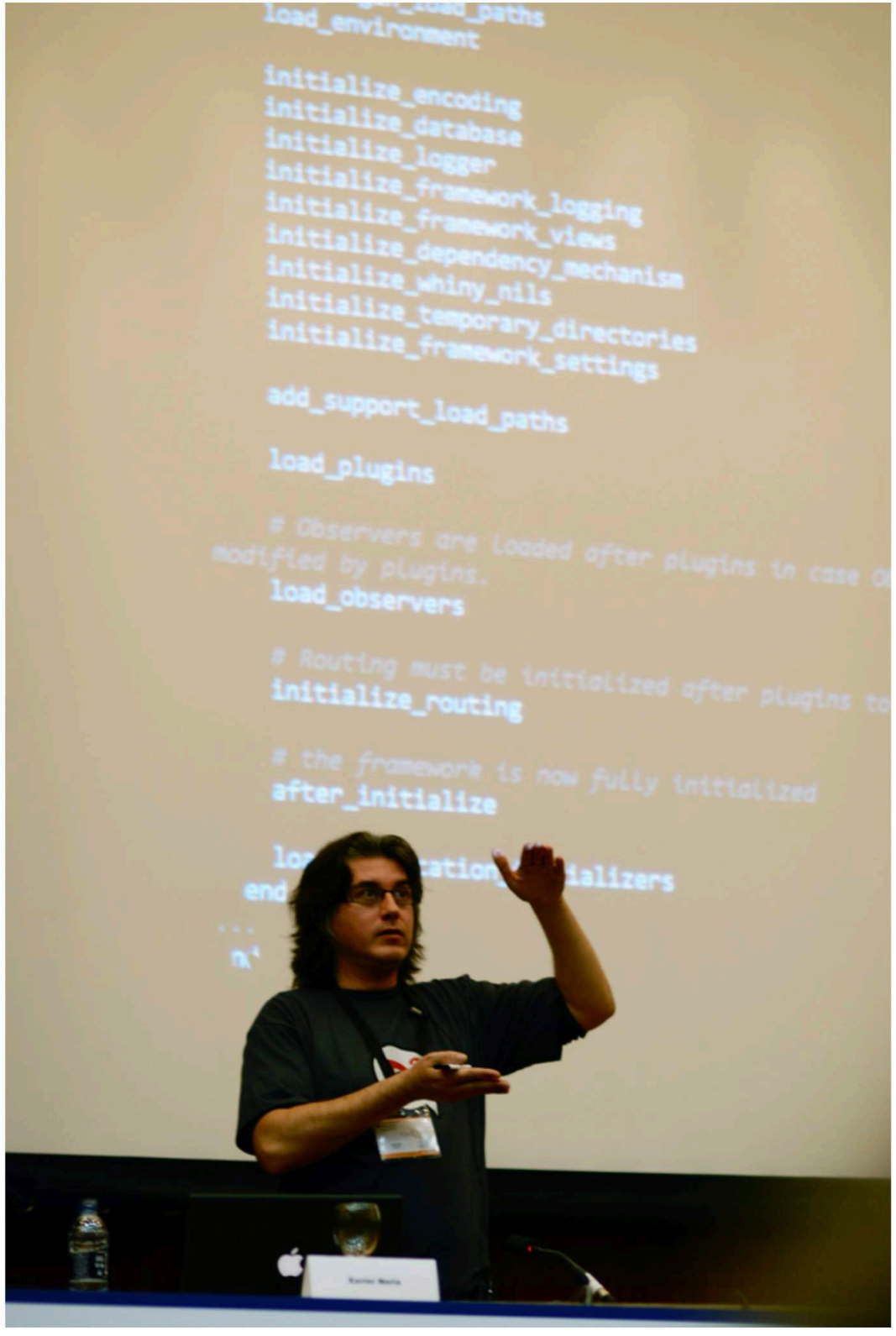
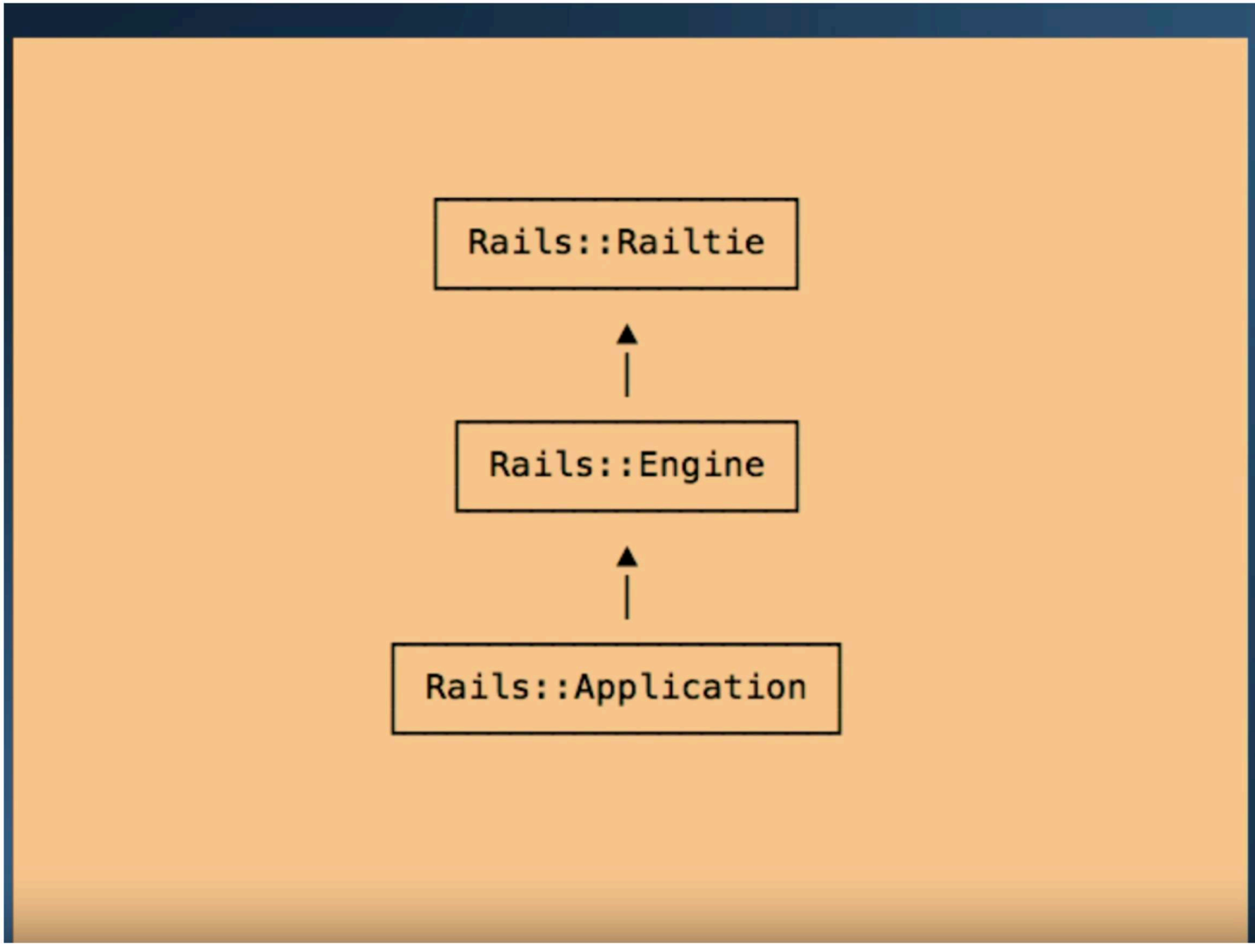
The Phoenix Boot Process

Xavier Noria
@fxn

ElixirConf EU 2018

A clas


load_environment
initialize_encoding
initialize_database
initialize_logger
initialize_framework_logging
initialize_framework_views
initialize_dependency_mechanism
initialize_whiny_nils
initialize_temporary_directories
initialize_framework_settings
add_support_load_paths
load_plugins
Observers are loaded after plugins in case
modified by plugins.
load_observers
Routing must be initialized after plugins
initialize_routing
the framework is now fully initialized
after_initialize


Rails::Railtie
Rails::Engine
Rails::Application


RC+KC
RAILSCONF | KANSAS CITY
RC+KC
RC
2016
2016
RC

Do it!

Talk Goal

```
mix phx.new chat
```

```
mix phx.server
```

Conceptual understanding

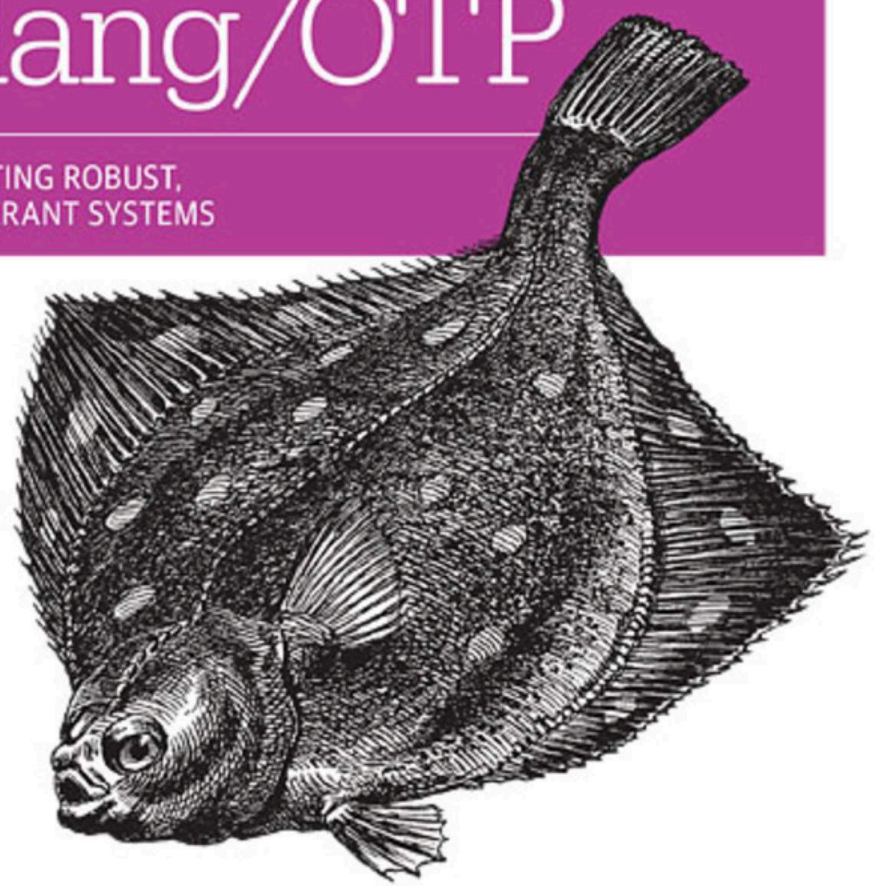
OTP Applications

OTP Design Principles User's Guide

O'REILLY®

Designing for Scalability with Erlang/OTP

IMPLEMENTING ROBUST, FAULT-TOLERANT SYSTEMS



Francesco Cesarini & Steve Vinoski

* Standardize **file structure**

* Standardize **configuration**

* Standardize **boot**

* Standardize **processes structure**

* Standardize **shutdown**

```
$ ls _build/prod/lib/chat/
consolidated
ebin
priv
```

Application resource file (Erlang notation):

```
  {application, Application,
    [{description,              Description},
     {id,                       Id},
     {vsn,                      Vsn},
     {modules,                  Modules},
     {maxT,                     MaxT},
     {registered,               Names},
     {included_applications,    Apps},
     {applications,             Apps},
     {env,                      Env},
     {mod,                      Start},
     {start_phases,             Phases},
     {runtime_dependencies,     RTDeps}]}.
```

```
{application,ecto,
  [{description,"A database wrapper ..."},
   {modules,['Elixir.Ecto.Multi',...]},
   {registered,[]},
   {vsn,"2.2.9"},
{applications,[kernel,stdlib,elixir,...]},
{env,[{json_library,'Elixir.Poison'},
      {postgres_map_type,<<"jsonb">>}]},
{mod,{'Elixir.Ecto.Application',[]}}]}.
```

```
{application, foo, []}.
```

application (Erlang)

Application (Elixir)

application_controller

```
Application.load(app):

  * Loads the application resource file,
    has to be somewhere in the code path

  * Merges environment variables from a
    config file given in -config, if any

    * Merges environment variables passed as
      -Application options, if any

    * Recurses over included applications
```

Loading an application is synchronous

Myth:

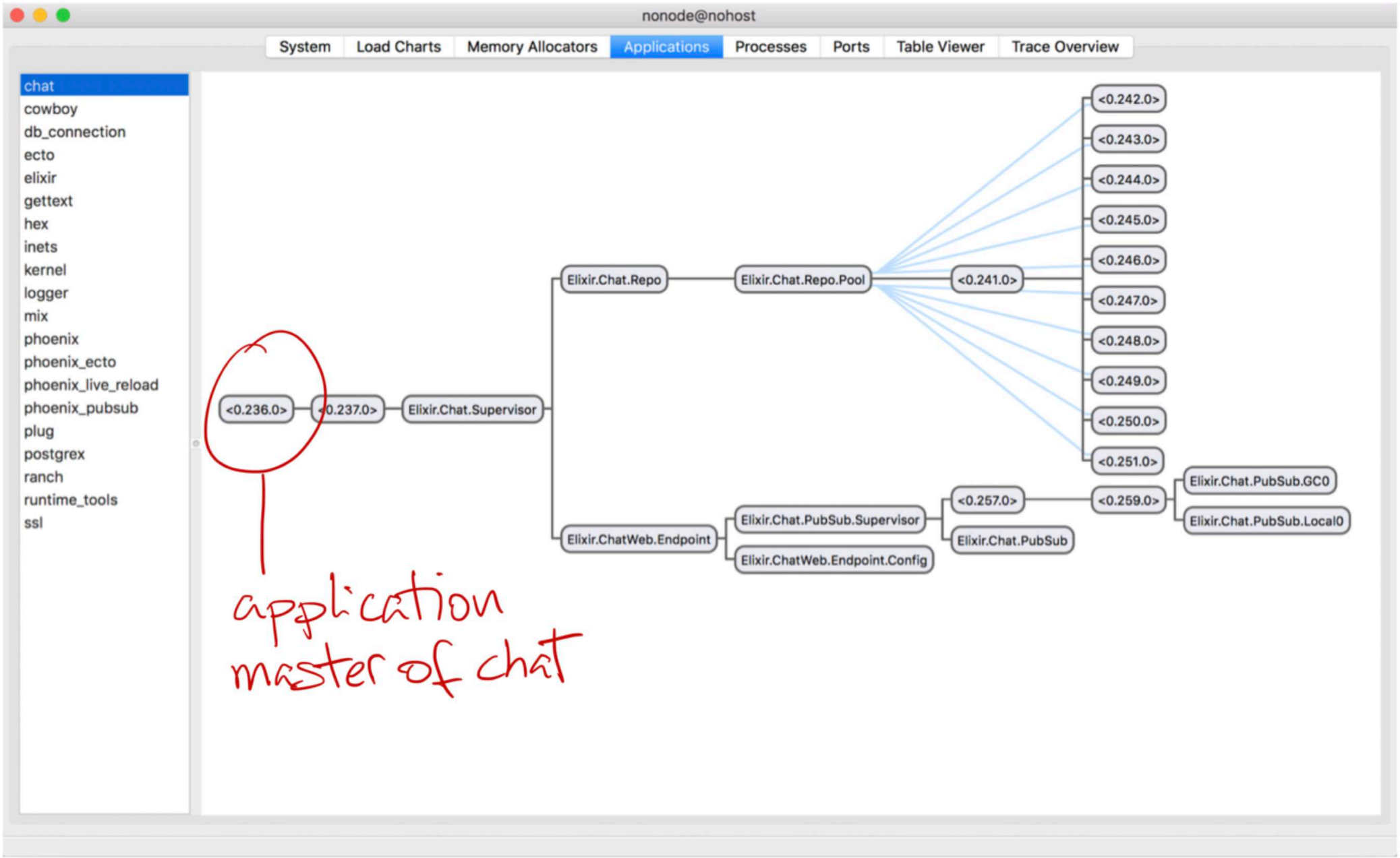
"*Loading an application loads its modules*"

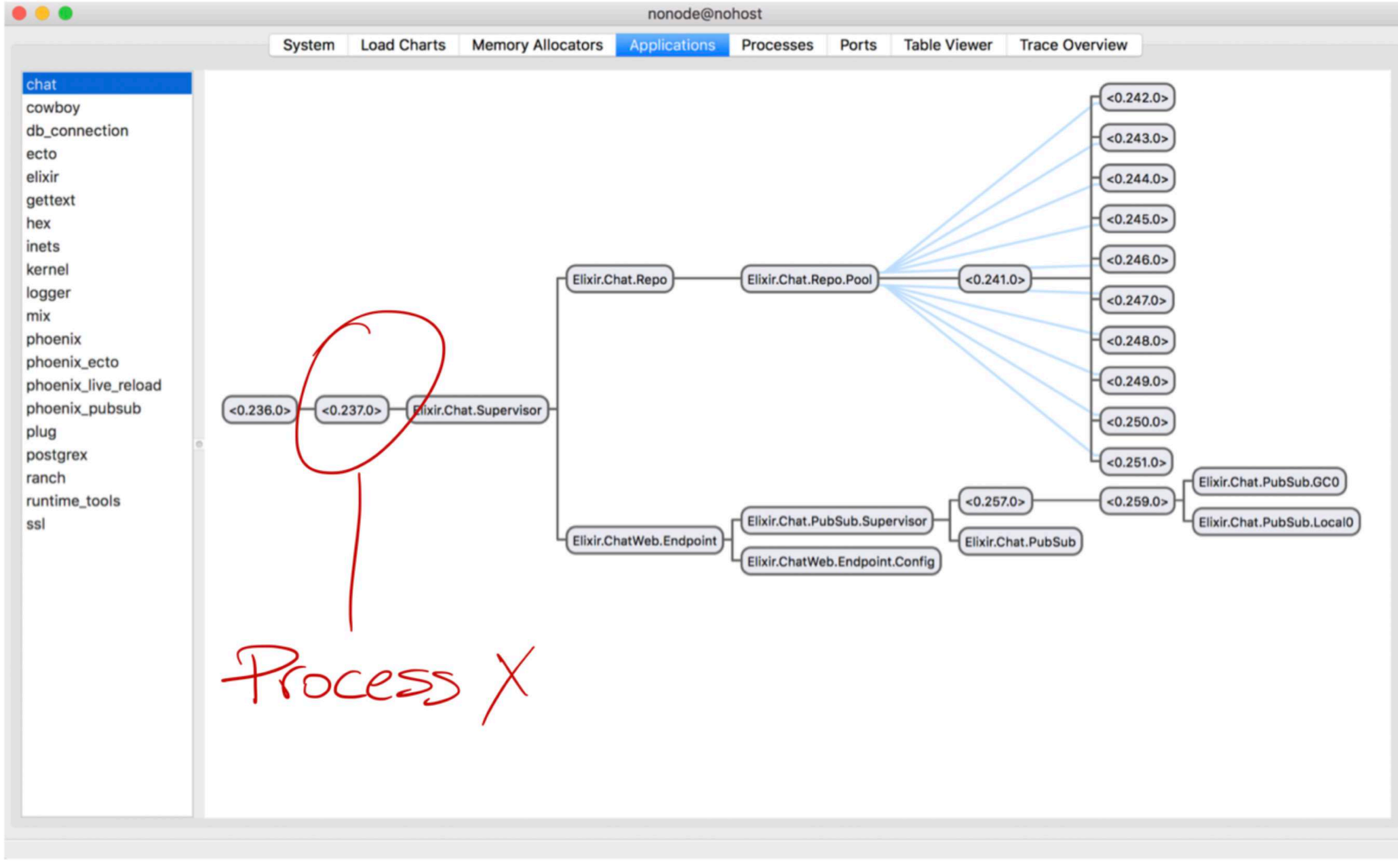
```
iex(1)> Application.load(:ex_unit)
:ok
iex(2)> :code.is_loaded(ExUnit)
false
```

```
Application.start(app, type \\ :temporary):

  * Loads the application if needed

  * Checks dependencies are started
    {:error,
      {:not_started, missing_dependency}}

  * Returns :ok if there is no application
    callback module (:mod key)

  * ...
```

```elixir
defmodule MyApplication do
  use Application

  def start(_type, args) do
    ...
    Supervisor.start_link(children, opts)
  end
end
```

* Creates an application master:
  + traps exits
  + becomes group leader
  + invokes start/2 in the callback module
  + stores the returned pid and state
  + enters a main loop


nonode@nohost
System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
chat
cowboy
db_connection
ecto
elixir
gettext
hex
inets
kernel
logger
mix
phoenix
phoenix_ecto
phoenix_live_reload
phoenix_pubsub
plug
postgrex
ranch
runtime_tools
ssl
<0.236.0>
<0.237.0>
Elixir.Chat.Supervisor
Elixir.Chat.Repo
Elixir.Chat.Repo.Pool
<0.241.0>
<0.242.0>
<0.243.0>
<0.244.0>
<0.245.0>
<0.246.0>
<0.247.0>
<0.248.0>
<0.249.0>
<0.250.0>
<0.251.0>
Elixir.ChatWeb.Endpoint
Elixir.Chat.PubSub.Supervisor
Elixir.ChatWeb.Endpoint.Config
<0.257.0>
Elixir.Chat.PubSub
<0.259.0>
Elixir.Chat.PubSub.GC0
Elixir.Chat.PubSub.Local0
application
master of chat


nonode@nohost
System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
chat
cowboy
db_connection
ecto
elixir
gettext
hex
inets
kernel
logger
mix
phoenix
phoenix_ecto
phoenix_live_reload
phoenix_pubsub
plug
postgrex
ranch
runtime_tools
ssl
<0.236.0>
<0.237.0>
Elixir.Chat.Supervisor
Elixir.Chat.Repo
Elixir.Chat.Repo.Pool
<0.241.0>
<0.242.0>
<0.243.0>
<0.244.0>
<0.245.0>
<0.246.0>
<0.247.0>
<0.248.0>
<0.249.0>
<0.250.0>
<0.251.0>
Elixir.ChatWeb.Endpoint
Elixir.Chat.PubSub.Supervisor
Elixir.ChatWeb.Endpoint.Config
<0.257.0>
Elixir.Chat.PubSub
<0.259.0>
Elixir.Chat.PubSub.GC0
Elixir.Chat.PubSub.Local0
Process X

Starting an application is synchronous

Myth:

"*Starting an application loads its modules*"

```
iex(1)> Application.start(:ex_unit)
:ok
iex(2)> :code.is_loaded(ExUnit.Case)
false
```

More information:

* *lib/kernel/src/application.erl*

* *lib/kernel/src/application_controller.erl*

* *lib/kernel/src/application_master.erl*

Intermission

Normal/Regular OTP application

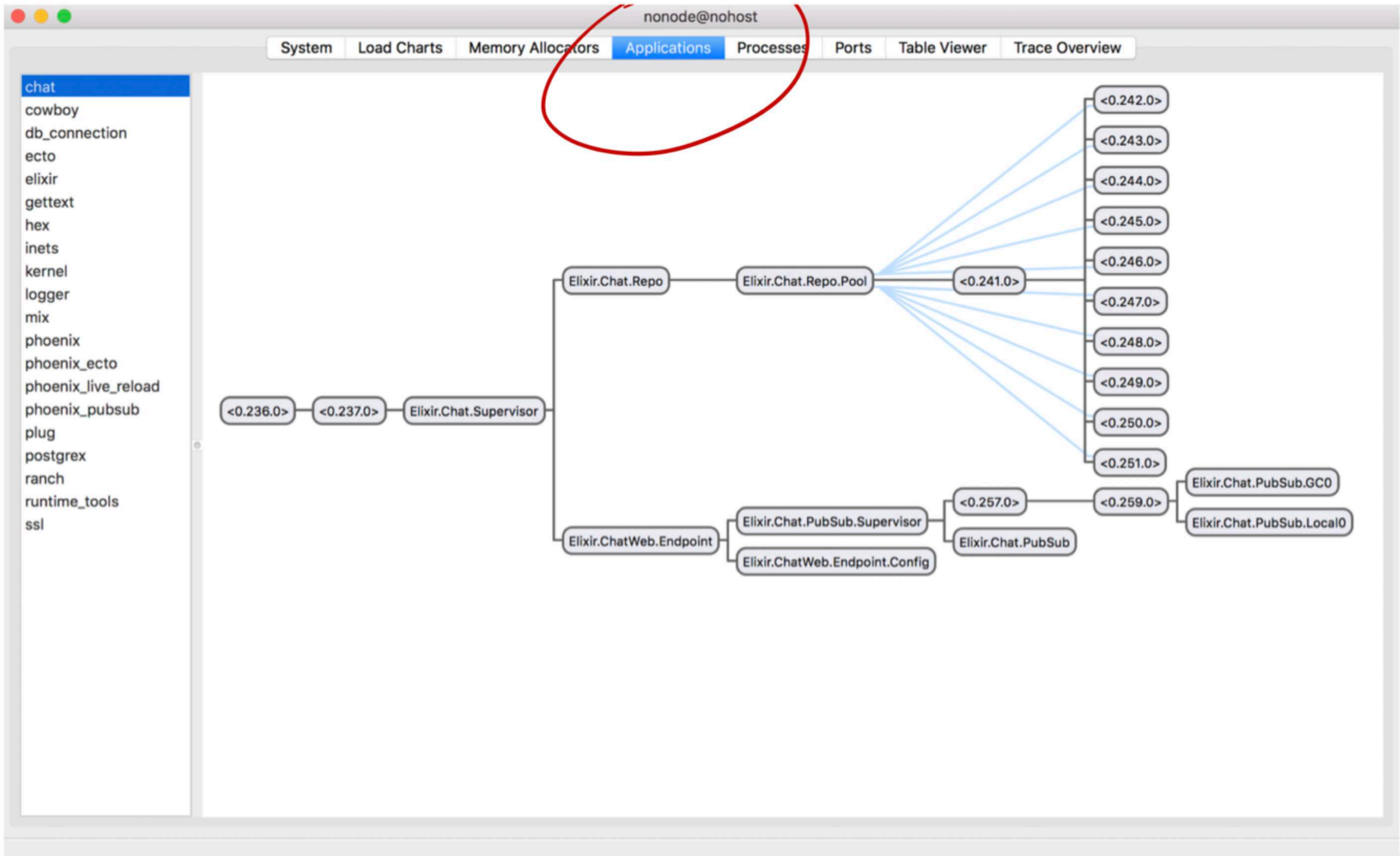
vs

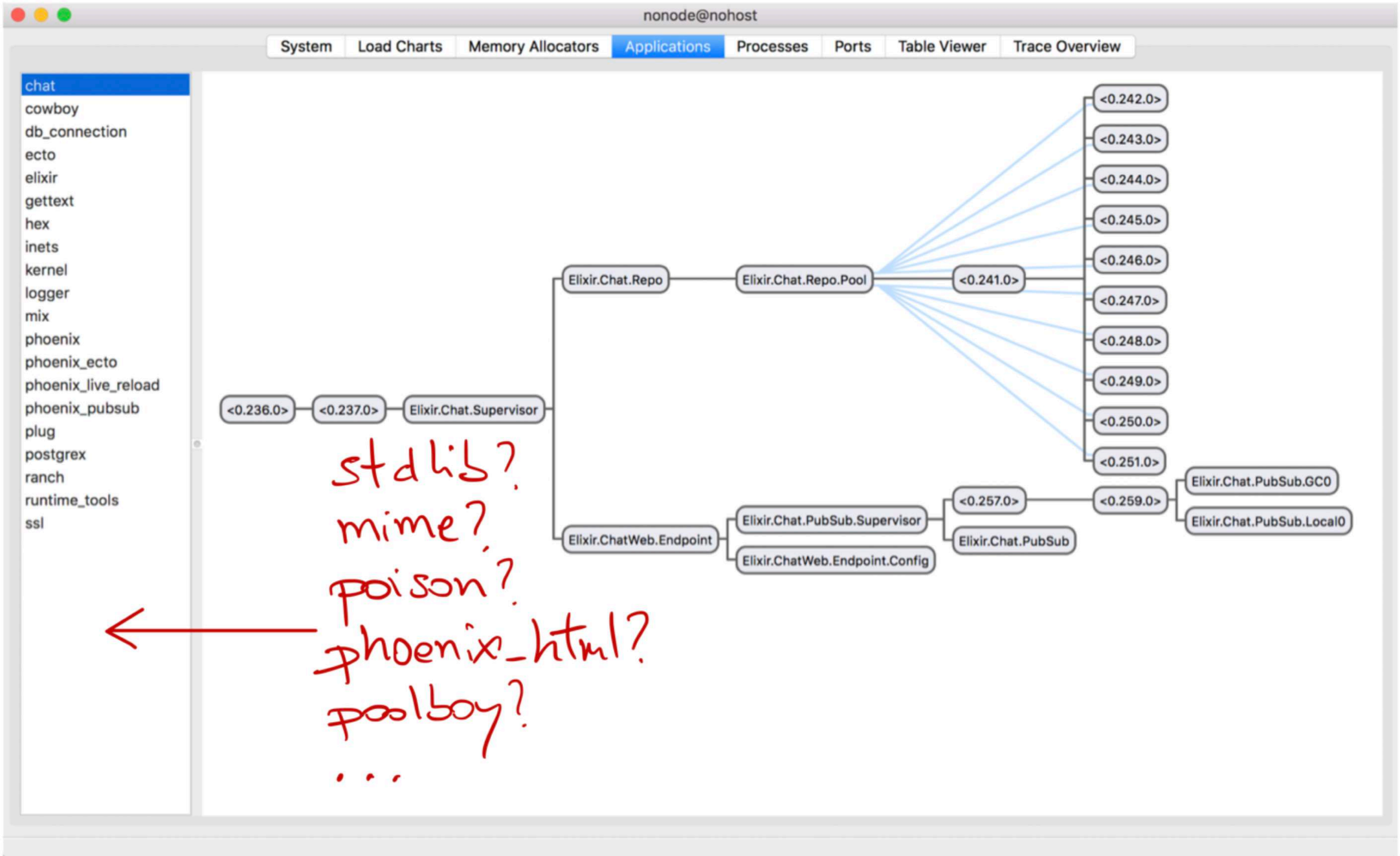
Library OTP application

A name is missing?

"Starting an application is done via the application module callback..."

"Like all OTP applications, COURTESY has a top supervisor."


nonode@nohost
System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
chat
cowboy
db_connection
ecto
elixir
gettext
hex
inets
kernel
logger
mix
phoenix
phoenix_ecto
phoenix_live_reload
phoenix_pubsub
plug
postgrex
ranch
runtime_tools
ssl
<0.242.0>
<0.243.0>
<0.244.0>
<0.245.0>
<0.246.0>
<0.247.0>
<0.248.0>
<0.249.0>
<0.250.0>
<0.251.0>
Elixir.Chat.Repo
Elixir.Chat.Repo.Pool
<0.241.0>
<0.236.0>
<0.237.0>
Elixir.Chat.Supervisor
Elixir.ChatWeb.Endpoint
Elixir.Chat.PubSub.Supervisor
<0.257.0>
<0.259.0>
Elixir.Chat.PubSub.GC0
Elixir.Chat.PubSub.Local0
Elixir.Chat.PubSub
Elixir.ChatWeb.Endpoint.Config


nonode@nohost
System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
chat
cowboy
db_connection
ecto
elixir
gettext
hex
inets
kernel
logger
mix
phoenix
phoenix_ecto
phoenix_live_reload
phoenix_pubsub
plug
postgrex
ranch
runtime_tools
ssl
<0.236.0>
<0.237.0>
Elixir.Chat.Supervisor
Elixir.Chat.Repo
Elixir.Chat.Repo.Pool
<0.241.0>
<0.242.0>
<0.243.0>
<0.244.0>
<0.245.0>
<0.246.0>
<0.247.0>
<0.248.0>
<0.249.0>
<0.250.0>
<0.251.0>
Elixir.ChatWeb.Endpoint
Elixir.Chat.PubSub.Supervisor
Elixir.ChatWeb.Endpoint.Config
<0.257.0>
Elixir.Chat.PubSub
<0.259.0>
Elixir.Chat.PubSub.GC0
Elixir.Chat.PubSub.Local0
stdlib?
mime?
poison?
phoenix_html?
poolboy?
...

Intermission Finale

Cool, but how is this related to Phoenix?

Mix projects manage OTP applications

```elixir
defmodule Chat.Mixfile do
  use Mix.Project

  def project do
    [app: :chat, version: "0.0.1", ...]
  end

  def application do
    [mod: {Chat.Application, []}, ...]
  end
end
```

```
{application, Application,
  [{description,              Description},
   {id,                       Id},
   {vsn,                      Vsn},
   {modules,                  Modules},
   {maxT,                     MaxT},
   {registered,               Names},
   {included_applications, Apps},
   {applications,             Apps},
   {env,                      Env},
   {mod,                      Start},
   {start_phases,             Phases},
   {runtime_dependencies,  RTDeps}]}.
```

```
$ ls _build/prod/lib/chat/ebin/*.app
_build/prod/lib/chat/ebin/chat.app
```

```
mix compile.app
```

```elixir
defmodule Chat.Application do
  use Application

  def start(_type, _args) do
    children = [
      Chat.Repo,
      ChatWeb.Endpoint
    ]
    opts = [
      strategy: :one_for_one,
      name: Chat.Supervisor
    ]
    Supervisor.start_link(children, opts)
  end
end
```

Phoenix projects are Mix projects

Modus Ponens

$$\frac{P \rightarrow Q \qquad P}{Q}$$

Phoenix projects manage OTP applications

```
mix run
```

run is the default Mix task by default

```
iex -S mix
```

~

```
iex -S mix run
```

```
mix run:

  * Boots Mix

  * Executes the run task
```

```
#!/usr/bin/env elixir

Mix.start
Mix.CLI.main
```

```
erl
  -pa ...
  -elixir ansi_enabled true
  -noshell
  -s elixir start_cli
  -extra bin/mix run
```

Mix.start:

  * Mix.start/0 is a regular function,
    not a callback.

  * Application.ensure_all_started(:mix)

Mix.CLI.main:

  * Loads *~/.mix/config.exs* if present

  * Loads *mix.exs*

  * Sets the Mix environment

  * Loads *config/config.exs*

  * Mix.Task.run(name, args)

```
mix run:
  * Adjusts the code path
  * Checks the project dependencies
  * Runs compilers, consolidates protocols
  * Starts the managed OTP application
    including all its dependencies
  * Preloads modules if --preload-modules
  * Process.sleep(:infinity) if --no-halt
```

```
elixir --erl '-boot start sasl' -S mix
```

```
elixir --logger-sasl-reports true -S mix
```

In our chat application, mix run starts 34 applications

Applications (34)		
kernel	plug	file_system
stdlib	poison	phoenix_live_reload
compiler	eex	connection
elixir	phoenix_pubsub	db_connection
mix	phoenix	decimal
crypto	gettext	postgrex
asn1	logger	poolboy
public_key	runtime_tools	ecto
ssl	ranch	phoenix_ecto
inets	cowlib	chat
hex	cowboy	
mime	phoenix_html	

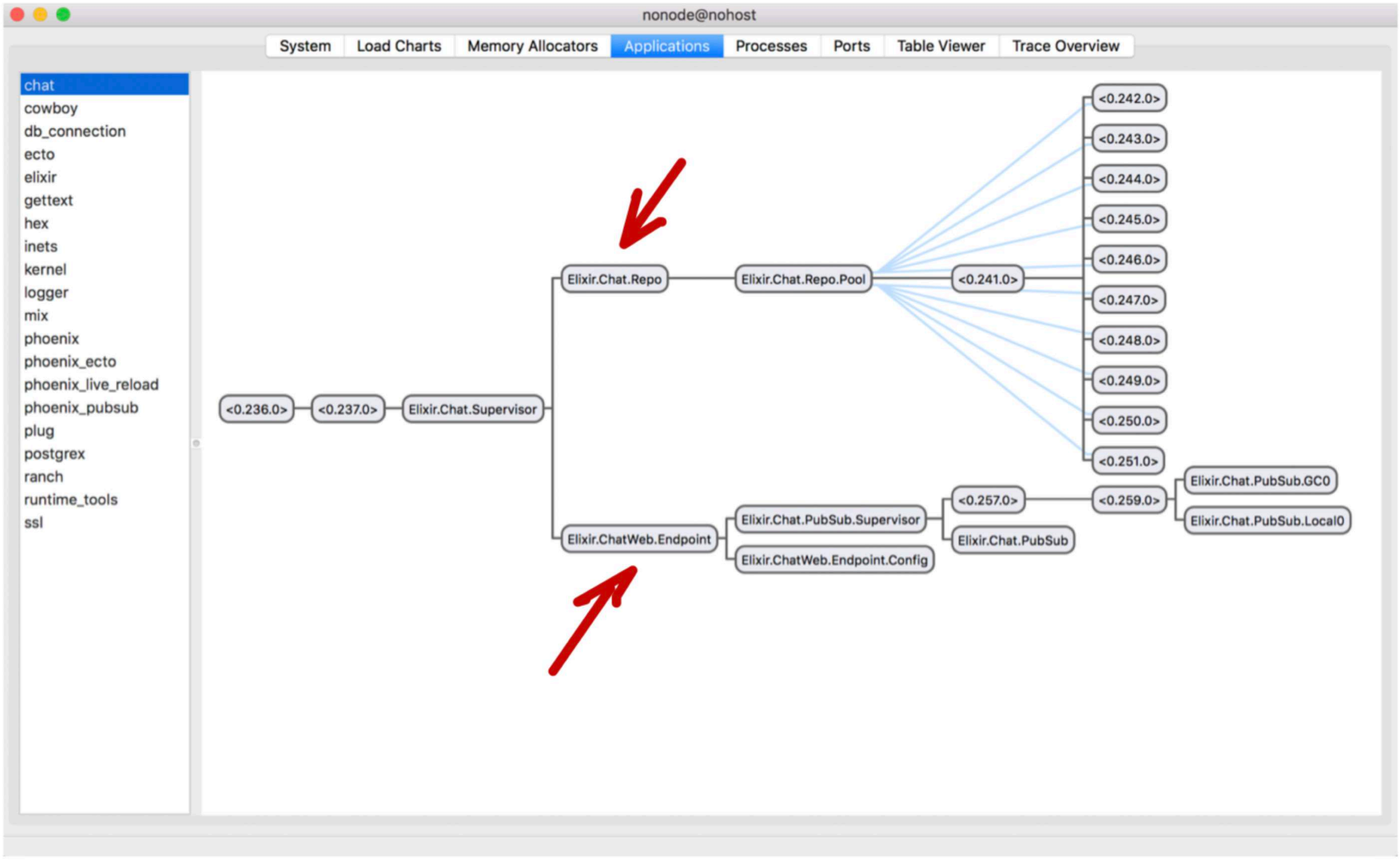
Erlang (14)	
kernel	ssl
stdlib	inets
compiler	runtime_tools
elixir	ranch
crypto	cowlib
asn1	cowboy
public_key	poolboy

Elixir (21)	
elixir	phoenix_html
mix	file_system
hex	phoenix_live_reload
mime	connection
plug	db_connection
poison	decimal
eex	postgrex
phoenix_pubsub	ecto
phoenix	phoenix_ecto
gettext	chat
logger	

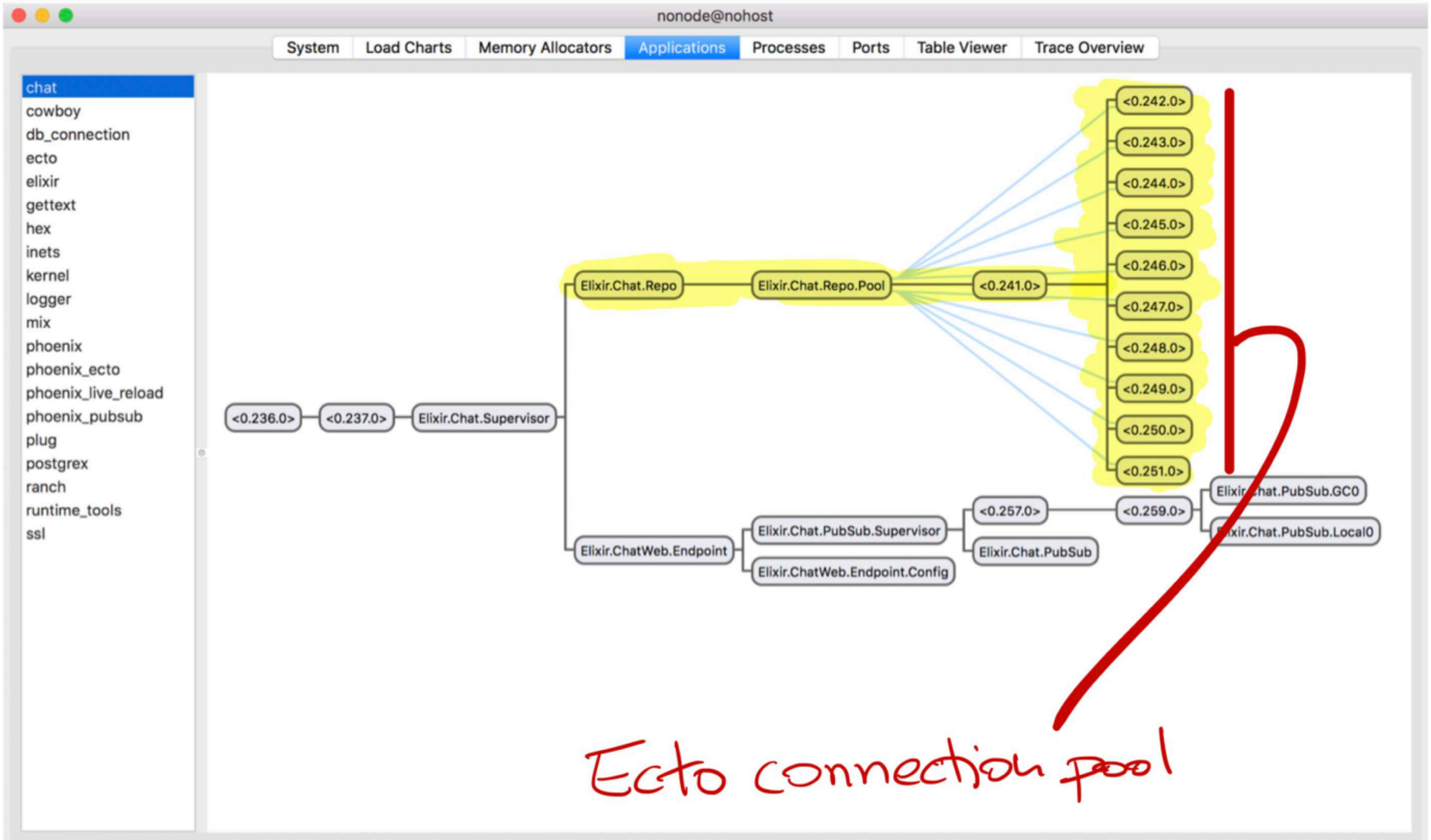
Library applications (14)	
stdlib	eex
compiler	cowlib
crypto	phoenix_html
asn1	file_system
public_key	connection
mime	decimal
poison	poolboy

Regular applications (20)	
kernel	logger
elixir	runtime_tools
mix	ranch
ssl	cowboy
inets	phoenix_live_reload
hex	db_connection
plug	postgrex
phoenix_pubsub	ecto
phoenix	phoenix_ecto
gettext	chat

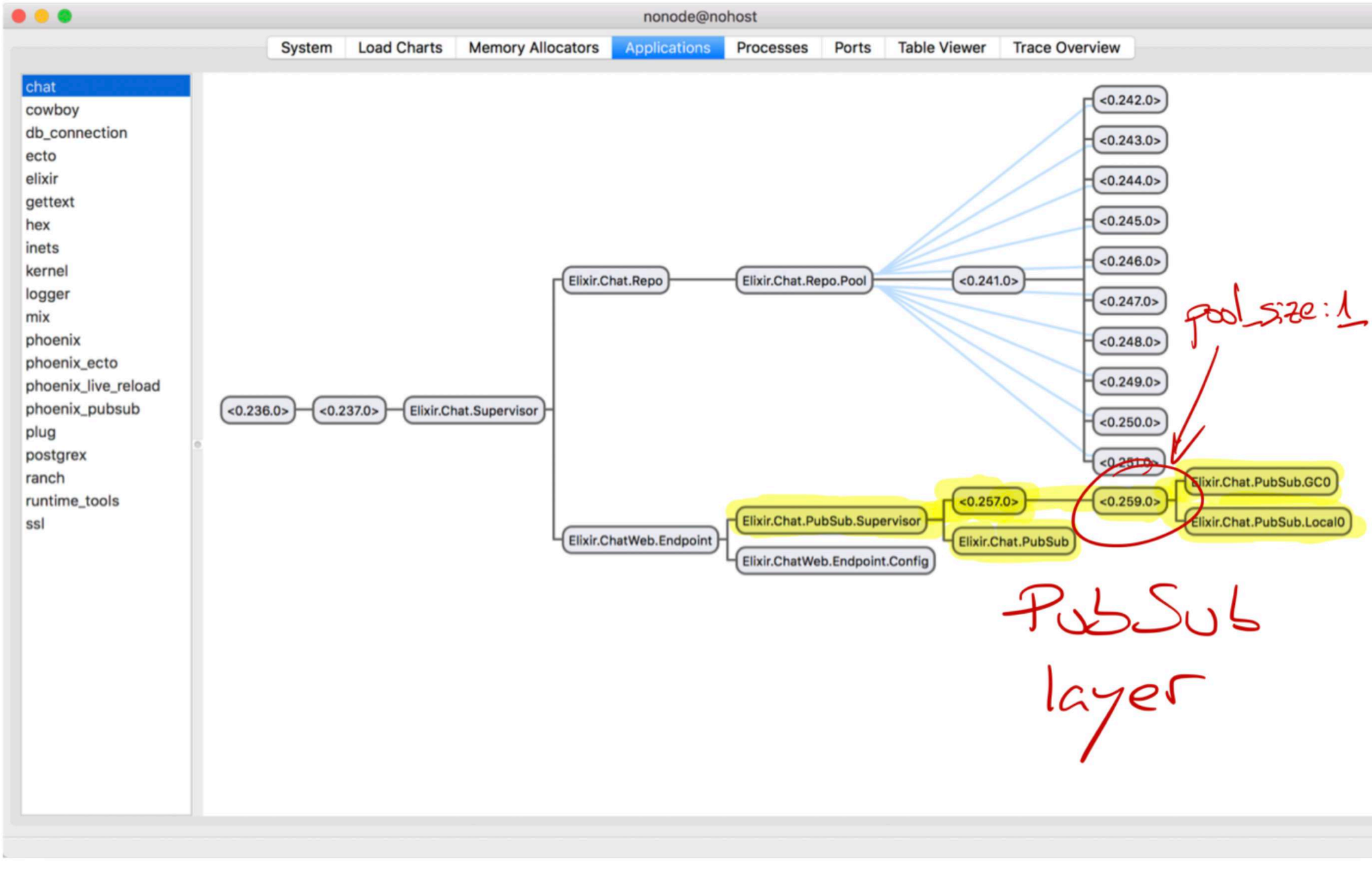
```elixir
defmodule Chat.Application do
  use Application

  def start(_type, _args) do
    children = [
      Chat.Repo,
      ChatWeb.Endpoint
    ]
    opts = [
      strategy: :one_for_one,
      name: Chat.Supervisor
    ]
    Supervisor.start_link(children, opts)
  end
end
```


nonode@nohost
System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
chat
cowboy
db_connection
ecto
elixir
gettext
hex
inets
kernel
logger
mix
phoenix
phoenix_ecto
phoenix_live_reload
phoenix_pubsub
plug
postgrex
ranch
runtime_tools
ssl
Elixir.Chat.Repo
Elixir.Chat.Repo.Pool
<0.241.0>
<0.242.0>
<0.243.0>
<0.244.0>
<0.245.0>
<0.246.0>
<0.247.0>
<0.248.0>
<0.249.0>
<0.250.0>
<0.251.0>
<0.236.0>
<0.237.0>
Elixir.Chat.Supervisor
Elixir.ChatWeb.Endpoint
Elixir.Chat.PubSub.Supervisor
Elixir.ChatWeb.Endpoint.Config
<0.257.0>
Elixir.Chat.PubSub
<0.259.0>
Elixir.Chat.PubSub.GC0
Elixir.Chat.PubSub.Local0

```elixir
defmodule Chat.Repo do
  use Ecto.Repo, otp_app: :chat

  def init(_, opts) do
    url = System.get_env("DATABASE_URL")
    {:ok, Keyword.put(opts, :url, url)}
  end
end
```


nonode@nohost
System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
chat
cowboy
db_connection
ecto
elixir
gettext
hex
inets
kernel
logger
mix
phoenix
phoenix_ecto
phoenix_live_reload
phoenix_pubsub
plug
postgrex
ranch
runtime_tools
ssl
<0.236.0>
<0.237.0>
Elixir.Chat.Supervisor
Elixir.Chat.Repo
Elixir.Chat.Repo.Pool
<0.241.0>
<0.242.0>
<0.243.0>
<0.244.0>
<0.245.0>
<0.246.0>
<0.247.0>
<0.248.0>
<0.249.0>
<0.250.0>
<0.251.0>
Elixir.ChatWeb.Endpoint
Elixir.Chat.PubSub.Supervisor
Elixir.ChatWeb.Endpoint.Config
<0.257.0>
Elixir.Chat.PubSub
<0.259.0>
Elixir.Chat.PubSub.GC0
Elixir.Chat.PubSub.Local0
Ecto connection pool

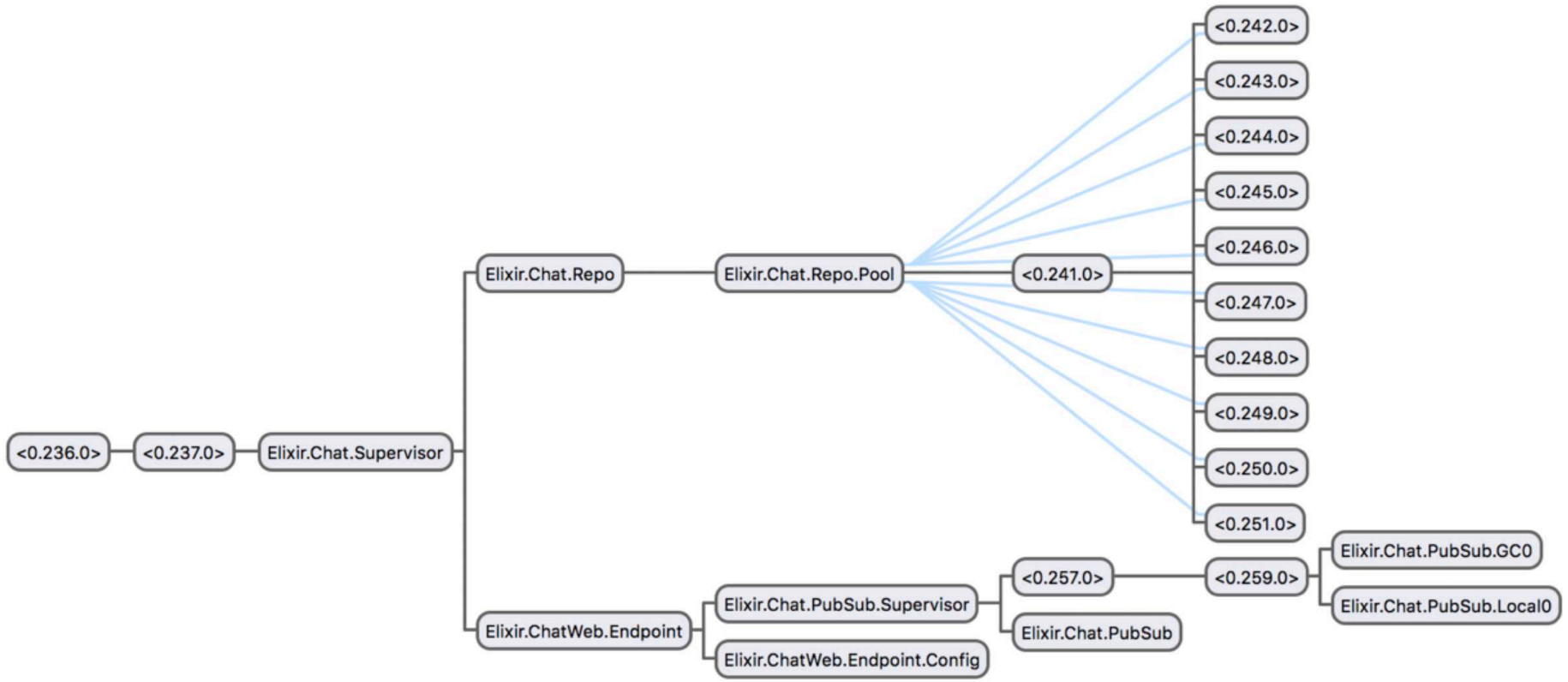
```
defmodule ChatWeb.Endpoint do
  use Phoenix.Endpoint, otp_app: :chat

  if code_reloading? do
    socket ...
    plug Phoenix.LiveReloader
    plug Phoenix.CodeReloader
  end
  ...
end
```


nonode@nohost
System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
chat
cowboy
db_connection
ecto
elixir
gettext
hex
inets
kernel
logger
mix
phoenix
phoenix_ecto
phoenix_live_reload
phoenix_pubsub
plug
postgrex
ranch
runtime_tools
ssl
<0.236.0>
<0.237.0>
Elixir.Chat.Supervisor
Elixir.Chat.Repo
Elixir.Chat.Repo.Pool
<0.241.0>
<0.242.0>
<0.243.0>
<0.244.0>
<0.245.0>
<0.246.0>
<0.247.0>
<0.248.0>
<0.249.0>
<0.250.0>
<0.251.0>
pool_size: 1
Elixir.ChatWeb.Endpoint
Elixir.Chat.PubSub.Supervisor
Elixir.ChatWeb.Endpoint.Config
<0.257.0>
Elixir.Chat.PubSub
<0.259.0>
Elixir.Chat.PubSub.GC0
Elixir.Chat.PubSub.Local0
PubSub
layer

nonode@nohost

System | Load Charts | Memory Allocators | Applications | Processes | Ports | Table Viewer | Trace Overview

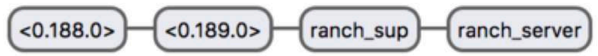
- chat
- cowboy
- db_connection
- ecto
- elixir
- gettext
- hex
- inets
- kernel
- logger
- mix
- phoenix
- phoenix_ecto
- phoenix_live_reload
- phoenix_pubsub
- plug
- postgrex
- ranch
- runtime_tools
- ssl

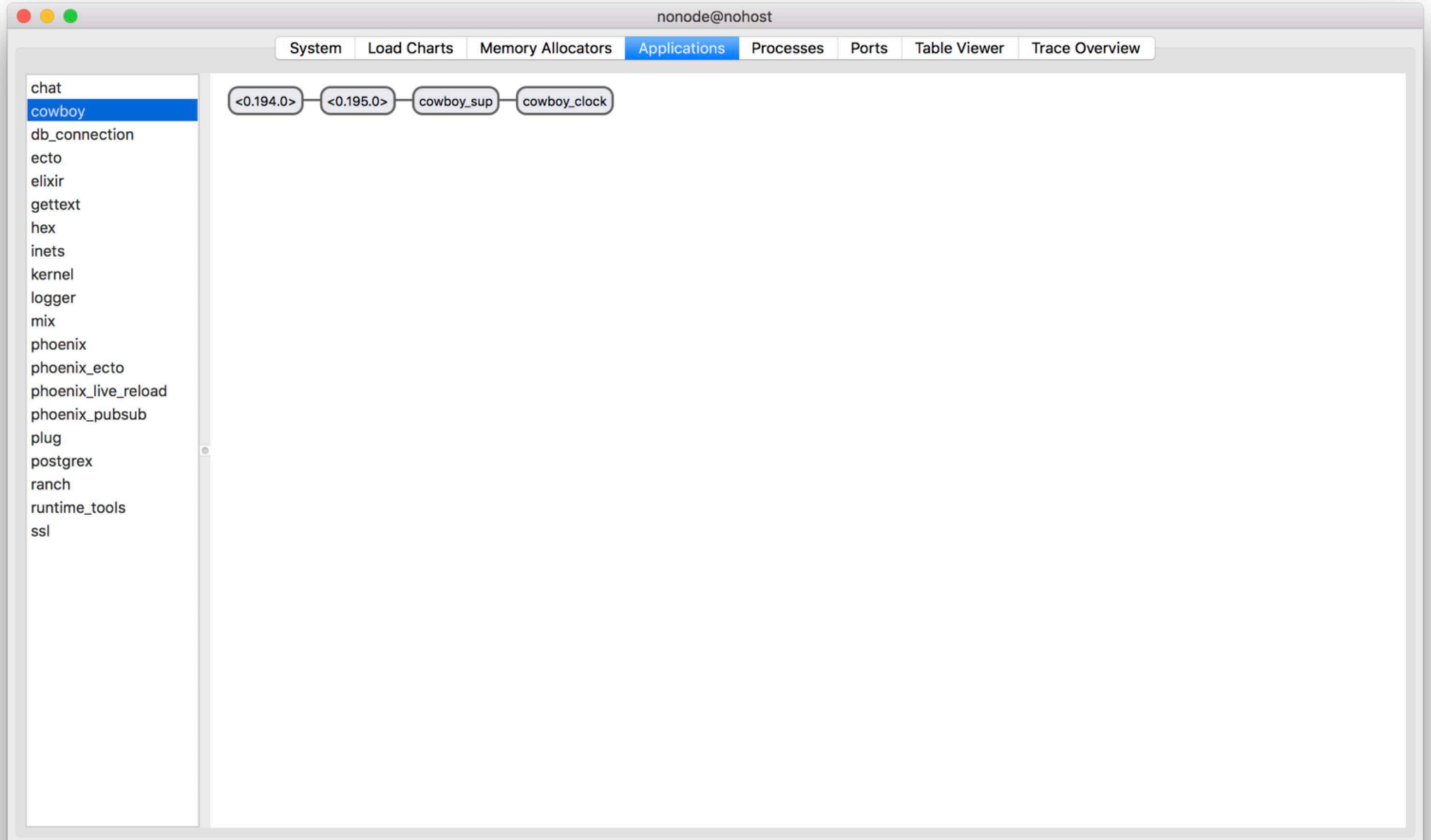


nonode@nohost

System | Load Charts | Memory Allocators | Applications | Processes | Ports | Table Viewer | Trace Overview

- chat
- cowboy
- db_connection
- ecto
- elixir
- gettext
- hex
- inets
- kernel
- logger
- mix
- phoenix
- phoenix_ecto
- phoenix_live_reload
- phoenix_pubsub
- plug
- postgrex
- ranch
- runtime_tools
- ssl




nonode@nohost
System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
chat
cowboy
db_connection
ecto
elixir
gettext
hex
inets
kernel
logger
mix
phoenix
phoenix_ecto
phoenix_live_reload
phoenix_pubsub
plug
postgrex
ranch
runtime_tools
ssl
<0.194.0>
<0.195.0>
cowboy_sup
cowboy_clock

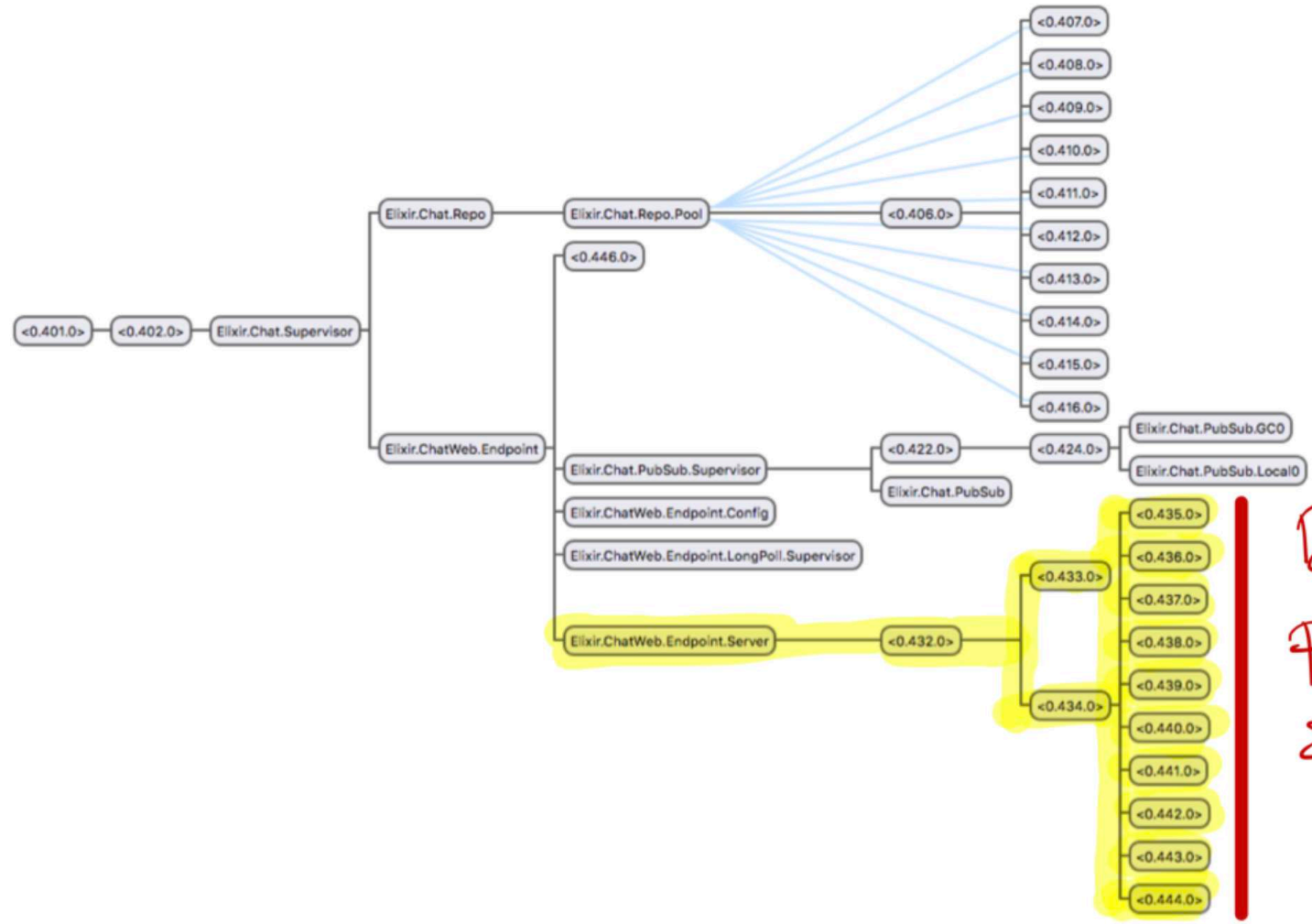
```
mix phx.server
```

```elixir
defmodule Mix.Tasks.Phx.Server do
  use Mix.Task

  def run(args) do
    Application.put_env(
      :phoenix,
      :serve_endpoints,
      true,
      persistent: true
    )
    Mix.Tasks.Run.run run_args() ++ args
  end
  ...
end
```

nonode@nohost

System | Load Charts | Memory Allocators | Applications | Processes | Ports | Table Viewer | Trace Overview

- chat
- cowboy
- db_connection
- ecto
- elixir
- gettext
- hex
- iex
- inets
- kernel
- logger
- mix
- phoenix
- phoenix_ecto
- phoenix_live_reload
- phoenix_pubsub
- plug
- postgrex
- ranch
- runtime_tools
- ssl



Ranch acceptor pool (default size is 100)

Summary of the Summary

* Load *mix.exs*

* Set the Mix environment

* Load *config/config.exs*

* Compile

* Start all applications

* On starting the chat application:
  + Create the db connection pool
  + Setup the PubSub layer
  + Launch Cowboy