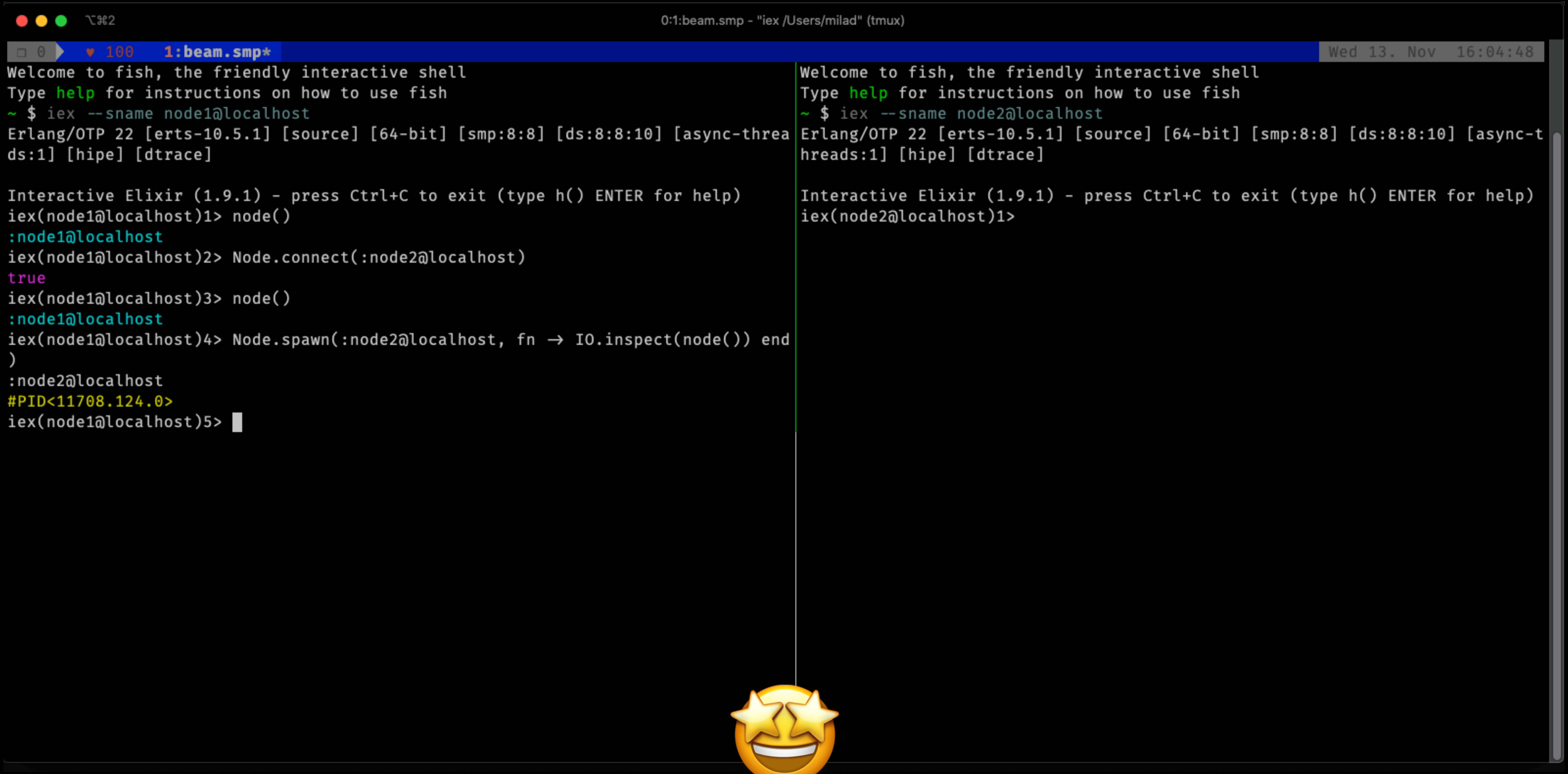
Patterns of Distributed Programming in BEAM

Milad(@slashmili)

❤️ Why do we love Elixir/BEAM ❤️

- Eloquent & Expressive
- Metaprogramming
- Concurrency
- Mature Environment
- Fault-tolerant
- Pattern Matching
- …

Distributed



```
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
~ $ iex --sname node1@localhost
Erlang/OTP 22 [erts-10.5.1] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threa
ds:1] [hipe] [dtrace]

Interactive Elixir (1.9.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(node1@localhost)1> node()
:node1@localhost
iex(node1@localhost)2> Node.connect(:node2@localhost)
true
iex(node1@localhost)3> node()
:node1@localhost
iex(node1@localhost)4> Node.spawn(:node2@localhost, fn -> IO.inspect(node()) end
)
:node2@localhost
#PID<11708.124.0>
iex(node1@localhost)5>
```

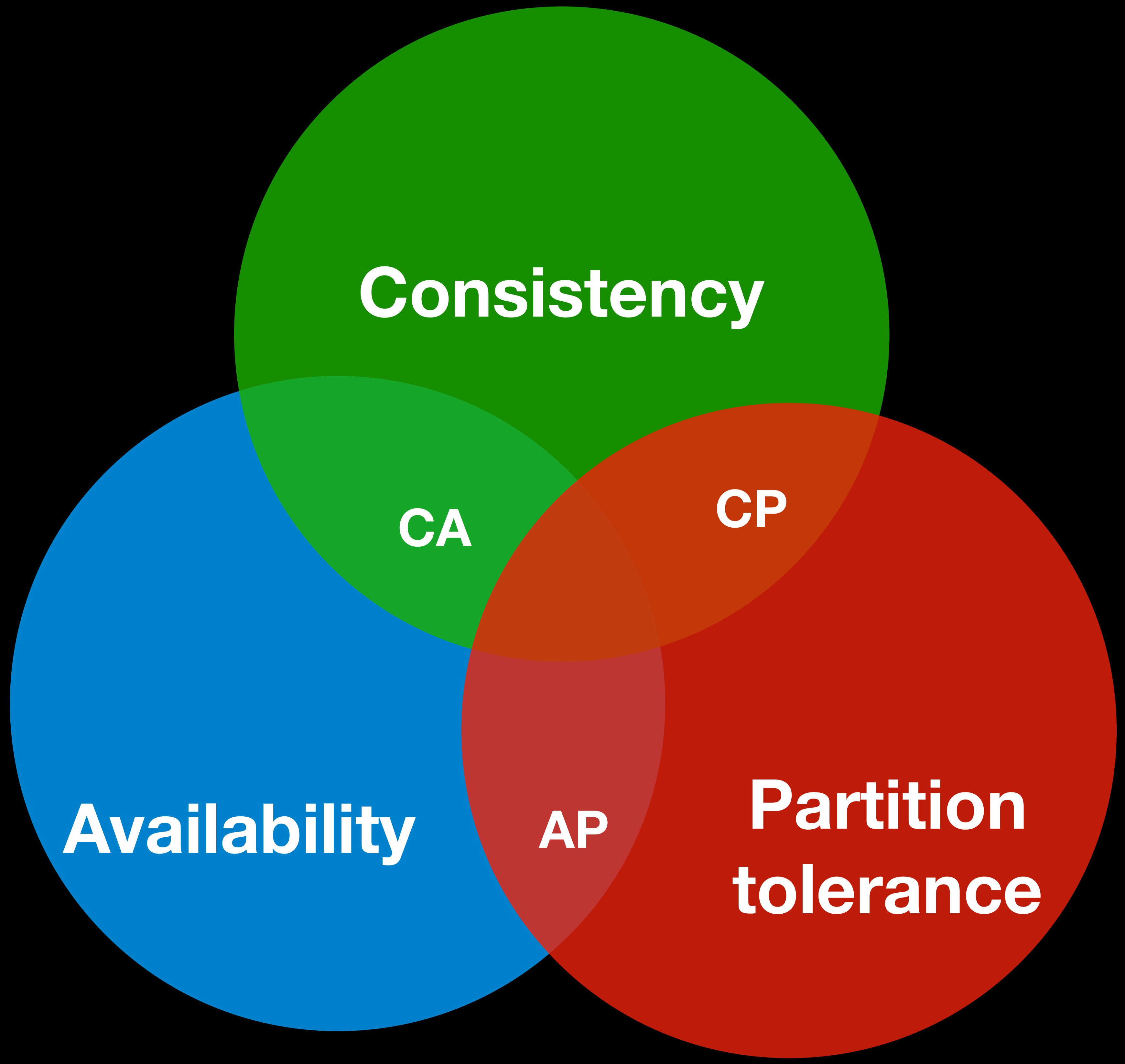
```
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
~ $ iex --sname node2@localhost
Erlang/OTP 22 [erts-10.5.1] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-t
hreads:1] [hipe] [dtrace]

Interactive Elixir (1.9.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(node2@localhost)1>
```

Distributed Programming

😰

📖 CAP theorem 📖



Topics

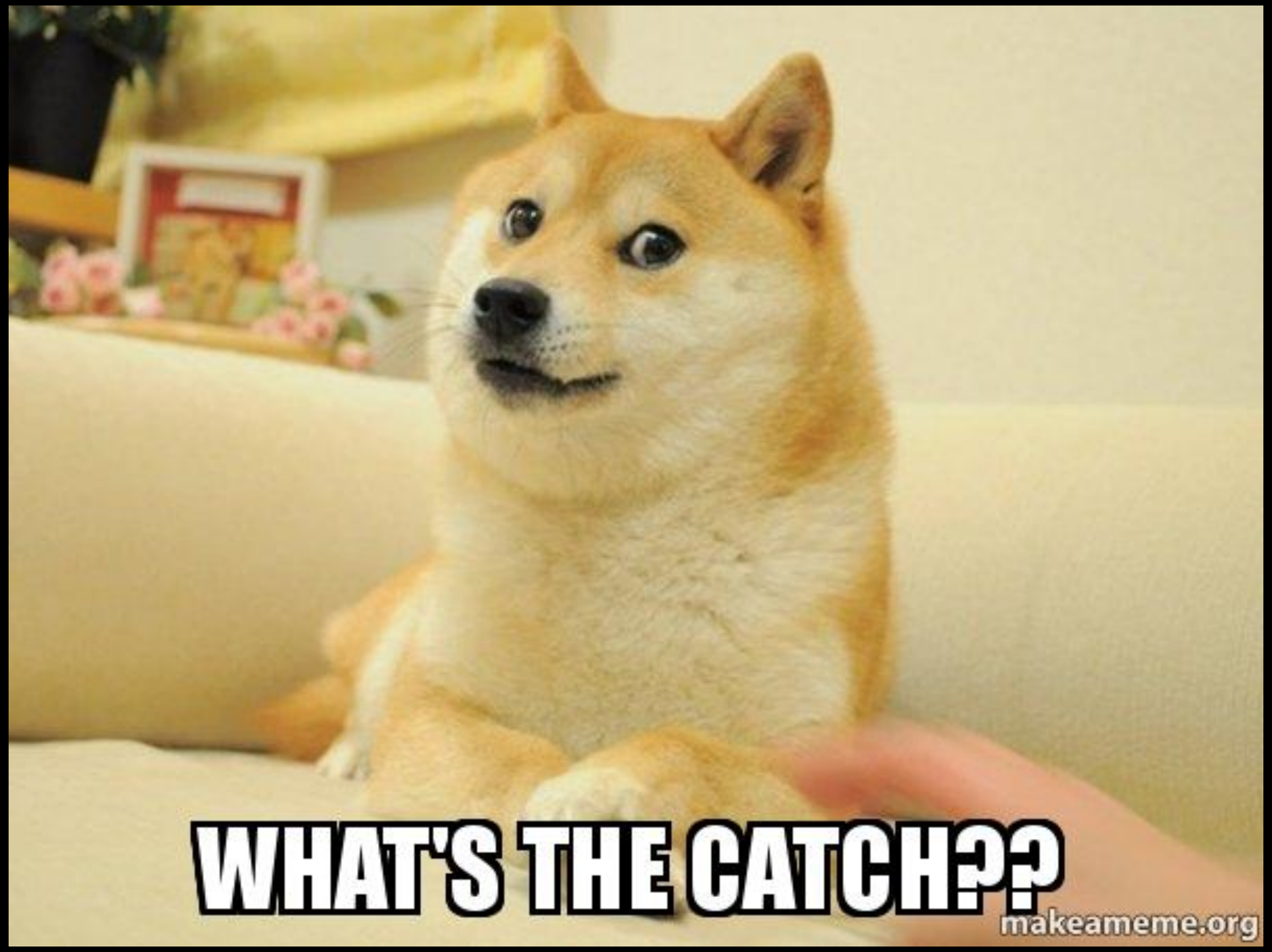
- Connecting Erlang Node
- Caching in Distributed nodes
- Process Discovery

Connecting Nodes

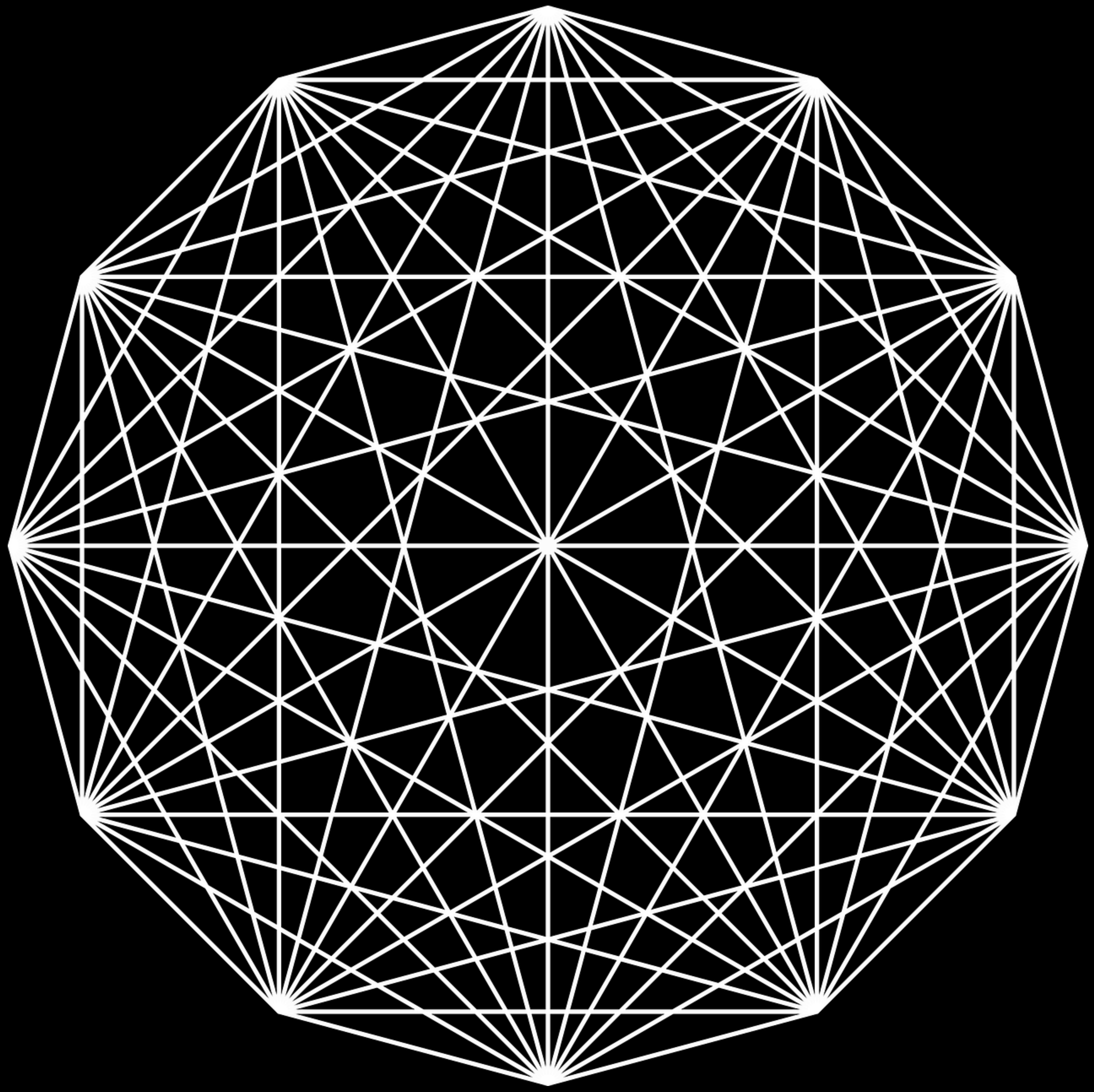
libcluster

- Cluster.Strategy.Epmd
- Cluster.Strategy.ErlangHosts
- Cluster.Strategy.Gossip
- Cluster.Strategy.Kubernetes
- Cluster.Strategy.Kubernetes.DNS
- Cluster.Strategy.Rancher

libcluster



Distributed Erlang Nodes



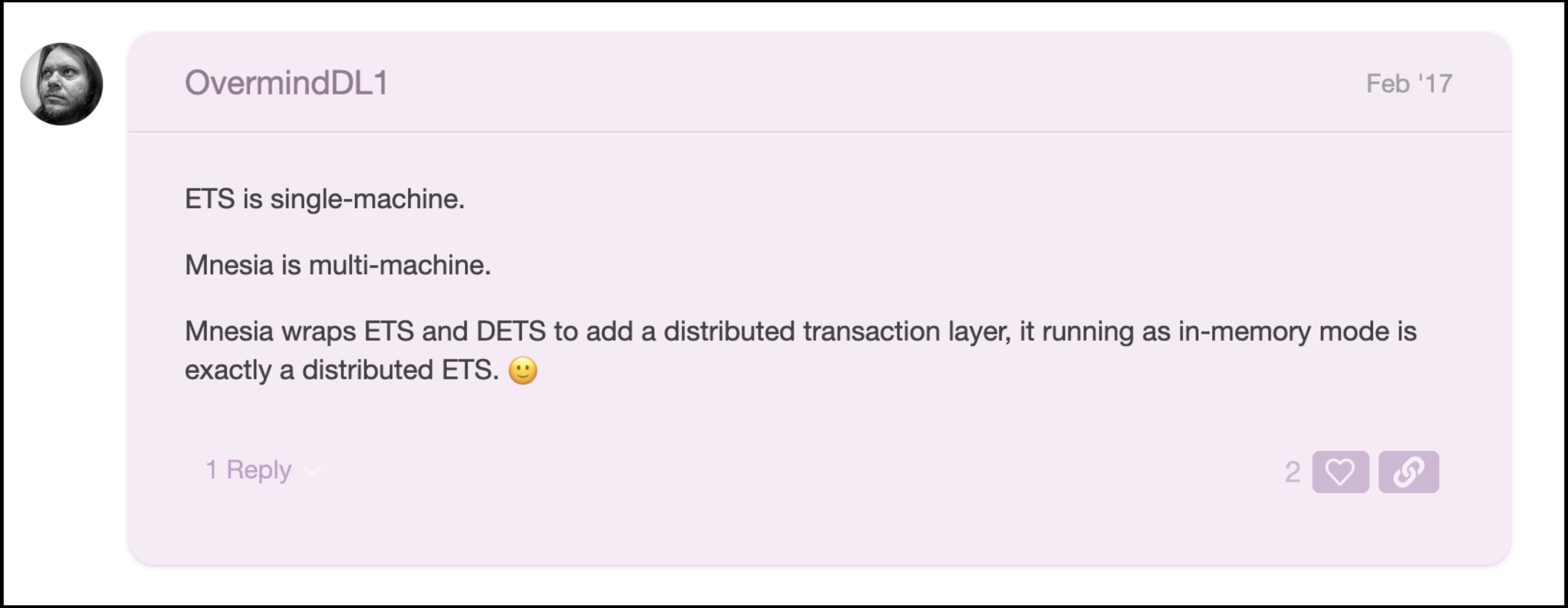
nkcluster

- A framework to manage jobs at huge Erlang clusters
- uses riak_core underneath
- Supports running worker nodes in WAN (using TCP or Websocket)
- Last commit in 2016!

Caching in Distributed nodes

MNESIA

CA



OvermindDL1 Feb '17

ETS is single-machine.

Mnesia is multi-machine.

Mnesia wraps ETS and DETS to add a distributed transaction layer, it running as in-memory mode is exactly a distributed ETS. 🙂

1 Reply 2

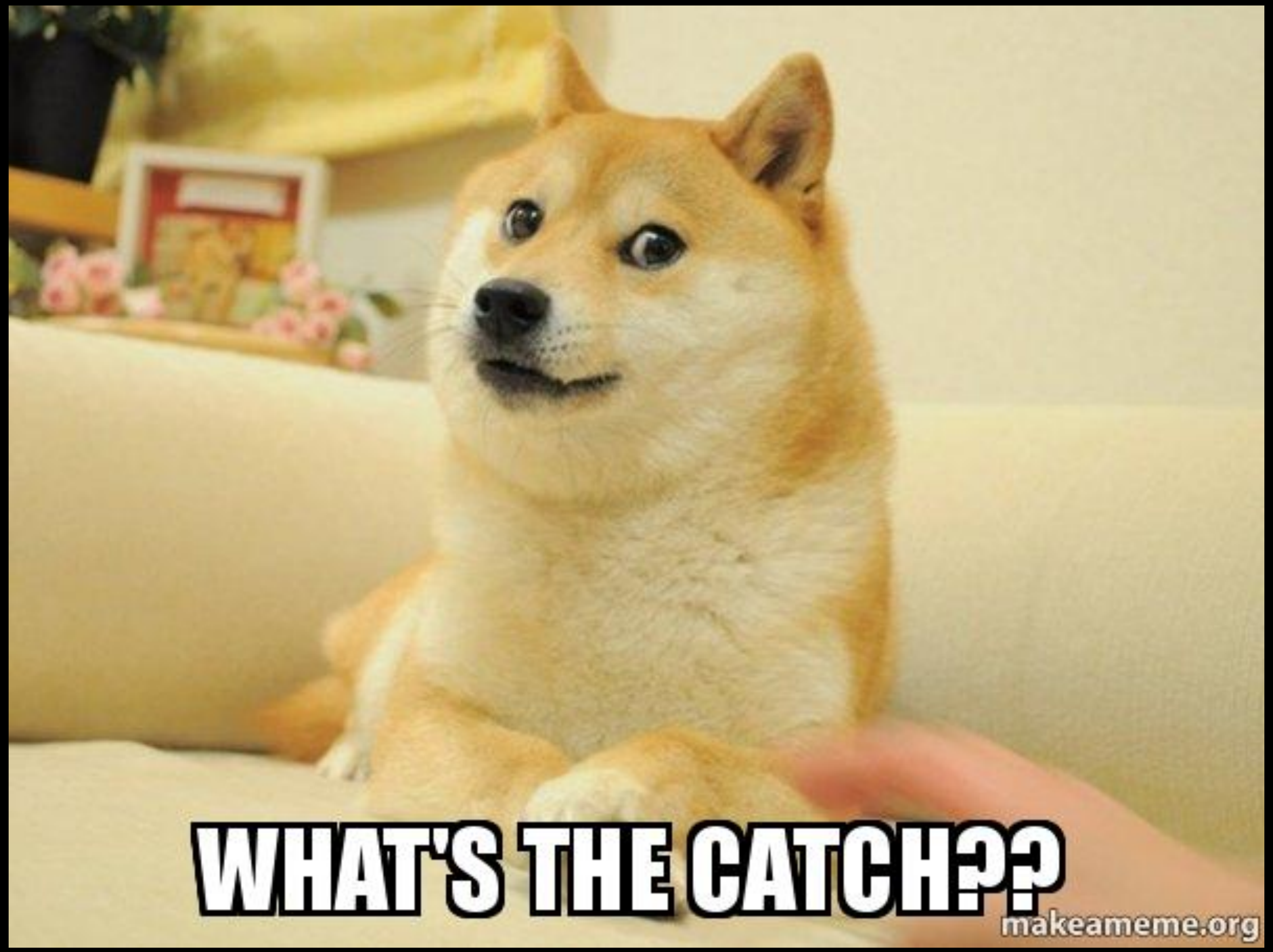
MNESIA

CA

- A relational/object hybrid data model that is suitable for telecommunications applications.
- A DBMS query language, Query List Comprehension (QLC) as an add-on library.
- Persistence. Tables can be coherently kept on disc and in the main memory.
- Replication. Tables can be replicated at several nodes.
- Atomic transactions. A series of table manipulation operations can be grouped into a single atomic transaction.

CA

MNESIA



MNESIA

CA

- Writes are expensive
- Prone to Split-Brain

MNESIA

CA

https://elixirschool.com/en/lessons/specifics/mnesia/

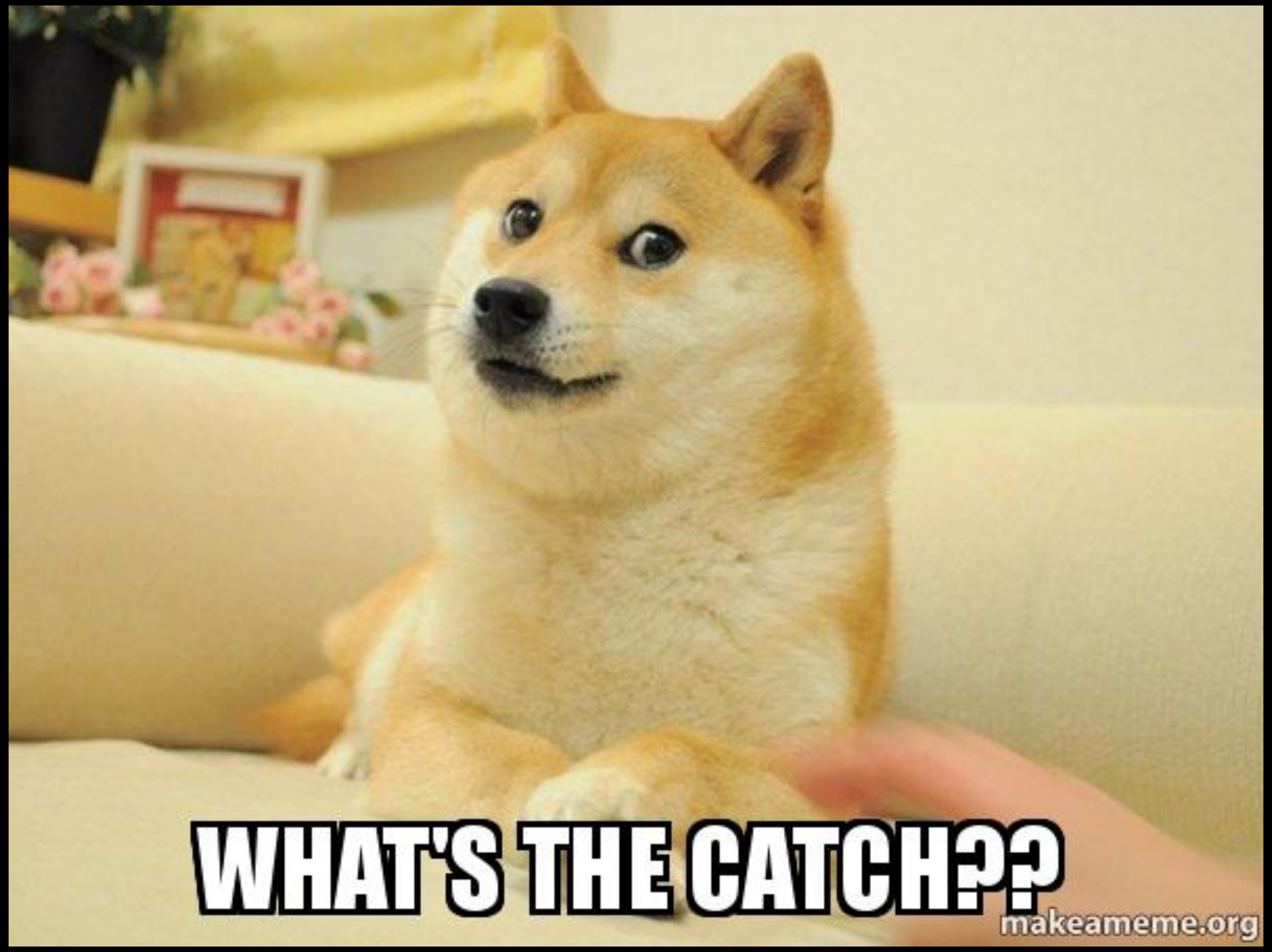
DeltaCrdt

AP

- Key/Value store
- Uses Conflict-free replicated data type

DeltaCrdt

AP



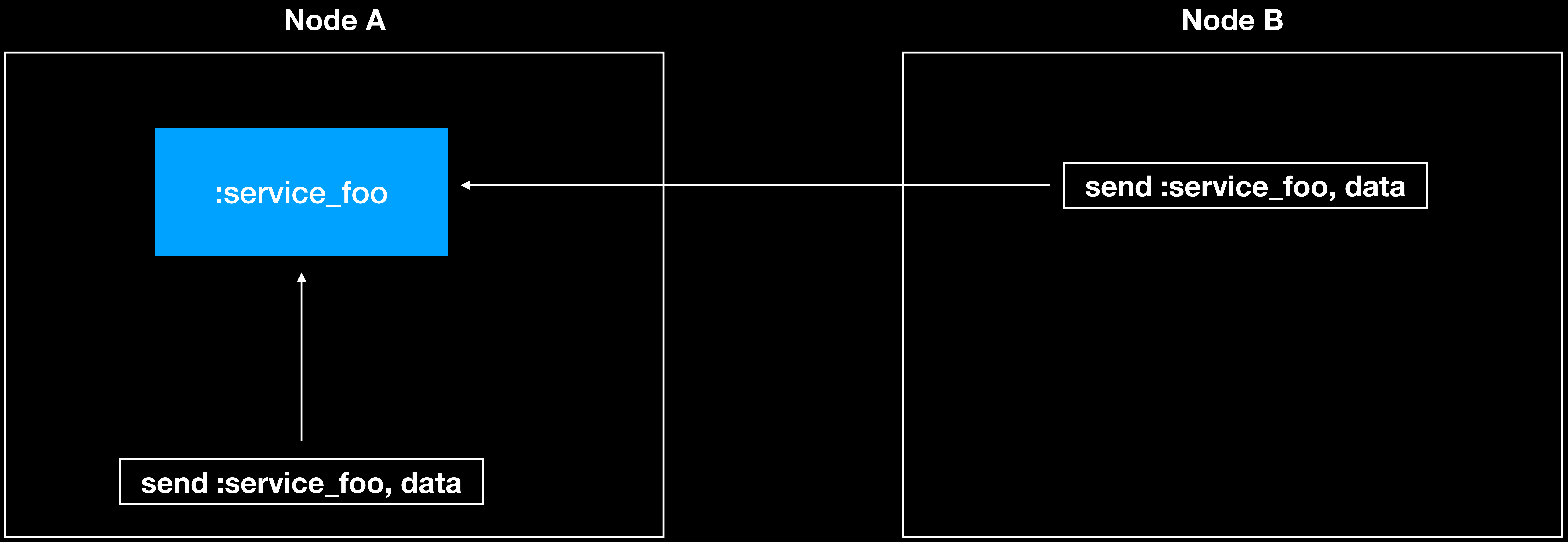
DeltaCrdt

AP

- Data might not be consistence(they will be though!)
- Setting neighbours is done by you

Process Discovery

Single Service



Single Service

- global (http://erlang.org/doc/man/global.html)
- swarm (https://github.com/bitwalker/swarm)
- horde (https://github.com/derekkraan/horde)

Single Service global

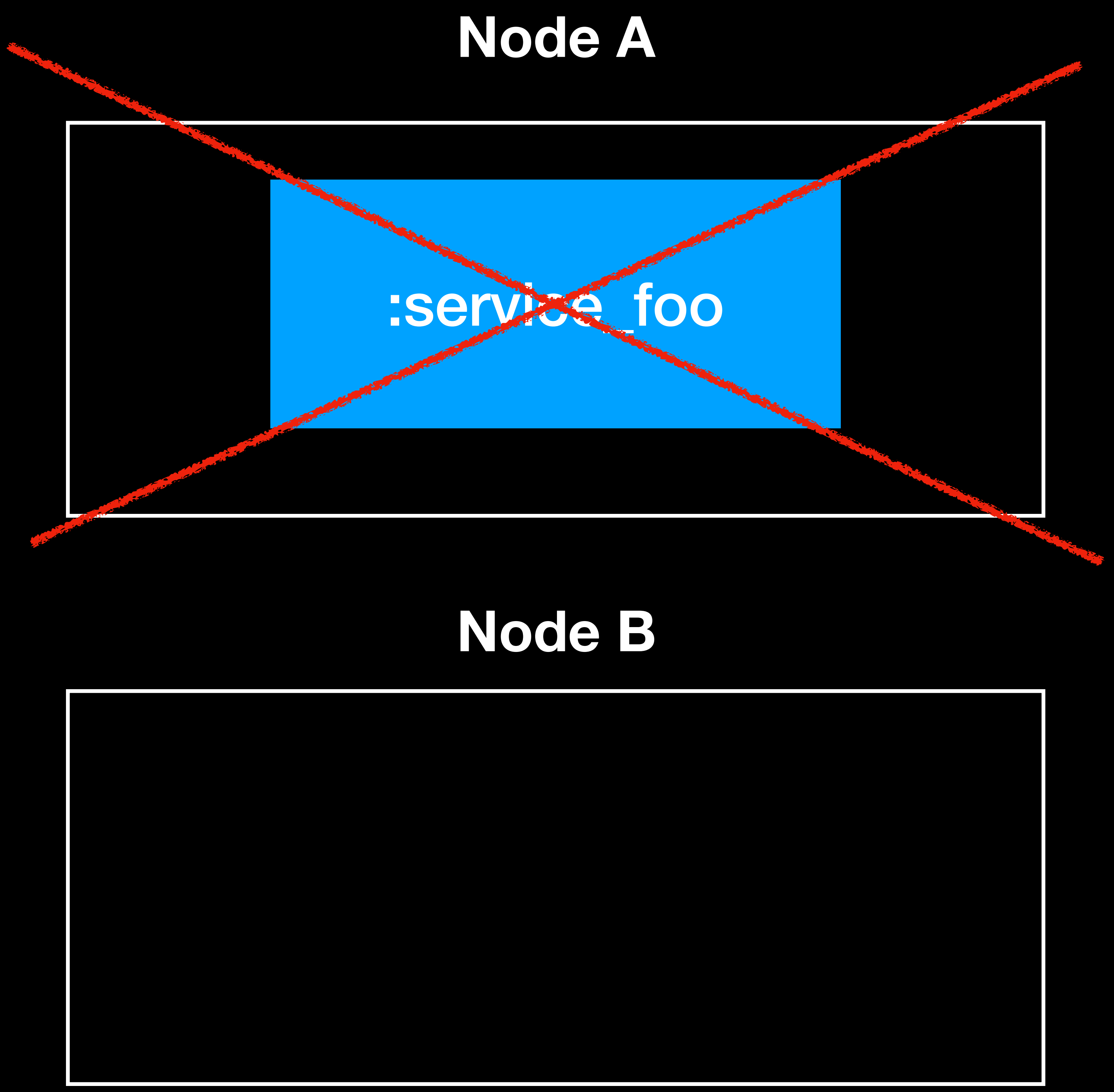
CA

- Built in Erlang
- Registration of global names
- Global locks

Single Service
global

CA

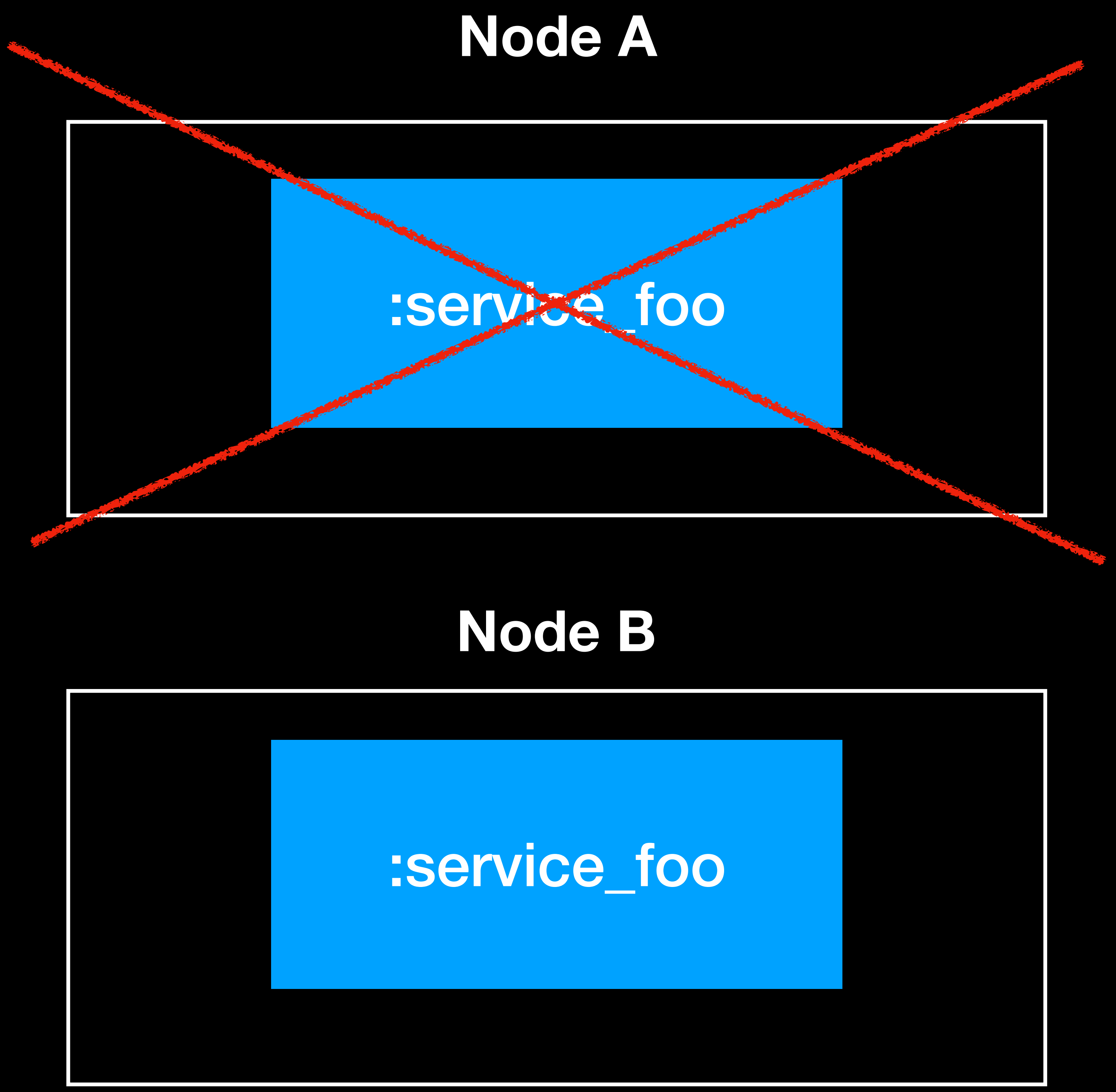
- Built in Erlang
- Registration of global names
- Global locks



Single Service
swarm

AP CP

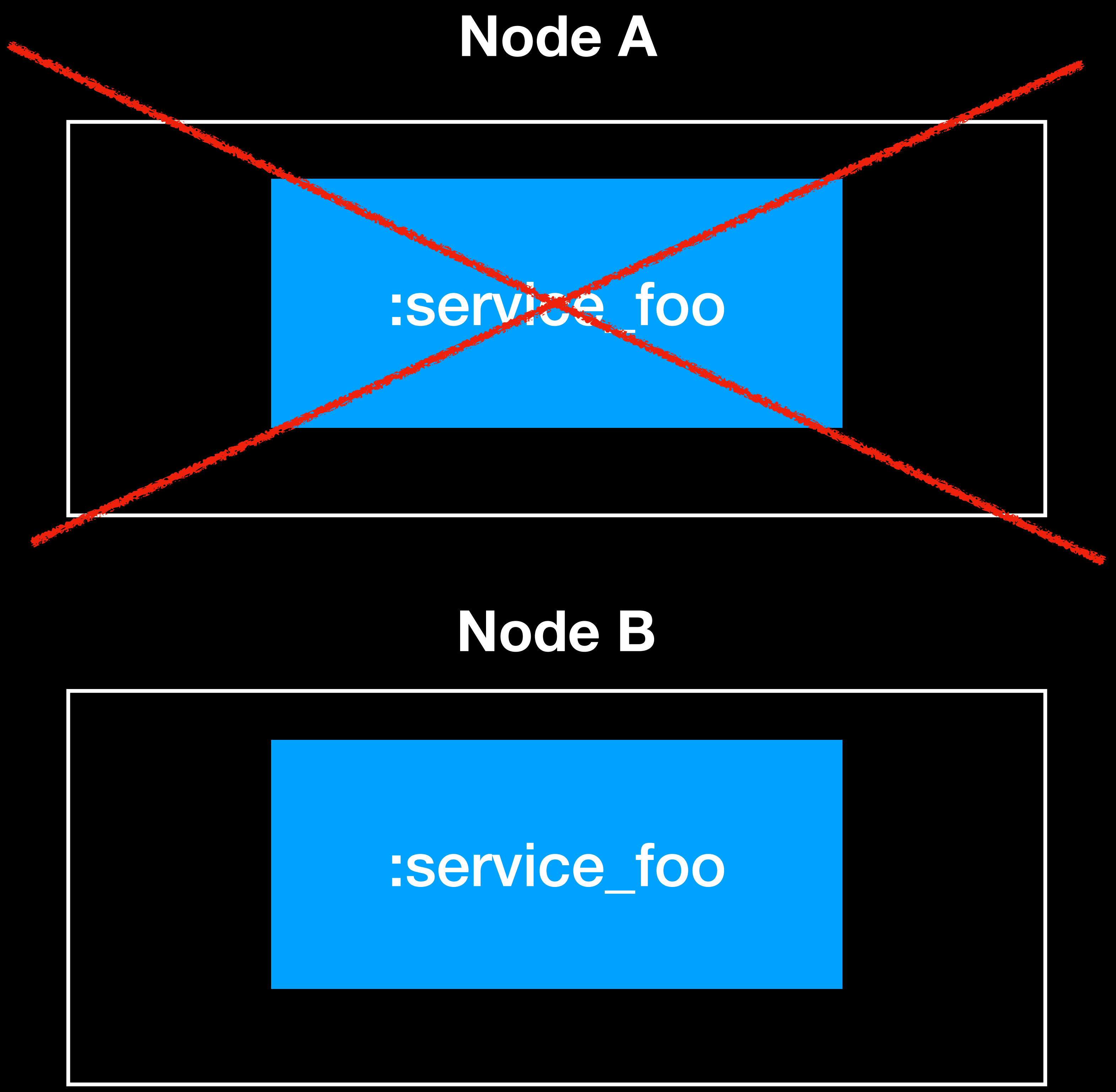
- Configure able to choose
  - Swarm.Distribution.Ring (AP)
  - Swarm.Distribution.StaticQuorumRing (CP)
- maintained by bitwalker
- Last release on Jan 2019



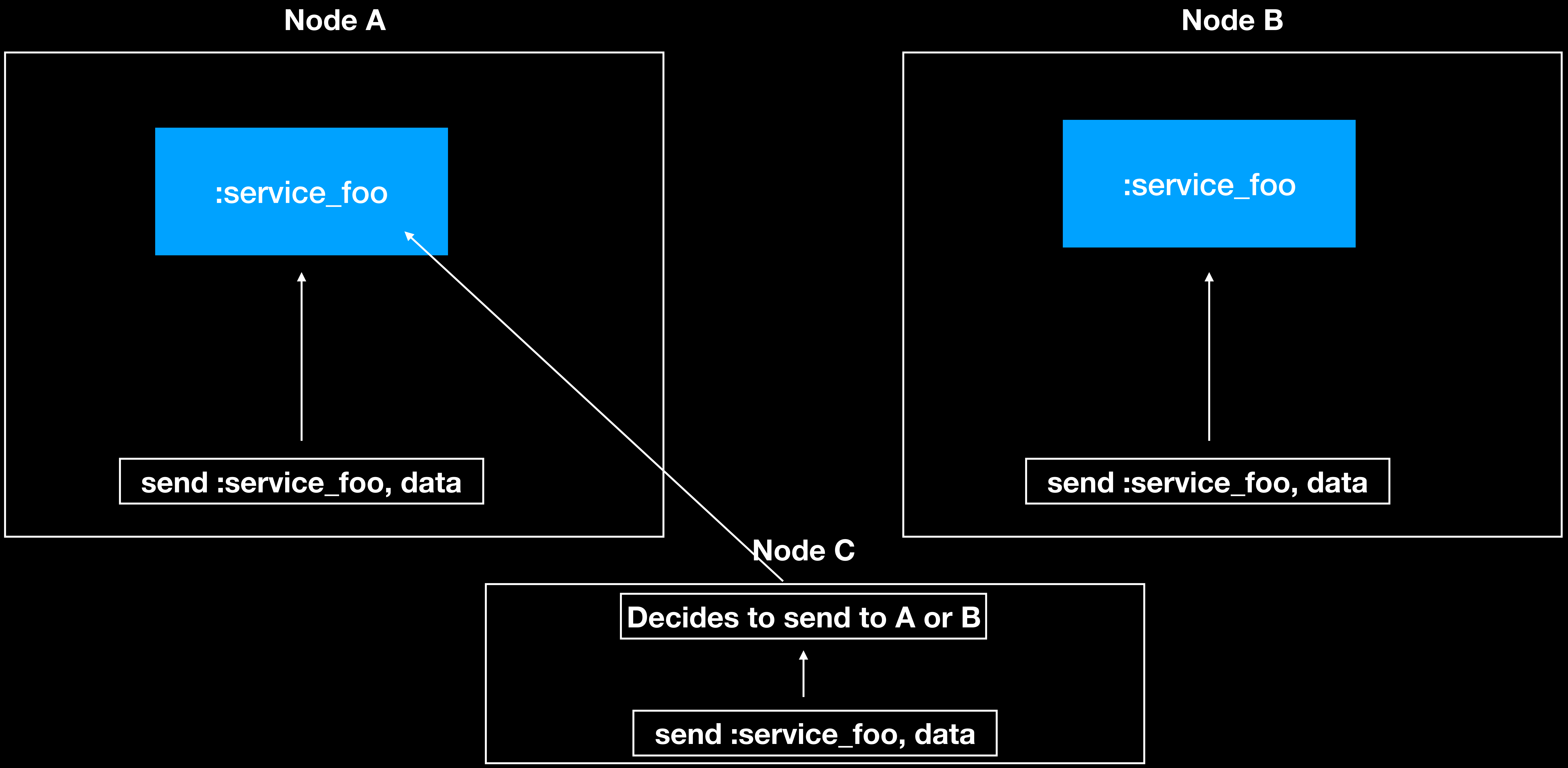
Single Service
horde

AP

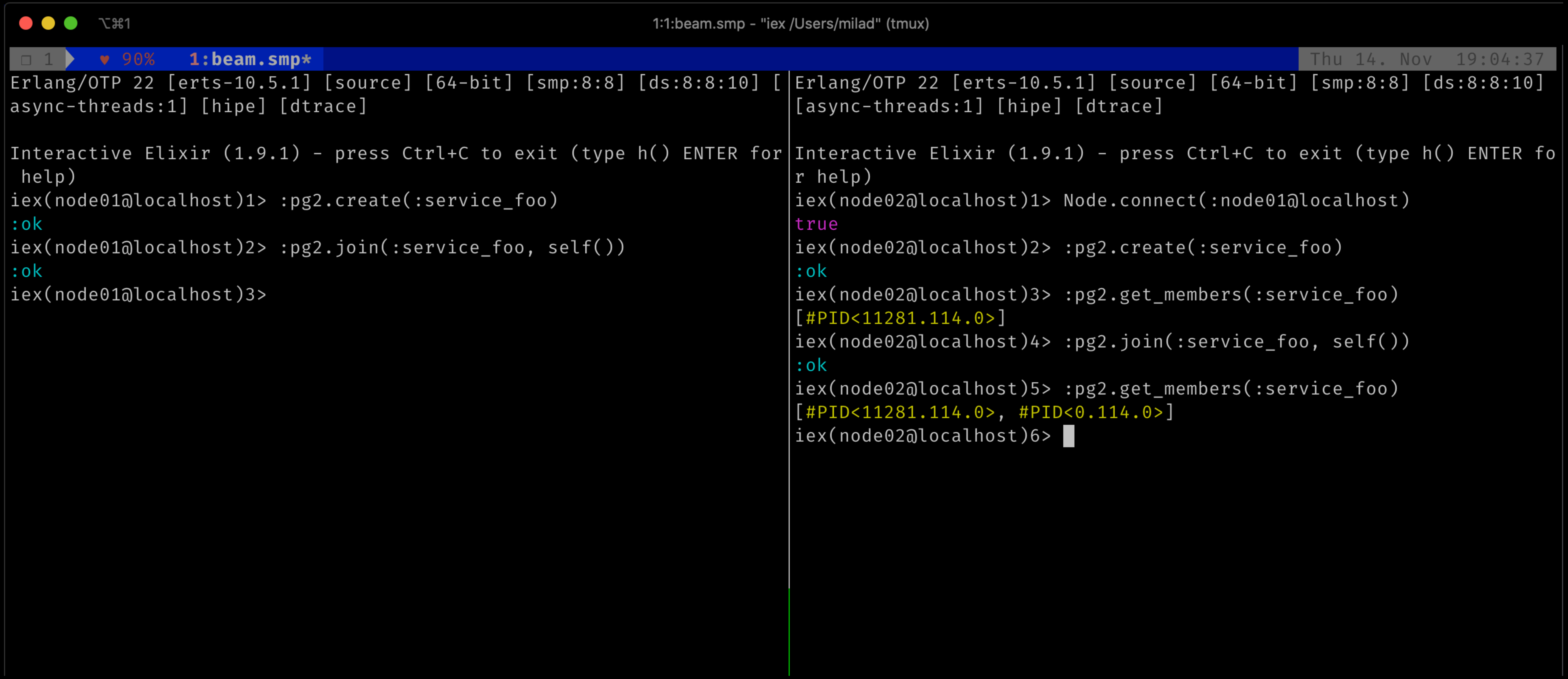
- guarantee availability and partition tolerancy. cannot guarantee consistency.
- has active community
- Current version 0.7.1 (close to 1.0 release)



Multiple Services



Multiple Services



```
Erlang/OTP 22 [erts-10.5.1] [source] [64-bit] [smp:8:8] [ds:8:8:10] [
async-threads:1] [hipe] [dtrace]

Interactive Elixir (1.9.1) - press Ctrl+C to exit (type h() ENTER for
 help)
iex(node01@localhost)1> :pg2.create(:service_foo)
:ok
iex(node01@localhost)2> :pg2.join(:service_foo, self())
:ok
iex(node01@localhost)3>
```

```
Erlang/OTP 22 [erts-10.5.1] [source] [64-bit] [smp:8:8] [ds:8:8:10]
[async-threads:1] [hipe] [dtrace]

Interactive Elixir (1.9.1) - press Ctrl+C to exit (type h() ENTER fo
r help)
iex(node02@localhost)1> Node.connect(:node01@localhost)
true
iex(node02@localhost)2> :pg2.create(:service_foo)
:ok
iex(node02@localhost)3> :pg2.get_members(:service_foo)
[#PID<11281.114.0>]
iex(node02@localhost)4> :pg2.join(:service_foo, self())
:ok
iex(node02@localhost)5> :pg2.get_members(:service_foo)
[#PID<11281.114.0>, #PID<0.114.0>]
iex(node02@localhost)6>
```

https://github.com/phoenixframework/phoenix_pubsub

Wrapping Up

- Distributed Erlang is cool 😎
- Don’t have to start with Distributed Erlang 💪
- It’s ok to use Redis/Postgres to keep your state 🙈

Thank You