Cell-Driven Development

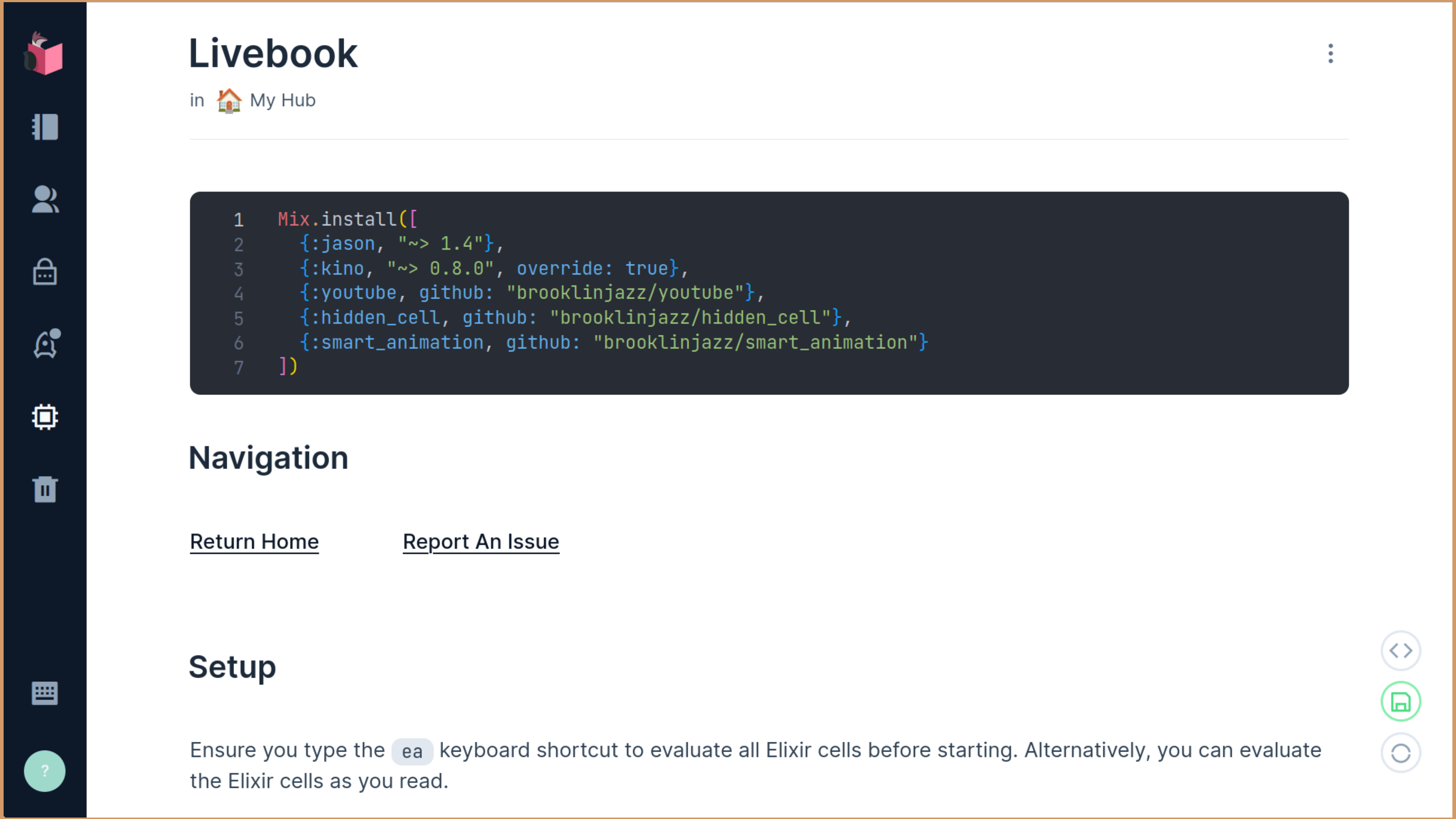
Three examples of custom smart cells in Livebook.

Marko Vukovic | mvk.vc | @mvkvc
Einar Engström | einariii.xyz | @einariii

who we are/why we're here

- two early-career Elixir developers
- friends & collaborators
- recent graduates of DockYard Academy
- livebook was our platform for learning
- let's give back to the community

livebook introduction



Livebook

in 🏠 My Hub

```elixir
Mix.install([
  {:jason, "~> 1.4"},
  {:kino, "~> 0.8.0", override: true},
  {:youtube, github: "brooklinjazz/youtube"},
  {:hidden_cell, github: "brooklinjazz/hidden_cell"},
  {:smart_animation, github: "brooklinjazz/smart_animation"}
])
```

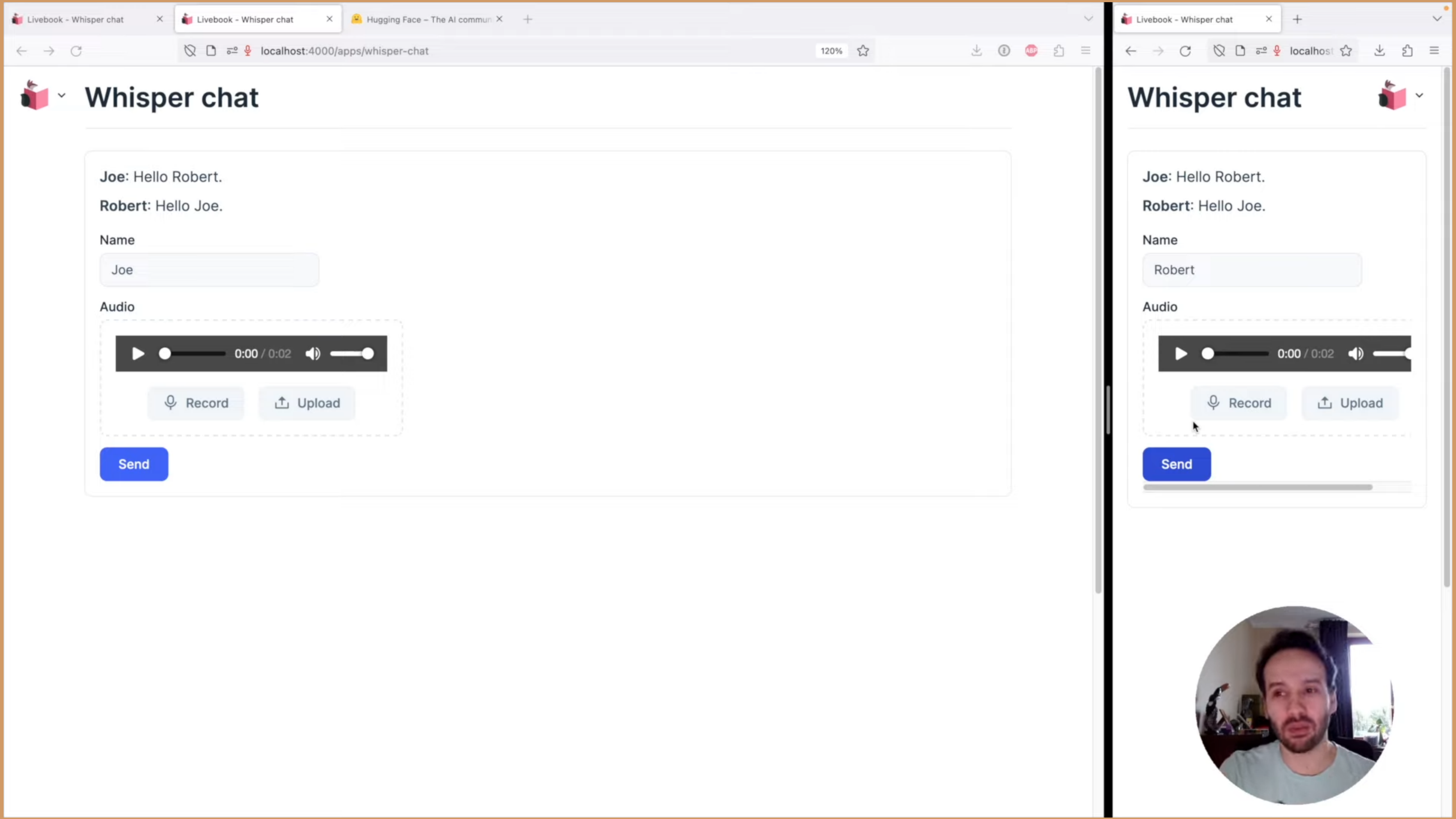
Navigation

Return Home   Report An Issue

Setup

Ensure you type the `ea` keyboard shortcut to evaluate all Elixir cells before starting. Alternatively, you can evaluate the Elixir cells as you read.

livebook apps



Livebook - Whisper chat
Hugging Face – The AI commun
localhost:4000/apps/whisper-chat

Whisper chat

Joe: Hello Robert.

Robert: Hello Joe.

Name: Joe

Audio

0:00 / 0:02

Record | Upload

Send

Whisper chat

Joe: Hello Robert.

Robert: Hello Joe.

Name: Robert

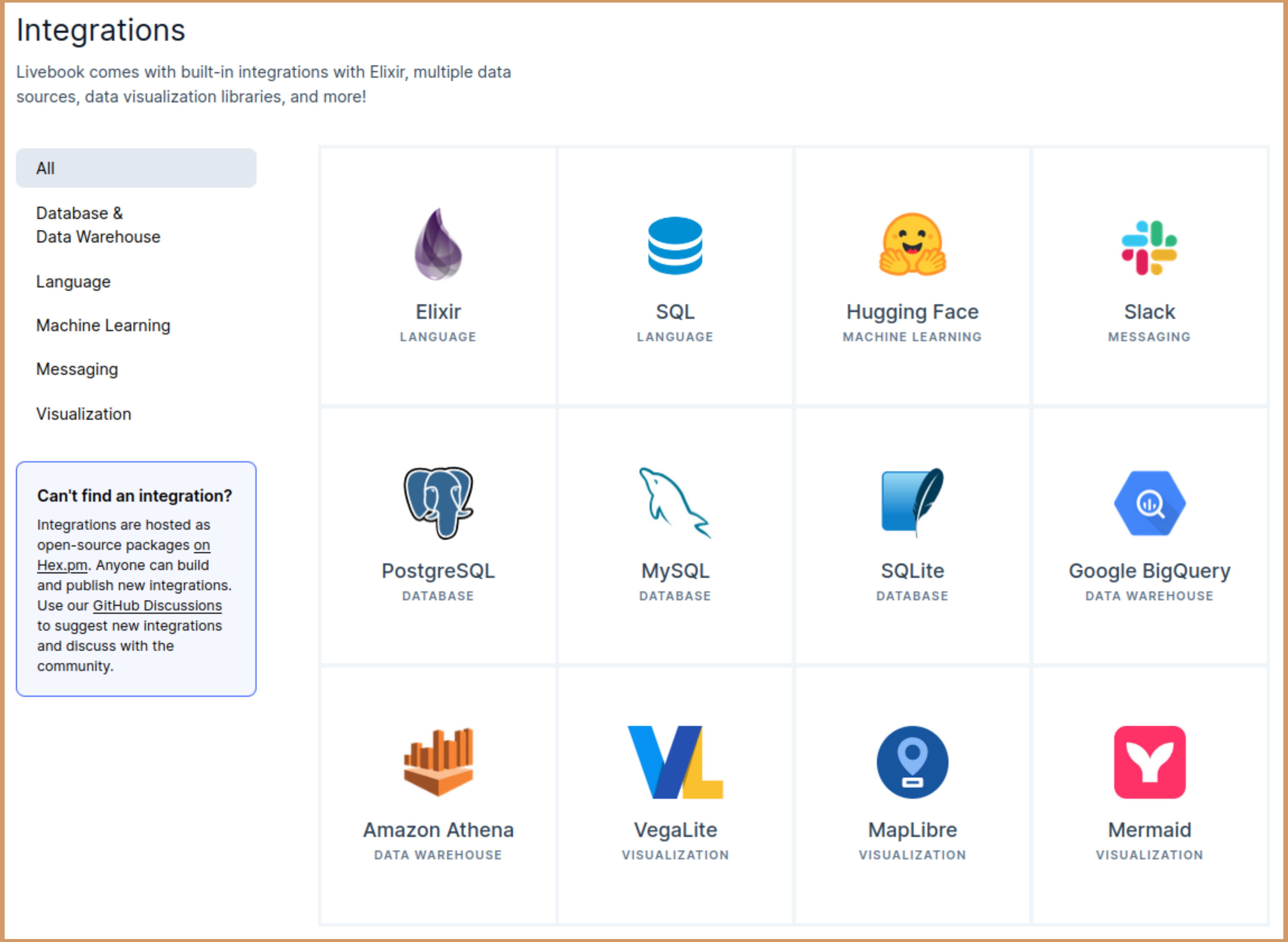
Audio

0:00 / 0:02

Record | Upload

Send

livebook integrations



Integrations

Livebook comes with built-in integrations with Elixir, multiple data sources, data visualization libraries, and more!

Can't find an integration?

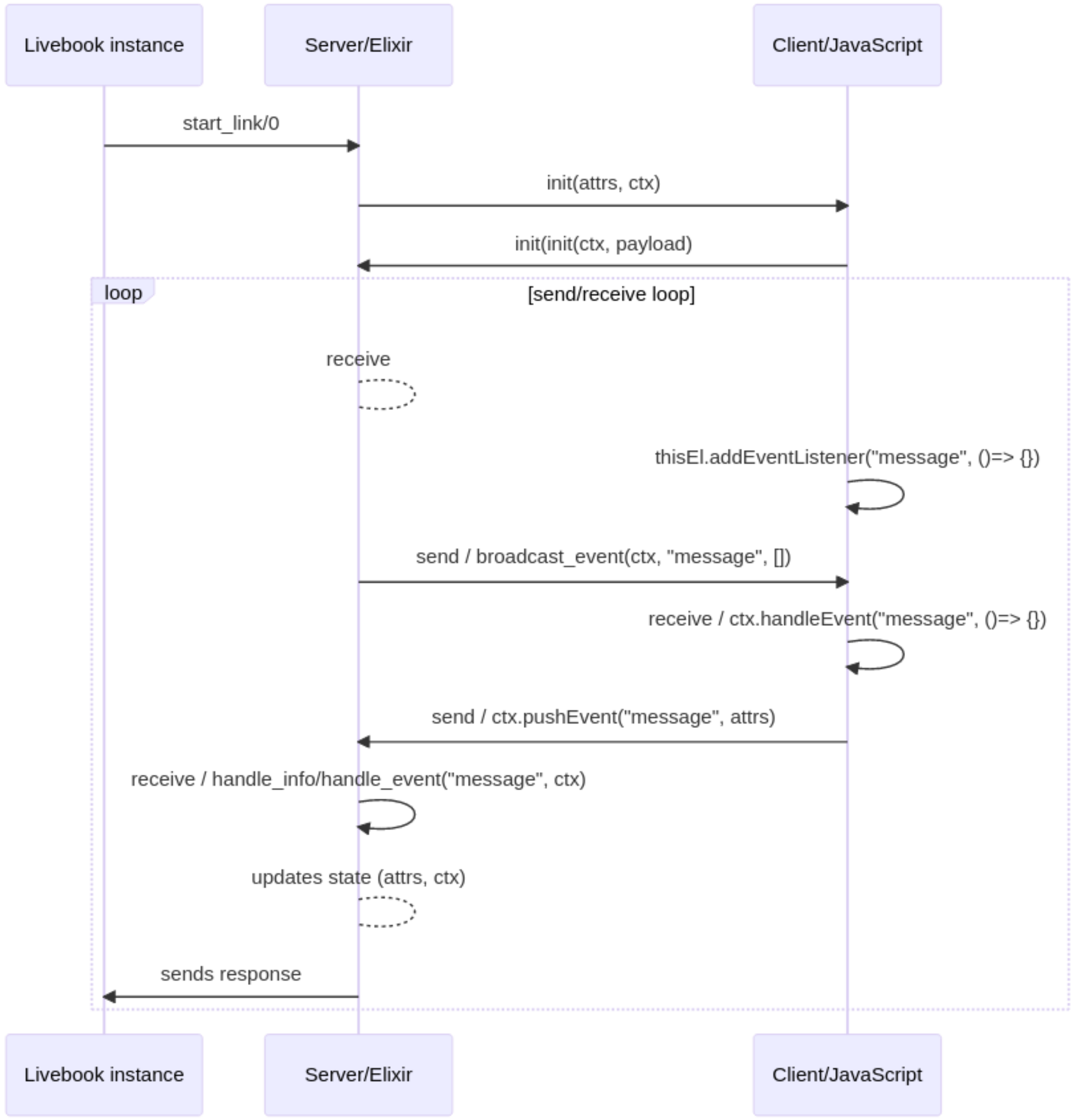
Integrations are hosted as open-source packages on Hex.pm. Anyone can build and publish new integrations. Use our GitHub Discussions to suggest new integrations and discuss with the community.

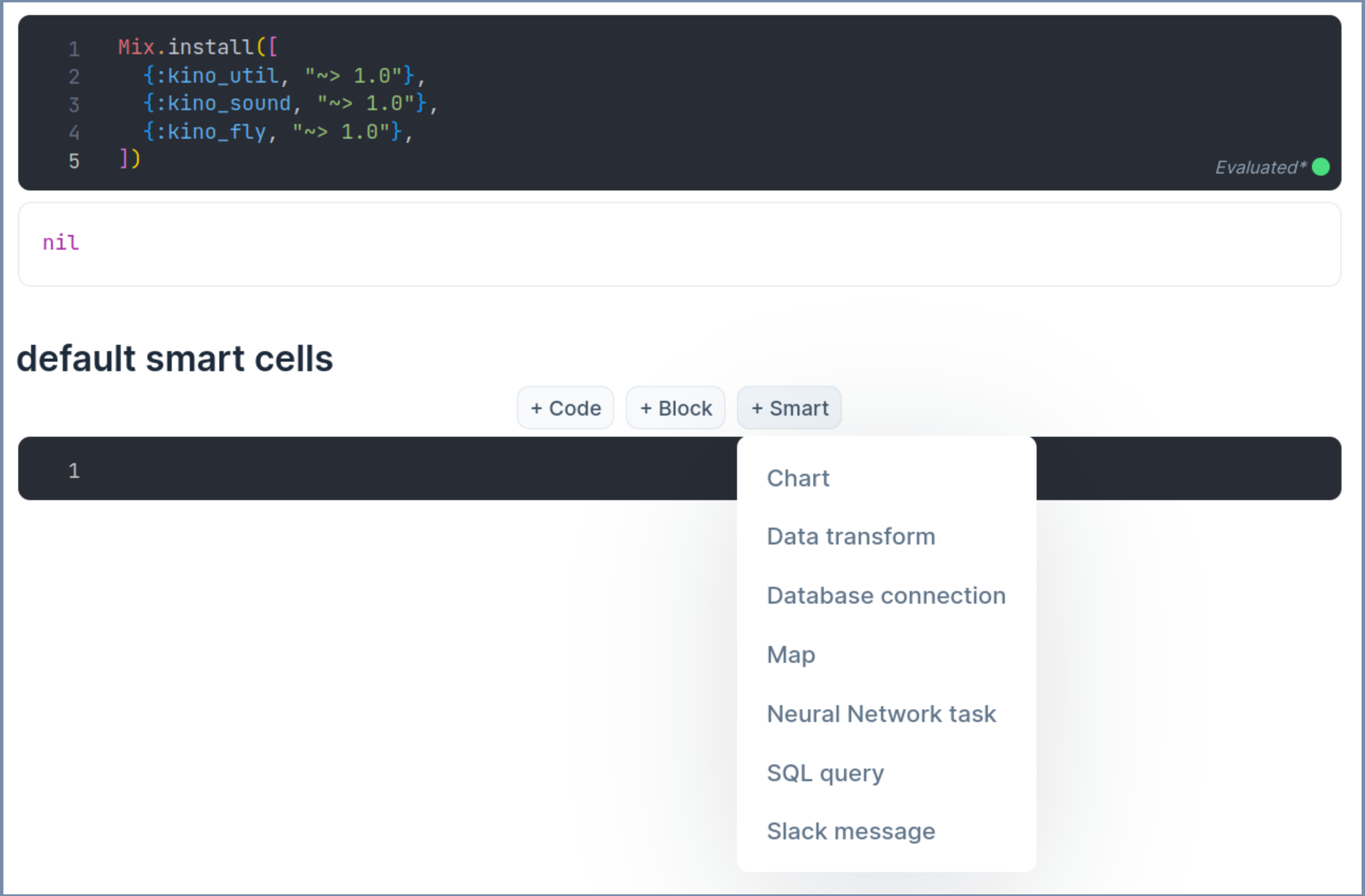
Elixir LANGUAGE	SQL LANGUAGE	Hugging Face MACHINE LEARNING	Slack MESSAGING
PostgreSQL DATABASE	MySQL DATABASE	SQLite DATABASE	Google BigQuery DATA WAREHOUSE
Amazon Athena DATA WAREHOUSE	VegaLite VISUALIZATION	MapLibre VISUALIZATION	Mermaid VISUALIZATION

smart cells

automated client-server programs
producing code in the background
great for testing & development

SMART CELL SEQUENCE

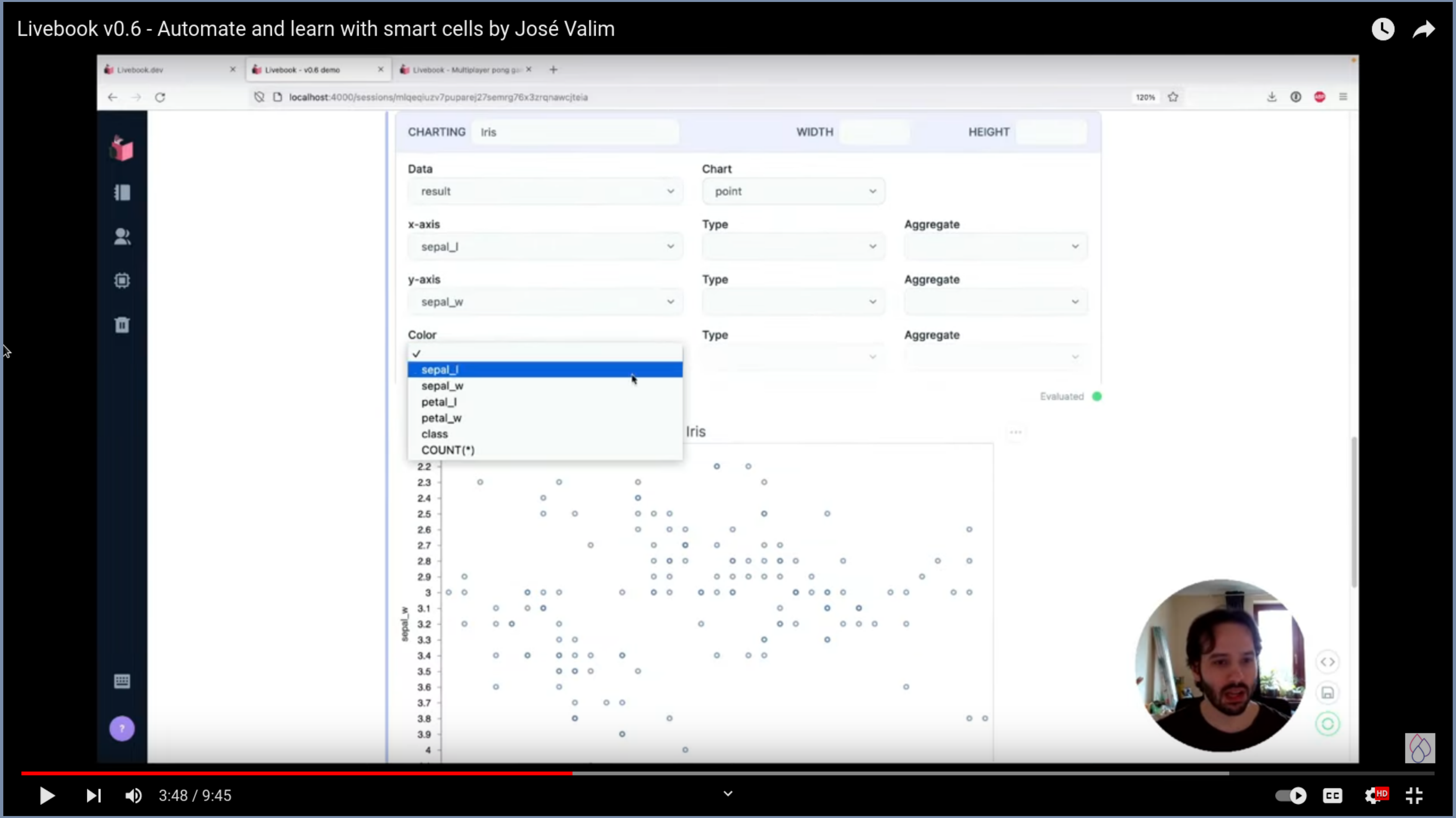

Livebook instance
Server/Elixir
Client/JavaScript
start_link/0
init(attrs, ctx)
init(init(ctx, payload)
loop
[send/receive loop]
receive
thisEl.addEventListener("message", ()=> {})
send / broadcast_event(ctx, "message", [])
receive / ctx.handleEvent("message", ()=> {})
send / ctx.pushEvent("message", attrs)
receive / handle_info/handle_event("message", ctx)
updates state (attrs, ctx)
sends response
Livebook instance
Server/Elixir
Client/JavaScript



```elixir
Mix.install([
  {:kino_util, "~> 1.0"},
  {:kino_sound, "~> 1.0"},
  {:kino_fly, "~> 1.0"},
])
```

```
nil
```

default smart cells


Livebook v0.6 - Automate and learn with smart cells by José Valim

https://livebook.dev

🧠🔨

```
mix new kino_util --sup
```

💻📊

kino_util

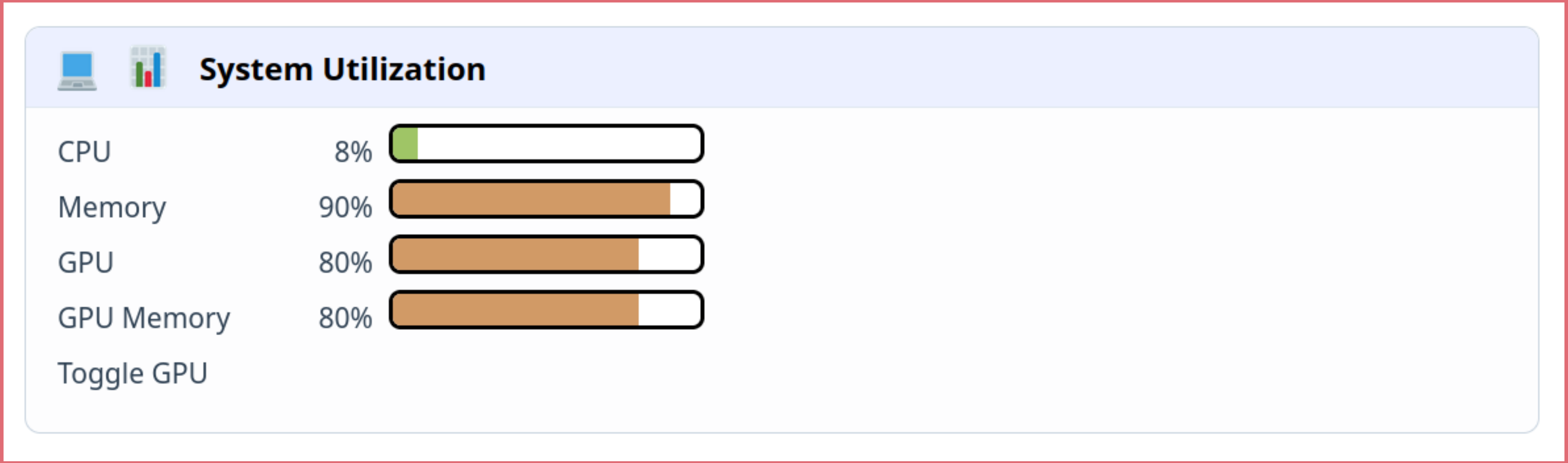
keep your system in sight.

https://hex.pm/packages/kino_util

```elixir
defmodule KinoUtil.Application do
  use Application

  @impl true
  def start(_type, _args) do
    Kino.SmartCell.register(KinoUtil)

    children = []
    opts = [strategy: :one_for_one, name: KinoUtil.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

```elixir
defmodule KinoUtil do
  use Kino.JS, assets_path: "lib/assets"
  use Kino.JS.Live
  use Kino.SmartCell, name: "System utilization"
  alias KinoUtil.Utils

  ...

  @impl true
  def init(attrs, ctx) do
    # ...
  end

  @impl true
  def handle_info("show_gpu", ctx) do
    # ...
    {:noreply, ctx}
  end

  @impl true
  def handle_info("update", ctx) do
    # ...
    {:noreply, ctx}
  end

  ...
  ...
  @impl true
  def handle_connect(ctx) do
    {:ok, %{fields: ctx.assigns.fields}, ctx}
  end

  @impl true
  def to_attrs(ctx) do
    ctx.assigns.fields
  end

  @impl true
  def to_source(_attrs) do
    quote do
      IO.puts("to_source not implemented")
    end
    |> Kino.SmartCell.quoted_to_string()
  end
end
```

```
import * as Vue from "https://cdn.jsdelivr.net/npm/vue@3.2.26/dist/vue.esm-browser.prod.js";

export function init(ctx, payload) {
  ctx.importCSS("output.css");

  const UtilBar = {...}};

  const app = Vue.createApp({...}).mount(ctx.root);

  ctx.handleEvent("show_gpu", (has_gpu) => {});

  ctx.handleEvent("update", (fields) => {});
}
```



System Utilization

CPU 8%

Memory 90%

GPU 80%

GPU Memory 80%

Toggle GPU

💻 📊

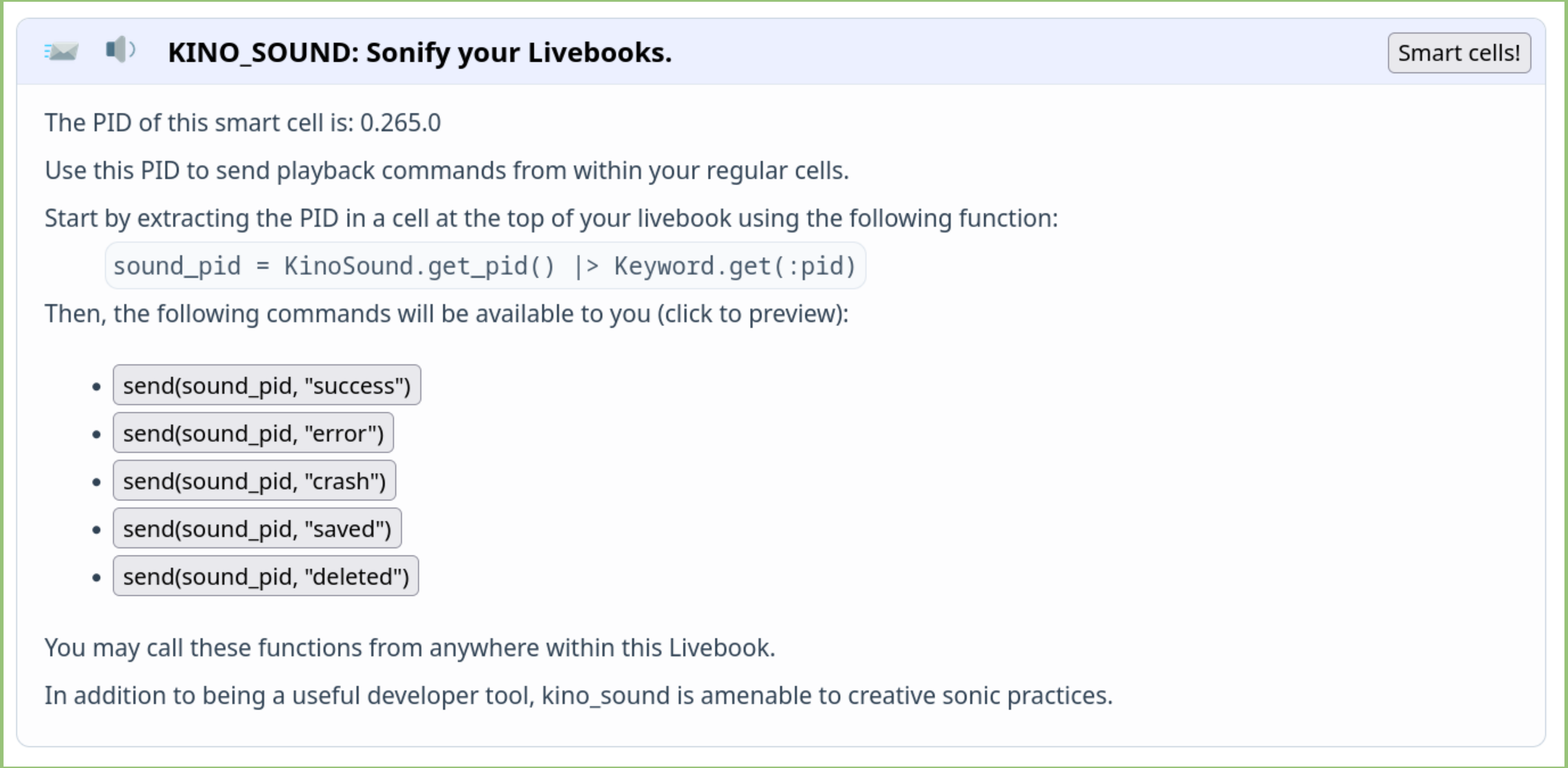
📨🔉

kino_sound

sonify your workflow.

https://hex.pm/packages/kino_sound

call & response:
a smart cell that allows regular cells
to send commands
to the howler.js framework
which sings back



KINO_SOUND: Sonify your Livebooks.

The PID of this smart cell is: 0.265.0

Use this PID to send playback commands from within your regular cells.

Start by extracting the PID in a cell at the top of your livebook using the following function:

```
sound_pid = KinoSound.get_pid() |> Keyword.get(:pid)
```

Then, the following commands will be available to you (click to preview):

- send(sound_pid, "success")
- send(sound_pid, "error")
- send(sound_pid, "crash")
- send(sound_pid, "saved")
- send(sound_pid, "deleted")

You may call these functions from anywhere within this Livebook.

In addition to being a useful developer tool, kino_sound is amenable to creative sonic practices.

📨 🔈

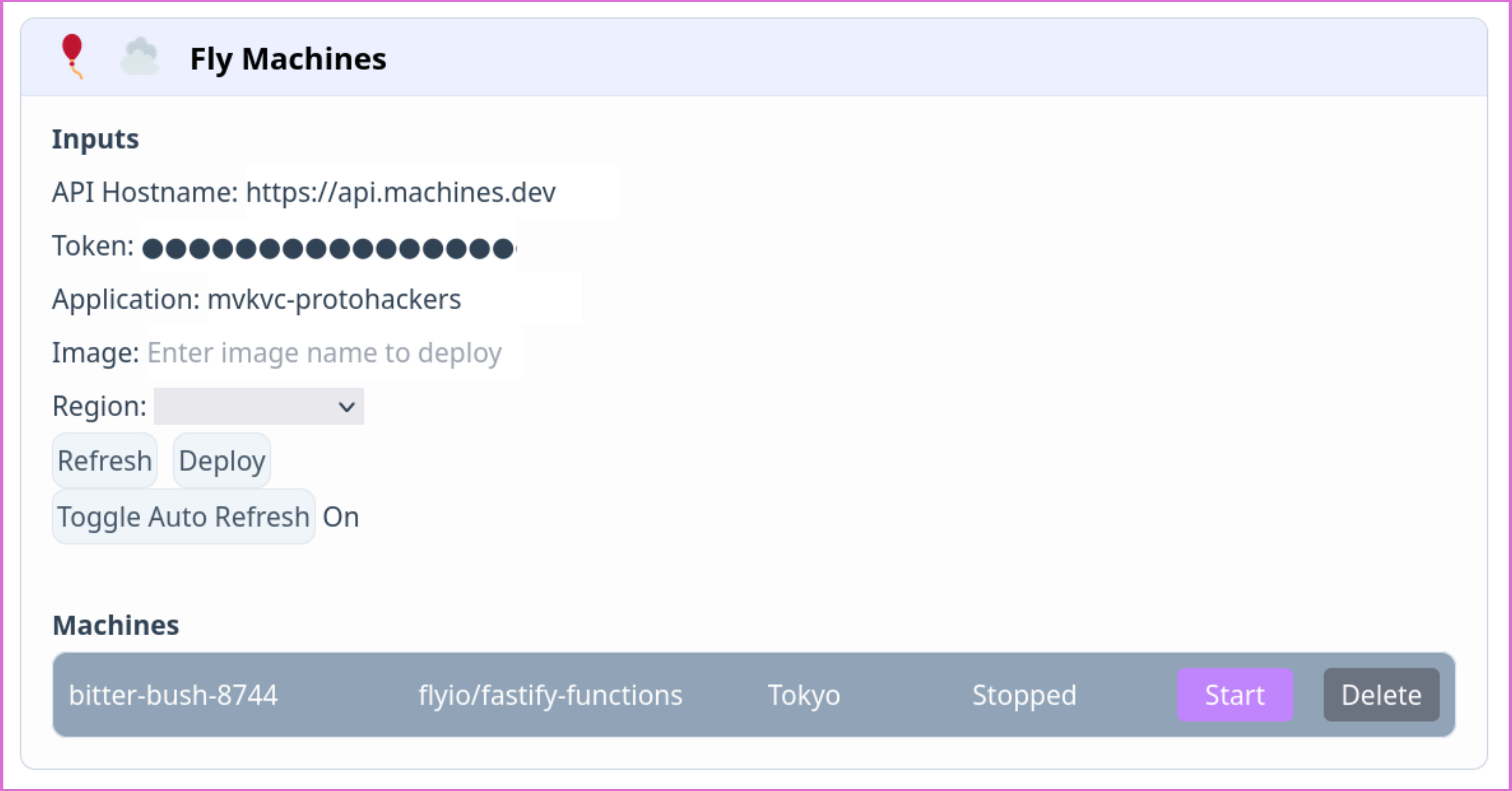
🎈☁️

kino_fly

cloud's-eye view with fly machines.

https://hex.pm/packages/kino_fly

inspect & control:
a smart cell that allows developers
to manage their fly.io applications
all from one spot
at a distance



Fly Machines

Inputs

API Hostname: https://api.machines.dev

Token: ●●●●●●●●●●●●●●●●

Application: mvkvc-protohackers

Image: Enter image name to deploy

Region:

Refresh Deploy

Toggle Auto Refresh On

Machines

bitter-bush-8744	flyio/fastify-functions	Tokyo	Stopped	Start	Delete

future work

- full user customization for `kino_sound`
- GPU detection on all platforms for `kino_util`
- built-in latency analysis for `kino_fly`
- improved test suites
- release/maintain on hex.pm
- split `kino_fly` client into own library

hopes and aspirations

- more smart cells
- more elixir
- more programmatic interaction
- more cell-to-cell interaction
- more collaboration

gratitude

to @ghedamat and the Toronto Elixir Meetup
to the Livebook team
to the Elixir community at large
to theScore for hosting us
to DockYard