Releases: Part 3

The Environment

Next:

1. The Basics
2. The Application

3. The Environment [you are here]
   A. Learn about the sys.config / vm.args files
   B. Investigate how environment variables interact with our app
   C. Enable use of environment variables in the API

4. Distillery

sys.config
/releases/0.3.0/sys.config

- Comes from the `config/config.exs` file
- Is used with the `-config PATH` option to `erl`
- Docs here: http://erlang.org/doc/man/config.html
- Looks like:

```
[{sasl,[{errlog_type,error}]},{simple_api,[{port,4000}]}].
```

sys.config
/releases/0.3.0/sys.config

- You can “chain” config files:

```
[{sasl,[{errlog_type,error}]},
{simple_api,[{port,4000}]},
"some/path/to/file.config"].
```

- It is also possible to pass config flags on the command line:

```
$ erl -simple_api port 4000
```

sys.config - Distributed Applications

/releases/0.3.0/sys.config

```
[{kernel,
  [
   {sync_nodes_optional,
     [‘a1@127.0.0.1’, ‘a2@127.0.0.1’]},
   {sync_nodes_timeout, 5000}
  ]
 }
].
```

vm.args
/releases/0.3.0/vm.args

- Uses the `-args_file` argument to `erl
- Can contain any VM arg from: http://erlang.org/doc/man/erl.html
- `-name` sets the fully qualified node name, eg: `my_app@127.0.0.1`
- `-cookie` sets the magic erlang communication cookie
- All of these can be passed as command line arguments
- Eg `+Bi`

Application.get_env vs System.get_env

- `Application.get_env` reads from the application config / sys.config file
- `System.get_env` reads from the runtime environment
- `System.get_env` within a mix config file is evaluated at compile time

Let’s talk about runtime configuration:

- Sys.config
- Application init callback
- An external library

Objectives:

- Support the `magic_env_var` in `./lib/simple_api/endpoint.ex` using the config file
- Start the app using the packaged config file, but referencing an environment variable instead of the hardcoded `4000`
- Run two instances of the app with different `name`s, and cluster them