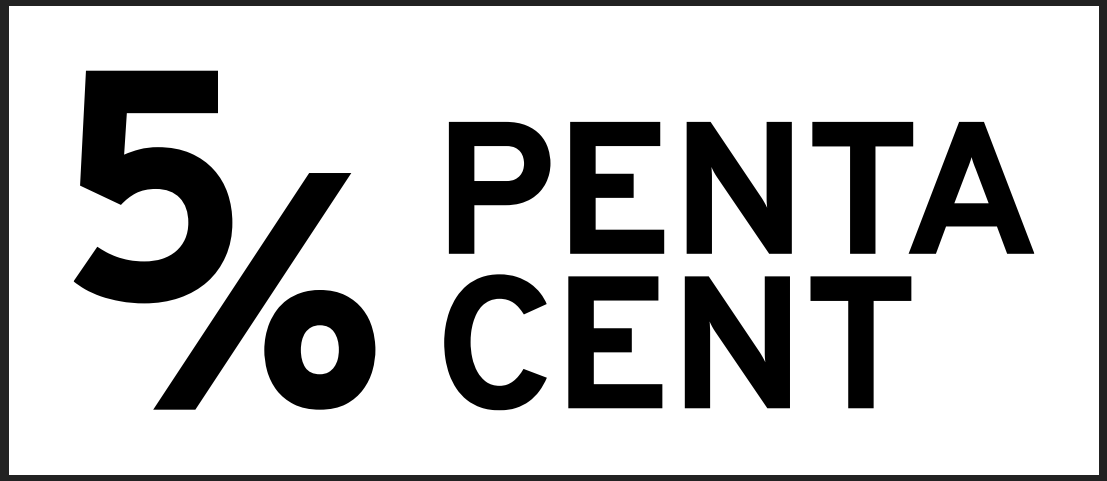
From Zero to Deployment With Elixir

Who am I?

Philipp Schmieder

- Founder of Pentacent
- Tech Lead @ Volt Europa


Volt


5% PENTA
CENT

now.do

Productivity powered by science

Who is This Talk For?

What is This Talk About?

What is This Talk About?

- Simple Atomic Deployments

What is This Talk About?

- Simple Atomic Deployments
- With OTP Releases and Docker

OTP Releases

When you have written one or more applications, you might want to create a complete system with these applications and a subset of the Erlang/OTP applications. This is called a release.

– Chapter 10, OTP Design Principles

The Clock App

From Zero

The Clock App

From Zero

```
$ mix new clock
```

The Clock App

From Zero

```
$ mix new clock
```

```
$ cd clock
$ find . -type f
./mix.exs
./.formatter.exs
./config/config.exs
./test/test_helper.exs
./test/clock_test.exs
./README.md
./.gitignore
./lib/clock.ex
```

The Clock App

OTP Application

The Clock App

OTP Application

```
# mix.exs

defmodule Clock.MixProject do
  def application do
    [
      mod: {Clock, []}
    ]
  end
  # ...
end
```

The Clock App

OTP Application

The Clock App

OTP Application

```
# lib/clock/clock.ex

defmodule Clock do
  use Application

  def start(_type, _args) do
    children = [{Clock.Server, []}]
    Supervisor.start_link(children, strategy: :one_for_one)
  end
end
```

The Clock App

Configuration

The Clock App

Configuration

```
#config/config.exs

use Mix.Config

config :clock,
  interval: 1000
```

Business Logic

```elixir
#lib/clock/server.ex

defmodule Clock.Server do
  use GenServer

  def start_link(_),
    do: GenServer.start_link(__MODULE__, %{})

  def init(_default) do
    state = %{interval: Application.get_env(:clock, :interval)}
    IO.puts("Initialized with #{state[:interval]} ms interval!")
    {:ok, state, 0}
  end

  def handle_info(:timeout, state) do
    date = DateTime.utc_now() |> DateTime.to_iso8601()
    version = Application.spec(:clock, :vsn)
    IO.puts("Clock Server [#{version}]: #{date}")
    {:noreply, state, state[:interval]}
  end
end
```

```
$ mix run --no-halt
```

Let’s Build a Release!

Let’s Build a Release!

```
# mix.exs

defp deps do
  [
    {:distillery, "~> 2.0"}
  ]
end
```

Let’s Build a Release!

```
# mix.exs

defp deps do
  [
    {:distillery, "~> 2.0"}
  ]
end
```

```
$ mix deps.get
```

Let’s Build a Release!

```
# mix.exs

defp deps do
  [
    {:distillery, "~> 2.0"}
  ]
end
```

```
$ mix deps.get
```

```
$ mix release.init
```

Let’s Build a Release!

```
# mix.exs

defp deps do
  [
    {:distillery, "~> 2.0"}
  ]
end
```

```
$ mix deps.get
```

```
$ mix release.init
```

```
$ find .
[…]
./rel/
./rel/config.exs
./rel/vm.args
```

```
$ mix release --verbose
```

What's in a Release?

```
$ find ./_build/prod/rel/clock
[…]
releases/
releases/0.1.0/clock.rel
releases/0.1.0/clock.script
releases/0.1.0/clock.boot
releases/0.1.0/vm.args
releases/0.1.0/hooks/
releases/0.1.0/hooks/pre_start.d/
releases/0.1.0/clock.tar.gz
releases/RELEASES
```

What's in a Release? (cont.)

```
$ find ./_build/prod/rel/clock
[…]
erts-10.3.1/
erts-10.3.1/bin/beam.smp

lib/
lib/iex-1.8.1/
lib/kernel-6.3/
lib/clock-0.1.0/ebin/Elixir.Clock.beam
lib/clock-0.1.0/ebin/clock.app
lib/clock-0.1.0/ebin/Elixir.Clock.Server.beam
lib/elixir-1.8.1/

bin/
bin/clock
```

Release Commands

```
$ bin/clock foreground
```

```
$ bin/clock start|stop|restart|reboot
```

```
$ bin/clock console|console_clean|remote_console
```

Release Configuration

Release Configuration

Environments & Profiles

```
#rel/config.exs

use Mix.Releases.Config,
  default_environment: Mix.env()

environment :dev do
  # ...
end

environment :prod do
  # ...
end

release :clock do
  # ...
end
```

Release Configuration

Options

```
#rel/config.exs

environment :prod do
  set include_src: false
  set dev_mode: false
  set include_erts: true
  set cookie: "${ERLANG_COOKIE}"
end

release :clock do
  set version: current_version(:clock)
  set applications: [
    :runtime_tools
  ]
end
```

Release Configuration

Hooks

Release Configuration

Hooks

- `pre_configure | post_configure`

Release Configuration

Hooks

- `pre_configure | post_configure`
- `pre_start | post_start`

Release Configuration

Hooks

- `pre_configure | post_configure`
- `pre_start | post_start`
- `pre_stop | post_stop`

Release Configuration

Hooks

- `pre_configure | post_configure`
- `pre_start | post_start`
- `pre_stop | post_stop`
- `pre_upgrade | post_upgrade`

```
#rel/hooks/pre_start/hello.sh

#!/bin/bash
echo "Running pre_start hook script ..."
```

```
#rel/hooks/pre_start/hello.sh

#!/bin/bash
echo "Running pre_start hook script ..."
```

```
#rel/config.exs

release :clock do
  set pre_start_hooks: "rel/hooks/pre_start"
  # ...
end
```

Release Configuration

Custom Commands

Release Configuration

Custom Commands

```
#rel/config.exs

release :clock do
  set commands: [
    my_command: 'rel/commands/my_command.sh'
  ]
  # ...
end
```

Application Configuration

REPLACE_OS_VARS

```
#rel/config.exs

environment :prod do
  set cookie: "${ERLANG_COOKIE}"
  # ...
end
```

Application Configuration

Configuration Providers

Application Configuration

Configuration Providers

```
#rel/config.exs

release :clock do
  set overlays: [
    {:copy, "rel/config/config.exs", "etc/config.exs"}
  ]
  set config_providers: [
    {
      Mix.Releases.Config.Providers.Elixir,
      ["${RELEASE_ROOT_DIR}/etc/config.exs"]
    }
  ]
  # ...
end
```

Application Configuration

Configuration Providers

```
#rel/config/config.exs

use Mix.Config

get_env = fn(varname, default) ->
  System.get_env(varname) || default
end

config :clock,
  interval: get_env.("INTERVAL", "1500") |> String.to_integer()
```

Hot Upgrades

Hot Upgrades

The .appup File

```
{Vsn,
 [{UpFromVsn, Instructions}, ...],
 [{DownToVsn, Instructions}, ...]}.
```

Hot Upgrades

The .appup File

```
{Vsn,
 [{UpFromVsn, Instructions}, ...],
 [{DownToVsn, Instructions}, ...]}.
```

```
{"0.2.0",
 [{"0.1.0",[{update,'Elixir.Clock.Server',{advanced,[]},[]}]}],
 [{"0.1.0",[{update,'Elixir.Clock.Server',{advanced,[]},[]}]}]}.
```

Hot Upgrades

Upgrading State

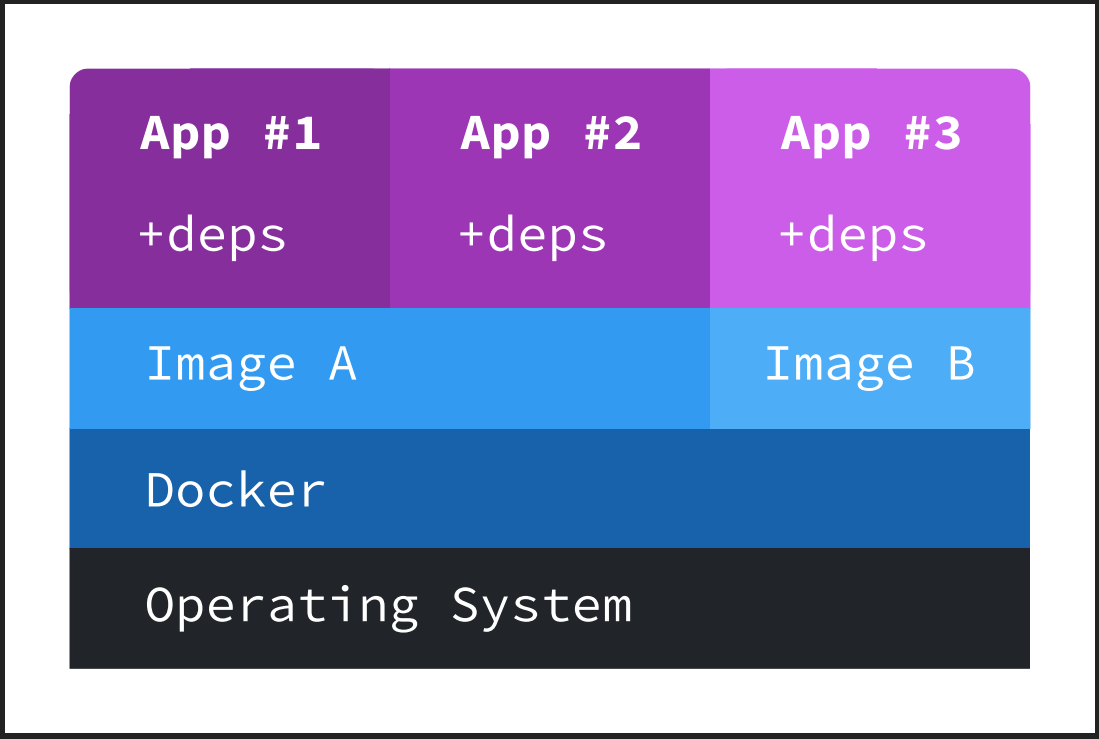
```
def code_change(_old_vsn, _state, _extra) do
  IO.puts("A new version of #{__MODULE__} was loaded!")
  {:ok, state, _} = init(:ok)
  {:ok, state}
end
```

```
$ mix release --upgrade
$ bin/clock foreground
$ bin/clock upgrade 0.2.0
```

Docker

Operating-System Level Virtualization

Operating-System Level Virtualization



Advantages & Disadvantages

Building a Docker Image

Building a Docker Image

```
FROM alpine:latest
RUN apk --no-cache add elixir
ENTRYPOINT ["iex"]
```

Building a Docker Image

```
FROM alpine:latest
RUN apk --no-cache add elixir
ENTRYPOINT ["iex"]
```

```
$ docker build -t pentacent/iex:latest .
```

Building a Docker Image

```
FROM alpine:latest
RUN apk --no-cache add elixir
ENTRYPOINT ["iex"]
```

```
$ docker build -t pentacent/iex:latest .
```

```
$ docker run -it pentacent/iex:latest
```

Building a Docker Image

```
FROM alpine:latest
RUN apk --no-cache add elixir
ENTRYPOINT ["iex"]
```

```
$ docker build -t pentacent/iex:latest .
```

```
$ docker run -it pentacent/iex:latest
```

```
$ docker push pentacent/iex:latest
```

Finding the Right Base

A Multi-Stage Dockerfile

A Multi-Stage Dockerfile

```
# ./Dockerfile

ARG MIX_ENV=prod
```

A Multi-Stage Dockerfile

Build Stage

```
# Build Container
FROM elixir:1.8-alpine AS builder
ARG MIX_ENV

RUN apk update && \
    apk upgrade --no-cache && \
    apk add build-base git && \
    mix do local.hex --force, local.rebar --force

COPY . .

RUN mix do deps.get, compile --force, release && \
    find _build/ -iname clock.tar.gz -exec cp {} . \; && \
    mkdir -p /opt/app && \
    tar -xf clock.tar.gz -C /opt/app
```

A Multi-Stage Dockerfile

Production Stage

```
# Production Container
FROM alpine:3.9
ARG MIX_ENV
ENV HOME=/opt/app

RUN apk update && \
    apk upgrade --no-cache && \
    apk add bash openssl ca-certificates zlib ncurses-libs && \
    mkdir -p ${HOME} && \
    adduser -s /bin/sh -u 1001 -G root -h ${HOME} -S -D default && \
    chown -R 1001:0 ${HOME}

USER default
WORKDIR ${HOME}
COPY --from=builder /opt/app .

ENTRYPOINT [ "/opt/app/bin/clock" ]
CMD ["foreground"]
```

```
docker build . -t clock
```

Let’s Deploy!

So many options!

- PaaS (Heroku, Cloud Foundry, OpenShift)
- Managed Container Services (DigitalOcean, Azure Kubernetes Service, AWS ECS)
- VPS + docker-compose

docker-compose

What is it?

Compose is a tool for defining and running multi-container Docker applications.

– Docker Documentation

docker-compose

The Setup

docker-compose

The Setup

```
$ find ./deployment
docker-compose.yml
.env
```

docker-compose

The Setup

```
$ find ./deployment
docker-compose.yml
.env
```

```
# deployment/docker-compose.yml

version: '3'
services:
  clock:
    image: clock.latest
    environment:
      - INTERVAL=${INTERVAL}
```

docker-compose

The Setup

```
$ find ./deployment
docker-compose.yml
.env
```

```
# deployment/docker-compose.yml

version: '3'
services:
  clock:
    image: clock.latest
    environment:
      - INTERVAL=${INTERVAL}
```

```
# deployment/.env

INTERVAL=1234
```

docker-compose

Integrating other services

```
version: '3'
services:
  mariadb:
    image: mariadb:10.3
    environment:
      - MYSQL_USER=${DB_USER}
      - MYSQL_PASSWORD=${DB_PASSWORD}
      - MYSQL_DATABASE=${DB_NAME}
  clock:
    image: clock:latest
    depends_on:
      - mariadb
    environment:
    - DB_URL=mysql://${DB_USER}:${DB_PASSWORD}@mariadb/${DB_NAME}
```

```
$ docker-compose up
```

Conclusion

- Releases are awesome
- Atomic deployments are awesome
- Docker-compose is awesome

Got questions?

https://pentacent.com/elixirconf