Low Ceremony Microservices with Elixir

Chris Nelson
chris@teamgaslight.com

Hi!

Agenda

- A little history
- Enough Elixir to be dangerous
- Refactoring Auctioneer to Microservices

Step into my time machine

- It’s 2011
- Mid-2000s startups are > 5 yrs old
- Simple Rails apps have grown to hundreds of dev and millions of users

Problems

- Dozens (hundreds?) of developers on a single code base
- Slow test suites
- Deploying monolith for every small change
- Performance

Solution: Microservices

- Single monolith becomes n apps
- Smaller, separately deployable
- Often polyglot
- Rewrite all the things!

Microservice challenges

- Deployment
- Routing
- Monitoring
- Versioning

“So my primary guideline would be **don't even consider microservices unless you have a system that's too complex to manage as a monolith**.” - Martin Fowler

What if

- We could start simply
- And refactor into microservices without rewriting or adding complex architecture

Microservices != HTTP

Meet Erlang

- Developed at Ericsson in 1987
- Prolog inspired syntax
- BEAM
- Designed to handle specific challenges of Telecom
  - High availability (like, real high)
  - High concurrency (like, real high)
  - Sound familiar?

Meet Elixir

- Developed in 2012 by José Valim
- Version 1.2 released in December
- Functional, immutable, process oriented
- More familiar looking syntax
- Better abstractions around Erlang platform
- Compiled to Erlang BEAM bytecode

Tools

- iex
  - Interactive Elixir
- mix
  - dependency management and build system
- observer
  - process inspector

Enough Elixir to be dangerous

Basic types

```
iex> 1                          # integer
iex> 0x1F                       # integer
iex> 1.0                        # float
iex> true                       # boolean
iex> :atom                      # atom / symbol
iex> "elixir"                   # string
iex> [1, 2, 3]                  # list
iex> {1, 2, 3}                  # tuple
iex> %{a: "b", c: "d"}          # map
```

ElixirExamples

Meet Phoenix

- MVC Web framework for Elixir
- Production ready (1.1 as of December)
- Very familiar coming from Rails (or similar)

Auctioneer

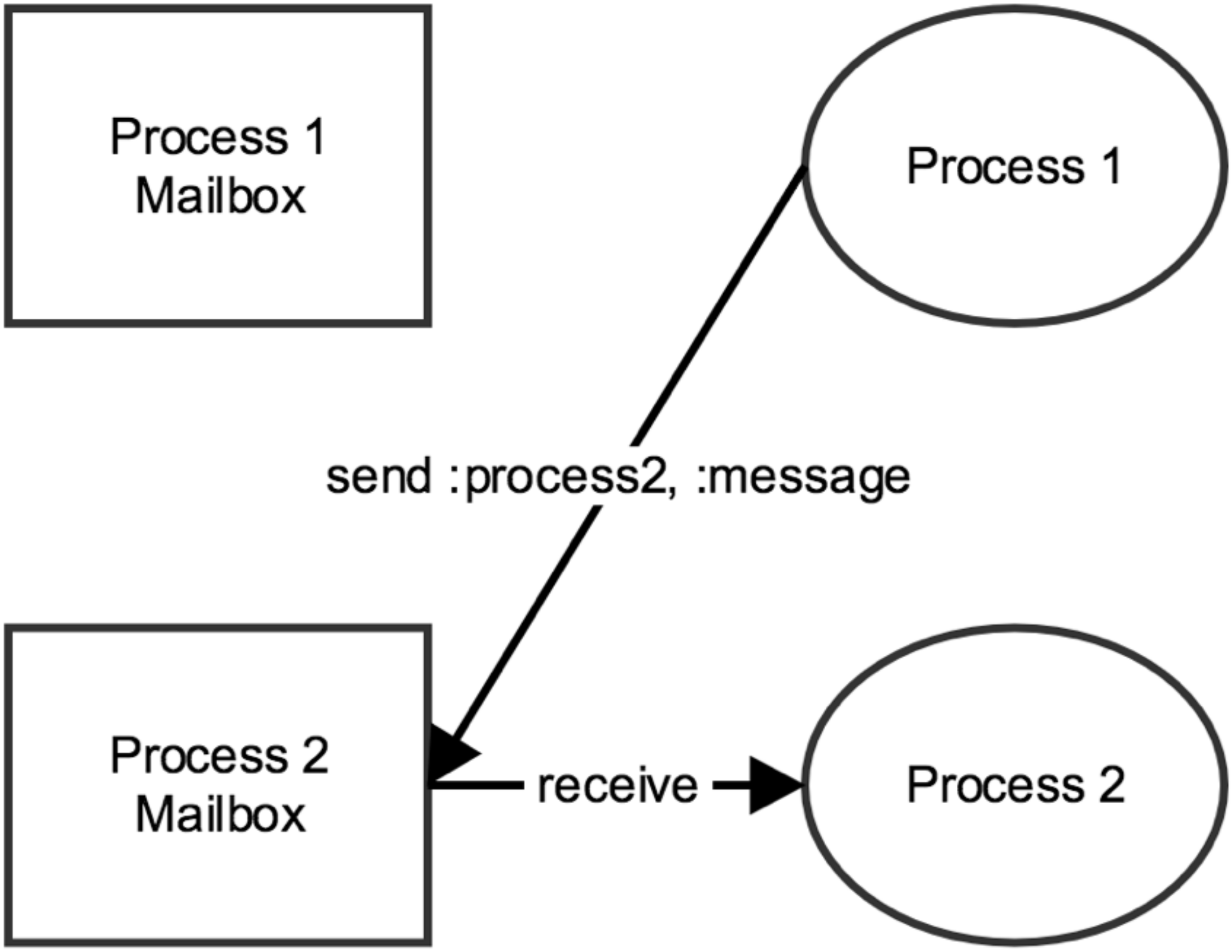
- Angular 2 Client app
- Phoenix backend app
  - REST JSON API
  - Channels (realtime websocket updates)

Processes in Elixir

- extremely lightweight
- share nothing
- can maintain state
- communicate via message passing

Process primitives

- spawn
- send
- receive


Process 1
Mailbox
Process 1
send :process2, :message
Process 2
Mailbox
receive
Process 2

Meet OTP

- Open Telecom Platform
- Patterns, libraries, and tools for managing large distributed systems
- Really Huge

OTP Behaviors

- gen_server
- supervisor
- application
- gen_event
- gen_fsm

GenServer

- Generic Server
- handle sync and async messages
- maintain state
- optionally implement terminate callback

AuctionServer

Let it crash

Supervisor

- Manage a “supervision tree”
- worker
  - module(s)
  - function (defaults to start_link)
  - restart

supervise options

- strategy
  - one_for_one (default)
  - one_for_all
  - one_for_rest
  - simple_one_for_one
- max_restarts
- max_seconds

OTP Application

- Service component
- Process(es) that can be started and stopped together
- Lets you share a supervision tree

AuctionServer Application

So much stuff I didn't get to

- OTP Releases
- deployment
- Hot code deploys
- exrm

Who’s using this stuff

- Pinterest
- Bleacher Report
- WhatsApp
- FanDuel
- You!