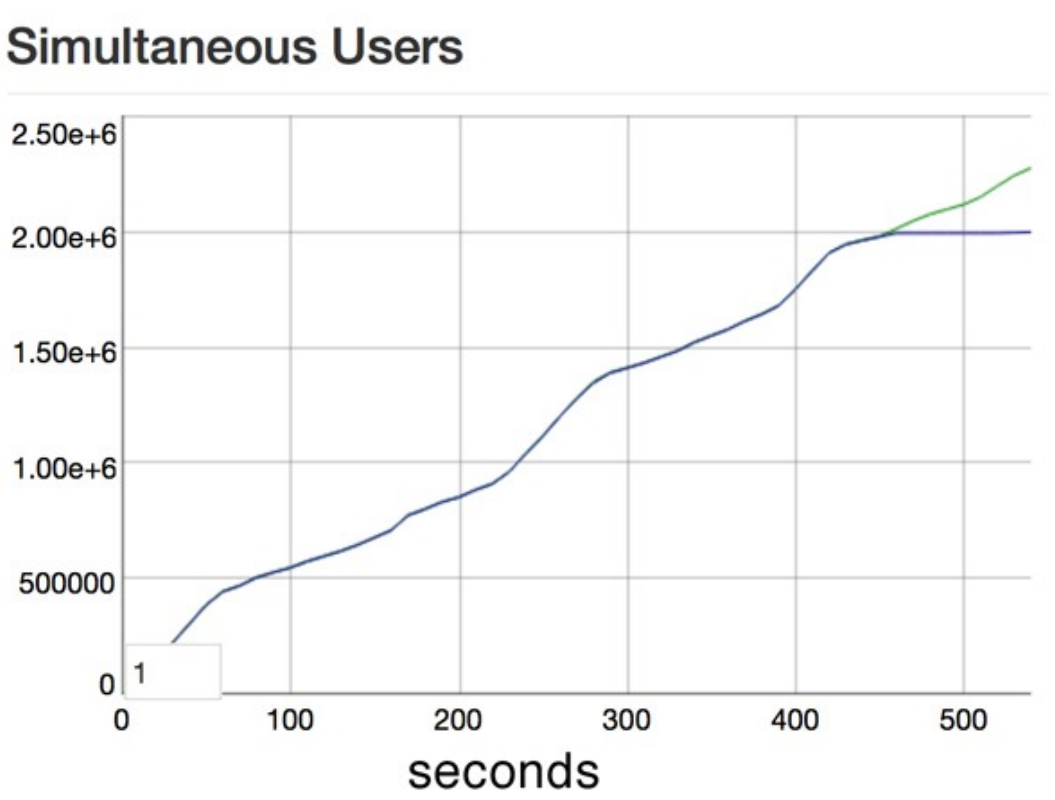
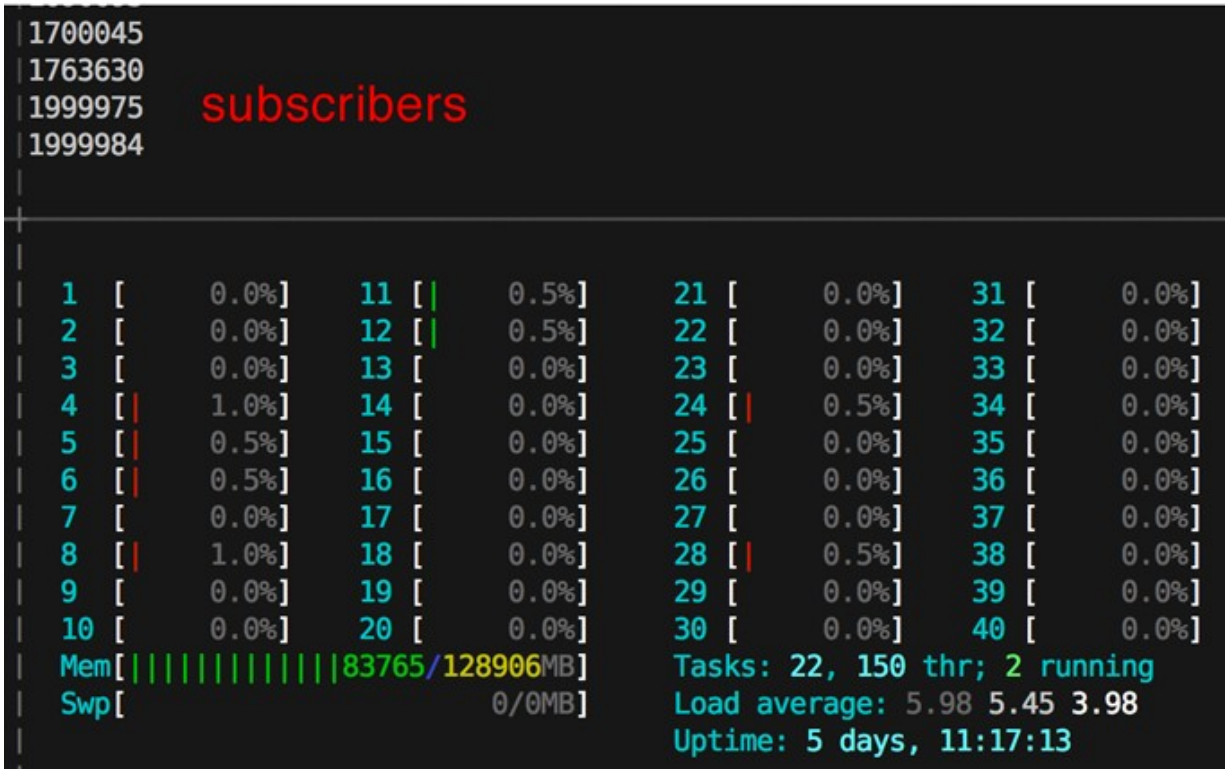


elixir


Simultaneous Users
2.50e+6
2.00e+6
1.50e+6
1.00e+6
500000
0
1
0
100
200
300
400
500
seconds


1700045
1763630
1999975
1999984
subscribers
1 [0.0%] 11 [| 0.5%] 21 [0.0%] 31 [0.0%]
2 [0.0%] 12 [| 0.5%] 22 [0.0%] 32 [0.0%]
3 [0.0%] 13 [0.0%] 23 [0.0%] 33 [0.0%]
4 [| 1.0%] 14 [0.0%] 24 [| 0.5%] 34 [0.0%]
5 [| 0.5%] 15 [0.0%] 25 [0.0%] 35 [0.0%]
6 [| 0.5%] 16 [0.0%] 26 [0.0%] 36 [0.0%]
7 [0.0%] 17 [0.0%] 27 [0.0%] 37 [0.0%]
8 [| 1.0%] 18 [0.0%] 28 [| 0.5%] 38 [0.0%]
9 [0.0%] 19 [0.0%] 29 [0.0%] 39 [0.0%]
10 [0.0%] 20 [0.0%] 30 [0.0%] 40 [0.0%]
Mem[||||||||||||83765/128906MB]
Swp[0/0MB]
Tasks: 22, 150 thr; 2 running
Load average: 5.98 5.45 3.98
Uptime: 5 days, 11:17:13

Elixir and Phoenix

fast, concurrent and explicit

Tobias Pfeiffer

@PragTob

pragtob.info

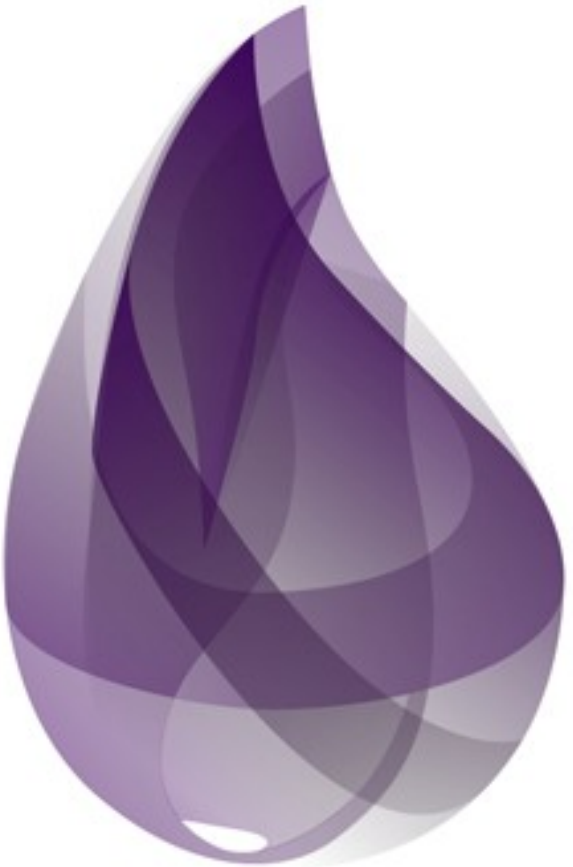
Elixir and Phoenix

fast, concurrent and explicit

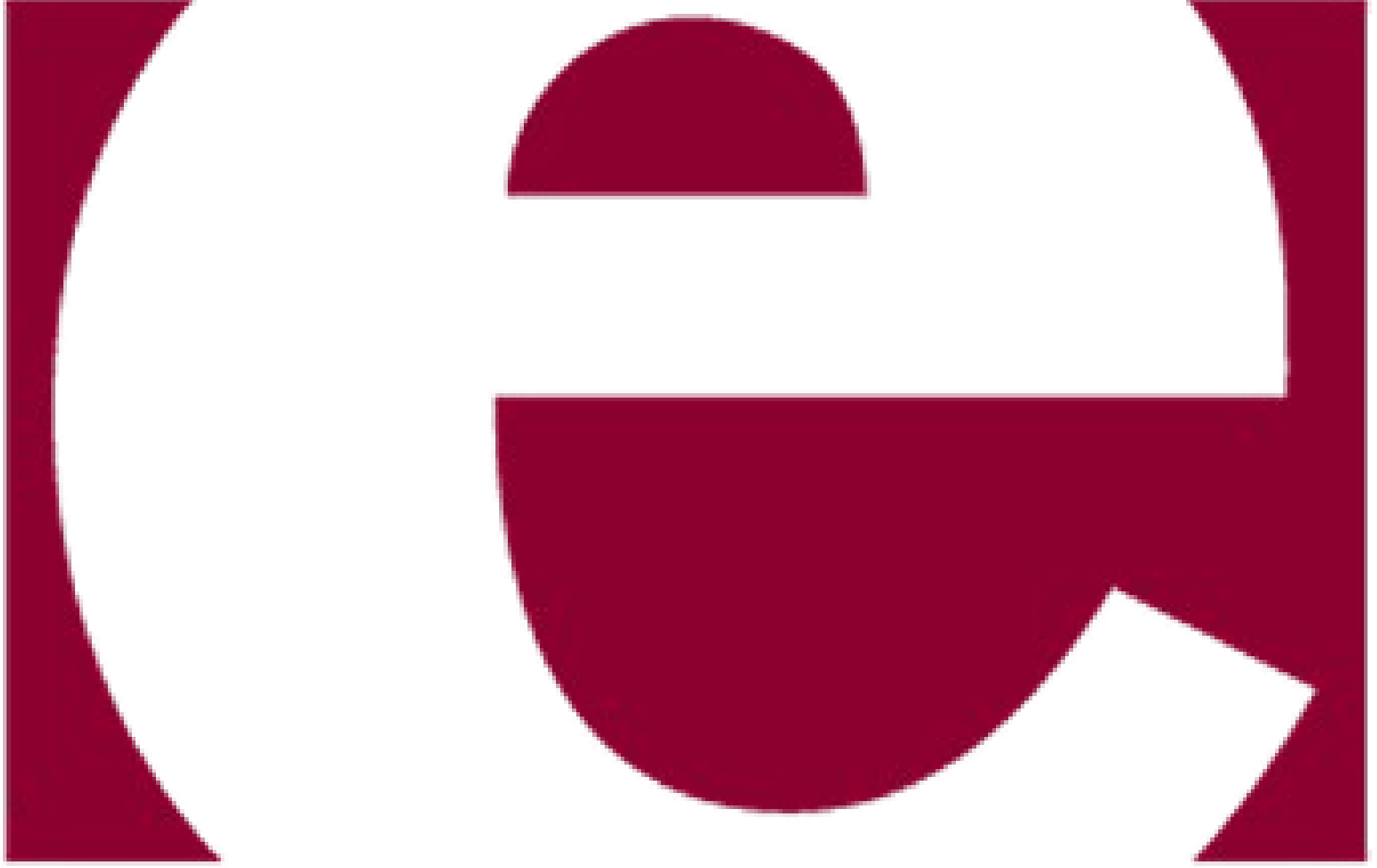
Tobias Pfeiffer

@PragTob

pragtob.info



elixir



ERLANG

José Valim
@josevalim

Folgen

Reminder it is 2016. Almost everything you do must be using all CPUs: compiling code, booting, running tests... Easy math on the wins here.

RETWEETS 56 GEFÄLLT 67

07:03 - 29. Feb. 2016

```
defmodule MyMap do
  def map([], _func), do: []

  def map([head | tail], func) do
    [func.(head) | map(tail, func)]
  end
end

MyMap.map [1, 2, 3, 4], fn(i) -> i * i end
```

Ruby-like Syntax

```
defmodule MyMap do
  def map([], _func), do: []

  def map([head | tail], func) do
    [func.(head) | map(tail, func)]
  end
end

MyMap.map [1, 2, 3, 4], fn(i) -> i * i end
```

Pattern Matching

```
defmodule MyMap do
  def map([], _func), do: []

  def map([head | tail], func) do
    [func.(head) | map(tail, func)]
  end
end

MyMap.map [1, 2, 3, 4], fn(i) -> i * i end
```

Pattern Matching

```
defmodule Patterns do
  def greet(%{name: name, age: age}) do
    IO.puts "Hi there #{name}, what's up at #{age}?"
  end

  def greet(%{name: name}) do
    IO.puts "Hi there #{name}"
  end

  def greet(_) do
    IO.puts "Hi"
  end
end

Patterns.greet %{name: "Tobi", age: 26}
Patterns.greet %{name: "Tobi"}
Patterns.greet ["Mop"]
```

Pipe

```
people = DB.find_customers
orders = Orders.for_customers(people)
tax    = sales_tax(orders, 2013)
filing = prepare_filing(tax)
```

Pipe

```
filing = DB.find_customers
            |> Orders.for_customers
            |> sales_tax(2013)
            |> prepare_filing
```

Pipe

```
filing =
prepare_filing(sales_tax(Orders.for_cusstomers(DB.find_c
ustomers), 2013))
```

Pipe

```
filing = DB.find_customers
            |> Orders.for_customers
            |> sales_tax(2013)
            |> prepare_filing
```

Optional Type Annotations

```
@spec all?(t) :: boolean
@spec all?(t, (element -> as_boolean(term))) :: boolean

def all?(enumerable, fun \\ fn(x) -> x end)

def all?(enumerable, fun) when is_list(enumerable) do
  do_all?(enumerable, fun)
end
```

Meta Programming

```
defmacro plug(plug, opts \\ []) do
  quote do
    @plugs {unquote(plug), unquote(opts), true}
  end
end
```

Functional Programming?


turbine
generator
coal
EXIT


EXIT


turbine
generator
coal
EXIT

Where to call functions

```
2.2.2 :001 > [1, 2, 3, 4].map { |i| i + 1 }
 => [2, 3, 4, 5]
```

VS

```
iex(2)> Enum.map [1, 2, 3, 4], fn(i) -> i + 1 end
[2, 3, 4, 5]
```

Transformation of Data

Minimize state

vs

Hiding State

Same Input,

Same Output

Testing++

Immutable Data

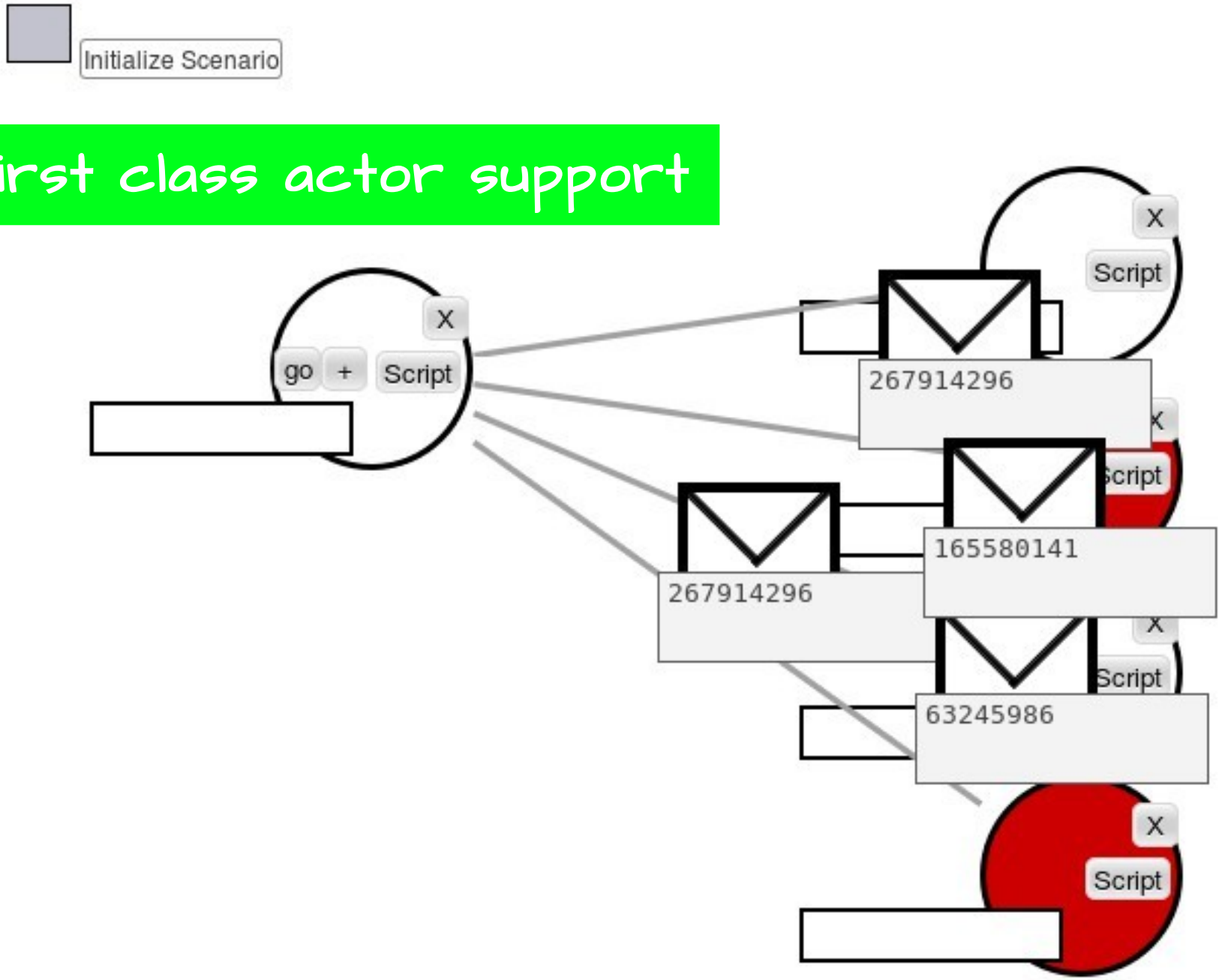
```
variable = 10
do_something(variable)
insert_in_db(variable)
```

Immutable Data

```
person = Person.new(attributes)
do_something(person)
insert_in_db(person)
```

Immutable Data

```
person = Person.new(attributes)
person = do_something(person)
insert_in_db(person)
```


Initialize Scenario
First class actor support
X
go
+
Script
X
Script
267914296
X
Script
165580141
267914296
X
Script
63245986
X
Script


TM

Flow

```
connection
|> endpoint
|> router
|> pipelines
|> controller
|> model
|> view
```

Controller

```
def new(conn, _params) do
  changeset = User.new_changeset(%User{})
  render conn, "new.html", changeset: changeset
end
```

Model

```
defmodule Rumbl.User do
  use Rumbl.Web, :model

  schema "users" do
    field :name,          :string
    field :username,      :string
    field :password,      :string, virtual: true
    field :password_hash, :string
    has_many              :videos, Rumbl.Video

    timestamps
  end

  # ...

end
```

View

```
defmodule Rumbl.UserView do
  use Rumbl.Web, :view
  alias Rumbl.User

  def first_name(%{name: name}) do
    name
    |> String.split(" ")
    |> Enum.at(0)
  end
end
```

Template

```
<%= form_for @changeset, user_path(@conn, :create), fn
form -> %>
  <div class="form-group">
    <%= text_input form, :name, placeholder: "Name",
class: "form-control" %>
    <%= error_tag form, :name %>
  </div>
  <div class="form-group">
    <%= text_input form, :username, placeholder:
"Username", class: "form-control" %>
    <%= error_tag form, :username %>
  </div>
  <div class="form-group">
    <%= password_input form, :password, placeholder:
"Password", class: "form-control" %>
    <%= error_tag form, :password %>
  </div>
  <%= submit "Create User", class: "btn btn-primary" %>
<% end %>
```


RAILS


TM

Changesets

```
def new_changeset(model, params \\ :empty) do
  model
  |> cast(params, ~w(name username), [])
  |> unique_constraint(:username)
  |> validate_length(:username, min: 1, max: 20)
end

def registration_changeset(model, params) do
  model
  |> new_changeset(params)
  |> cast(params, ~w(password), [])
  |> validate_length(:password, min: 6, max: 100)
  |> put_pass_hash()
end
```

Changesets

```
def create(conn, %{"user" => user_params}) do
  changeset = User.registration_changeset(%User{},
user_params)
  case Repo.insert changeset do
    {:ok, user} ->
      conn
      |> Rumbl.Auth.login(user)
      |> put_flash(:info, "You successfully registered
as #{user.name}!")
      |> redirect(to: user_path(conn, :index))
    {:error, changeset}->
      render conn, "new.html", changeset: changeset
  end
end
```

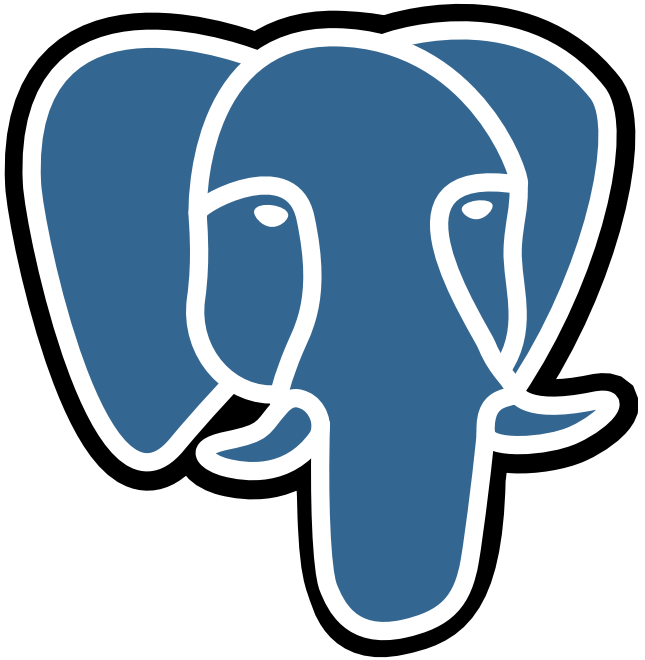
Channels

```
defmodule Rumbl.VideoChannel do
  use Rumbl.Web, :channel

  def join("videos:" <> video_id, _params, socket) do
    {:ok, socket}
  end

  def handle_in("new_annotation", params, socket) do
    broadcast! socket, "new_annotation", %{
      user: %{username: "anon"},
      body: params["body"],
      at: params["at"]
    }

    {:reply, :ok, socket}
  end
end
```




Brunch

```
iex(13)> user = Repo.get_by(User, name: "Homer")
iex(14)> user.videos
#Ecto.Association.NotLoaded<association :videos is not
loaded>
iex(15)> Repo.preload(user, :videos)
iex(16)> user.videos
#Ecto.Association.NotLoaded<association :videos is not
loaded>
iex(17)> user = Repo.preload(user, :videos)
iex(18)> user.videos
[%Rumbl.Video{__meta__: #Ecto.Schema.Metadata<:loaded>,
  category: #Ecto.Association.NotLoaded<association
:category is not loaded>,
  category_id: nil, description: "such great many wow", id:
3,
  inserted_at: #Ecto.DateTime<2016-02-28T18:42:41Z>, title:
"Hubidubiee",
  updated_at: #Ecto.DateTime<2016-02-28T18:42:41Z>, url:
"www.lol.com",
  user: #Ecto.Association.NotLoaded<association :user is
not loaded>,
  user_id: 5}]
```

Explicit preloading

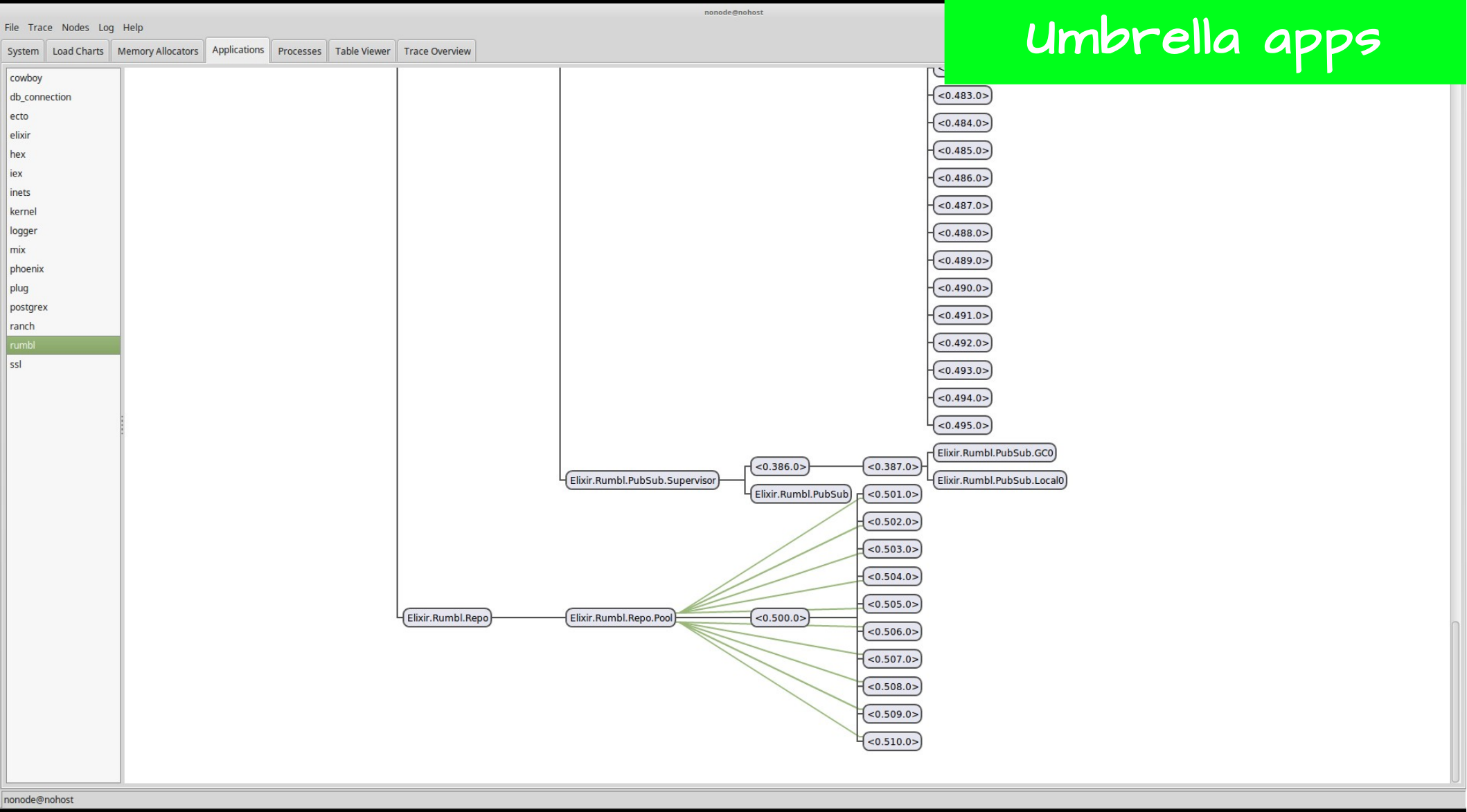
```
iex(13)> user = Repo.get_by(User, name: "Homer")
iex(14)> user.videos
#Ecto.Association.NotLoaded<association :videos is not
loaded>
iex(15)> Repo.preload(user, :videos)
iex(16)> user.videos
#Ecto.Association.NotLoaded<association :videos is not
loaded>
iex(17)> user = Repo.preload(user, :videos)
iex(18)> user.videos
[%Rumbl.Video{__meta__: #Ecto.Schema.Metadata<:loaded>,
  category: #Ecto.Association.NotLoaded<association
:category is not loaded>,
  category_id: nil, description: "such great many wow", id:
3,
  inserted_at: #Ecto.DateTime<2016-02-28T18:42:41Z>, title:
"Hubidubiee",
  updated_at: #Ecto.DateTime<2016-02-28T18:42:41Z>, url:
"www.lol.com",
  user: #Ecto.Association.NotLoaded<association :user is
not loaded>,
  user_id: 5}]
```

Explicit preloading

```
iex(13)> user = Repo.get_by(User, name: "Homer")
iex(14)> user.videos
#Ecto.Association.NotLoaded<association :videos is not
loaded>
iex(15)> Repo.preload(user, :videos)
iex(16)> user.videos
#Ecto.Association.NotLoaded<association :videos is not
loaded>
iex(17)> user = Repo.preload(user, :videos)
iex(18)> user.videos
[%Rumbl.Video{__meta__: #Ecto.Schema.Metadata<:loaded>,
  category: #Ecto.Association.NotLoaded<association
:category is not loaded>,
  category_id: nil, description: "such great many wow", id:
3,
  inserted_at: #Ecto.DateTime<2016-02-28T18:42:41Z>, title:
"Hubidubiee",
  updated_at: #Ecto.DateTime<2016-02-28T18:42:41Z>, url:
"www.lol.com",
  user: #Ecto.Association.NotLoaded<association :user is
not loaded>,
  user_id: 5}]
```

Explicit preloading

Umbrella apps



So we all go and do Elixir and Phoenix now?

Baggage


Eco-System

Thanks & Enjoy Elixir

Tobias Pfeiffer
@PragTob
pragtob.info

Photo Attribution

- CC BY-ND 2.0
  - https://www.flickr.com/photos/mmmswan/8918529543/
- CC BY 2.0
  - https://flic.kr/p/eKGRRJ
- CC BY-NC 2.0
  - https://www.flickr.com/photos/-jule/2728475835/
- CC BY-NC-ND 2.0
  - https://flic.kr/p/eyC7ZT
- CC BY-SA 2.0
  - https://commons.wikimedia.org/wiki/File:Heckert_GNU_white.svg