
Erlang in
Cl
leroku

Who are we?

Geoff Cant

@archaelus

Blake Gentry

@blakegentry

What do we do?

Software Engineers

Heroku Routing Team

What is Heroku?

Cloud Application PaaS

We manage servers so you don't have to.

Keeps your apps running.

Painless deploys

Simple, instant scaling.

Getting Started in the Cloud

Launch EC2 Instance

Install OS packages

Configure & Secure

Clone your app

Add monitoring

When you grow, repeat.

Getting Started in the Cloud

Launch EC2 Instance

Install OS packages

Configure & Secure

Clone your app

Add monitoring

When you grow, repeat.

The Heroku Way

Provision an app

```
$ heroku create -s cedar

Creating cold-river-2049... done, stack is
http://cold-river-2049.herokuapp.com/ | gi
Git remote heroku added
```

Instant Deployment

```
$ git push heroku master

-----> Heroku receiving push
-----> ...
-----> Launching... done, v5
       http://myapp.herokuapp.com deployed
```

Instant Scaling

```
$ heroku scale web+30 worke
Scaling web processes... done, now running
Scaling worker processes... done, now runn
```

HTTP Routing Mesh

Balances requests between "dynos"

Intelligent fail-over

Written in Erlang!

Supported Languages

Ruby, Node.js, Python, Java,
Scala, Clojure

And…

Supported Languages

Anything. Including Erlang.

Language support comes

from "buildpacks"

Buildpacks

A buildpack is an adapter between an app and Heroku's runtime [1]

Existing Buildpacks

C Clojure Erlang Go Lisp Mono NES
Node.js Perl Python Ruby Scala
Wordpress Ø (null)

and dozens more. A good list is here

more on Buildpacks later...

More about Erlang!

Buildpacks for R14B03, R15B

My First Heroku Erlang Deploy

Git init...

```
$ mkdir ef_example
$ cd ef_example
$ git init
```

```
Initialized empty Git repository in /Users
```

Rebar ALL THE BOILERPLATE

```
$ rebar create-app appid=ef
==> ef_example (create-app)
Writing src/ef.erl
Writing src/ef.app.src
Writing src/ef_app.erl
Writing src/ef_sup.erl
Writing include/ef_log.hrl
Writing rebar.config
Writing .gitignore
```

```
$ rebar compile
==> ef_example (compile)
Compiled src/ef_app.erl
Compiled src/ef.erl
Compiled src/ef_sup.erl
```

Make sure it works locally

```
$ erl -pa ebin -s ef_app
Erlang R15B01 (erts-5.9.1) [source] [64-b

Eshell V5.9.1  (abort with ^G)
1> application:which_applications().
[{ef,[],"1"},
 {stdlib,"ERTS  CXC 138 10","1.18.1"},
 {kernel,"ERTS  CXC 138 10","2.15.1"}]
2>
User switch command
 --> q
```

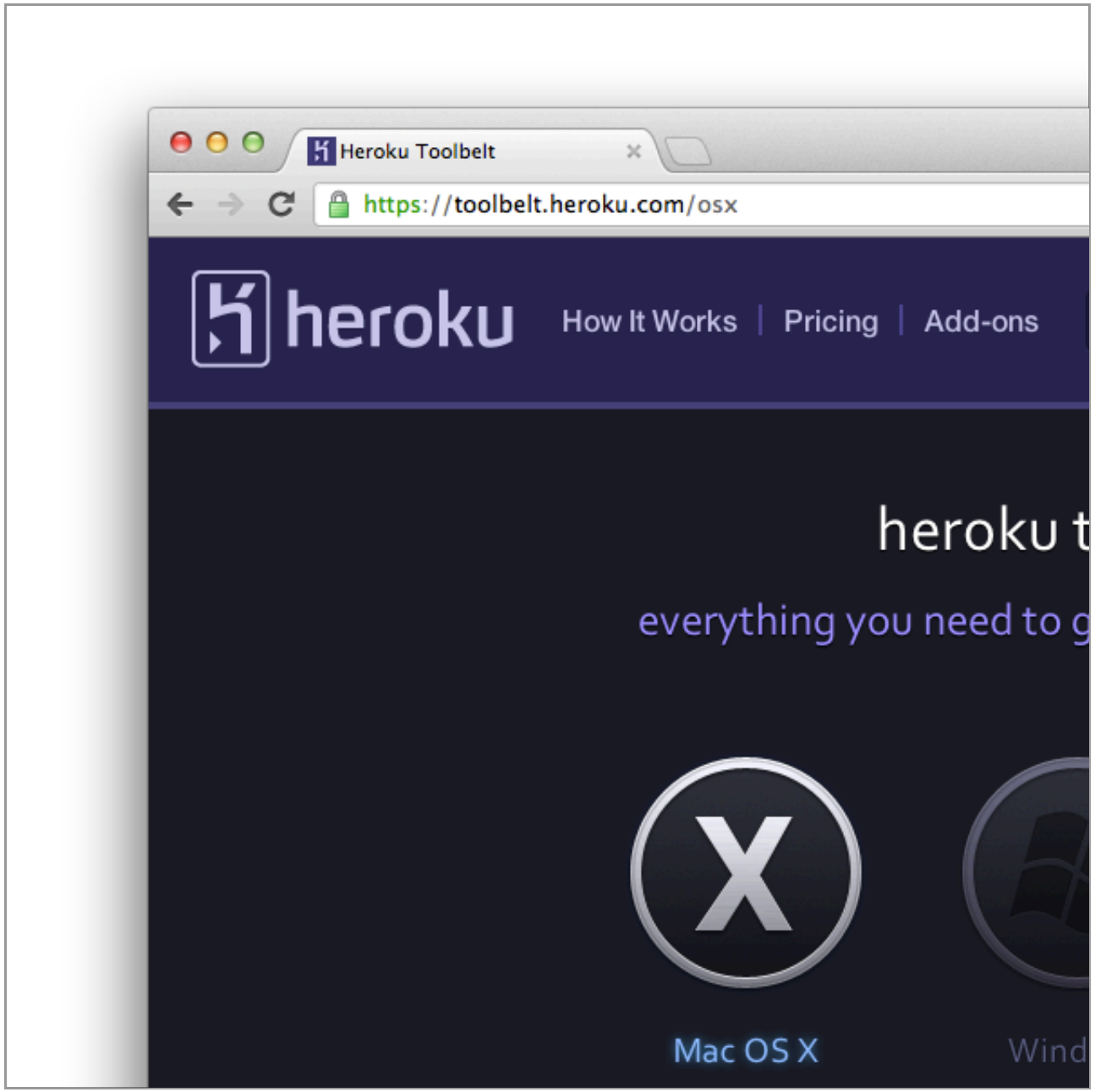
Commit

```
$ git commit -m "rebar crea
[master (root-commit) 3836e22] rebar creat
 7 files changed, 171 insertions(+), 0 del
 create mode 100644 .gitignore
 create mode 100644 include/ef_log.hrl
 create mode 100644 rebar.config
 create mode 100644 src/ef.app.src
 create mode 100644 src/ef.erl
 create mode 100644 src/ef_app.erl
 create mode 100644 src/ef_sup.erl
```

Install the Heroku client

Use the rubygem or Heroku toolbelt:

```
gem install heroku
    #  OR
open https://toolbelt.herok
```


Heroku Toolbelt
https://toolbelt.heroku.com/osx
heroku
How It Works
Pricing
Add-ons
heroku t
everything you need to g
Mac OS X
Wind

Create the heroku app

```
$ heroku help create
Usage: heroku apps:create [NAME]

 create a new app

     --addons ADDONS          # a comma-deli
 -b, --buildpack BUILDPACK    # a buildpack
 -r, --remote REMOTE          # the git remo
 -s, --stack STACK            # the stack on

$ heroku create erlang-factory-2012-exampl
    --stack cedar --buildpack \
    https://github.com/archaelus/heroku-bu
Creating erlang-factory-2012-example... do
http://erlang-factory-2012-example.herokua
Git remote heroku added
```

Deploy your application

Launch EC2 Instance

Install erlang

Clone your app

Build it

Write some init scripts

Debug your init scripts. For ever.

Trolololololol

Launch EC2 Instance

Install erlang

Clone your app

Build it

Write some init scripts

Debug your init scripts. For ever.

J/K

```
$ git push heroku master
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 1.99 KiB, done.
Total 11 (delta 1), reused 0 (delta 0)
-----> Heroku receiving push
-----> Fetching custom buildpack... done
-----> Erlang app detected
-----> Using Erlang/OTP r15b
-------> Fetching Erlang/OTP r15b
-------> Unpacking Erlang/OTP r15b
-------> Installing Erlang/OTP r15b
-----> Installing Rebar from buildpack
-----> Building with Rebar
       ==> build_32vbusz5o9hkq (get-deps)
       ==> build_32vbusz5o9hkq (compile)
       Compiled src/ef.erl
       Compiled src/ef_app.erl
       Compiled src/ef_sup.erl
-----> Discovering process types
       Procfile declares types -> (none)
-----> Compiled slug size is 53.3MB
-----> Launching... done, v4
       http://erlang-factory-2012-example.herokuapp.
```

```
$ heroku ps:scale web=1
$ heroku open
Opening http://erlang-factory-2012-example.herokuapp
$ heroku logs
2012-03-28T03:30:26+00:00 heroku[api]: Config add BU
2012-03-28T03:30:26+00:00 heroku[api]: Release v2 cr
2012-03-28T03:31:06+00:00 heroku[slugc]: Slug compil
2012-03-28T03:31:20+00:00 heroku[slugc]: Slug compil
2012-03-28T03:33:01+00:00 heroku[slugc]: Slug compil
2012-03-28T03:33:40+00:00 heroku[api]: Config add PA
2012-03-28T03:33:40+00:00 heroku[api]: Release v3 cr
2012-03-28T03:33:41+00:00 heroku[api]: Release v4 cr
2012-03-28T03:33:41+00:00 heroku[api]: Deploy 3836e2
2012-03-28T03:33:41+00:00 heroku[web.1]: State chang
2012-03-28T03:33:42+00:00 heroku[slugc]: Slug compil
2012-03-28T03:35:15+00:00 heroku[router]: Error H14
2012-03-28T03:35:16+00:00 heroku[router]: Error H14
2012-03-28T03:35:16+00:00 heroku[router]: Error H14
2012-03-28T03:35:16+00:00 heroku[router]: Error H14
```

Let's add a web server

```
$ cat rebar.config

{erl_opts, [debug_info]}.
{deps, [ {cowboy, "", {git,
  "git://github.com/extend/cowboy.git", {branch,
}]}.%
```

Add cowboy to our app.src

```
$ rebar get-deps compile
...
```

Make sure it still works

```
$ erl -pa ebin -env ERL_LIB
   -s ef_app -ef http_port 1

Erlang R15B01 (erts-5.9.1) [source] [64-b

Eshell V5.9.1  (abort with ^G)
1> application:which_applications().
[{ef,[],"1"},
 {cowboy,"Small, fast, modular HTTP server.","0.5
 {stdlib,"ERTS  CXC 138 10","1.18.1"},
 {kernel,"ERTS  CXC 138 10","2.15.1"}]
2> application:get_env(ef, http_port).
{ok,16526}
3> ef_app:config(http_port).
16526
```

Add a basic http handler

```
-module(ef_http_handler).
-behaviour(cowboy_http_handler).
-export([init/3, handle/2, termina

init({_Any, http}, Req, []) ->
    {ok, Req, undefined}.
handle(Req, State) ->
    {ok, R2} = cowboy_http_req:re
      200, [{'Content-Type', "text/plai
      <<"Hello world!">>, Req),
    {ok, R2, State}.
terminate(_Req, _State) -> ok.
```

Configure cowboy in ef_app

```
start_phase(listen, _Type, _Args)
    Dispatch = [{'_', [
                       {'_', ef_htt
                      ]}
                ],
    cowboy:start_listener(
      http, 100, cowboy_tcp_transp
      [{port, config(http_port)}],
      cowboy_http_protocol,
      [{dispatch, Dispatch}]
    ),
    ok.
```

Add start phases to app.src

```
,{start_phases,
  [{listen, []}
  ]}
```

Check it all works

```
$ erl -pa ebin -env ERL_LIB
  -s ef_app -ef http_port 1
Erlang R15B01 (erts-5.9.1) ...
...
$ curl localhost:16526
Hello world!
```

Adding a Procfile

```
$ cat Procfile

web: erl -pa ebin -env ERL_LIBS deps -s ef
     -noshell -noinput
```

Tedious deployment

```
git push heroku master
```

Enjoy frozen margarita

```
heroku open ; heroku
logs -t
```

Profit.

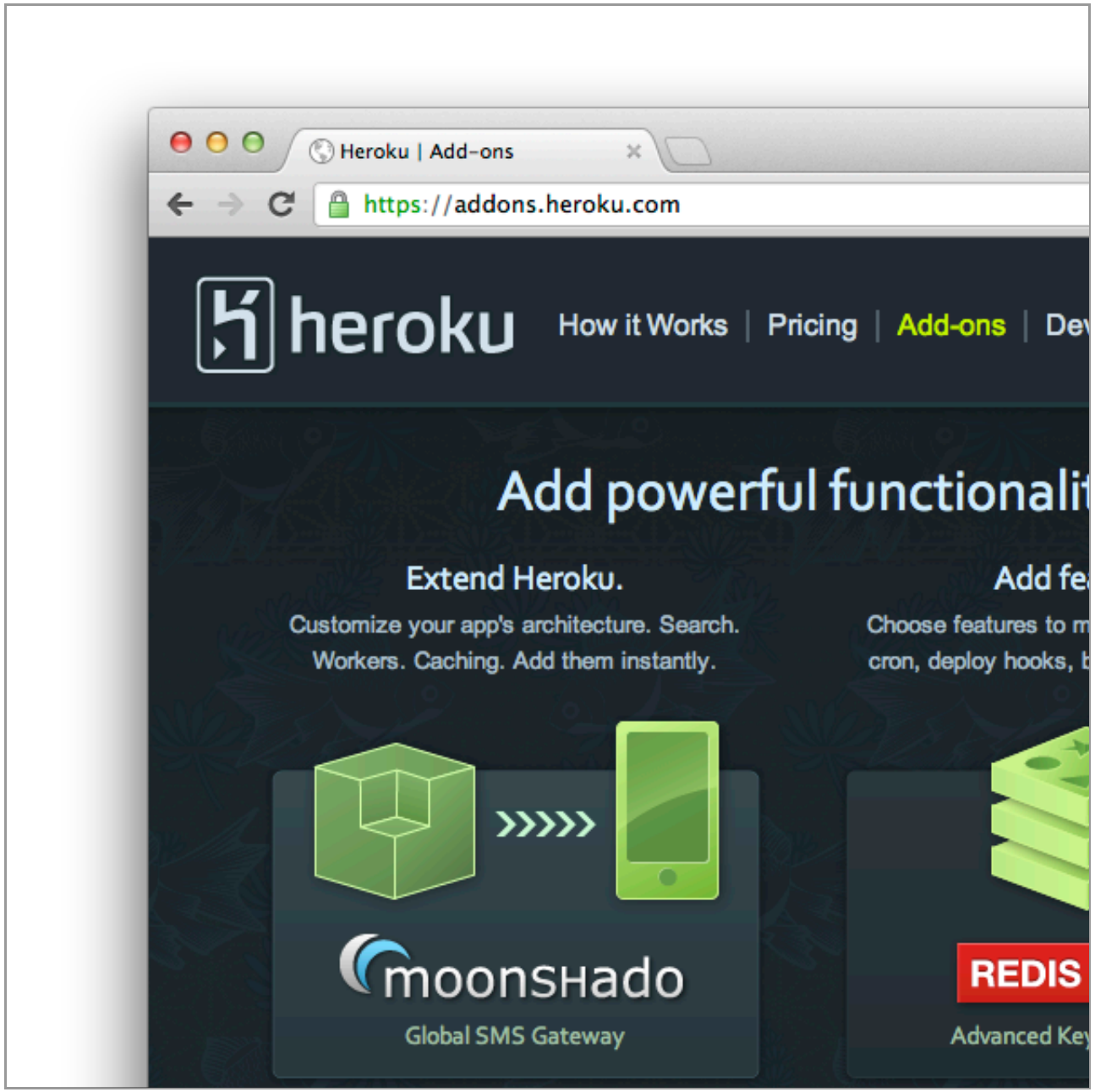
tl;dr

1. Take an Erlang web server
2. Add a Procfile (simplified init script)
3. heroku create –arguments
4. git push heroku master

github.com/archaelus/erlang-factory-2012-example

What else can I do??

Add-ons!


Heroku | Add-ons
https://addons.heroku.com
heroku
How it Works
Pricing
Add-ons
Add powerful functionali
Extend Heroku.
Customize your app's architecture. Search.
Workers. Caching. Add them instantly.
Add fe
Choose features to m
cron, deploy hooks, b
moonSHado
Global SMS Gateway
REDIS
Advanced Ke

Installing an addon

```
$ heroku addons:add redisto
----> Adding redistogo to warm-ice-9184...
$ heroku config
...
REDISTOGO_URL => redis://redistogo:...@cod
...
```

So you said something about Buildpacks...

Buildpacks

What is a Buildpack?

A buildpack is responsible for language

and framework-specific details.

What is a Buildpack?

Installs language binaries

Resolves dependencies

Builds / compiles your code

Buildpack Documentation

https://devcenter.heroku.com/articles/buildpacks

The Erlang Buildpack

(condensed version)

The Erlang Buildpack

1. Fetch language binaries:

```
#!/bin/sh
# usage: bin/compile <build-dir> <cache
...
(
  cd ${cache}
  echo "-------> Fetching Erlang/OTP $version"
  curl -s0 https://s3.amazonaws.com/her
    /${version}.tgz
)
```

Erlang Buildpack (cont.)

2. Unpack + install OTP

```
#!/bin/sh
...
echo "-------> Unpacking Erlang/OTP $version"
...
tar jxf ${cache}/${version}.tgz -C ${ER
echo "-------> Installing Erlang/OTP $version"
ln -s ${ERLROOT} /app/otp
${ERLROOT}/Install -minimal /app/otp
...
```

Erlang Buildpack (cont.)

3. Install Rebar & get-deps

```
#!/bin/sh
...
PATH=/app/otp/bin:$PATH
cd $build
if [ ! -e "rebar" ]; then
  echo "-----> Installing Rebar from buildpack"
  cp ${buildpack_dir}/opt/rebar ./
fi

echo "-----> Building with Rebar"
unset GIT_DIR
(./rebar get-deps compile 2>&1 |
  sed -u 's/^/       /') || exit 1
```

Awesome! What *can't* I do??

Limitations

HTTP-only routing

No direct inter-process communication

Single TCP port available to bind

So what's it good for?

"Traditional" HTTP server apps

Batch processing that pulls off a queue

Anything that uses an intermediary for communication

The Future

(no promises)

The Future

Multi-protocol routing

Bind to multiple/many ports

Dyno-dyno networking

Run any Erlang app you like

We're Hiring!

We're Hiring!

We're, er, dining.

Drop by the office for
lunch/coffee/drinks sometime

We're (Er)Lounging

- We're trying to get into the habit of regular Erlounges with talks and workshops.
- Stay tuned on meetup.com

The End.