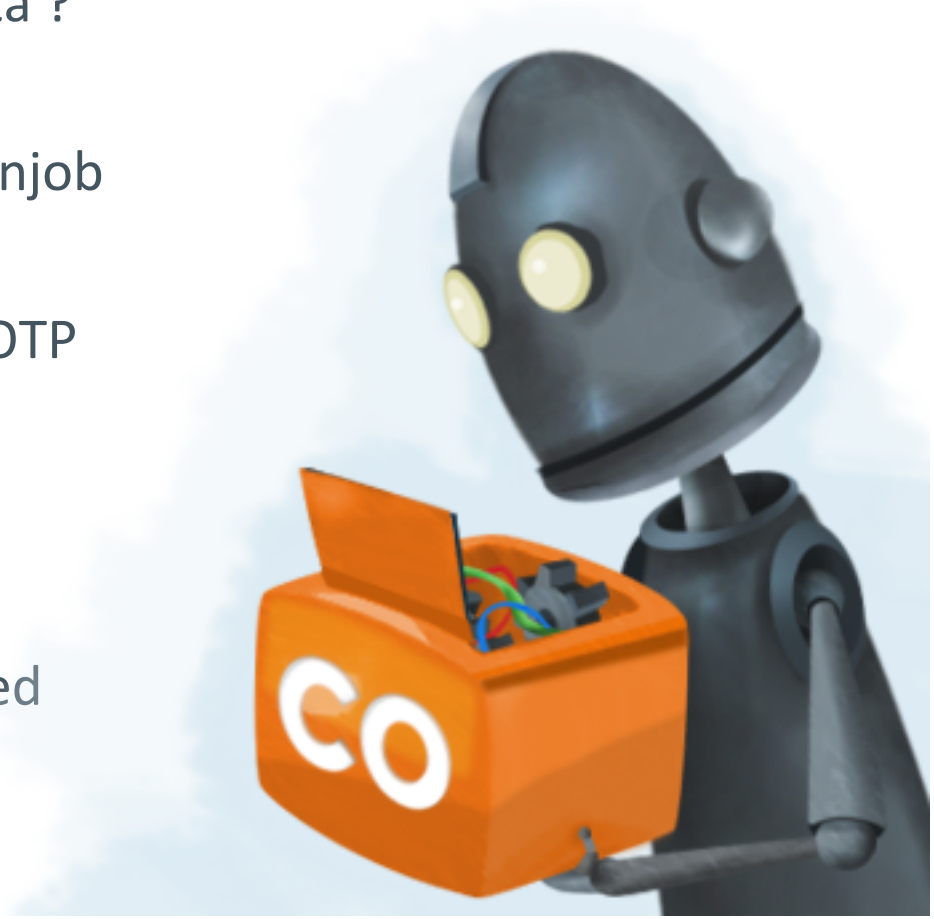
ExactTarget

MAGICBE.AM

HISTORICAL CONTEXT

THE COTWEET SOCIAL DATA STORY

- How to retrieve users social data ?
- First incarnation was a ruby cronjob
- Second incarnation was a non-OTP Erlang daemon
  - Nomura was born
- Current incarnation is distributed (via RabbitMQ) and largely OTP



WHAT IS A NOMURA

- A very large jellyfish, the largest cnidarian in the world
- 野村 “Field Village”
- A distributed platform for providing social data streams throughout the ExactTarget software ecosystem
- Integrates with Twitter / Facebook / Others coming soon
- Multiple components including scheduling, api workers & data workers, command & control

HOW IS WORKER POOL FORMED

- Three key requirements
  - Dynamically adjust pool size up / down in response to ever-changing circumstances
  - Faults isolated between individual requests and customers
  - Transactional / stateless
- All our workers wrap a RabbitMQ consumer
- Find a supervisory structure that fits

CHAOS BEAMS

ENTER THE THUNDERDOME

- Took a cue from Netflix Chaos Monkeys
- How better to test supervisory structures….
- How better to cause mild concern in the QA department….

THUNDERBEAM IN PRINCIPLE

- Hey what happens when we kill off random Erlang processes
  - They are supervised right?
- Simulates unexpected function calls and inbound messages
  - We call these “typos”
- Fine line between not handling common errors and matching all the catch patterns
  - try 1 / 0 catch _:_ -> ok end.
- Who supervises the supervisors application masters ?

THUNDERBEAM IN ACTION

- Tunable timing of process death
- Blacklist of registered processes or OTP applications to not kill
  - Mnesia is easily upset
- gen_server handle_info handler for bespoke trap_exit'd processes
- Ability to force-kill supervisors and non-OTP processes

HOT LOADED BEAMS

WAITING IS BORING

- No more sword fights
  - No more excuses
- Reload mod on beam file change
  - Nothing new here
- Recompile on source file change
  - Finally caught up to PHP

HOTBEAM IN ACTION

- Extrapolate source location for normal beam via compile attribute
- Control which OTP applications are monitored for change
- Selectively enable / disable automagic compilation / reloading
- Manually reload single module, OTP application or everything

HOTBEAM IMPROVEMENTS

- Support additional source file types
- Monitor header files and recompile dependent source files
- code_change callbacks to support run-time state changes

REPL FOR ALL

WHO DOESN'T LIKE RUNTIME INTROSPECTION

- Habitual implementation of info/0 functions
  - Simpler than sys:get_status/1
- Cultural tendency towards live configuration changes
  - Mea Maxima Culpa; Over-engineered config logic
- Manual control of running systems
  - Not every button gets a web interface
- Friendlier than the typical Erlang shell
  - Less dangerous as well

SHELLBEAM IN PRINCIPLE

- Simple behavio(u)r for implementing a bespoke and easy to use REPL shell
- Abstraction around logic with run-time help
- Hierarchical sub-shells with shortcut access

SHELLBEAM IN ACTION

- Basic magicbeam/Erlang functionality as default callback
- Application Environment editing subshell part of default callback
- Passes parsed arguments into callback implementation functions
- Integrated SSHD support. Handles generation of host keys.

SHELLBEAM MODULE SAMPLE (APPENV)

```
-module(magicbeam_shell_appenv).
-behaviour(shellbeam).

-export([commands/0]).
-export([list/0, list/1, get/2, set/3]).

commands() ->
    [
     {["list"], "Lists loaded OTP applications", fun list/0},
     {["list", {"application", atom}], "Lists environment for an application", fun list/1},
     {["get", {"application", atom}, {"key", atom}], "Displays a key for an OTP Application", fun get/2},
     {["set", {"application", atom}, {"key", atom}, {"value", auto}], "Sets an OTP Application environment key", fun set/3}
    ].

list() ->
    F = fun({A, _, _}) ->
            atom_to_list(A) ++ "~n" end,
    {ok, lists:flatten("Available Applications.~n" ++ lists:map(F, application:loaded_applications()))}.

list(A) when is_atom(A) ->
    F = fun({included_application, _}) ->
            "";
        ({K, V}) -> io_lib:format("~p : ~p~n", [K, V])
        end,
    {ok, lists:flatten("Application Environment for " ++ atom_to_list(A) ++ "~n" ++ lists:map(F, application:get_all_env(A)))}.

get(A, K) ->
    V = case application:get_env(A, K) of
            undefined -> undefined;
            {ok, T} -> T
        end,
    {ok, "~p:~p is ~p", [A, K, V]}.

set(A, K, V) ->
    ok = application:set_env(A, K, V),
    {ok, "Set ~p:~p to ~p", [A, K, V]}.
```

SHELLBEAM USAGE SAMPLE (APPENV)

```
foo@ctberries.local magicbeam shell 0 > appenv list
Available Applications.
ssh
kernel
crypto
sasl
magicbeam
stdlib

foo@ctberries.local magicbeam shell 1 > appenv
config ^_^ 0 > list kernel
Application Environment for kernel
included_applications : []
error_logger : tty

config ^_^ 1 > get magicbeam thunderbeam_enabled
magicbeam:thunderbeam_enabled is undefined
config ^_^ 2 > set magicbeam thunderbeam_enabled true
Set magicbeam:thunderbeam_enabled to true
config ^_^ 3 > get magicbeam thunderbeam_enabled
magicbeam:thunderbeam_enabled is true
config ^_^ 4 >
```

MAGICBEAM 4 DEVOPS

MAGICBEAM SUMMARY

- Catch-all project for devops-y bits which we've found useful
- Comprised of three key pieces of functionality
  - thunderbeam
  - hotbeam
  - shellbeam
- Designed with ease of integration in mind

MAGICBEAM INTEGRATION

- Available as rebar dependency. May be included in release as permanent, temporary or load-only application
- Inject into already running node. Note that this enables the use of magicbeam in already-deployed environments
- Project-specific configuration and event handling via implementation of magicbeam behavior

USE CASES

DEPLOYMENT

- Determine which applications you wish to reload on the fly
  - All bespoke code
- Create an initscript hook for hotbeam:all()
  - project_ctl
- Integrate into deployment pipeline
  - Choice between restart / hotload

EXTREME PAIR PROGRAMMING

- Two developers, one editor, one running release
  - Hotbeam reloads code on save
- Robustify software as it's being created
  - Thunderbeam does it's thing
- Tweak runtime parameters to test all facets
  - Shellbeam shortcuts for config and such

OPS SUPPORT

- Ops isn't going to learn Erlang
  - Yet still needs to tweak knobs and pull levers
- Shellbeam implementations for common operations
  - Configuration & Scaling & Metrics Oh My
  - Distributed component control
- Direct SSH access
  - Remove OS distractions

ERRATA

ERRATA

- SSH access only supports one private key
- Magicbeam injection requires compatible bytecode
- Hotbeam does not/will never support upgrading Erlang VM

FIN

JONATHAN FREEDMAN
MAGICBE.AM