

Configuring Your Release

...

Release the Kraken... er Elixir - Part 2

Part 2.

- Environment variable extravaganza
- Clustering
- Release.init / general config
- Phoenix setup

Environmental Extravaganza

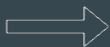
How do environment vars make it into our Application?

Build Time

config.exs

dev.exs

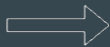
prod.exs



Run Time

releases.exs

env.sh.eex



Call Time

```
Application.get_env(:web_app, :key)
```

A Tale of Two Config Modules.

- ``use Mix.Config`` is only available at build time.
- ``import Config`` is NEW and can access environment variables at run time!

All the Gotchas - compile time vs run time is tricky

```
config :phx_example_app, :var, System.get_env("VAR")
```

```
defmodule PhxExampleApp.Example do
  @var Application.get_env(:phx_example_app, :var)

  def read do
    @var
  end
end
```

Config Providers - mix.exs

```
releases: [  
  web_app: [  
    config_providers: [{JSONConfigProvider, "/etc/config.json"}]  
    include_executables_for: [:unix],  
  ]  
]
```

Config Providers - the provider implementation

```
defmodule JSONConfigProvider do
  @behaviour Config.Provider

  # Let's pass the path to the JSON file as config
  def init(path) when is_binary(path), do: path

  def load(config, path) do
    # We need to start any app we may depend on.
    {:ok, _} = Application.ensure_all_started(:jason)

    json = path |> File.read!() |> Jason.decode!()

    Config.Reader.merge(
      config,
      my_app: [
        some_value: json["my_app_some_value"],
        another_value: json["my_app_another_value"],
      ]
    )
  end
end
```


Distributed Releases

Requirements.

- Shared cookie (can be set with RELEASE_COOKIE)
- A chosen distribution type (sname, name)
- A node name (RELEASE_NAME)
- Open communication on EPMD ports (4369, et al)

Helpful Commands.

- `Node.list` - lists connected known nodes
- `Node.ping` - tries to communicate with a node (and clusters them together if it can)

Release Config

mix.exs - release settings, umbrella config, scripts


```
def project do
  [
    app: :phx_example_app,
    version: "0.1.0",
    elixir: "~> 1.5",
    elixirc_paths: elixirc_paths(Mix.env()),
    compilers: [:phoenix, :gettext] ++ Mix.compilers(),
    start_permanent: Mix.env() == :prod,
    deps: deps(),
    releases: [
      web_app: [
        include_executables_for: [:unix],
        applications: [runtime_tools: :permanent],
        steps: [:assemble, &copy_scripts/1],
      ]
    ]
  ]
end
```



<https://hexdocs.pm/mix/Mix.Tasks.Release.html#module-customization>

`mix release.init` - generates basic config niceties

`rel`



- `env.bat.eex`
- `env.sh.eex`
- `vm.args.eex`

vm.args - configure the Erlang runtime

```
## Customize flags given to the VM: http://erlang.org/doc/man/erl.html
## -mode/-name/-sname/-setcookie are configured via env vars, do not set them here

-kernel inet_dist_listen_min 9100 inet_dist_listen_max 9155

## Number of dirty schedulers doing IO work (file, sockets, etc)
##+SDio 5

## Increase number of concurrent ports/sockets
##+Q 65536

## Tweak GC to run more often
##-env ERL_FULLSWEEP_AFTER 10
```

env.sh(bat) - some environment variable setup

```
#!/bin/sh

# Sets and enables heart (recommended only in daemon mode)
# case $RELEASE_COMMAND in
#   daemon*)
#     HEART_COMMAND="$RELEASE_ROOT/bin/$RELEASE_NAME $RELEASE_COMMAND"
#     export HEART_COMMAND
#     export ELIXIR_ERL_OPTIONS="-heart"
#     ;;
#   *)
#     ;;
# esac

# Set the release to work across nodes. If using the long name format like
# the one below (my_app@127.0.0.1), you need to also uncomment the
# RELEASE_DISTRIBUTION variable below.
# export RELEASE_DISTRIBUTION=name
export RELEASE_NODE=$NAME
```

<https://hexdocs.pm/mix/Mix.Tasks.Release.html#module-vm-args-and-env-sh-env-bat>

Phoenix

Required Changes.

- Set the server option for your endpoint
(`config/prod.secret.exs`)
- Update ``use Mix.Config`` -> ``import Config``
(`config/prod.secret.exs`)
- Rename `config/prod.secret.exs` -> `config/releases.exs`
- Remove the ``import_config`` line from `config/prod.exs`

Additionally Required Build Commands for Phoenix.

```
# Compile assets  
$ npm run deploy --prefix ./assets  
$ mix phx.digest
```

Objectives.

- Setup a new Phoenix app and build a release for it
- Add / use some environment variables within the new app
- Cluster two running releases of the phoenix app

Resources.

- General Release Setup:
<https://hexdocs.pm/mix/Mix.Tasks.Release.html#module-customization>
- Config Module: <https://hexdocs.pm/elixir/Config.html>
- Config Providers: <https://hexdocs.pm/elixir/Config.Provider.html>
- Phoenix Release Prep: <https://hexdocs.pm/phoenix/releases.html#releases-assemble>