# Scaling Work Queues with Oban

Andrei Zvonimir Crnkovic - **`0x7f d.o.o.`**

## whoami

- elixir product developer/freelancer
- writing a book on Oban
- licensed accountant & co-founder of SmartAccount
- vice-president at Open.hr

0x7f SmartAccount

HR OPEN

## Agenda

- Why Elixir?
- What is a background job?
- What is Oban?
- Using Oban and some cool use cases
- Oban Pro

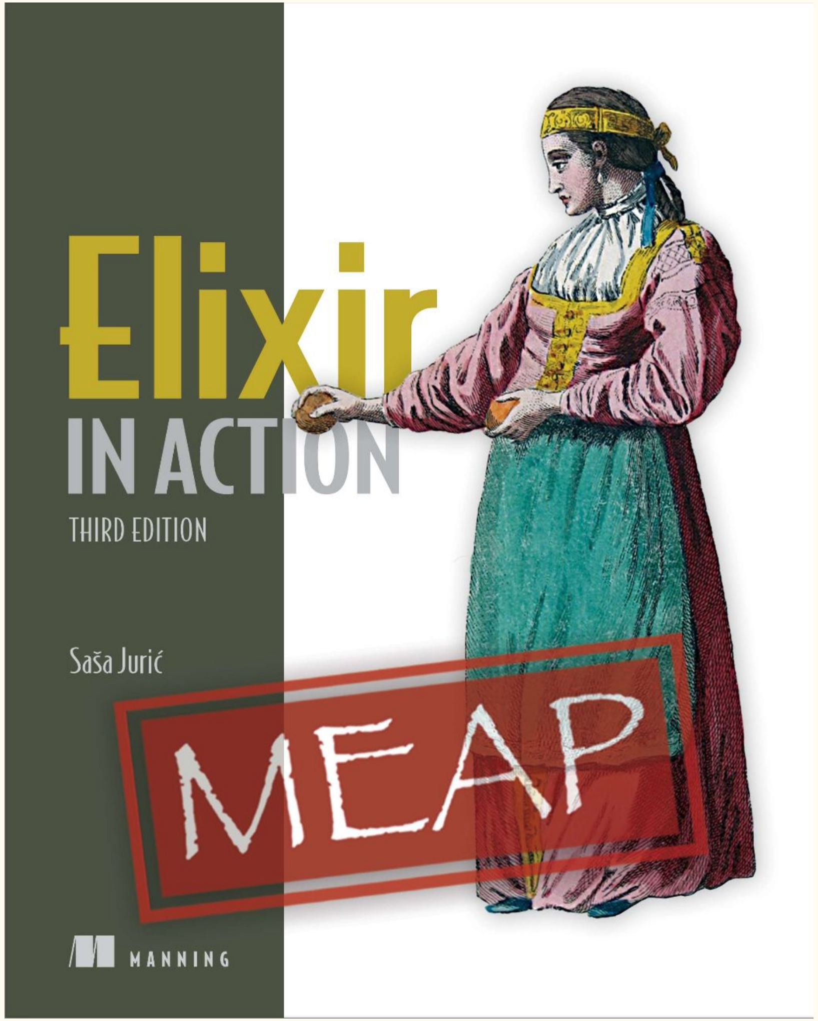
## Why Elixir?

# Functional programming & the Erlang VM

1. **Concurrent and Scalable**
   - Handles thousands of concurrent processes efficiently.
   - Ideal for applications with high parallel processing needs.
2. **Fault-Tolerant**
   - Designed for systems requiring high uptime.
   - Processes are isolated, ensuring that failure in one doesn't affect others.
3. **Hot Code Swapping**
   - Allows code updates without stopping the system.
   - Critical for systems needing continuous operation.
4. **Soft Real-Time Capabilities**
   - Supports applications with real-time processing needs.
   - Ensures timely execution of tasks.
5. **Robust Ecosystem**
   - Strong support for Erlang and Elixir, with growing libraries and tools.
   - Benefits from a vibrant, supportive community.

1. **Lightweight and Efficient**
    - BEAM processes are extremely lightweight, often using just a few kilobytes of memory.
    - This allows the creation of millions of processes on a single machine.
2. **Isolated Execution**
    - Each process runs in complete isolation with no shared memory.
    - This design prevents processes from interfering with each other, enhancing fault tolerance.
3. **Garbage Collection**
    - Each process has its own garbage collector, which runs independently.
    - This localized garbage collection minimizes pause times and does not stop other processes.

1. **Deployment**
   - No runtimes needed on the server
   - Single file* deployment
2. **Multi-node friendly**
   - Built-in support for multi-node operations
   - Very easy to scale up and down
3. **Awesome library support**
   - Developed with scale and ease of use in mind
   - Great community support
   - Open source by default



For deep dive into Elixir I recommend:

Elixir in action, 3rd edition by Saša Jurić

https://www.manning.com/books/elixir-in-action-third-edition

## What is a background job?

**Sending emails**

All transactional emails you send to the user

**Processing uploads**

Long running processes going thru user uploaded data, e.g. resizing photos

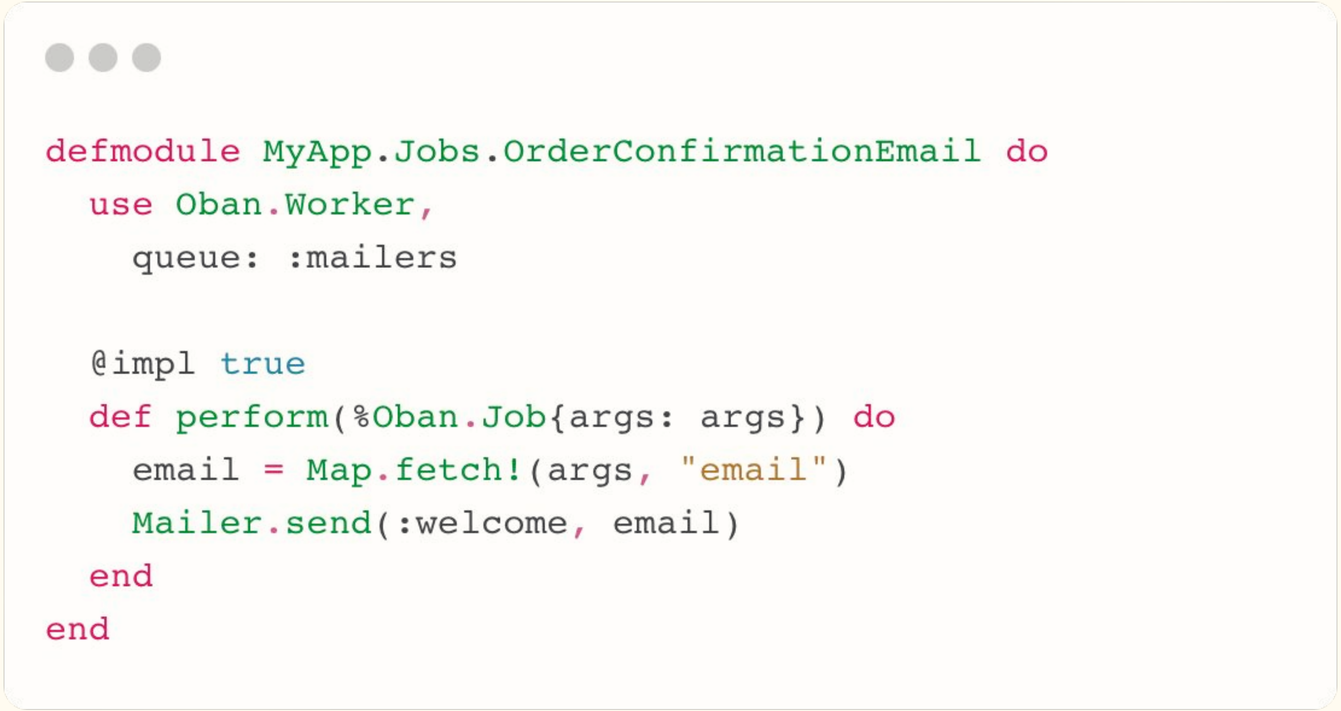
**Generating documents**

Generating PDFs, esp. if you have more than one

**Scheduled jobs**

Clean up jobs for sessions or uploaded files, etc.

Stop wasting your users time...



```elixir
defmodule MyApp.Jobs.OrderConfirmationEmail do
  use Oban.Worker,
    queue: :mailers

  @impl true
  def perform(%Oban.Job{args: args}) do
    email = Map.fetch!(args, "email")
    Mailer.send(:welcome, email)
  end
end
```

```elixir
args = %{"email" => order.customer_email}
Oban.insert!(%OrderConfirmationEmail{args: args})
```

## PostgreSQL

```elixir
args = %{"email" => order.customer_email}
Oban.insert(%OrderConfirmationEmail{args: args})
```

```elixir
Repo.transaction(fn ->
  User.create(order.customer_email)
  args = %{"email" => order.customer_email}
  Oban.insert(%OrderConfirmationEmail{args: args})
end)
```

## Using Oban and some cool use cases

```elixir
defmodule MyApp.Jobs.OrderConfirmationEmail do
  use Oban.Worker,
    queue: :mailers

  @impl true
  def perform(%Oban.Job{args: args}) do
    email = Map.fetch!(args, "email")
    Mailer.send(:welcome, email)
  end
end
```

```elixir
defmodule MyApp.Jobs.OrderConfirmationEmail do
  use Oban.Worker,
    queue: :mailers,
    max_attempts: 3

  @impl true
  def perform(%Oban.Job{args: args}) do
     email = Map.fetch!(args, "email")
     Mailer.send(:welcome, email)
  end
end
```

```elixir
defmodule MyApp.Jobs.OrderConfirmationEmail do
  use Oban.Worker,
    queue: :mailers,
    max_attempts: 3,
    unique: [period: 30]

  @impl true
  def perform(%Oban.Job{args: args}) do
     email = Map.fetch!(args, "email")
     Mailer.send(:welcome, email)
  end
end
```

## Digest emails

```elixir
defmodule MyApp.Jobs.OrderDigestEmail do
   use Oban.Worker,
    queue: :mailer,
    max_attempts: 3,
    unique: [period: :infinity, states: [:scheduled]]
end
```

```elixir
def enqueue(order_id) do
  %{order_id: order_id}
   |> new(schedule_in: 600, replace: [scheduled: [:scheduled_at]])
   |> Oban.insert()
end
```

# Oban Pro

1. **Oban Web**
    - A beautiful dashboard to manage your jobs
    - You can schedule jobs manually, re-try them, inspect them, ...
2. **Oban Pro**
    - Plugins
    - Better Job control

## Use case

```elixir
# config/config.exs

config :api, Oban,
  repo: MyApp.Repo,
  engine: Oban.Pro.Engines.Smart,
  queues: [
    mailer: 5
  ]
```

```
# config/config.exs

config :api, Oban,
  repo: MyApp.Repo,
  engine: Oban.Pro.Engines.Smart,
  queues: [
    mailer: [
      local_limit: 5,
      rate_limit: [
        allowed: 1, period: 3
      ]
    ]
  ]
```

```elixir
# config/config.exs

config :api, Oban,
  repo: MyApp.Repo,
  engine: Oban.Pro.Engines.Smart,
  queues: [
    mailer: [
      local_limit: 5,
      rate_limit: [
        allowed: 1, period: 3,
        partition: [args: [:email]]
      ]
    ]
  ]
```

# Thank you!

## Links

- https://0x7f.dev/weblica2024

# DORS/CLUC

29. DANI OTVORENIH RAČUNARSKIH SUSTAVA / CROATIAN LINUX USERS' CONFERENCE 2024.

Open for business:

**15. SVIBNJA 2024.**

Hackaton:

**18.-19. SVIBNJA 2024.**

Predavanja i radionice:

**16.-17. SVIBNJA 2024.**

Sveučilište Algebra, Gradišćanska 24, Zagreb

https://dorscluc.org

Use code **ELIXIR24** for 40% off