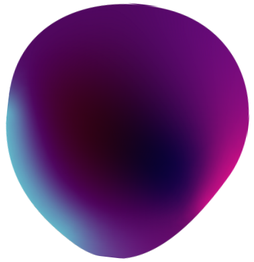
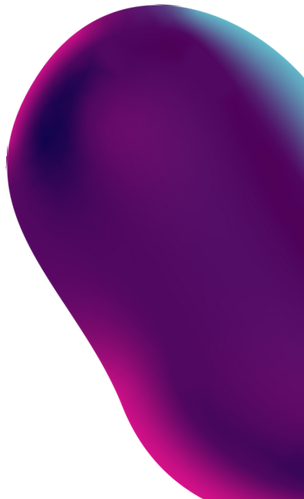
Alexander Sysoev |> 2020





Distributed Elixir with Horde

and tears :(

Hi folks, I am Alex (Saña)

I am a developer at Let's Talk. We provide digital attendance platform for big companies. And a part of this platform is now running on Elixir.

I have 9 year experience working on ruby and almost 4 years now I worked with Elixir and during this time became a big fan of this language.

Distributed systems

Distributed systems also known as distributed computing and distributed databases, a distributed system is a collection of independent components located on different machines that share messages with each other in order to achieve common goals.

As such, the distributed system will appear as if it is one interface or computer to the end-user. The hope is that together, the system can maximize resources and information while preventing failures, as if one system fails, it won't affect the availability of the service.

δ-CRDT

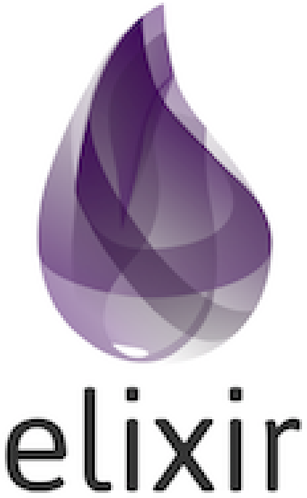
A CRDT is a conflict-free replicated data-type. That is to say, it is a distributed data structure that automatically resolves conflicts in a way that is consistent across all replicas of the data. In other words, your distributed data is guaranteed to eventually converge globally.

Delta means that only state's delta is transmitted between nodes. Thanks for anti-entropy algorithm from this research

Distributed Erlang/Elixir

Node

A distributed Erlang system consists of a number of Erlang runtime systems communicating with each other. Each such runtime system is called a node.



Demos

Simple Counter

In this demo we build elixir application which counts how many times particular user entered to the url. We will use Plug, GenServer, DynamicSupervisor and Registry.

No cluster, no nodes, It's a deal, it's a steal.

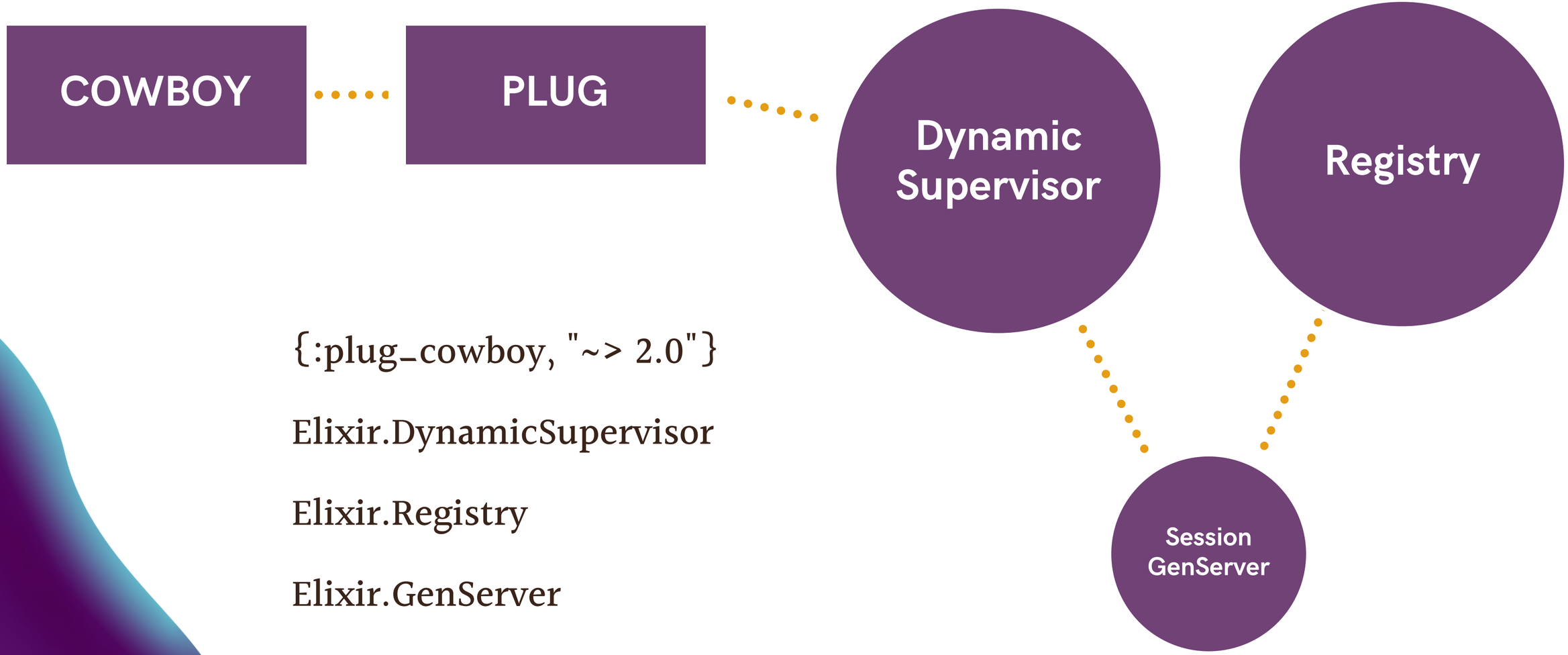
Distributed Counter

Here we will distribute our sessions across the cluster of 3 nodes.

With hand-off

What's about the state. We will never? loose the state again implementing a hand-off strategy.

Simple counter




```
{:plug_cowboy, "~> 2.0"}
```

```
Elixir.DynamicSupervisor
```

```
Elixir.Registry
```

```
Elixir.GenServer
```

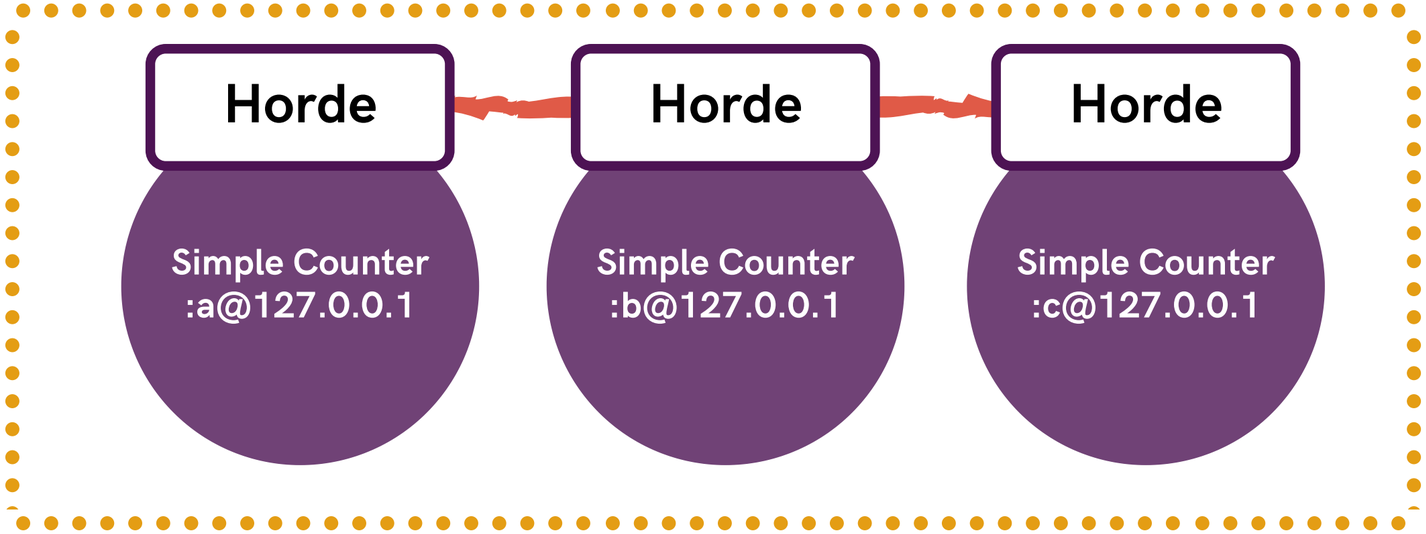
Clustered Counter

Adding {:libcluster, "~> 3.1"} and defining a simple topology's strategy we now automatically form a cluster but eventually sessions are not aware about each other across the nodes.

COWBOY_PORT=4005 iex --name a@127.0.0.1 -S mix

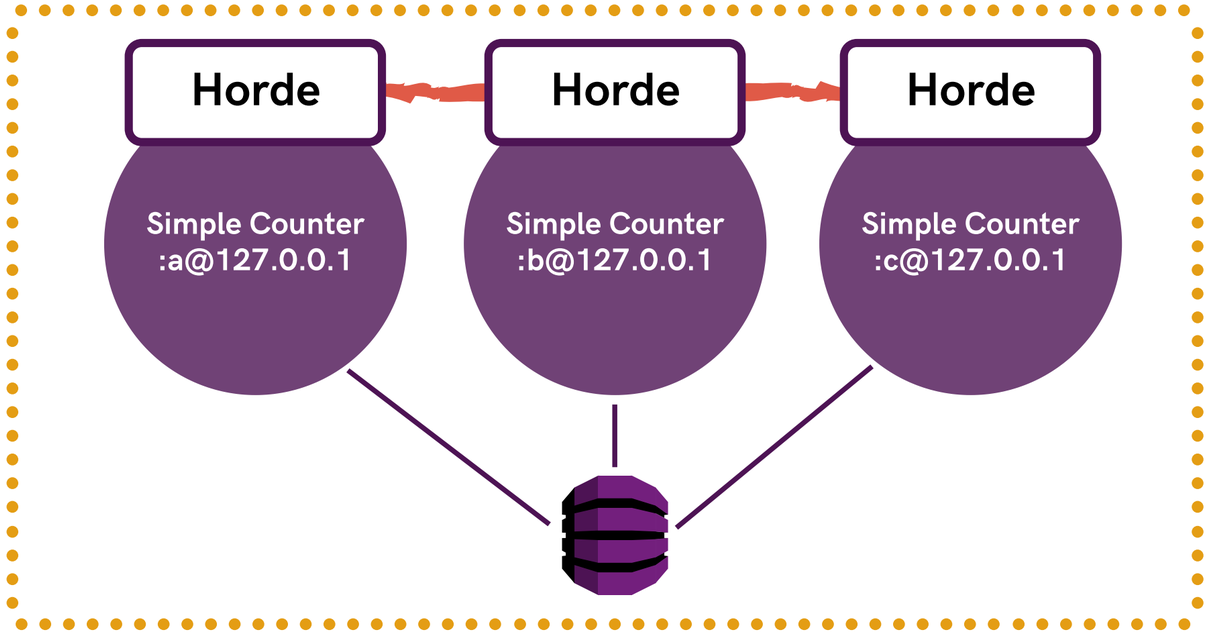
Distributed Counter

With {:horde, "~> 0.7.0"} we connect nodes under a Horde cluster and get distributed supervisor and global distributed registry almost for free. Under the hood Horde uses concepts from Delta CRDTs, and relies on MerkleMap for efficient synchronization.



Counter with Hand-off

By adding {:ecto_sql, "~> 3.0"} and {:postgrex, ">= 0.0.0"} we connect the app to database that will keep session's state between death and resurrection and never loose it again.



Tears

New class of problems - race conditions.

Latency on process registration you are speaking about micro seconds (one must pay attention).

Timeout on set_members by supervisor on setting cluster members (overload).

Architecture paradigm change.

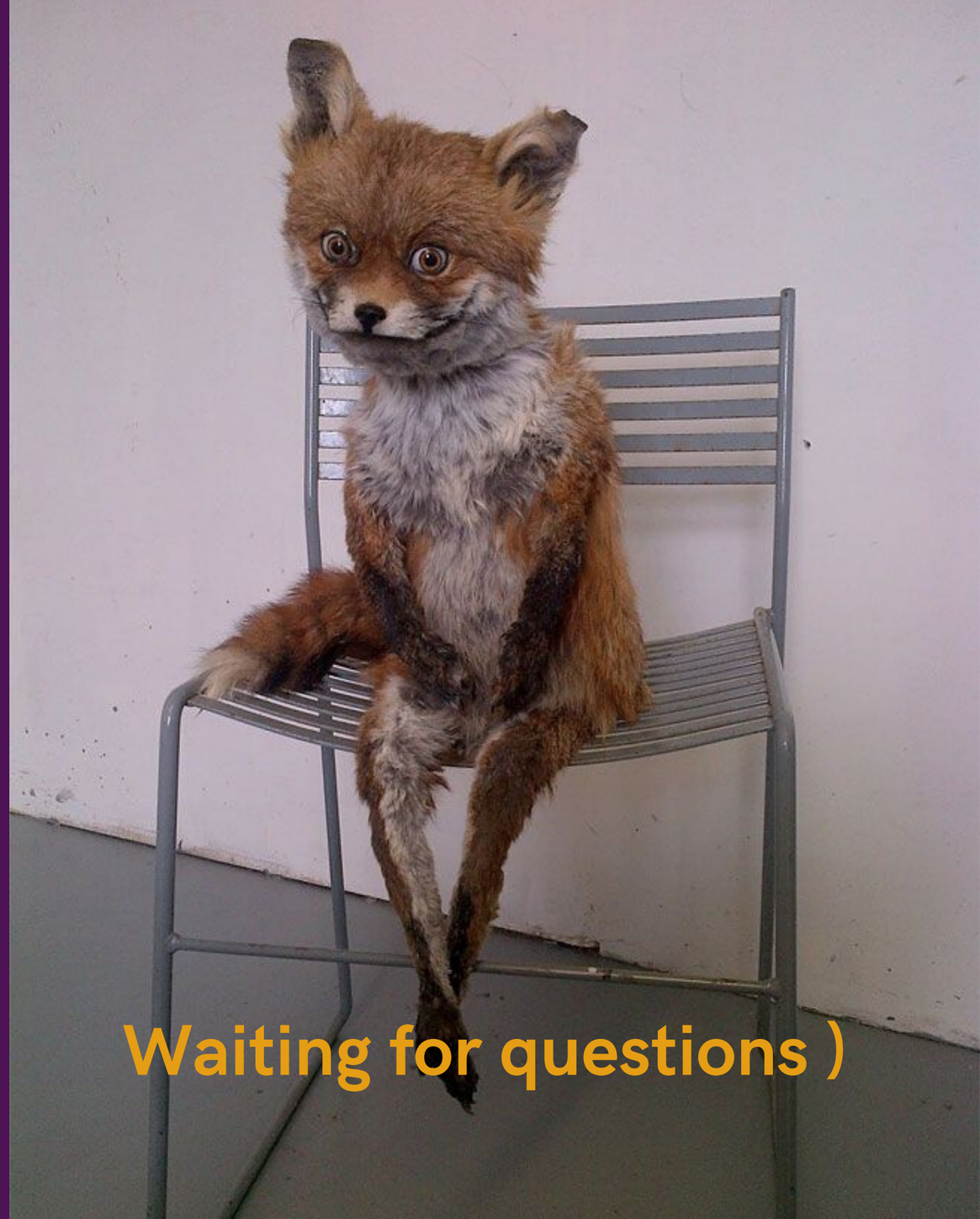
Deployment paradigm change.

Thank ya :)

email: alexander@letsta.lk

github: https://github.com/alex-sysoev

demo: https://github.com/alex-sysoev/distributed_elixir_demo



Waiting for questions)