COWBOY 2.0

THE SHAPE OF THINGS TO COME

Loïc Hoguin (@lhoguin), Nine Nines

COWBOY CHANGES... AGAIN?!

WTF man, don't you understand what *stable* means?

WTF is wrong with you?!

IN THIS TALK

1. Why Cowboy 2
2. Goals
3. Low-level Cowboy
4. High-level Cowboy
5. Websocket
6. Related projects

WHY COWBOY 2

COWBOY 1 SUCKS

If you are watching this talk, you probably agree.

HTTP/2 REQUIRES A NEW MODEL

HTTP/2 is concurrent, so Cowboy must also be.

LONG TERM VISION

Even if we are required to make small backward incompatible changes, overall the Cowboy 2 design will stand the test of time.

You can quote me on that.

INFLUENCES

Misultin, HTTP/2, Windows

GOALS

PROTOCOLS SUPPORTED

HTTP/2, HTTP/1.1, Websocket

HTTP/1.0, SPDY/3.1, SPDY/3

MORE POWER TO USERS

- Everything special processes (proc_lib/sys)
- Pluggable low-level interface
- Tons of options (like Windows!)

SIMPLER INTERFACE

- Less, simpler callbacks
- No more dealing with Req object
- Extract, validate and convert input in one step
- Maps. Maps everywhere!

BETTER CODE

All parsing code will be in Cowlib.

Cowboy will deal strictly with protocol logic and its own features.

TESTS TESTS TESTS

- Property based testing of Cowlib
- Functional testing of Cowboy protocols and features

LOW-LEVEL COWBOY

PERFORMANCE

Do as little as possible, allocate as little as possible, and provide a pluggable interface for power users.

USE CASES

- Proxies (CONNECT or otherwise)
- Frameworks
- Low-level protocols (Websocket)
- Hooks
- Handling high demand resources early

THE BIG PICTURE

Connection → Protocol → Streams

CONNECTION

- cowboy_clear
- cowboy_tls

LISTENER STARTUP

- cowboy:start_clear(Name, Nb, TransOpts, ProtoOpts)
- cowboy:start_tls(Name, Nb, TransOpts, ProtoOpts)

CHOICE OF PROTOCOL

cowboy_clear → cowboy_http

cowboy_tls → ALPN → Protocol

- cowboy_http
- cowboy_http2
- cowboy_spdy

ALPN?

1. Client advertises list of protocols it supports
2. Server chooses protocol and informs client

Erlang/OTP 18+

PROTOCOL MODULES

- Common interface
- Non-blocking
- Handles system messages
- Able to act as a supervisor if stream-based

PROTOCOL UPGRADES

Same mechanism for:

- Upgrading from HTTP/1.1 to HTTP/2
- Upgrading from HTTP/1.1 to Websocket
- Upgrading HTTP/1.1 connections from TCP to TLS

STREAMS

A stream is an HTTP request/response pair identified by a unique stream identifier.

A stream can be initiated by the server using a promise.

STREAM-BASED PROTOCOLS

- cowboy_http
- cowboy_http2
- cowboy_spdy
- (cowboy_coap?)
- NOT cowboy_websocket

STREAM HANDLER

- init(ID, IsFin, Method, Scheme, Host, Path, Headers, Opts)
- data(ID, IsFin, Data, State)
- info(ID, Msg, State)
- terminate(ID, Reason, State)

STREAM-SPECIFIC MESSAGES

- Messages are filtered per-stream
- {{Handler, ID}, Msg}
- {{Handler, ID}, From, Msg}

STREAM COMMANDS

- {response, IsFin, StatusCode, Headers}
- {data, IsFin, Data}
- {promise, Method, Scheme, Authority, Path, Headers}
- {flow, auto | Size}
- {spawn, Pid}
- {upgrade, Mod, Opts}

DEFAULT STREAM HANDLER

- Creates a new process per stream
- Communicates with process using Cowboy stream protocol
- High-level Cowboy

ONE PROCESS PER REQUEST

- High-level Cowboy uses 1 proc/conn + 1 proc/stream
- *BUT*
- Low-level Cowboy uses 1 proc/conn *only*

COWBOY STREAM PROTOCOL

- {response_header, Key, Value}
- {response_cookie, Key, Value, Opts}
- {response_body, Body}
- {response, IsFin, StatusCode, Headers}
- {data, IsFin, Data}
- {promise, Method, Scheme, Authority, Path, Headers}
- {data_request, Size}

LAYERED STREAM HANDLERS

- A stream handler can call another stream handler
- Use handler-specific options to define the next layer
- This makes Cowboy 1 hooks worthless

HIGH-LEVEL COWBOY

CONVENIENCE

Pie tastes good, right? Just like Cowboy 2.

Favor convenience, elegance and simplicity. No pitfalls. Straightforward.

THE BIG PICTURE

Middlewares → User handlers

Default: cowboy_router → cowboy_handler

MIDDLEWARES

No more 'error' tuple; ok, suspend, stop

ROUTING

I would like to solve the fact that routing rules are copied to all connection processes. Perhaps ets?

REVERSE ROUTING

Required for better HATEOAS support.

Give module name and bindings, get URL.
Add query string to URL (optional).

CONSTRAINTS

Use constraints all across high-level Cowboy.

Improve error handling interface, add human errors.

HANDLERS

Unify init and terminate callbacks

Simplify init return value: {ok | Mod, State}

Do everything in init/2

REST HANDLERS

Add behaviour with optional callbacks

Remove known_content_type callback

REQ OBJECT

Immutable

3 levels of access to values:

- Raw value
- Parsed value
- Matched value

MATCH FUNCTIONS

Extract, validate and convert values in one step.

```
#{lang := Lang} = cowboy_req:match_qs(
        [{lang, nonempty, <<"en-US">>}], Req)
```

ADDITIONAL NOTES

Handler suffix becomes _h

Settle on 'stop' instead of 'shutdown' or 'halt'

Cowlib provides parsers for nearly everything

WEBSOCKET

UPGRADE

- From stream handler
- From request process

Connection process switches to Websocket protocol

FEATURES

Websocket permessage-deflate support added.

Websocket UTF-8 validation optimized.
An option to disable validation will be added.

WEBSOCKET HANDLERS

- No more websocket_init
- No more websocket_terminate
- Optional terminate
- No keeping track of Req

RELATED PROJECTS

RANCH 2

- Merge acceptor and supervisor functionality
- Use the async accept mechanism

GUN

Gun is an asynchronous HTTP client with support for HTTP/1.1, HTTP/2, SPDY/3 and Websocket, designed for long-running connections.

ERLANG.MK

Erlang.mk is a Makefile based build tool that *just works*.

No Makefile knowledge required to use it

WHY ERLANG.MK

My users need a build tool that actually works.

CONVINCE ME

- Compatible with a lot more projects than rebar
- Dependencies can be in any language (C, Javascript...)
- It's just a text file

COMPLEXITY COMPARISON

Rebar feature	Erlang.mk equivalent
rebar.config	variables
rebar.config.script	variables and/or rules
rebar hook	rules
rebar2 plugin	rules
rebar3 plugin	rules

ERLANG.MK INDEX

Getting close to 450 projects

DEPS = cowboy cpg erlydtl riak_core

ALL PACKAGES 1/6

aberth active aleppo alog annotations antidote apns azdht backoff barrel basho_bench bcrypt beam beanstalk bear bertconf bifrost binpp bisect bitcask bitstore bootstrap boss_db boss bson bullet cache cake carotene cberl cecho cferl chaos_monkey check_node chronos classifier clique cloudi_core cloudi_service_* cluster_info color confetti couchbeam couch covertool cowboy cowdb cowlib cpg cqerl cr cuttlefish damocles debbie decimal detergent dh_date dhtcrawler dirbusterl dispcount dlhttpc dns dnssd dtl dynamic_compile e2 eamf eavro ecapnp econfig edate edgar edis edns

ALL PACKAGES 2/6

edown eep_app eep efene eganglia egeoip ehsa ejabberd ej ekaf elarm eleveldb elli elvis emagick emysql enm entop epcap eper epgsql episcina eplot epocxy epubnub eqm eredis eredis_pool erlang_cep erlang_js erlang_localtime erlang_smtp erlasticsearch erlastic_search erlbrake erlcloud erlcron erldb erldis erldns erldocker erlfsmon erlgit erlguten erlmc erlmongo erlog erlpass erlport erlsha2 erlsh erlsom erl_streams erlubi erlvolt erlware_commons erlydtl errd erserve erwa espec estatsd etap etest etest_http etoml eunit_formatters eunit euthanasia evum exec exml exometer exs1024 exs64 exsplus116 exsplus128 ezmq ezmtp

ALL PACKAGES 3/6

fast_disk_log feeder fix flower fn folsom_cowboy folsom folsomite fs fuse gcm gcprof geas geef gen_cycle gen_icmp gen_nb_server gen_paxos gen_smtp gen_tracker gen_unix getopt gettext giallo gin gitty gold_fever gpb gproc grapherl gun hackney hamcrest hanoidb hottub hyper ibrowse ierlang iota ircd irc_lib iris iso8601 itweet jerg jesse jiffy jiffy_v jobs joxa jsonerl json jsonpath json_rec jsonx jsx kafka kai katja kdht kinetic kjell kraken kucumberl kvc kvlists kvs lager_amqp_backend lager lager_syslog lambdapad lasp lasse ldap lethink lfe ling live lmq locker locks log4erl lol lucid luerl luwak lux mad mavg mcd mcerlang mc_erl meck

ALL PACKAGES 4/6

mekao memo merge_index merl mimetypes mixer mochiweb mochiweb_xpath mockgyver modlib mongodb mongooseim moyo msgpack mustache myproto mysql n2o nat_upnp neo4j neotoma newrelic nifty nitrogen_core nkbase nkdocker nkpacket nodefinder nprocreg oauth2c oauth2 oauth of_protocol openflow openid openpoker pal parse_trans parsexml pegjs percept2 pgsql pkgx pkt plain_fsm plumtree pmod_transform pobox ponos poolboy pooler poxa pqueue procket proper props protobuffs psycho ptrackerl purity push_service qdate qrcode quest rabbit_exchange_type_riak rack radierl rafter ranch

ALL PACKAGES 5/6

rbeacon rebar rebus rec2json recon record_info redgrid redo relx resource_discovery restc rfc4627_jsonrpc riakc riak_core riak_dt riak_ensemble riakhttpc riak_kv riaknostic riak_pg riak_pipe riakpool riak_sysmon riak_test rivus_cep rlimit safetyvalve seestar service setup sext sfmt sgte sheriff shotgun sidejob sieve sighandler simhash simple_bridge simple_oauth2 skel social spapi_router sqerl srly sshrpc stable statebox statebox_riak statman statsderl stdinout_pool stockdb stripe surrogate swab swarm switchboard sync syntaxerl syslog taskforce tddreloader tempo ticktick tinymq tinymt traffic_tools trane transit trie

ALL PACKAGES 6/6

triq tunctl twerl twitter_erlang ucol_nif unicorn unsplit uuid ux vert verx vmq_acl vmq_bridge vmq_graphite vmq_passwd vmq_server vmq_snmp vmq_systree vmstats walrus webmachine websocket_client worker_pool wrangler wsock xref_runner yamerl yamler yaws zab_engine zeta zippers zlists zraft_lib zucchini

ERLANG.MK PLANS 1/2

1. Compile everything
2. Keep track of versions
3. Provide curated packages

ERLANG.MK PLANS 2/2

1. Add support for LFE, Elixir projects
2. Add Concuerror, Chaos Monkey, Smother, RefactorErl...
3. Generate the .app file without a .app.src

EVERYTHING IN THE MAKEFILE

- PROJECT → application name
- PROJECT_DESCRIPTION → description
- PROJECT_VERSION → vsn
- PROJECT_ID → id
- PROJECT_TYPE → mod, registered
- PROJECT_REGISTERED → registered
- PROJECT_ENV or PROJECT_ENV_FILE → env
- modules list automatically filled
- applications list automatically filled from DEPS

EVERYTHING OPTIONAL

All the variables that ultimately build the .app file are either optional or automatically defined when bootstrapping.

Adding a dependency becomes a one step process.

STARTING UP GOALS

- mkdir kitty; cd kitty
- wget $ERLANG_MK_URL
- make -f bootstrap bootstrap-rel
- append DEPS = cowboy cpg erlydtl riak_core
- make run

STARTING UP LATER GOALS

- erlang-mk new kitty
- append DEPS = cowboy cpg erlydtl riak_core
- make run

ERLANG.MK ON WINDOWS

Make 4 makes supporting Windows possible without needing the whole Unix toolchain.

Before this there are much bigger Windows issues to solve to make the experience smooth, unrelated to Erlang.mk

THE ERLANGER PLAYBOOK

This book is the missing developer manual. It contains advice from my experience working with Cowboy.

Price: 50€

Ebook preorder available soon on ninenines.eu

Want to buy it today and get it *right now*? Come talk to me or send an email later at essen@ninenines.eu

TERMINATE

TL;DR

Cowboy 2's design is long term.

QUESTIONS?

To look at an early release of the code in this talk:
tag 2.0.0-pre.2

To buy the book, access projects:
ninenines.eu

To follow my exciting adventures:
@lhoguin on Twitter

To ask questions, talk about hats:
#ninenines on Freenode