

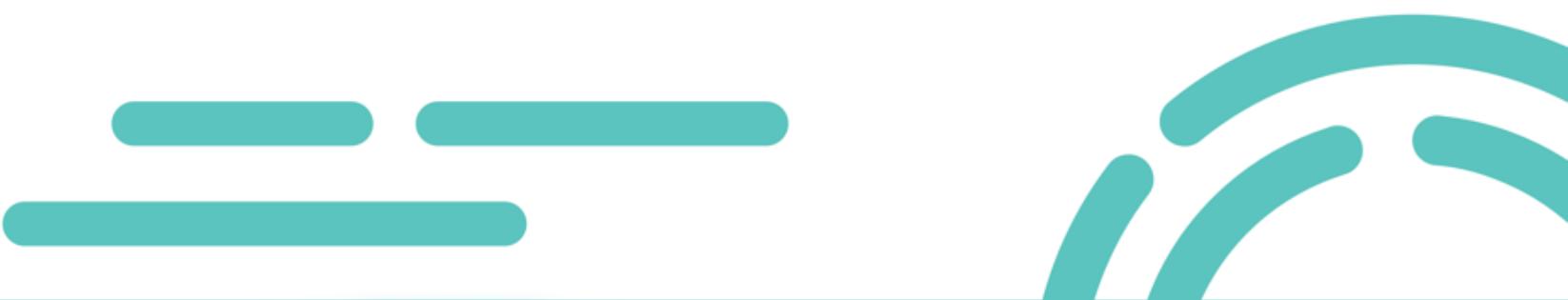


DEVCONF

Fending off Zombies with OTP

Dave Long

Platinum sponsor:



BUILDING TELEPHONY APPLICATIONS WITH TWILIO AND ELIXIR



DAVE LONG

DIRECTOR OF DEVELOPMENT @ CAGE DATA

Husband & Father
Missionary to Uganda

[@davejlong](https://twitter.com/davejlong) on Twitter / GitHub / Keybase
davejlong.com on Web

A dark, moody photograph of a man with a zombie-like appearance. He has a large, gory wound on his forehead and is wearing a dark suit and tie. He is looking over his shoulder, and the background is blurred.

**SUBSCRIBE TO THE APPLICATION AT
ZOMBIES.DAVEJLONG.COM**

FENDING OFF ZOMBIES

1. What is Elixir?
2. What is OTP?
3. What is Twilio?
4. Let's build something!



WHAT IS ELIXIR?

Elixir is a dynamic, functional language designed
for building scalable and maintainable
applications.

– elixir-lang.org



ELIXIR IS...

- ▶ Scalable
- ▶ Fault-Tolerant
- ▶ Functional
- ▶ Extensible

```
current_process = self()

# Spawn an Elixir process (not an operating system one!)
spawn_link(fn ->
  send current_process, {:msg, "Must eat brains!"}
end)

# Block until the message is received
receive do
  {:msg, contents} -> IO.puts contents
end
```

```
current_process = self()

# Spawn an Elixir process (not an operating system one!)
spawn_link(fn ->
  send current_process, {:msg, "Must eat brains!"}
end)

# Block until the message is received
receive do
  {:msg, contents} -> IO.puts contents
end
```

```
current_process = self()

# Spawn an Elixir process (not an operating system one!)
spawn_link(fn ->
  send current_process, {:msg, "Must eat brains!"}
end)

# Block until the message is received
receive do
  {:msg, contents} -> IO.puts contents
end
```

```
current_process = self()

# Spawn an Elixir process (not an operating system one!)
spawn_link(fn ->
  send current_process, {:msg, "Must eat brains!"}
end)

# Block until the message is received
receive do
  {:msg, contents} -> IO.puts contents
end
```

```
import Supervisor.Spec

children = [
    supervisor(Zombie.Hoard, []),
    worker(Hunter, [])
]

Supervisor.start_link(children, strategy: :one_for_one)
```



 @davejlong PGP (4FF9ED7C)

```
import Supervisor.Spec

children = [
    supervisor(TCP.Pool, []),
    worker(TCP.Acceptor, [4040])
]

Supervisor.start_link(children, strategy: :one_for_all)
```

```
def shoot(%Weapon{uses: ammo, type: "rifle"}) when ammo > 0 do
  # You shot a zombie!
end

def shoot(%Weapon{uses: uses, type: "knife"}) when uses > 0 do
  # You shot a zombie, but were too close and
  # got eaten by others
end

def shoot(_), do: # You're a zombie now.

shoot Weapon.gets("Dessert Eagle")
#=> Fails if the weapon has no more uses
```

OBJECT ORIENTED CODE

```
title. # <= "Top 3 Reasons To Love Zombies!"  
downcase.  
gsub(/\W/, " "). # convert non-word chars (like - , !) into spaces  
split. # drop extra whitespace  
join("-") # join words with dashes
```

OBJECT ORIENTED CODE

```
title. # <= "Top 3 Reasons To Love Zombies!"  
downcase.  
gsub(/\W/, " "). # convert non-word chars (like - , !) into spaces  
split. # drop extra whitespace  
join("-") # join words with dashes
```

OBJECT ORIENTED CODE

```
title. # <= "Top 3 Reasons To Love Zombies!"  
downcase.  
gsub(/\W/, " "). # convert non-word chars (like - , !) into spaces  
split. # drop extra whitespace  
join("-") # join words with dashes
```

OBJECT ORIENTED CODE

```
title. # <= "Top 3 Reasons To Love Zombies!"  
downcase.  
gsub(/\W/, " "). # convert non-word chars (like - , !) into spaces  
split. # drop extra whitespace  
join("-") # join words with dashes
```

OBJECT ORIENTED CODE

```
title. # <= "Top 3 Reasons To Love Zombies!"  
downcase.  
gsub(/\W/, " "). # convert non-word chars (like - , !) into spaces  
split. # drop extra whitespace  
join("-") # join words with dashes
```

top-3-reasons-to-love-zombies"

FUNCTIONAL CODE

```
title # <= Top 3 Reasons To Love Zombies!
|> String.downcase
|> String.replace(~r/\W/, " ") # convert non-word chars (like -, !) into spaces
|> String.split                  # drop extra whitespace
|> Enum.join("-")                # join words with dashes
```

FUNCTIONAL CODE

```
title # <= Top 3 Reasons To Love Zombies!
|> String.downcase
|> String.replace(~r/\W/, " ") # convert non-word chars (like -, !) into spaces
|> String.split                      # drop extra whitespace
|> Enum.join("-")                     # join words with dashes
```

FUNCTIONAL CODE

```
title # <= Top 3 Reasons To Love Zombies!
|> String.downcase
|> String.replace(~r/\W/, " ") # convert non-word chars (like -, !) into spaces
|> String.split                      # drop extra whitespace
|> Enum.join("-")                     # join words with dashes
```

FUNCTIONAL CODE

```
title # <= Top 3 Reasons To Love Zombies!
|> String.downcase
|> String.replace(~r/\W/, " ") # convert non-word chars (like -, !) into spaces
|> String.split                      # drop extra whitespace
|> Enum.join("-")                     # join words with dashes
```

FUNCTIONAL CODE

```
title # <= Top 3 Reasons To Love Zombies!
|> String.downcase
|> String.replace(~r/\W/, " ") # convert non-word chars (like - ,!) into spaces
|> String.split                      # drop extra whitespace
|> Enum.join("-")                    # join words with dashes
```

top-3-reasons-to-love-zombies

FUNCTIONAL

```
title # <= Top 3 Reasons To Love Zombies!
|> String.downcase
|> String.replace(~r/\W/, " ") # convert non-word chars (like -,!) into spaces
|> String.split                  # drop extra whitespace
|> Enum.join("-")                # join words with dashes
```

OBJECT ORIENTED

```
title. # <= Top 3 Reasons To Love Zombies!
downcase.
gsub(/\W/, " "). # convert non-word chars (like -,!) into spaces
split.            # drop extra whitespace
join("-")        # join words with dashes
```

MIX

- ▶ Elixir's build tool
- ▶ Tasks for creating, compiling, testing applications
- ▶ Easy to extend
- ▶ Manages dependencies

HEX

- ▶ Elixir's package manager
- ▶ Like Rubygems, NPM, Pypi
- ▶ 4905 packages
- ▶ ~500,000 daily downloads
- ▶ ~127,000,000 all time downloads



@davejlong PGP (4FF9ED7C)

WHAT IS OTP?

If half of Erlang's greatness comes from its concurrency and distribution and the other half comes from its error handling capabilities, then the OTP framework is the third half of it.

– LearnYouSomeErlang.com/what-is-otp

A dark, atmospheric photograph of several zombies walking through a field. In the foreground, a woman with long dark hair and a man with a white face and red lips are prominent. The ground is covered in dry, brown grass and fallen leaves.

WHAT IS OTP?

Modules for managing:

- ▶ State
- ▶ Processes
- ▶ Async task

AGENTS

SIMPLE STATE MANAGER

```
defmodule Weapon do
  use Agent

  @initial 10

  def start_link(), do: Agent.start_link(fn _ &gt; @initial end, name: __MODULE__)

  def attack(task, project), do: Agent.update(__MODULE__, &(&1 - 1))

  def reload(), do: Agent.update(__MODULE__, fn _ &gt; @initial end)
end
```

```
defmacro __using__(opts) do
  quote location: :keep, bind_quoted: [opts: opts] do
    spec = [
      id: opts[:id] || __MODULE__,
      start: Macro.escape(opts[:start]) || quote(do: {__MODULE__, :start_link, [arg]}),
      restart: opts[:restart] || :permanent,
      shutdown: opts[:shutdown] || 5000,
      type: :worker
    ]
  end

  @doc false
  def child_spec(arg) do
    %{unquote_splicing(spec)}
  end

  defoverridable child_spec: 1
end
end
```

AGENTS

```
defmodule Weapon do
  use Agent

  @initial 10

  def start_link(), do: Agent.start_link(fn -> @initial end, name: __MODULE__)

  def attack(task, project), do: Agent.update(__MODULE__, &(&1 - 1))

  def reload(), do: Agent.update(__MODULE__, fn -> @initial end)
end
```

AGENTS

```
defmodule Weapon do
  use Agent

  @initial 10

  def start_link(), do: Agent.start_link(fn -> @initial end, name: __MODULE__)

  def attack(task, project), do: Agent.update(__MODULE__, &(&1 - 1))

  def reload(), do: Agent.update(__MODULE__, fn -> @initial end)
end
```

AGENTS

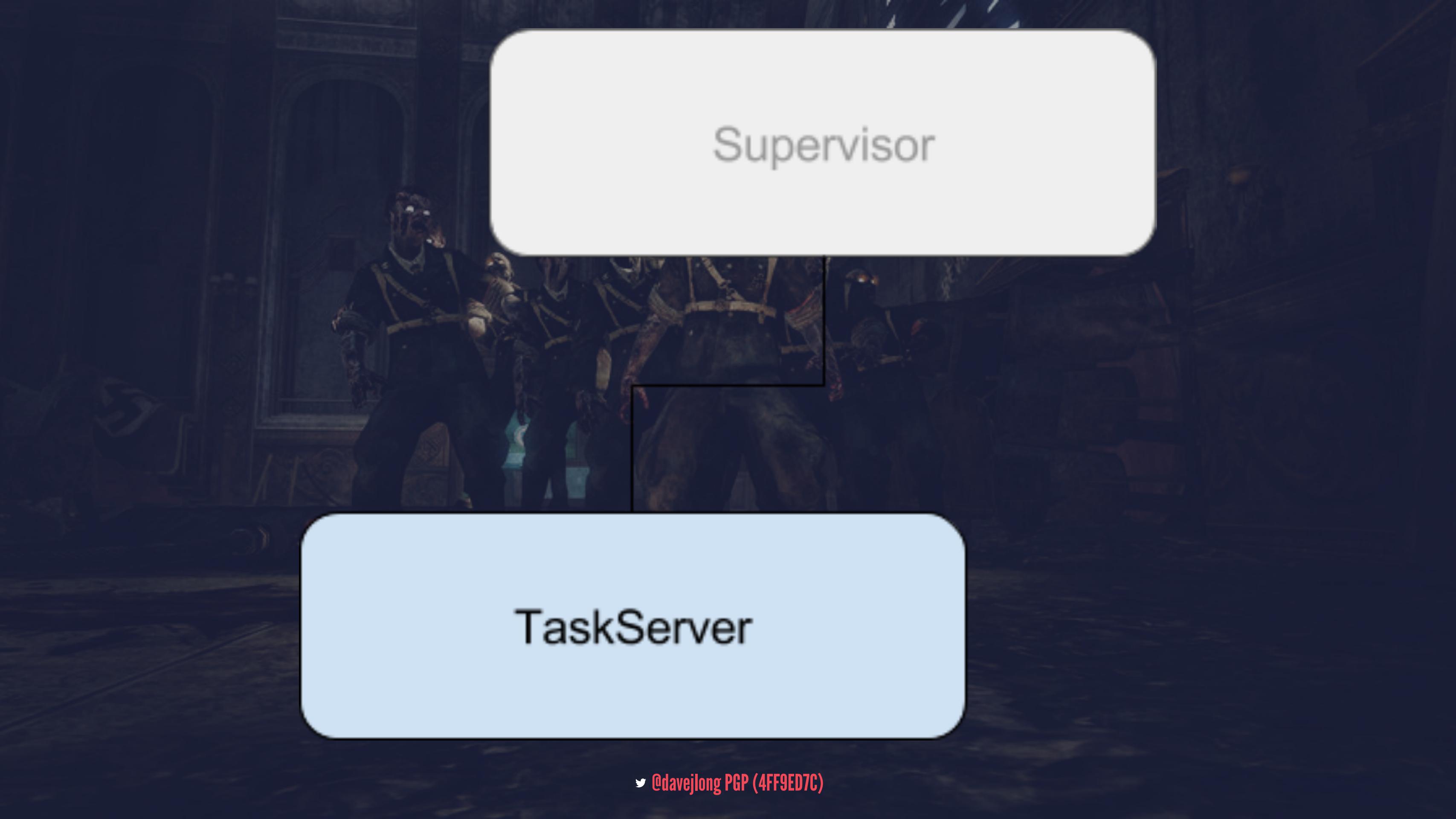
```
defmodule Weapon do
  use Agent

  @initial 10

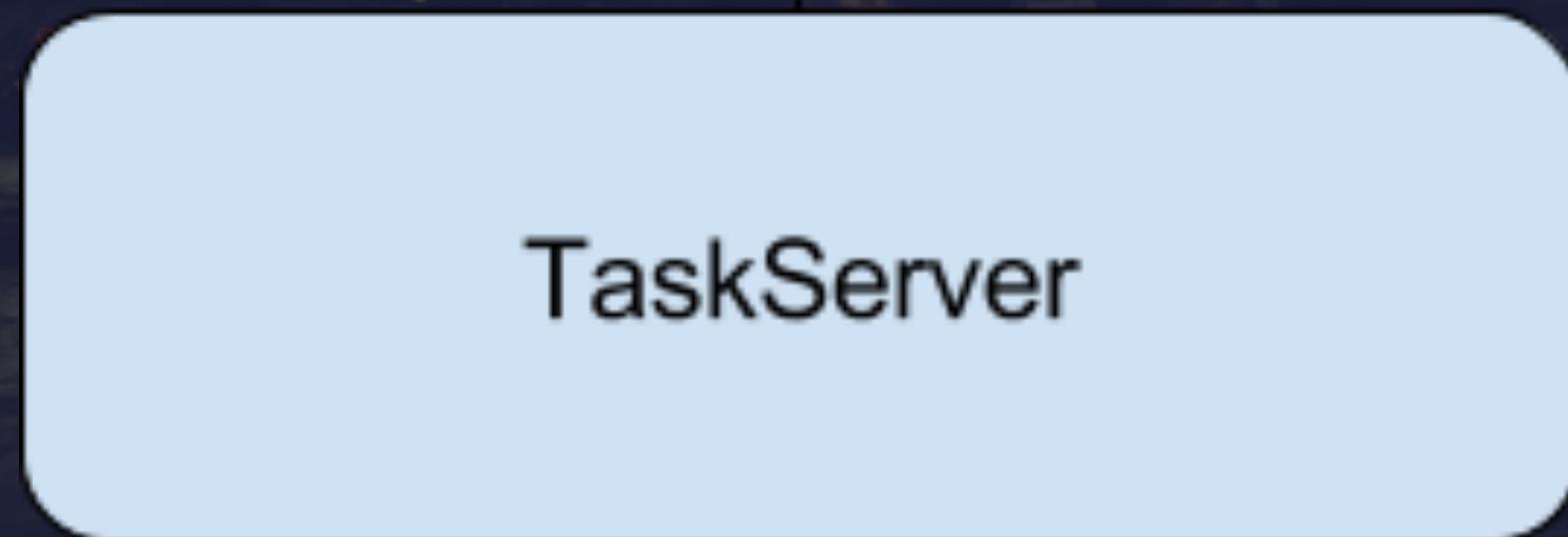
  def start_link(), do: Agent.start_link(fn -> @initial end, name: __MODULE__)

  def attack(task, project), do: Agent.update(__MODULE__, &(&1 - 1))

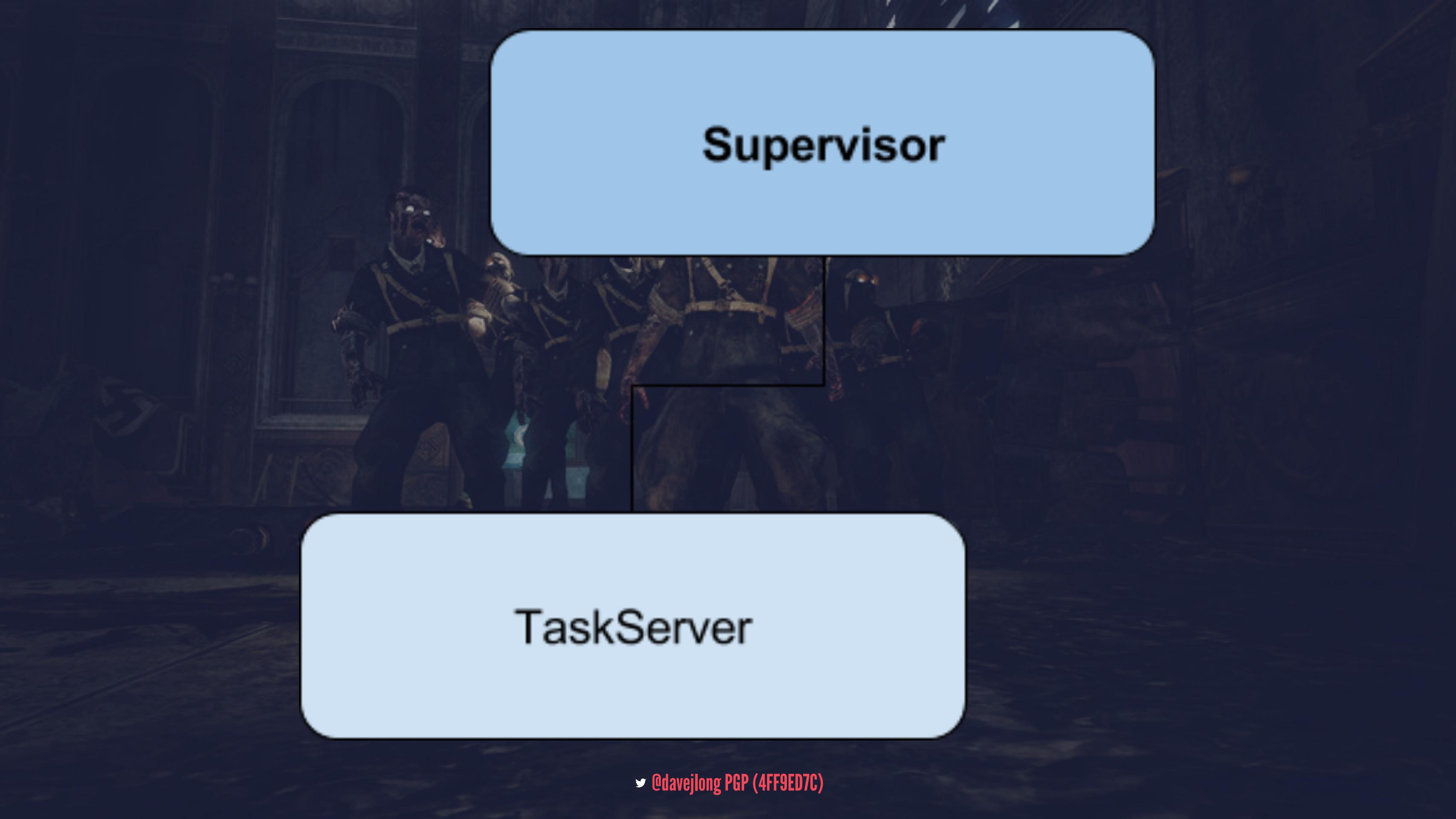
  def reload(), do: Agent.update(__MODULE__, fn -> @initial end)
end
```



Supervisor



TaskServer



A dark, atmospheric scene from a video game featuring several skeletal figures in a stone structure. In the foreground, a large, ornate doorway is visible, through which skeletal figures are emerging. The scene is dimly lit, with shadows and firelight illuminating the bones and stone walls.

Supervisor

TaskServer

CREATING A SUPERVISOR

```
$ mix new walking_dead --sup
```

```
# lib/walking_dead/application.ex
defmodule WalkingDead.Application do
  use Application

  def start(_type, _args) do
    children = [
      Weapon.child_spec(nil)
    ]

    opts = [strategy: :one_for_one, name: MyApplication.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

```
# lib/walking_dead/application.ex
defmodule WalkingDead.Application do
  use Application

  def start(_type, _args) do
    children = [
      Weapon.child_spec(nil)
    ]

    opts = [strategy: :one_for_one, name: MyApplication.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

```
# lib/walking_dead/application.ex
defmodule WalkingDead.Application do
  use Application

  def start(_type, _args) do
    children = [
      Weapon.child_spec(nil)
    ]

    opts = [strategy: :one_for_one, name: MyApplication.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

```
# lib/walking_dead/application.ex
defmodule WalkingDead.Application do
  use Application

  def start(_type, _args) do
    children = [
      Weapon.child_spec(nil)
    ]

    opts = [strategy: :one_for_one, name: MyApplication.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

```
# lib/walking_dead/application.ex
defmodule WalkingDead.Application do
  use Application

  def start(_type, _args) do
    children = [
      Weapon.child_spec(nil)
    ]

    opts = [strategy: :one_for_one, name: MyApplication.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

```
# lib/walking_dead/application.ex
defmodule WalkingDead.Application do
  use Application

  def start(_type, _args) do
    children = [
      Weapon.child_spec(nil)
    ]

    opts = [strategy: :one_for_one, name: MyApplication.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

RUNNING OUR APPLICATION

```
# mix.exs
defmodule WalkingDead.Mixfile do
  use Mix.Project

  def project do
    [ ... ]
  end

  def application do
    [
      extra_applications: [:logger],
      mod: {WalkingDead.Application, []}
    ]
  end
end
```

A dark, moody background featuring a man's face in profile, looking towards the right. He has short, light-colored hair and a serious, intense expression. There are small amounts of red liquid, possibly blood, visible on his white shirt. The lighting is dramatic, with strong shadows.

RUNNING OUR APPLICATION

```
$ mix run --no-halt
```

RUNNING WITH REPL

```
$ iex -S mix run
```

WHAT IS TWILIO?



A photograph showing several people of diverse ages and ethnicities standing behind a chain-link fence. They are looking over the fence with expressions ranging from concern to hope. Some are holding hands or reaching towards the fence. The background shows some foliage and a clear sky.

WHAT IS TWILIO?

- ▶ API for Phone and SMS
- ▶ TaskRouter for communication workflows
- ▶ Two Factor Authentication Platform
- ▶ APIs for Communication

A large, diverse crowd of people is gathered in an open field. They are standing in various groups, some looking towards the camera and others looking away. The sky above is filled with heavy, grey clouds, suggesting an overcast day. The overall atmosphere is one of a public event or gathering.

LET'S BUILD
SOMETHING

THE APPLICATION

- ▶ Zombies are invading
- ▶ World leaders need to trigger alerts
- ▶ People must be able to subscribe to get alerted



```
$ mix new zombie_alerter --sup
```

HOW IS THIS
REVIEWER
A MAN?

EMERGENCY WARNING SYSTEM



@davejlong PCP (4FF9ED7C)

AFP

QUESTIONS?

**SLIDES AND CODE AT
GITHUB.COM/DAVEJLONG/ZOMBIES**

THANK YOU

