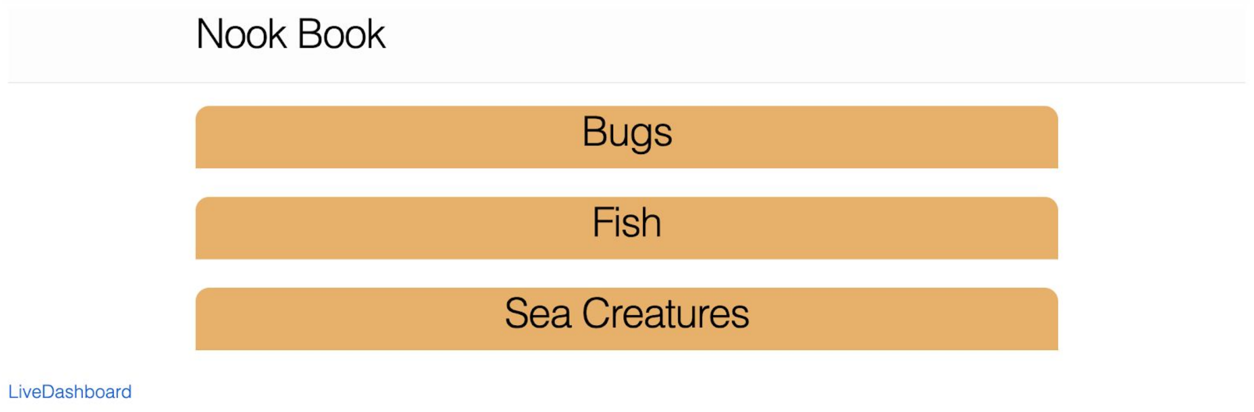
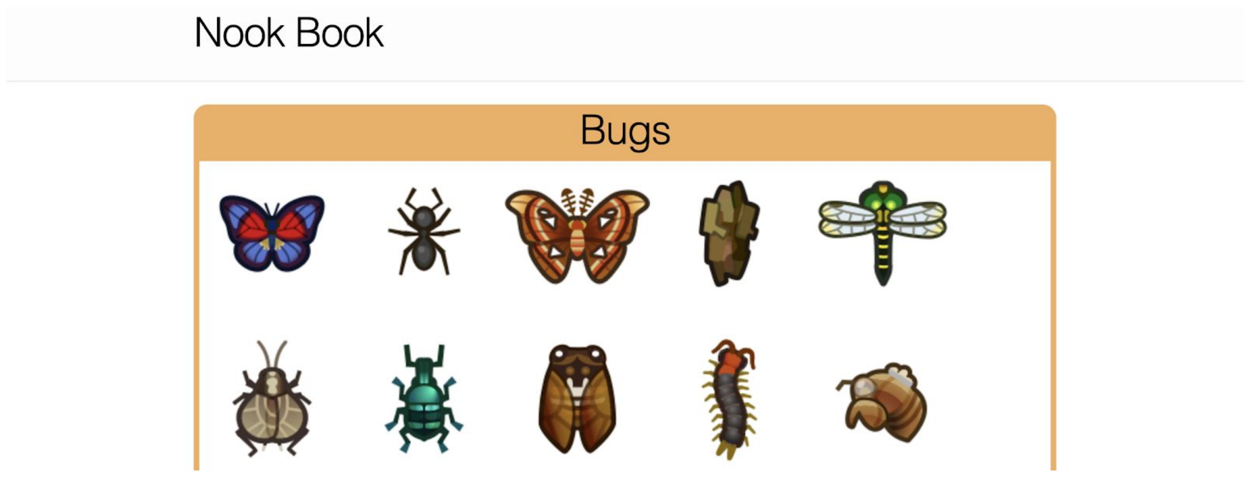
Mnesia: Concept to Reality

During this course we will be building a simple web application with Phoenix 1.5 and LiveView. This is not a course on LiveView but as it is now part of Phoenix and this is a simple web application, we will be using it. The application will not have a traditional database such as PostgreSQL, instead it will use a public API to pull information about the game Animal Crossing New Horizons for display. As we don't want to hit the API on every request, we will use Mnesia to build a simple cache. If you find copy/pasting from this PDF difficult you may access a GoogleDoc instead.





Build the Nook Book Application

Step 1: Project Setup

```
mix phx.new nook_book --no-ecto --live
cd nook_book
iex -S mix phx.server
^c^c
mkdir -p mnesia/data
```

Edit “mnesia/data/.gitignore”

```
*
!.gitignore
```

Edit “.gitignore” and add to the bottom:

```
.elixir_ls/
```

Add to “config/config.exs” before loading the environment specific file:

```
config :mnesia, :dir, 'mnesia/data'
```

Initialize git and commit what we have so far.

```
git init -b main
git add .
git commit -m "Step 1"
```

(Note if you have an older version of git without the -b option you can do this)

```
git init
git branch -m main
git add .
git commit -m "Step 1"
```

(Note in either method you can choose a main branch name that suits you or keep the default)

Step 2: Mnesia Schema Setup

Create a new file “lib/nook_book/data/setup.ex”

```
defmodule NookBook.Data.Setup do
  def setup() do
    :mnesia.start()
    create_schema()
  end

  def create_schema() do
    case schema_exists_anywhere?() do
      true ->
        {:ok, :already_created}

      false ->
        Enum.each(Node.list([:visible]), fn n -> Node.spawn_link(n,
&:mnesia.stop/0) end)
        :mnesia.stop()
        :mnesia.create_schema(nodes())
        :mnesia.start()
        Enum.map(Node.list([:visible]), fn n -> Node.spawn_link(n,
&:mnesia.start/0) end)
    end
  end

  def nodes(), do: [node() | Node.list([:visible])]

  def schema_exists_anywhere?() do
    {answers, _} = :rpc.multicall(nodes(), NookBook.Data.Setup,
:schema_exists?, [])
    Enum.any?(answers, fn x -> x end)
  end

  def schema_exists?() do
    :mnesia.table_info(:schema, :disc_copies) != []
  end
end
```

Edit “lib/nook_book/application.ex” and add this line to the beginning of the `start/2` function:

```
NookBook.Data.Setup.setup()
```

Step 3: Table Setup

Create a new file “lib/nook_book/data/table_behaviour.ex”

```elixir
defmodule NookBook.Data.TableBehaviour do
  @callback table_name() :: atom()
  @callback table_type() :: atom()
  @callback table_fields() :: [atom()]
  @callback table_indexes() :: [atom()]
end
```

Create a new file “lib/nook_book/data/generic_cache.ex”

```elixir
defmodule NookBook.Data.GenericCache do
  @behaviour NookBook.Data.TableBehaviour
  require Record

  Record.defrecord(:generic_cache, key: nil, value: nil)

  def table_name(), do: :generic_cache
  def table_type(), do: :set
  def table_fields(), do: [:key, :value]
  def table_indexes(), do: []
end
```

Edit “lib/nook_book/data/setup.ex”
Add near the top:

```elixir
@tables [
  NookBook.Data.GenericCache
]
```

Add to the end of “setup/0”

```elixir
create_tables()
```

Add these functions to the module:

```elixir
def create_tables() do
  @tables
  |> Enum.each(&create_table/1)
end

def create_table(module) do
  case table_exists?(module.table_name()) do
    true ->
      {:ok, :already_created}
    false ->
      :mnesia.create_table(
        module.table_name(),
        attributes: module.table_fields(),
        type: module.table_type(),
        index: module.table_indexes(),
        disc_copies: nodes()
      )
  end
end

def table_exists?(table_name) do
  Enum.member?(:mnesia.system_info(:tables), table_name)
end

def wait_for_tables() do
  :mnesia.wait_for_tables(table_names(), 10_000)
end

def table_names(), do: @tables |> Enum.map(&apply(&1, :table_name, []))
```

Step 4: Create a Repo for Easier Mnesia Access

Create a new file: “lib/data/repo.ex”

```
defmodule NookBook.Data.Repo do
  def read(table, key) do
    :mnesia.transaction(fn -> :mnesia.read(table, key) end)
  end

  def write(record) do
    :mnesia.transaction(fn -> :mnesia.write(record) end)
  end

  def delete(table, key) do
    :mnesia.transaction(fn -> :mnesia.delete({table, key}) end)
  end

  def clear(table) do
    :mnesia.clear_table(table)
  end

  def all(table) do
    :mnesia.transaction(fn ->
      :mnesia.match_object(table, :mnesia.table_info(table, :wild_pattern),
:read)
    end)
  end

  def filter(table, pattern) do
    :mnesia.transaction(fn ->
      :mnesia.match_object(table, pattern, :read)
    end)
  end
end
```

Step 5: Expand the GenericCache Module

Edit “lib/nook_book/data/generic_cache.ex” and add these functions

```
alias NookBook.Data.Repo

def get(key) do
  case Repo.read(table_name(), key) do
    {:atomic, [data]} -> generic_cache(data, :value)
    {:atomic, []} -> nil
  end
end

def set(_key, nil), do: nil

def set(key, value) do
  Repo.write(generic_cache(key: key, value: value))
  value
end

def remove(key) do
  Repo.delete(table_name(), key)
end

def clear() do
  Repo.clear(table_name())
end

def all() do
  table_name()
  |> Repo.all()
  |> case do
    {:atomic, list} ->
      list
      |> Enum.map(&generic_cache(&1, :value))

    _ ->
      []
  end
end

def filter(pattern) do
```

```
  table_name()
  |> Repo.filter({table_name(), pattern, :_})
  |> case do
    {:atomic, list} ->
      list
      |> Enum.map(&generic_cache(&1, :value))

    _ ->
      []
  end
end
```

Step 6: A Simple API Client

We'll use Tesla to build an API client for the API at https://acnhapi.com/
Add to "mix.exs" in the "deps/0" function:

```
{:tesla, "~> 1.3.0"}
```

Get the dependency

```
mix deps.get
```

Create a new file "lib/nook_book/acnh/api/client.ex"

```
defmodule NookBook.ACNH.API.Client do
  require Logger
  alias NookBook.ACNH.API.Client.Private
  use Tesla
  plug(Tesla.Middleware.BaseUrl, "https://acnhapi.com/v1a")
  plug(Tesla.Middleware.JSON)

  def bugs() do
    "/bugs/"
    |> get()
    |> Private.unwrap_response()
  end

  def bug(id) do
    "/bugs/#{id}"
    |> get()
    |> Private.unwrap_response()
  end

  def bug_icon(id) do
    "/icons/bugs/#{id}"
    |> get()
    |> Private.unwrap_response()
  end

  def bug_image(id) do
    "/images/bugs/#{id}"
    |> get()
    |> Private.unwrap_response()
  end
```

```elixir
def fish() do
  "/fish/"
  |> get()
  |> Private.unwrap_response()
end

def fish(id) do
  "/fish/#{id}"
  |> get()
  |> Private.unwrap_response()
end

def fish_icon(id) do
  "/icons/fish/#{id}"
  |> get()
  |> Private.unwrap_response()
end

def fish_image(id) do
  "/images/fish/#{id}"
  |> get()
  |> Private.unwrap_response()
end

def sea_creatures() do
  "/sea/"
  |> get()
  |> Private.unwrap_response()
end

def sea_creature(id) do
  "/sea/#{id}"
  |> get()
  |> Private.unwrap_response()
end

def sea_creature_icon(id) do
  "/icons/sea/#{id}"
  |> get()
  |> Private.unwrap_response()
end
```

```elixir
  def sea_creature_image(id) do
    "/images/sea/#{id}"
    |> get()
    |> Private.unwrap_response()
  end

  defmodule Private do
    def unwrap_response({:ok, %Tesla.Env{body: body, status: 200}}), do:
body

    def unwrap_response(response) do
      response
      |> inspect()
      |> Logger.info()

      nil
    end
  end
end
```

Step 7: Create a Simple Cache

We will use our Mnesia cache table and the API client to create a simple cache.
Create a new file “lib/nook_book/acnh/cache.ex”

```
defmodule NookBook.ACNH.Cache do
  alias NookBook.ACNH.Cache.Private

  def bugs() do
    Private.list_from_cache(:bug, :bugs)
  end

  def bug(id) do
    Private.record_from_cache(:bug, :bug, id)
  end

  def bug_icon(id) do
    Private.record_from_cache(:bug_icon, :bug_icon, id)
  end

  def bug_image(id) do
    Private.record_from_cache(:bug_image, :bug_image, id)
  end

  def fish() do
    Private.list_from_cache(:fish, :fish)
  end

  def fish(id) do
    Private.record_from_cache(:fish, :fish, id)
  end

  def fish_icon(id) do
    Private.record_from_cache(:fish_icon, :fish_icon, id)
  end

  def fish_image(id) do
    Private.record_from_cache(:fish_image, :fish_image, id)
  end

  def sea_creatures() do
    Private.list_from_cache(:sea_creature, :sea_creatures)
```

```elixir
  end

  def sea_creature(id) do
    Private.record_from_cache(:sea_creature, :sea_creature, id)
  end

  def sea_creature_icon(id) do
    Private.record_from_cache(:sea_creature_icon, :sea_creature_icon, id)
  end

  def sea_creature_image(id) do
    Private.record_from_cache(:sea_creature_image, :sea_creature_image, id)
  end

  defmodule Private do
    alias NookBook.ACNH.API.Client, as: Client
    alias NookBook.Data.GenericCache, as: Cache

    def list_from_cache(namespace, function) do
      {namespace, :_}
      |> Cache.filter()
      |> case do
        [] ->
          apply(Client, function, [])
          |> Enum.reject(fn record -> is_nil(record["id"]) end)
          |> Enum.map(&Cache.set({namespace, &1["id"]}, &1))

        list ->
          list
      end
      |> Enum.sort_by(fn record ->
String.capitalize(record["name"]["name-USen"]) end)
    end

    def record_from_cache(namespace, function, id) do
      {namespace, id}
      |> Cache.get()
      |> case do
        nil ->
          record = apply(Client, function, [id])
          Cache.set({namespace, id}, record)
        record ->
          record
      end
    end
  end
end
```

Step 8: Create a LiveView for the Application

Create a new file “lib/nook_book_web/live/nook_book_live.ex”

```
defmodule NookBookWeb.NookBookLive do
  use NookBookWeb, :live_view
  alias NookBook.ACNH.Cache, as: Cache

  @collection_functions %{
    "bugs" => %{list: &Cache.bugs/0, record: &Cache.bug/1},
    "fish" => %{list: &Cache.fish/0, record: &Cache.fish/1},
    "sea_creatures" => %{list: &Cache.sea_creatures/0, record:
&Cache.sea_creature/1}
  }

  @impl true
  def mount(_params, _session, socket) do
    {:ok,
     assign(socket, %{
       toggles: %{
         "bugs" => false,
         "fish" => false,
         "sea_creatures" => false
       },
       collections: %{
         "bugs" => [],
         "fish" => [],
         "sea_creatures" => []
       },
       record: nil
     })}
  end

  @impl true
  def handle_event("toggle", %{"collection" => name}, socket) do
    section_collection =
      if !socket.assigns.toggles[name] do
        @collection_functions[name].list.()
        |> Enum.map(fn record ->
          %{
            id: record["id"],
            name: String.capitalize(record["name"]["name-USen"]),
```

```elixir
            icon: "/icons/#{name}/#{record["id"]}"
          }
        end)
      else
        []
      end

    {:noreply,
     assign(socket, %{
       toggles: Map.put(socket.assigns.toggles, name,
!socket.assigns.toggles[name]),
       collections: Map.put(socket.assigns.collections, name,
section_collection)
     })}
  end

  @impl true
  def handle_event("show", %{"collection" => name, "id" => id}, socket) do
    record =
      case @collection_functions[name].record.(String.to_integer(id)) do
        nil ->
          nil

        record ->
          %{
            name: String.capitalize(record["name"]["name-USen"]),
            image: "/images/#{name}/#{record["id"]}",
            price: record["price"]
          }
      end

    {:noreply, assign(socket, %{record: record})}
  end

  @impl true
  def handle_event("hide", _params, socket) do
    {:noreply, assign(socket, %{record: nil})}
  end
end
```

Step 9: Create a Controller for Our Images

We will be serving our images straight out of the Mnesia Cache. A special controller is needed to do this.
Create a new file “lib/nook_book_web/controllers/image_controller.ex”

```
defmodule NookBookWeb.ImageController do
  use NookBookWeb, :controller
  alias NookBook.ACNH.Cache, as: Cache

  @collection_functions %{
    "bugs" => %{icon: &Cache.bug_icon/1, image: &Cache.bug_image/1},
    "fish" => %{icon: &Cache.fish_icon/1, image: &Cache.fish_image/1},
    "sea_creatures" => %{icon: &Cache.sea_creature_icon/1, image:
&Cache.sea_creature_image/1}
  }

  def icon(conn, %{"namespace" => namespace, "id" => id}) do
    file =
      id
      |> String.to_integer()
      |> @collection_functions[namespace].icon.()

    conn
    |> put_resp_content_type("image/png", "utf-8")
    |> send_resp(200, file)
  end

  def image(conn, %{"namespace" => namespace, "id" => id}) do
    file =
      id
      |> String.to_integer()
      |> @collection_functions[namespace].image.()

    conn
    |> put_resp_content_type("image/png", "utf-8")
    |> send_resp(200, file)
  end
end
```

Step 10: Update Our Router

Edit “lib/nook_book_web/router.ex”
In the scope “/”, NookBookWeb
Delete:

```
live "/", PageLive, :index
```

Add:

```
live("/", NookBookLive, :index)
get("/icons/:namespace/:id", ImageController, :icon)
get("/images/:namespace/:id", ImageController, :image)
```

Step 11: Build Out the HTML

Edit “lib/nook_book_web/templates/layout/root.html.leex”
Replace the body with:

```
<body>
  <header>
    <section class="container">
      <h1>Nook Book</h1>
    </section>
  </header>
  <%= @inner_content %>
  <footer>
    <%= if function_exported?(Routes, :live_dashboard_path, 2) do %>
      <%= link "LiveDashboard", to: Routes.live_dashboard_path(@conn,
:home) %></li>
    <% end %>
  </footer>
</body>
```

Delete the files:
“lib/nook_book_web/live/page_live.ex”
“lib/nook_book_web/live/page_live.html.leex”
Create a new file “lib/nook_book_web/live/nook_book_live.html.leex”

```
<div>
  <div>
    <h1 class="collection-header" phx-click="toggle"
phx-value-collection="bugs">Bugs</h1>
    <div>
      <ul class="collection">
        <%= for bug <- @collections["bugs"] do %>
          <li><img src="<%= bug.icon %>" alt="<%= bug.name %>" title="<%=
bug.name %>" phx-click="show" phx-value-collection="bugs" phx-value-id="<%=
bug.id %>"/></li>
        <% end %>
      </ul>
    </div>
  </div>
  <div>
    <h1 class="collection-header" phx-click="toggle"
phx-value-collection="fish">Fish</h1>
    <div>
```

```
      <ul class="collection">
        <%= for fish <- @collections["fish"] do %>
          <li><img src="<%= fish.icon %>" alt="<%= fish.name %>" title="<%=
fish.name %>" phx-click="show" phx-value-collection="fish"
phx-value-id="<%= fish.id %>" /></li>
        <% end %>
      </ul>
    </div>
  </div>
  <div>
    <h1 class="collection-header" phx-click="toggle"
phx-value-collection="sea_creatures">Sea Creatures</h1>
    <div>
      <ul class="collection">
        <%= for sea_creature <- @collections["sea_creatures"] do %>
          <li><img src="<%= sea_creature.icon %>" alt="<%=
sea_creature.name %>" title="<%= sea_creature.name %>" phx-click="show"
phx-value-collection="sea_creatures" phx-value-id="<%= sea_creature.id
%>"/></li>
        <% end %>
      </ul>
    </div>
  </div>
<%= if @record do %>
  <div class="record">
    <h2><%= @record.name %> (<%= @record.price %> Bells)</h2><h2
class="close" phx-click="hide">X</h2>
    <img src="<%= @record.image %>"/>
  </div>
<% end %>
</div>
```

Edit the file “assets/css/app.scss”
Add to the bottom:

```
.collection-header {
  background-color: #F2AF5E;
  border-radius: 12px 12px 0px 0px;
  text-align: center;
  cursor: pointer;
  margin-bottom: 0px;
}
```

```css
.collection {
  list-style-type: none;
  border-color: #F2AF5E;
  border-width: 5px;
  border-style: solid;
}

.collection li {
  display: inline-block;
  width: 130px;
  cursor: pointer;
}

.record {
  border-color: #000000;
  border-width: 5px;
  border-style: solid;
  position: absolute;
  top: 100px;
  left: 50px;
  background-color: #FFFFFF;
  border-radius: 12px;
  width: 600px;
}

.record h2 {
  background-color: #F2AF5E;
  text-align: center;
  width: 550px;
  display: inline-block;
}

.record .close {
  background-color: #000000;
  color: #FFFFFF;
  border-color: #000000;
  text-align: center;
  width: 40px;
  display: inline-block;
  cursor: pointer;
}
```

Step 12: Setup Local Multi-Node Mnesia

We can run mnesia locally in a multi-node cluster but it takes some additional setup.
Create 2 new directories:

```
mkdir mnesia/data/n1
mkdir mnesia/data/n2
```

Copy the “mnesia/data/.gitignore” to both new directories:

```
cp mnesia/data/.gitignore mnesia/data/n1/.gitignore
cp mnesia/data/.gitignore mnesia/data/n2/.gitignore
```

Edit “config/config.exs”
Add add a line before the env specific import

```
config :nook_book, cluster_role: :primary, primary_node: :"n1@127.0.0.1"
```

Edit “config/dev.exs”
Change the NookBookWeb.Endpoint configuration:

```
config :nook_book, NookBookWeb.Endpoint,
  http: [port: System.get_env("PORT") || 4000],
  debug_errors: true,
  code_reloader: true,
  check_origin: false,
  watchers: [
    node: [
      "node_modules/webpack/bin/webpack.js",
      "--mode",
      "development",
      "--watch-stdin",
      cd: Path.expand("../assets", __DIR__)
    ]
  ]
```

Allow getting the role from the environment

```
config :nook_book, cluster_role: System.get_env("CLUSTER_ROLE", "primary")
|> String.to_atom()
```

Override the mnesia data location based on node name

```
config :mnesia,
       :dir,
       'mnesia/data' ++
         '/' ++
         (node() |> Atom.to_string() |> String.split("@") |> List.first()
|> String.to_charlist())
```

Edit “mix.exs”
Modify “application/0” to add a start_phase

```
def application do
  [
    mod: {NookBook.Application, []},
    extra_applications: [:logger, :runtime_tools],
    start_phases: [init: []]
  ]
end
```

Edit “lib/nook_book/application.ex”
Remove the call to “NookBook.Data.Setup.setup/1” from “start/2”
Add a new function

```
def start_phase(:init, :normal, _) do
  Application.get_env(:nook_book, :cluster_role)
  |> NookBook.Data.Setup.setup()
end
```

Edit “lib/nook_book/data/setup.ex”
Add an import

```
import Logger
```

Replace the “setup/0” function with two “setup/1” functions

```
def setup(:primary) do
  Logger.info("Setting up mnesia for primary node")
  :mnesia.start()
  create_schema()
  create_tables()
end

def setup(:member) do
  Node.connect(Application.get_env(:nook_book, :primary_node))
  Logger.info("Setting up mnesia for member node, cluster peers:")
  Logger.info(inspect(nodes()))
  [existing_node | _] = Node.list([:visible])
  name = Node.self()

  :mnesia.start()
  {:ok, _} = :rpc.call(existing_node, :mnesia, :change_config,
[:extra_db_nodes, [name]])

  :mnesia.change_table_copy_type(:schema, name, :disc_copies)
  :mnesia.add_table_copy(:schema, name, :disc_copies)
  sync_remote_tables_to_local_disk()
end
```

The new ":member" version calls another new function so we'll add it

```
def sync_remote_tables_to_local_disk() do
  name = Node.self()

  :mnesia.system_info(:tables)
  |> Enum.each(fn table ->
    case Node.self() in :mnesia.table_info(table, :disc_copies) do
      true ->
        :ok

      false ->
        Logger.info("Syncing #{table}")
        :mnesia.add_table_copy(table, name, :disc_copies)
    end
  end)
end
```

Finally, let's fire up the two nodes in two separate console windows!

```
PORT=4000 CLUSTER_ROLE=primary iex --name n1@127.0.0.1 --cookie secret -S
mix phx.server
PORT=4001 CLUSTER_ROLE=member iex --name n2@127.0.0.1 --cookie secret -S
mix phx.server
```

Deploy Nook Book to AWS

Step 13: Setup Release

From the command line, run:

```
mix release.init
rm rel/env.bat.eex
```

Edit “config/prod.exs” and remove the last line, which references “prod.secrets.exs”

Create and edit the file “config/releases.exs”

```
import Config

config :nook_book, NookBookWeb.Endpoint,
  server: true,
  http: [port: 4000],
  url: [host: "localhost"],
  secret_key_base:
"p6Ws6Q1cFzoCUjkOhiBsKwjXiRpJ/E26ryJ7tTzXVrZsavqS2RG3eznc4Mp2KubF"
```

Update “mix.exs” by adding “releases” to the project keyword list,

```
  def project do
    [
      ...
      aliases: aliases(),
      deps: deps(),
      releases: releases()
    ]
  end
```

and by implementing the “releases” function,

```
  defp releases do
    [
      nook_book: [
        include_executables_for: [:unix],
        steps: [:assemble, :tar]
      ]
    ]
  end
```

and by adding “mnesia” to the “included_applications” under the application keyword list:

```
def application do
  [
    mod: {NookBook.Application, []},
    extra_applications: [:logger, :runtime_tools],
    included_applications: [:mnesia],
    start_phases: [init: []]
  ]
end
```

Step 14: Setup Circle

Create and edit the file ".circleci/config.yml"

```
version: 2
jobs:
  build:
    docker:
      - image: ckhrysze/elixir_for_amazonlinux2

    environment:
      MIX_ENV: prod
    working_directory: ~/repo
    steps:
      - checkout

      - run: mix local.hex --force
      - run: mix local.rebar
      - run: mix deps.get
      - run: mix compile
      - run: npm install --prefix ./assets
      - run: npm run deploy --prefix ./assets
      - run: mix phx.digest
      - run: mix release
      - run: cp _build/prod/nook_book-0.1.0.tar.gz nookbook.tgz

      - store_artifacts:
          path: nookbook.tgz
```

On the console, locally commit the changes

```
git add .circleci
git add rel
git commit -am "Step 2, add release setup and circleci config"
```

The following steps depend heavy on previous actions...with a fresh github user:

- In a browser at github, create a new empty repo called nook_book
- On the console:

```
git remote add origin <repo link>
git push origin main
```

- In a browser at circleci, sign in with your github account

- If prompted, click authorize circleci
- If prompted, click your github user account at “Select an organization”
- Click the blue “Set Up Project” button
- Click the gray “Use Existing Config” button
- Click the blue “Start Building” button
- If prompted, click the blue “Proceed to New UX” button

Step 15: Setup AWS

Using the credentials emailed beforehand, sign in to:
https://797420293962.signin.aws.amazon.com/console

If you haven't already, install the AWS CLI tool:
https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html

Create access key for the cli tool:

- In a browser, navigate to the IAM Dashboard
- On the left sidebar, click "Users"
- Click on your user name
- In the central tab bar, click "Security credentials"
- Under the "Access keys" header, click the "Create access key" button
- Save the key and secret, or download the csv file, to setup the CLI tool

Back on a console, run 'aws configure', which will prompt for those credentials as well as region and output format:

```
aws configure
AWS Access Key ID [None]: < access >
AWS Secret Access Key [None]: < secret >
Default region name [None]: us-east-2
Default output format [None]: json
```

Step 16: Setup Terraform

Edit “.gitignore”, add non versioned Terraform files at the bottom:

```
# Terraform files
.terraform
terraform.tfstate
terraform.tfvars
*.tfstate*
terraform-provider*v*
current.plan
```

Create and edit “infrastructure/main.tf”

```
variable "name" { default = "< name >" }
variable "nookbook_vpc_id" { default = "vpc-0dc86c5a01fcfc699" }

provider "aws" { region = "us-east-2" }

resource "aws_security_group" "nookbook" {
  name   = "${var.name}-nookbook-application"
  vpc_id = var.nookbook_vpc_id
}
```

On the console:

```
cd infrastructure
terraform init
alias tfp='terraform plan -out=current.plan'
alias tfa='terraform apply -input=true current.plan'
tfp
tfa
```

Step 17: Implement Security Group

Edit “infrastructure/main.tf”, update the aws_security_group nookbook resource:

```
resource "aws_security_group" "nookbook" {
  name   = "${var.name}-nookbook-application"
  vpc_id = var.nookbook_vpc_id

  ingress {
    protocol    = "tcp"
    from_port   = 22
    to_port     = 22
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    protocol    = "tcp"
    from_port   = 4000
    to_port     = 4000
    cidr_blocks = ["10.0.0.0/16"]
  }

  ingress {
    protocol    = "tcp"
    from_port   = 4369
    to_port     = 4369
    cidr_blocks = ["10.0.0.0/16"]
    description = "Default EPMD port"
  }

  ingress {
    protocol    = "tcp"
    from_port   = 20000
    to_port     = 20099
    cidr_blocks = ["10.0.1.0/24"]
    description = "EPMD post connect mapping"
  }

  egress {
    protocol    = "-1"
    from_port   = 0
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

On the console, from the infrastructure directory, apply the changes:

```
tfp
tfa
```

Step 18: Update Release Files

Edit “rel/env.sh.eex”, add to the bottom:

```
mkdir -p mnesia/data
export RELEASE_DISTRIBUTION=name
export RELEASE_NODE=<%= @release.name %>@`(curl
http://169.254.169.254/latest/meta-data/local-ipv4)`
```

Edit “rel/vm.args.eex”, add to the bottom:

```
-kernel inet_dist_listen_min 20000
-kernel inet_dist_listen_max 20099
```

Step 19: Finish Infrastructure Setup

Edit “infrastructure/main.tf”, near the top add the additional variables:

```
variable "nookbook_alb_listener" { default =
"arn:aws:elasticloadbalancing:us-east-2:797420293962:listener/app/app-alb/0
6ba40177a02a85d/8970cd65f589522b" }
variable "nookbook_private_subnet" { default = "subnet-00deae77515e96e8c" }
```

and at the bottom add:

```
resource "aws_lb_target_group" "nookbook_group" {
  name     = "${var.name}-nookbook-target"
  port     = 4000
  protocol = "HTTP"
  vpc_id   = var.nookbook_vpc_id
}

resource "aws_lb_listener_rule" "static" {
  listener_arn = var.nookbook_alb_listener

  action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.nookbook_group.arn
  }

  condition {
    host_header {
      values = ["${var.name}.nookbook.online"]
    }
  }
}

resource "aws_instance" "nookbook1" {
  instance_type          = "t2.micro"
  ami                    = "ami-016b213e65284e9c9"
  vpc_security_group_ids = ["${aws_security_group.nookbook.id}"]
  subnet_id              = var.nookbook_private_subnet
  tags = {
    Name = "${var.name}-instance-1"
  }
}

resource "aws_instance" "nookbook2" {
  instance_type          = "t2.micro"
  ami                    = "ami-016b213e65284e9c9"
  vpc_security_group_ids = ["${aws_security_group.nookbook.id}"]
  subnet_id              = var.nookbook_private_subnet
  tags = {
    Name = "${var.name}-instance-2"
  }
}

resource "aws_lb_target_group_attachment" "attachment1" {
  target_group_arn = aws_lb_target_group.nookbook_group.arn
  target_id        = aws_instance.nookbook1.id
  port             = 4000
}

resource "aws_lb_target_group_attachment" "attachment2" {
  target_group_arn = aws_lb_target_group.nookbook_group.arn
  target_id        = aws_instance.nookbook2.id
  port             = 4000
}
```

On the console, from the infrastructure directory, apply the changes:

```
tfp
tfa
```

Step 20: Use Terraform Output for Hosts

Add and edit “infrastructure/output.tf”:

```
output "libcluster-hosts" {
value = <<HOSTS

hosts: [
  :"nook_book@${aws_instance.nookbook1.private_ip}",
  :"nook_book@${aws_instance.nookbook2.private_ip}"
]
HOSTS
}
```

On the console, from the infrastructure directory, apply the changes:

```
tfp
tfa
```

The output from the terraform apply will be used in the next step.

Step 21: Setup Libcluster

Edit “mix.exs”, add to the deps keyword list:

```
{:libcluster, "~> 3.2"}
```

Edit “lib/nook_book/application.ex”, add to children array within start/2:

```
{Cluster.Supervisor, [Application.get_env(:libcluster, :topologies), [name:
NookBook.ClusterSupervisor]]}
```

Edit “lib/nook_book/data/setup.ex”, from the “:member” setup function, remove the line:

```
Node.connect(Application.get_env(:nook_book, :primary_node))
```

Edit “config/releases.exs”, add:

```
config :libcluster,
  topologies: [
    nook_book: [
      strategy: Cluster.Strategy.Epmd,
      config: [
        <output from step 20>
      ]
    ]
  ]
```

Also add:

```
name = "<name>"

config :nook_book,
  cluster_role: System.get_env("CLUSTER_ROLE", "member") |>
String.to_atom(),
  base_uri: "http://#{name}.nookbook.online"
```

And update the endpoint url host:

```
config :nook_book, NookBookWeb.Endpoint,
  server: true,
  http: [port: 4000],
  url: [host: "#{name}.nookbook.online"],
  secret_key_base:
"p6Ws6Q1cFzoCUjkOhiBsKwjXiRpJ/E26ryJ7tTzXVrZsavqS2RG3eznc4Mp2KubF"
```

Step 22: Use Terraform Output for SSH Config

On the console:

```
ssh-keygen -t rsa -b 4096 -C "<key name>" -f $HOME/.ssh/<key name>
```

Edit “infrastructure/output.tf”, add:

```
variable "sshkey" { default = "<key name>" }
variable "bastion_id" { default = "i-0f592d9fe5e38f33c" }

output "sshconfig" {
  value = <<SSHCONF

Host nookbook_bastion
  User ec2-user
  Hostname 18.188.0.20
  IdentityFile ~/.ssh/${var.sshkey}

Host nookbook1
  User ec2-user
  Hostname ${aws_instance.nookbook1.private_ip}
  ProxyCommand ssh -q -W %h:%p nookbook_bastion
  IdentityFile ~/.ssh/${var.sshkey}

Host nookbook2
  User ec2-user
  Hostname ${aws_instance.nookbook2.private_ip}
  ProxyCommand ssh -q -W %h:%p nookbook_bastion
  IdentityFile ~/.ssh/${var.sshkey}
SSHCONF
}

output "ssh-commands" {
  value = <<SSHCMDS

aws ec2-instance-connect send-ssh-public-key --instance-id
${var.bastion_id}  --instance-os-user ec2-user --ssh-public-key
file://~/.ssh/${var.sshkey}.pub --availability-zone us-east-2a | cat
aws ec2-instance-connect send-ssh-public-key --instance-id
${aws_instance.nookbook1.id}  --instance-os-user ec2-user --ssh-public-key
file://~/.ssh/${var.sshkey}.pub --availability-zone us-east-2c | cat
aws ec2-instance-connect send-ssh-public-key --instance-id
${aws_instance.nookbook2.id}  --instance-os-user ec2-user --ssh-public-key
file://~/.ssh/${var.sshkey}.pub --availability-zone us-east-2c | cat
SSHCMDS
}
```

On the console, from the infrastructure directory:

```
tfp
tfa
terraform output sshconfig
```

Take the output from the last command and add it to the end of “~/.ssh/config”

Step 23: Build and Download Release

On the console, from the base directory, commit to github:

```
git add infrastructure/
git commit -am "Step 23, build the final release"
git push origin main
```

In a browser, go to the build [at circleci]()

Once the build is complete, click on build

Click on the ‘ARTIFACTS’ tab

Click “nookbook.tgz” to download the release

Step 24: Upload and Run the Release

On the console from the infrastructure directory, use the terraform output to setup aws ssh connectivity

```
terraform output ssh-commands | bash
```

scp the release to each server

```
scp ~/Downloads/nookbook.tgz nookbook1:~/
scp ~/Downloads/nookbook.tgz nookbook2:~/
```

On the console in two windows:

```
ssh nookbook1
```

```
ssh nookbook2
```

From nookbook1

```
tar -zxf nookbook.tgz
CLUSTER_ROLE=primary ./bin/nook_book start
```

From nookbook2

```
tar -zxf nookbook.tgz
./bin/nook_book start
```