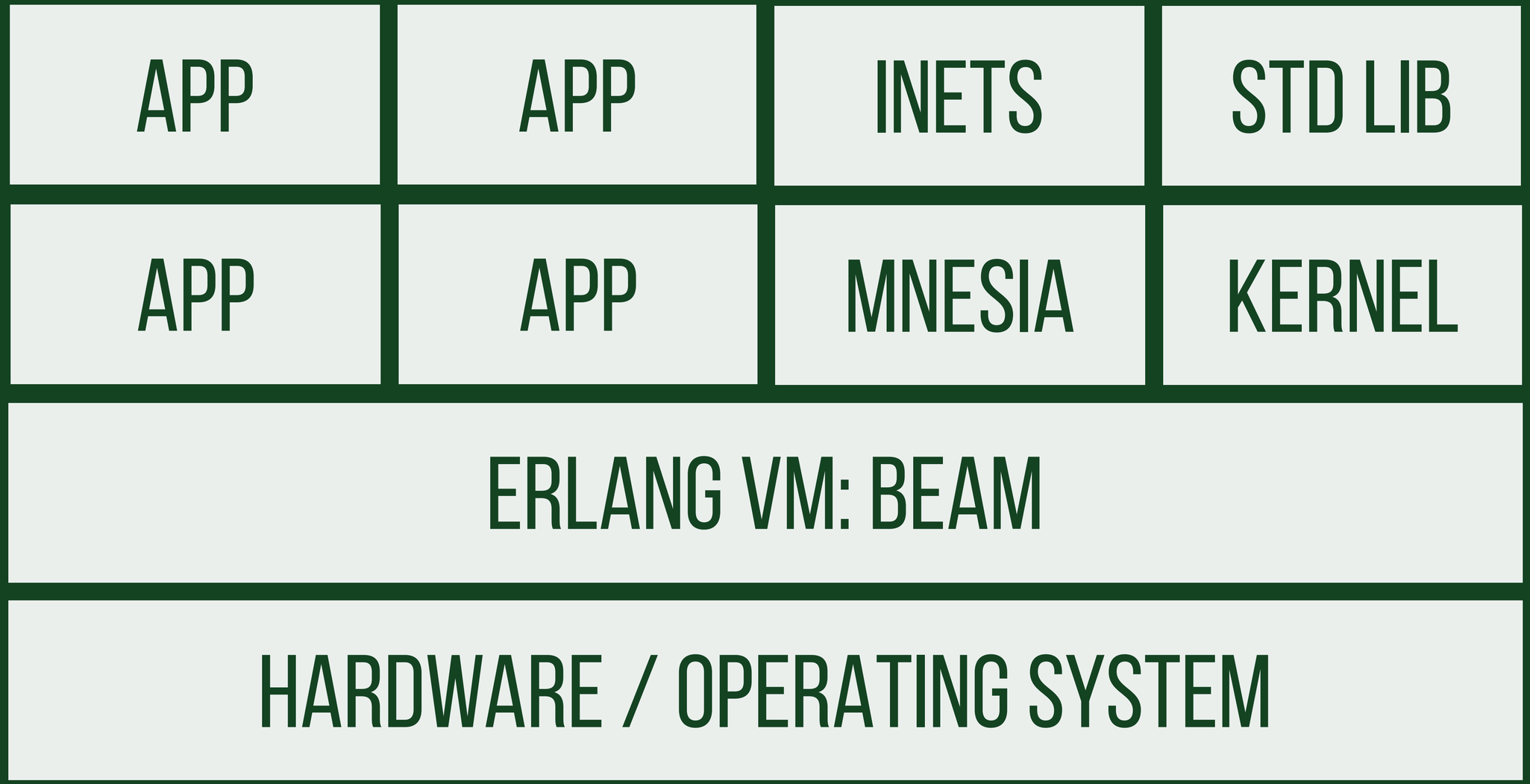
ELIXIR.

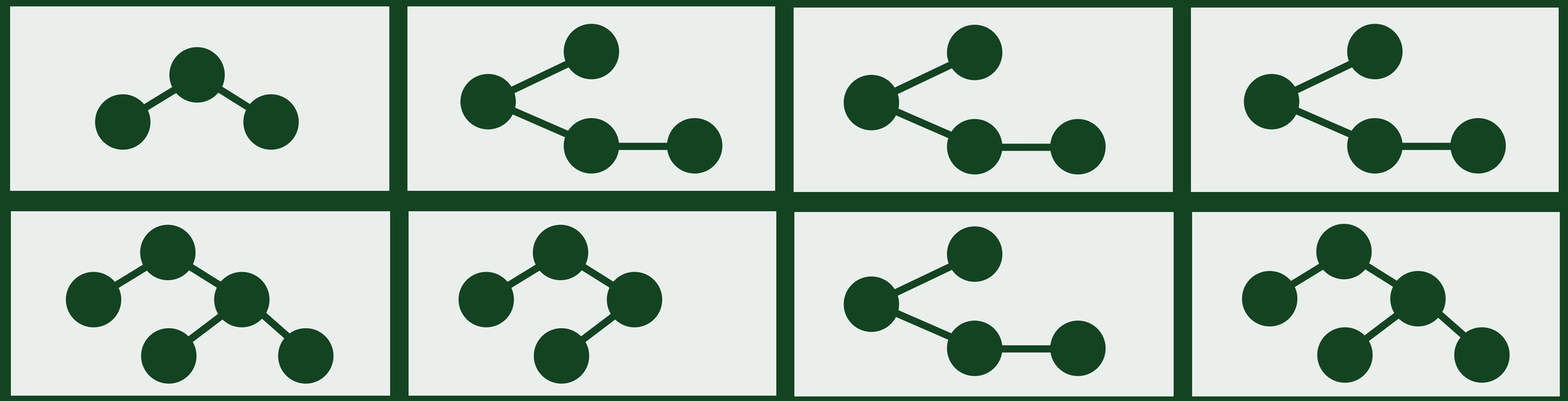
MUCH DISTRIBUTED.

SUCH RELIABLE.


WHY


ERLANG


APP
APP
INETS
STD LIB
APP
APP
MNESIA
KERNEL
ERLANG VM: BEAM
HARDWARE / OPERATING SYSTEM



ERLANG VM: BEAM

HARDWARE / OPERATING SYSTEM

# SCALABILITY: WHATSAPP

~2M CONNECTIONS

~1B ACTIVE USERS DAILY

~50B MESSAGES DAILY

~50 ENGINEERS

# RELIABILITY: TELECOMS

HOT CODE RELOADING

VIRTUAL PROCESSES

OTP: CONCURRENCY FRAMEWORK

CLUSTERING


ME


TOOLING

# MIX

```
# Ruby's Bundler
$ mix new cat_feeder
$ mix deps.get

# Ruby's Rake
$ mix test
$ mix phoenix.new

# Ruby's Rubygems
$ mix archive.install
```

IEX

```
iex(1)> IO.puts "WOW"
WOW
:ok
iex(2)> "Much impress"
"Much impress"
iex(3)> so_interactive!
** (CompileError) iex:3: undefined function
so_interactive!/0
```

# EXUNIT

```elixir
# test.exs

ExUnit.start

defmodule TruthTest do
  use ExUnit.Case

  test "the truth" do
    assert 1 + 1 == 2
  end
end
```

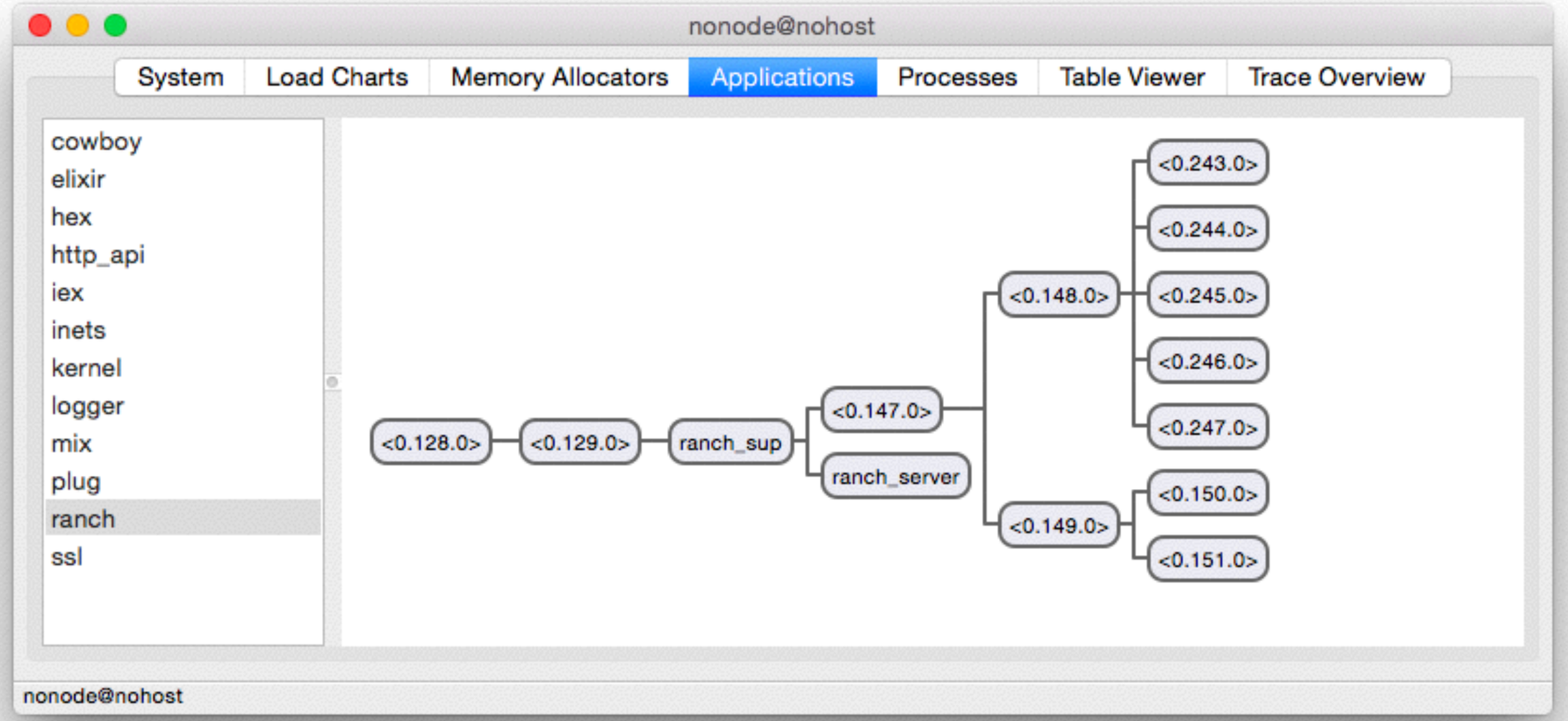
```
$ elixir test.exs
.

Finished in 0.04 seconds
  (0.04s on load, 0.00s on tests)
1 test, 0 failures

Randomized with seed 547571
```

```elixir
# test.exs

ExUnit.start

defmodule TruthTest do
  use ExUnit.Case

  test "the truth" do
    assert 1 + 1 == 2_000_000
  end
end
```

```
$ elixir test.exs

  1) test the truth (TruthTest)
     test.exs:6
     Assertion with == failed
     code: 1 + 1 == 2000000
     lhs:  2
     rhs:  2000000
     stacktrace:
       test.exs:7

Finished in 0.05 seconds
  (0.04s on load, 0.01s on tests)
1 test, 1 failure

Randomized with seed 1481
```

# OBSERVER

# TOOLING


nonode@nohost
System
Load Charts
Memory Allocators
Applications
Processes
Table Viewer
Trace Overview
cowboy
elixir
hex
http_api
iex
inets
kernel
logger
mix
plug
ranch
ssl
<0.128.0>
<0.129.0>
ranch_sup
<0.147.0>
ranch_server
<0.148.0>
<0.149.0>
<0.243.0>
<0.244.0>
<0.245.0>
<0.246.0>
<0.247.0>
<0.150.0>
<0.151.0>
nonode@nohost

# BATTERIES INCLUDED

**Table 1.1 Comparison of technologies used in two real-life web servers**

| Technical requirement | Server A | Server B |
|---|---|---|
| HTTP server | Nginx and Phusion Passenger | Erlang |
| Request processing | Ruby on Rails | Erlang |
| Long-running requests | Go | Erlang |
| Server-wide state | Redis | Erlang |
| Persistable data | Redis and MongoDB | Erlang |
| Background jobs | Cron, Bash scripts, and Ruby | Erlang |
| Service crash recovery | Upstart | Erlang |

(ELIXIR IN ACTION BY SAŠA JURIĆ)


FUNCTIONAL
from NATHANIEL
APOTHECA
Extract of WITCH
From NATHANIEL B WOOD
APOTHECARY,
Extract of WITCH HAZEL
J. MARSON
Extract of
SLIPPERY
Ulmus Lubrica
APOTHEC
A LA CROIX ROUGE
Rue du Vieux-Colombier
PARIS
Mon. MARES
BONZOL
Phen
NATHANIEL B. WOOD
APOTHECARY,
and Peabody Streets

# FUNCTIONS

```
iex> greeter = fn name -> "Hello, #{name}" end
#Function<6.54118792/1 in :erl_eval.expr/5>
iex> greeter.("Simon")
"Hello, Simon"
```

# IMMUTABILITY

```
iex> a = "oh hai"
"oh hai”

iex> a = "ham sandwhich"
"ham sandwhich”

iex> ^a = "turkey bun"
** (MatchError) no match of right hand side value: "turkey bun"
```

# PATTERN MATCHING

```
iex> list = [1, 2, [ 3, 4, 5 ] ]
[1, 2, [3, 4, 5]]
iex> [a, b, c] = list
[1, 2, [3, 4, 5]]
iex> a
1
iex> b
2
iex> c
[3, 4, 5]
```

# TRANSFORMATIONS

# FUNCTIONAL

```
iex> Enum.sum(Enum.filter(Enum.map(1..100_000, &(&1 * 3)), &(rem(&1, 2) != 0)))
7500000000
```

# FUNCTIONAL

```
iex> odd? = fn x -> rem(x, 2) != 0 end
#Function<6.54118792/1 in :erl_eval.expr/5>
iex> multiply_by_3 = fn x -> x * 3 end
#Function<6.54118792/1 in :erl_eval.expr/5>
iex> Enum.sum(Enum.filter(Enum.map(1..100_000, multiply_by_3), odd?))
7500000000
```

# FUNCTIONAL

```
iex> odd? = fn x -> rem(x, 2) != 0 end
#Function<6.54118792/1 in :erl_eval.expr/5>
iex> multiply_by_3 = fn x -> x * 3 end
#Function<6.54118792/1 in :erl_eval.expr/5>

iex> 1..100_000 |> Enum.map(multiply_by_3) |> Enum.filter(odd?) |> Enum.sum
7500000000
```

```elixir
defmodule Transforms do
  def multiply_by_3(x) do
    x * 3
  end

  def odd?(x) do
    rem(x, 2) != 0
  end

  def triple_the_odds(list) do
    list
      |> Enum.map(&Transforms.multiply_by_3/1)
      |> Enum.filter(&Transforms.odd?/1)
      |> Enum.sum
  end
end

iex> Transforms.triple_the_odds(1..100_000)
7500000000
```


DYNAMIC

# TYPES

DYNAMIC

```
defmodule Greeter do
  def greet(name) do
    IO.puts("Hello " <> name)
  end
end

iex> Greeter.greet "Simon"
"Hello Simon"
nil
iex> Greeter.greet 12
** (ArgumentError) argument error
    :erlang.byte_size(12)
    iex:4: Greeter.greet/1
```

DYNAMIC

```elixir
defmodule Greeter do
  @spec greet(String.t) :: String.t
  def greet(name) do
    IO.puts("Hello " <> name)
  end
end
```

DYNAMIC

```elixir
defmodule Greeter do
  @spec greet(String.t) :: String.t
  def greet(name) do
    IO.puts("Hello " <> name)
  end
end
```

# PROTOCOLS

```elixir
defprotocol Blank do
  def blank?(data)
end
```

```
iex> Blank.blank?(0)
false
iex> Blank.blank?([])
true
iex> Blank.blank?([1, 2, 3])
false
```

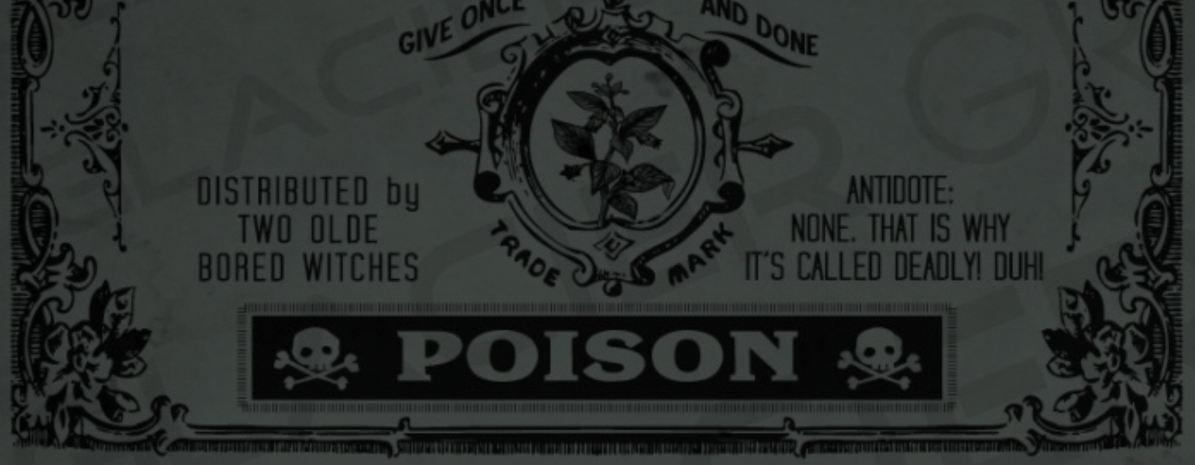
```elixir
# Integers are never blank
defimpl Blank, for: Integer do
  def blank?(_), do: false
end

# Just empty list is blank
defimpl Blank, for: List do
  def blank?([]), do: true
  def blank?(_),  do: false
end
```

# BEHAVIOURS

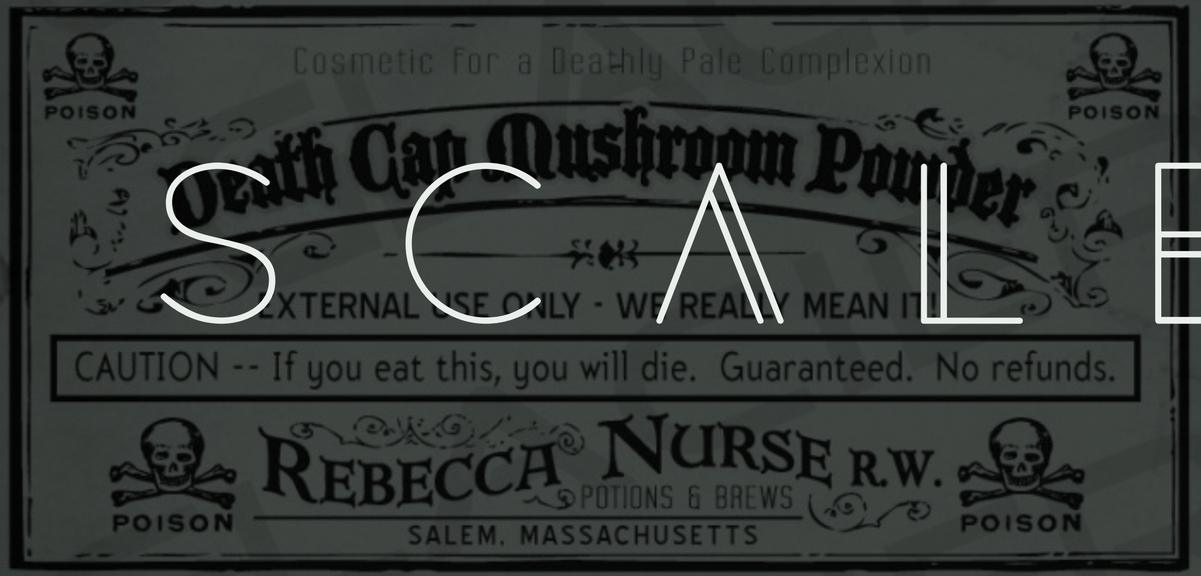
```elixir
defmodule Parser do
  @callback parse(String.t) :: any
  @callback extensions() :: [String.t]
end


defmodule YAMLParser do
  @behaviour Parser

  def parse(str), do: # ... parse YAML
  def extensions, do: ["yml", "yaml"]
end
```

# MACROS

```
iex> sum(1, 2, 3)
** (CompileError) iex:7: undefined function sum/3

iex> quote do
...>   sum(1, 2, 3)
...> end
{:sum, [], [1, 2, 3]}
```


GIVE ONCE
AND DONE
DISTRIBUTED by
TWO OLDE
BORED WITCHES
TRADE MARK
ANTIDOTE:
NONE. THAT IS WHY
IT'S CALLED DEADLY! DUH!
POISON


IT'S PRETTY DEADLY
TRADE
MARK
ANTIDOTE: Really? The pictures didn't give you a clue?
CAULDRON
APOTHECARY


Cosmetic For a Deathly Pale Complexion
POISON
POISON
Death Cap Mushroom Powder
EXTERNAL USE ONLY - WE REALLY MEAN IT!
CAUTION -- If you eat this, you will die. Guaranteed. No refunds.
POISON
REBECCA NURSE R.W.
POTIONS & BREWS
SALEM, MASSACHUSETTS
POISON

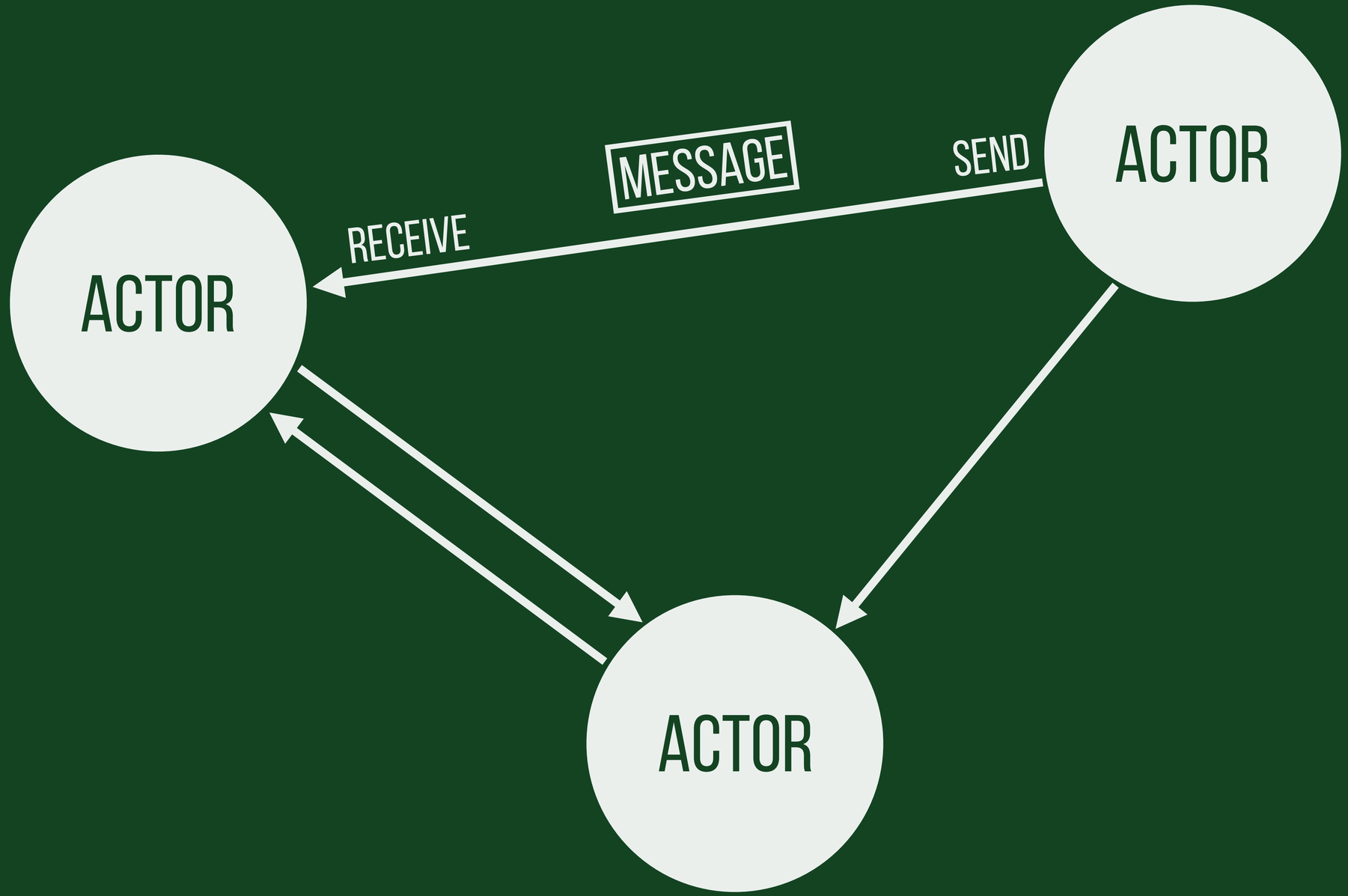

POISON
MANDRAKE ROOT
COBWEB CLEANSER
ANTIDOTE: Mix tincture of Serpent Spit and Ravens Tongue. Dilute with
Sodium Bicarb and drink freely during full moon. Otherwise kiss your butt goodbye.
Ye Olde Magick
Apothecary
100% PURE
HANDPICKED
NATURAL
A
Product of
THE HAUNTED FOREST

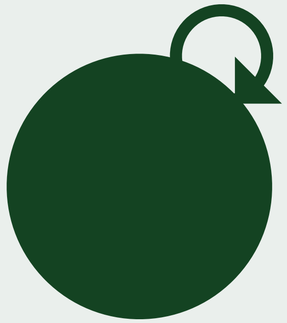

When you start
to prowl...
We silence the
howl!
TINCTURE OF WOLFSBANE
POISON


CASTENADA
DRUG CO.
ESSENCE of DEVIL'S APPLE

SCALEABLE

# ACTOR MODEL


ACTOR
MESSAGE
SEND
ACTOR
RECEIVE
ACTOR



SCALEABLE

```
iex> self
#PID<0.57.0>
iex> send(self, {:hello, "world"})
{:hello, "world"}
iex> receive do
...>   {:hello, msg} -> msg
...>   {:world, msg} -> "won't match"
...> end
"world"
```


# DISTR IBUTED

# NODES

# DISTRIBUTED

```
$ iex --name ben
iex(ben@simon.local)> Node.list
[]
```

```
iex(ben@simon.local)>
Node.connect(:"jerry@simon.local")
true
iex(ben@simon.local)> Node.list
[:"jerry@simon.local"]
```

```
$ iex --name jerry
iex(jerry@simon.local)> Node.list
[]
```

```
iex(jerry@simon.local)> files = fn ->
IO.puts(Enum.join(File.ls!, ", ")) end
#Function<20.54118792/0
in :erl_eval.expr/5>
```

# DISTRIBUTED

```
iex(jerry@simon.local)> files.()
cats.txt
:ok
iex(jerry@simon.local)> Node.spawn(:"ben@simon.local", files)
dynamic.ex, hello.exs, scablable.ex, shell.sh, test.exs, yo.ex
#PID<8207.68.0>
```

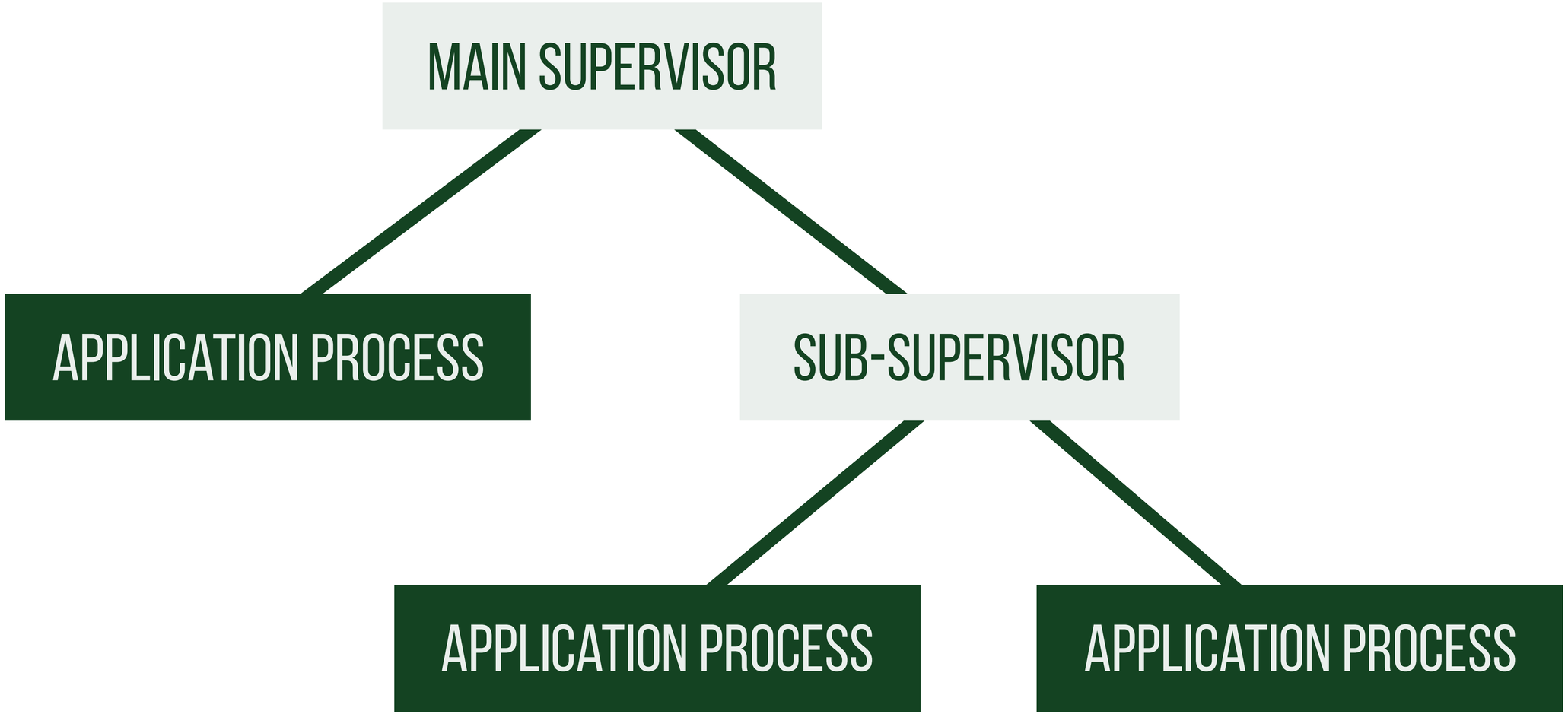
# REMOTE BYTE CODE

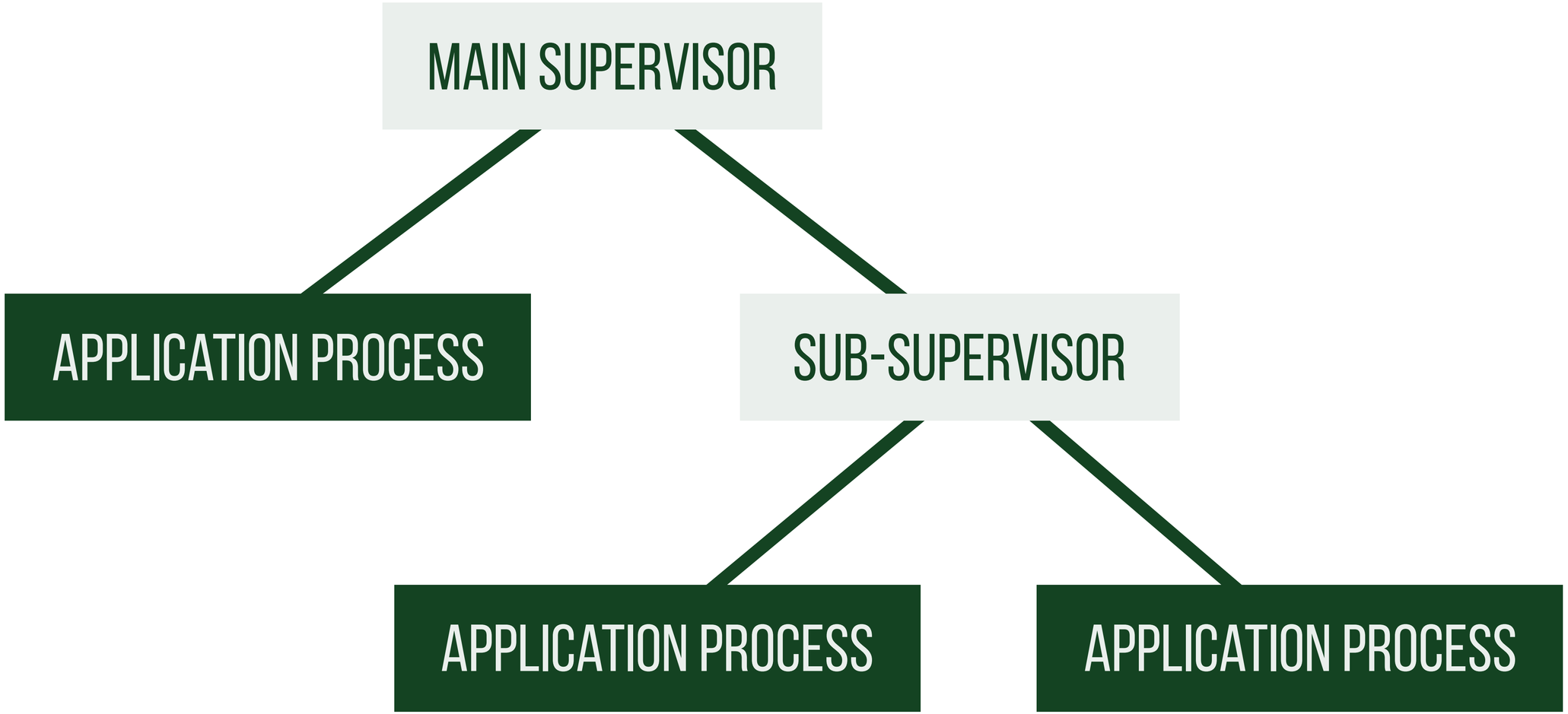
# DISTRIBUTED

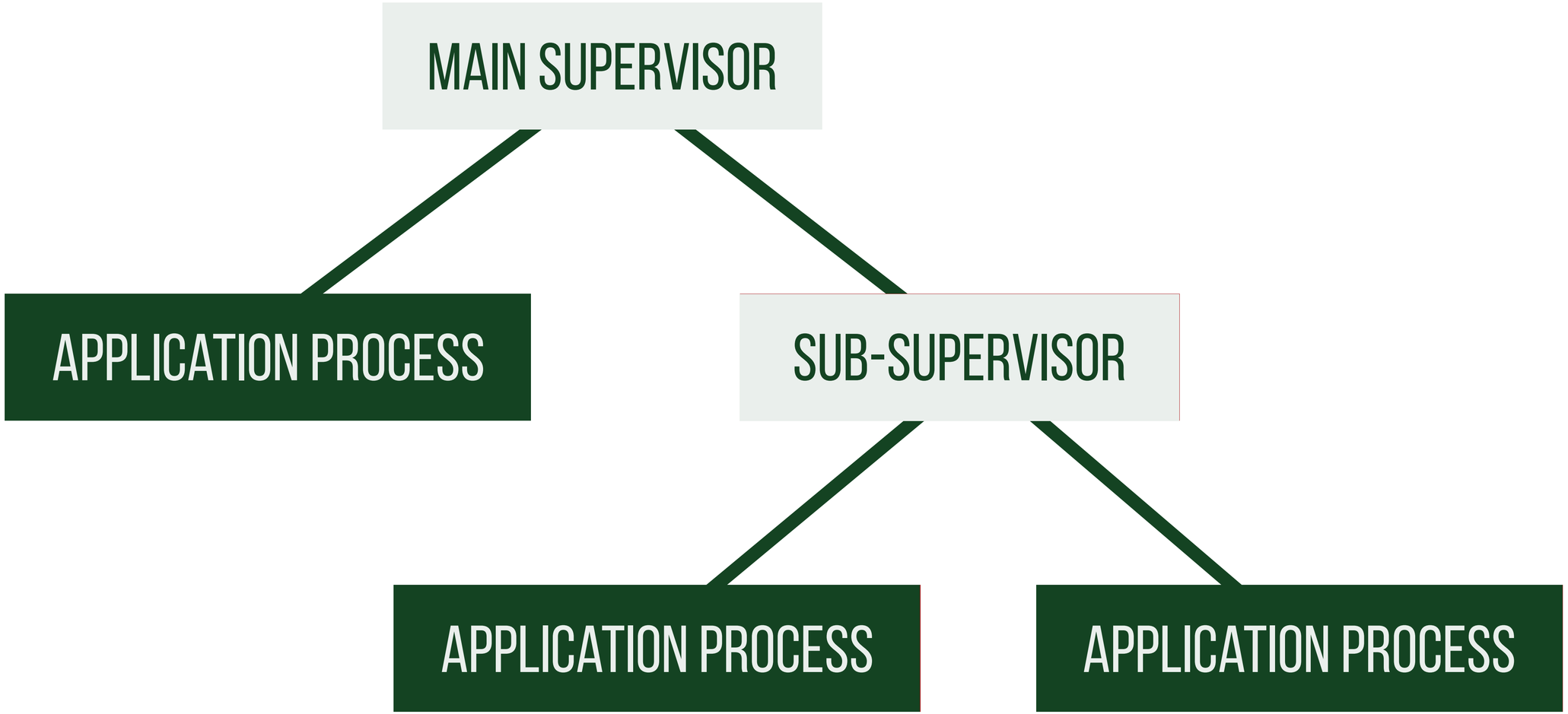
```
iex(1)> defmodule Math do
...(1)>   def sum(a, b) do
...(1)>     a + b
...(1)>   end
...(1)> end
{:module, Math,
 <<70, 79, 82, 49, 0, 0, 4, 212, 66, 69, 65, 77, 69, 120, 68, 99, 0,
0, 0, 157, 131, 104, 2, 100, 0, 14, 101, 108, 105, 120, 105, 114,
95, 100, 111, 99, 115, 95, 118, 49, 108, 0, 0, 0, 4, 104, 2, ...>>,
 {:sum, 2}}
```


RELIABLE

# SUPERVISORS


MAIN SUPERVISOR
APPLICATION PROCESS
SUB-SUPERVISOR
APPLICATION PROCESS
APPLICATION PROCESS


MAIN SUPERVISOR
APPLICATION PROCESS
SUB-SUPERVISOR
APPLICATION PROCESS
APPLICATION PROCESS


MAIN SUPERVISOR
APPLICATION PROCESS
SUB-SUPERVISOR
APPLICATION PROCESS
APPLICATION PROCESS

# DEMO

# EXAMPLE: PHOENIX CHANNELS

“Erlang makes difficult things easy
and easy things difficult”

“Elixir makes difficult things easy
and easy things easy, too.”

Platform 45

```
{
    “” => “@siefi”,
    “” => “@sighmin”,
    “” => “simon@platform45.com”
}
```

# RESOURCES

| | |
|---|---|
| INTRO | HTTP://ELIXIR-LANG.ORG/GETTING-STARTED |
| CODE | HTTPS://GITHUB.COM/SIGHMIN/CHATTER |
| SLIDES | HTTPS://SPEAKERDECK.COM/SIMONVANDYK/ELIXIR-MUCH-DISTRIBUTED-SUCH-RELIABLE |