Cuttlefish

easing the pain of erlang application configuration

Joe DeVivo
erlanger @ basho
@joedevivo

app.config (sys.config)?

vm.args

That’s how it happened with Riak

```
%% -*- tab-width: 4;erlang-indent-level: 4;indent-tabs-mode: nil -*-
%% ex: ts=4 sw=4 et
[
 %% Riak Core config
 {riak_core, [
    %% Default location of ringstate
    {ring_state_dir, "data/ring"},

    %% riak_web_ip is the IP address that Riak's HTTP interface will
    %%  bind to.  If this is undefined, the HTTP interface will not run.
    {web_ip, "127.0.0.1"},

    %% riak_web_port is the TCP port that Riak's HTTP interface will
    %% bind to.
    {web_port, 8098}
]},

%% Riak KV config
{riak_kv, [
    %% Storage_backend specifies the Erlang module defining the storage
    %% mechanism that will be used on this node.
    {storage_backend, riak_kv_dets_backend},

    %% Different storage backends can use other configuration variables.
    %%  For instance, riak_dets_backend_root determines the directory
    %%  under which dets files will be placed.
    {riak_kv_dets_backend_root, "data/dets"},
```

```
## Name of the riak node
-name dev1@127.0.0.1

## Cookie for distributed erlang
-setcookie riak

## Heartbeat management; auto-restarts VM if it dies or
becomes unresponsive
## (Disabled by default..use with caution!)
##-heart

## Enable kernel poll and a few async threads
+K true
+A 5

## Increase number of concurrent ports/sockets
-env ERL_MAX_PORTS 4096

## Tweak GC to run more often
-env ERL_FULLSWEEP_AFTER 10
```

```
%% -*- tab-width: 4;erlang-indent-level: 4;indent-tabs-mode: nil -*-
%% ex: ts=4 sw=4 et
[
 %% Riak Core config
 {riak_core, [
    %% Default location of ringstate
    {ring_state_dir, "data/ring"},

    %% riak_web_ip is the IP address that Riak's HTTP interface will
    %%  bind to.  If this is undefined, the HTTP interface will not run.
    {web_ip, "127.0.0.1"},

    %% riak_web_port is the TCP port that Riak's HTTP interface will
    %% bind to.
    {web_port, 8098}
 ]},

 %% Riak KV config
 {riak_kv, [
    %% Storage_backend specifies the Erlang module defining the storage
    %% mechanism that will be used on this node.
    {storage_backend, riak_kv_dets_backend},

    %% Different storage backends can use other configuration variables.
    %%  For instance, riak_dets_backend_root determines the directory
    %%  under which dets files will be placed.
    {riak_kv_dets_backend_root, "data/dets"},

            %% riak_handoff_port is the TCP port that Riak uses for
            %% intra-cluster data handoff.
            {handoff_port, 8099},

            %% pb_ip is the IP address that Riak's Protocol Buffers interface
            %% will bid to.  If this is undefined, the interface will not run.
            {pb_ip,   "0.0.0.0"},

            %% pb_port is the TCP port that Riak's Protocol Buffers interface
            %% will bind to
            {pb_port, 8087},

            %% raw_name is the first part of all URLS used by Riak's raw HTTP
            %% interface.  See riak_web.erl and raw_http_resource.erl for
            %% details.
            %{raw_name, "riak"},

            %% mapred_name is URL used to submit map/reduce requests to Riak.
            {mapred_name, "mapred"},

            %% js_vm_count is the number of Javascript VMs to start per Riak
            %% node.  8 is a good default for smaller installations. A larger
            %% number like 12 or 16 is appropriate for installations handling
            %% lots of map/reduce processing.
            {js_vm_count, 8},

            %% js_source_dir should point to a directory containing Javascript
            %% source files which will be loaded by Riak when it initializes
            %% Javascript VMs.
            %{js_source_dir, "/tmp/js_source"}

            %% riak_stat enables the use of the "riak-admin status" command to
            %% retrieve information the Riak node for performance and debugging
needs
            {riak_kv_stat, true}
           ]},

 %% SASL config
 {sasl, [
         {sasl_error_logger, {file, "log/sasl-error.log"}},
         {errlog_type, error},
         {error_logger_mf_dir, "log/sasl"},      % Log directory
         {error_logger_mf_maxbytes, 10485760},   % 10 MB max file size
         {error_logger_mf_maxfiles, 5}           % 5 files max
         ]}
].
```

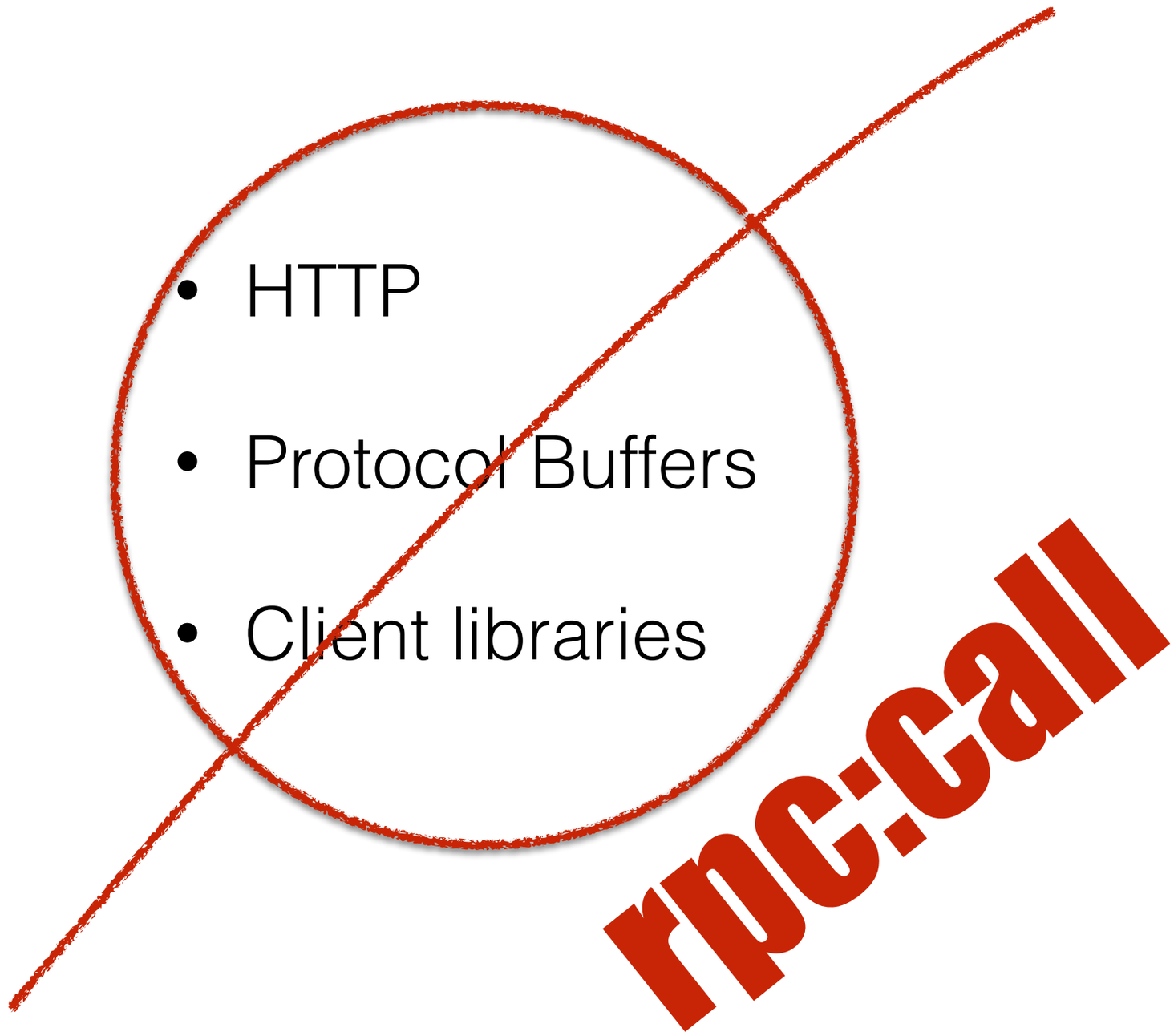
```
## Name of the riak node
-name dev1@127.0.0.1

## Cookie for distributed erlang
-setcookie riak

## Heartbeat management; auto-restarts VM if it dies or becomes unresponsive
## (Disabled by default..use with caution!)
##-heart

## Enable kernel poll and a few async threads
+K true
+A 5

## Increase number of concurrent ports/sockets
-env ERL_MAX_PORTS 4096

## Tweak GC to run more often
-env ERL_FULLSWEEP_AFTER 10
```

Review all files 🐟


OTP

Configuration is a part of your application's user experience

Riak's API



It’s not enough because we don’t want to force users into an Erlang vm.

You shouldn’t have to know Erlang to use Riak

Why do we feel it’s acceptable to force users to Erlang with it comes to configuration?

It’s not

Welcome to Riak 2.0

app.config + vm.args +  -  = riak.conf

There’s only ONE file

It only has ONE syntax

Everything you need to know about a single configuration setting is on a single line

That line can be anywhere in the file

That line doesn’t change if it’s in a different place

Key/Value is a pretty good idea

```
## Enable consensus subsystem. Set to 'on' to enable the
## consensus subsystem used for strongly consistent Riak operations.
##
## Default: off
##
## Acceptable values:
##   - on or off
## strong_consistency = on

## listener.http.<name> is an IP address and TCP port that the Riak
## HTTP interface will bind.
##
## Default: 127.0.0.1:8098
##
## Acceptable values:
##   - an IP/port pair, e.g. 127.0.0.1:10011
listener.http.internal = 127.0.0.1:8098

## listener.protobuf.<name> is an IP address and TCP port that the Riak
## Protocol Buffers interface will bind.
##
## Default: 127.0.0.1:8087
##
## Acceptable values:
##   - an IP/port pair, e.g. 127.0.0.1:10011
listener.protobuf.internal = 127.0.0.1:8087
```

It looks like this

If you set something more than once,
the last one wins

```
→  riak git:(develop) ✗ cat etc/riak.conf | grep ^anti_entropy
anti_entropy = active
anti_entropy = passive
→  riak git:(develop) ✗ bin/riak config effective | grep ^anti_entropy
anti_entropy = passive
...
```

Shoutout to
Last Write Wins!

Built in documentation

```
→  riak git:(develop) ✗ ./bin/riak config describe object.size.maximum
Documentation for object.size.maximum
Writing an object bigger than this will send a failure to the
client.

   Datatype     : [bytesize]
   Default Value: "50MB"
   Set Value    : 75MB
   app.config   : riak_kv.max_object_size
```

Configuration values have datatypes

```
➜  riak git:(develop) ✗ ./bin/riak config describe search.solr.start_timeout
Documentation for search.solr.start_timeout
How long Riak will wait for Solr to start. The start sequence
will be tried twice. If both attempts timeout, then the Riak node
will be shutdown. This may need to be increased as more data is
indexed and Solr takes longer to start. Values lower than 1s will
be rounded up to the minimum 1s.

   Datatype     : [{duration,s}]
   Default Value: "30s"
   Set Value    : 45s
   app.config   : yokozuna.solr_startup_wait
```

Configuration values can be validated

If there’s an error in the configuration,
you’ll get a decent explanation

unknown key

```
➜  riak git:(develop) ✗ cat etc/riak.conf | grep ^anti_ent
anti_entrophy = passive
➜  riak git:(develop) ✗ ./bin/riak console
10:20:20.466 [error] You've tried to set anti_entrophy, but there is no
setting with that name.
10:20:20.466 [error]   Did you mean one of these?
10:20:20.486 [error]     anti_entropy
10:20:20.486 [error]     anti_entropy.throttle
10:20:20.486 [error]     riak_control
Error generating config with cuttlefish
```

invalid value

```
## Number of partitions in the cluster (only valid when first
## creating the cluster). Must be a power of 2, minimum 8 and maximum
## 1024.
##
## Default: 64
##
## Acceptable values:
##   - an integer
ring_size = 42
```

```
➜  riak git:(develop) ✗ ./bin/riak console
13:16:11.933 [error] Error generating configuration in phase validation
13:16:11.933 [error] ring_size invalid, not a power of 2
Error generating config with cuttlefish
```

If there's an error in the configuration,
Riak won't start

You can view the effective configuration

```
➜  riak git:(develop) ✗ ./bin/riak config effective
anti_entropy = active
anti_entropy.bloomfilter = on
anti_entropy.concurrency_limit = 2
anti_entropy.data_dir = $(platform_data_dir)/anti_entropy
anti_entropy.max_open_files = 20
anti_entropy.throttle = on
anti_entropy.tree.build_limit.number = 1
anti_entropy.tree.build_limit.per_timespan = 1h
anti_entropy.tree.expiry = 1w
anti_entropy.trigger_interval = 15s
anti_entropy.write_buffer_size = 4MB
bitcask.data_root = $(platform_data_dir)/bitcask
bitcask.expiry = off
bitcask.expiry.grace_time = 0
bitcask.fold.max_age = unlimited
bitcask.fold.max_puts = 0
bitcask.hintfile_checksums = strict
bitcask.io_mode = erlang
bitcask.max_file_size = 2GB
bitcask.merge.policy = always
bitcask.merge.thresholds.dead_bytes = 128MB
bitcask.merge.thresholds.fragmentation = 40
```

There’s a definitive place to look for default configuration values

No Erlang Required

You can use app.config and vm.args when upgrading from 1.4

Command Line Fu

```
→  riak git:(develop) ✗ cat etc/riak.conf | grep ^anti_entropy
anti_entropy = active

→  riak git:(develop) ✗ bin/riak config effective | grep ^anti_entropy
anti_entropy = active
anti_entropy.bloomfilter = on
anti_entropy.concurrency_limit = 2
anti_entropy.data_dir = $(platform_data_dir)/anti_entropy
anti_entropy.max_open_files = 20
anti_entropy.throttle = on
anti_entropy.tree.build_limit.number = 1
anti_entropy.tree.build_limit.per_timespan = 1h
anti_entropy.tree.expiry = 1w
anti_entropy.trigger_interval = 15s
anti_entropy.write_buffer_size = 4MB


→  riak git:(develop) ✗ ./bin/riak config effective | grep ^anti_entropy.con
anti_entropy.concurrency_limit = 2
→  riak git:(develop) ✗ echo anti_entropy.concurrency_limit = 4 >> etc/riak.conf
→  riak git:(develop) ✗ ./bin/riak config effective | grep ^anti_entropy.con
anti_entropy.concurrency_limit = 4
```

What is  ?

cuttlefish |ˈkətlˌfiSH| (noun)

1. An Erlang library for human friendly configuration files.

2. A pun on sysctl & babelfish

If you don’t know Erlang, you probably don’t like the syntax

Configuration files are your application's first impression on the user

You are not your application's average user

If your user doesn’t know about your app.config, your ops team is your user

It generates the **app.config** and **vm.args** that growing erlang vms love

Bad configuration should fail fast

A validated configuration is an easy way to prevent early user issues

Product Upgrayedds are smoother

It’s not that hard to add cuttlefish to your application, and that work is mostly in Erlang

The time spent integrating cuttlefish is time saved in customer support

Cuttlefish lets you unit test your configuration interface

There’s Erlang when you need it

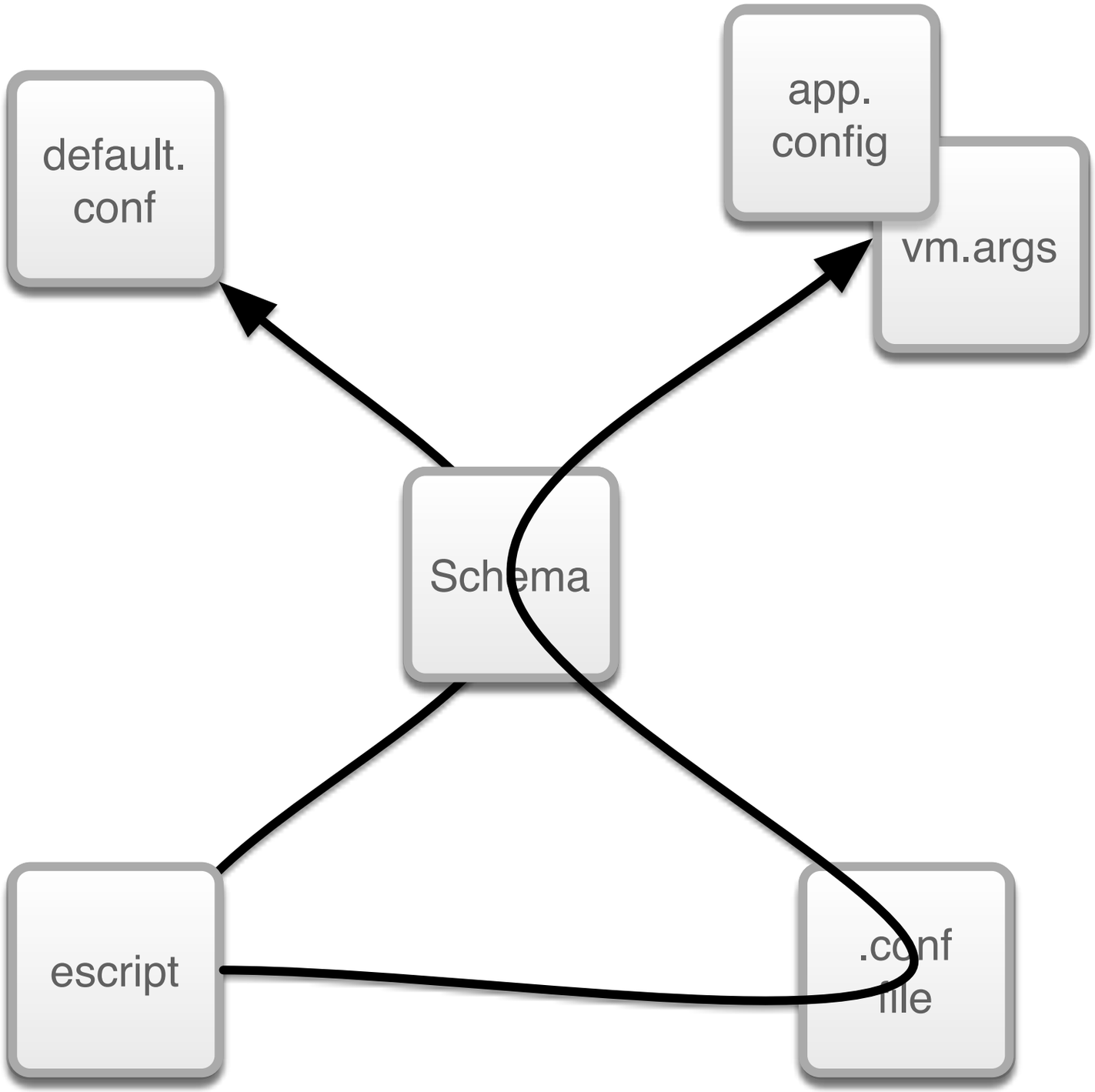
Phased Error Handling

```
## Acceptable values:
##   - one of: off, file, console, both
log.console = penguin

## Acceptable values:
##   - one of: debug, info, warning, error
log.console.level = polite

## Acceptable values:
##   - a byte size with units, e.g. 10GB
log.crash.maximum_message_size = 64DB

ring_size = 42

## Acceptable values:
##   - an IP/port pair, e.g. 127.0.0.1:10011
listener.http.internal = 127.0.0.1:port
```

```
➜  riak git:(develop) ✗ ./bin/riak console
 [error] Error generating configuration in phase transform_datatypes
 [error] Error transforming datatype for: listener.http.internal
 [error] "127.0.0.1:port" cannot be converted into an IP
 [error] Error transforming datatype for: log.crash.maximum_message_size
 [error] 64DB
 [error] Error transforming datatype for: log.console.level
 [error] polite is not a valid enum value, acceptable values are
                          [debug, info, warning, error].
 [error] Error transforming datatype for: log.console
 [error] penguin is not a valid enum value, acceptable values are
                          [off, file, console, both].
Error generating config with cuttlefish
```

Ok, I’m in. How?

The Schema


default.
conf
app.
config
vm.args
Schema
escript
.conf
file

Fun Facts

- They’re written in Erlang
- Each schema element is a tuple
- The first element is the type of schema element it is

Schema Elements

- Mappings
- Translations
- Validators

Mappings

- Simplest Element

- Most Options

```
{mapping,
    "my.setting",
    "erlang_app.setting",
[]}.
```

1. ‘mapping’
2. Name
3. Destination
4. Other Options

```
{mapping,
    "my.setting",
    "erlang_app.setting",
    []}.
```

```
my.setting = foo
```

```
[
    {erlang_app, [
        {setting, "foo"}
    ]}
].
```

Strings? Really?

- Default, because anything in a file could be a string
- Need more? Use Datatypes!

Datatypes

- string
- integer
- atom
- file
- directory
- flag
- ip
- enum
- duration
- bytesize
- extended

```
{mapping,   "my.setting",
            "erlang_app.setting", [
            {datatype, atom}
]}.
```

```
[
                    {erlang_app, [
my.setting = foo  ------------>       {setting, foo}
                    ]}
].
```

More Interesting Datatypes

```
{enum, [one, two, three]}
```

- becomes an atom
- but only one of those valid atoms

flag
{flag, On, Off}

- default: on -> true, off -> false

- On, Off: On -> true, Off -> false

`{duration, Unit}`

- Takes a string with units, e.g. 1h30m
- converts that string to an integer in `Unit`
  - 1h30m `{duration, m}` -> 90
  - 1h30m `{duration, ms}` -> 5400000

`bytesize`

- Takes a string with units, e.g. 12MB
- converts that string to an integer in bytes
  - 12MB -> 12582912
  - 512KB -> 524288

Extended types

- a list of datatypes: `[integer, atom]`
- a list of specific values:
  - `[{duration, ms}, {atom, never}]`

```
%% @doc By default, Bitcask keeps all of your data around. If your
%% data has limited time-value, or if for space reasons you need to
%% purge data, you can set the `expiry` option. If you needed to
%% purge data automatically after 1 day, set the value to `1d`.
%%
%% Default is: `off` which disables automatic expiration
{mapping,
  "bitcask.expiry",
  "bitcask.expiry_secs", [
  {datatype, [{atom, off}, {duration, s}]},
  hidden,
  {default, off}
]}.
```

```
{default, Value}
```

- A default setting for the configuration key
- Included* in packaged .conf file
- If you don’t include a value in the user’s .conf file, this will be used
- Almost everything should have a default

```
true = proplists:is_defined(x, [{x, undefined}])
```

```
{commented, Value}
```

- A commented setting for the configuration key in a packaged .conf file

- This value will only be present in a packaged .conf file as an example

```
hidden,
{hidden, true}
```

- Setting will not be included in a packaged .conf file

- This is a way of creating hidden or “advanced” knobs

```
%% @doc MultiLine Comment DocString
```

- You can write documentation for a setting in the schema
- This gets included in the packaged .conf file
- The cuttlefish escript can also display it

```
%% @see other.config.key
```

- You just wrote the same @doc for another setting.
- This tells cuttlefish to reuse that @doc here too
- Unless this one has it’s own @doc, then it uses that and displays a reference to this other key’s @doc

Translations

- When mappings aren't enough
- When many .conf settings contribute to a single app.config setting

```
{translation,
 "riak_kv.anti_entropy",
 fun(Conf) ->
       {on, [debug]}
  end
}.
```

1. ‘translation’
2. destination
3. fun/1

```
{mapping, "anti_entropy", "riak_kv.anti_entropy", [
  {datatype, {enum, [active, passive, 'active-debug']}},
  {default, active}
]}.

{translation,
 "riak_kv.anti_entropy",
 fun(Conf) ->
     Setting = cuttlefish:conf_get("anti_entropy", Conf),
     case Setting of
       active -> {on, []};
       'active-debug' -> {on, [debug]};
       passive -> {off, []};
       _Default -> {on, []}
     end
   end
}.
```

```
anti_entropy = active
```

```
[
    {riak_kv, [
        {anti_entropy, {on, []}}
    ]}
].
```

```
{mapping,
  "anti_entropy.tree.build_limit.number",
  "riak_kv.anti_entropy_build_limit", [
  {default, 1},
  {datatype, integer},
  hidden
]}.

{mapping,
  "anti_entropy.tree.build_limit.per_timespan",
  "riak_kv.anti_entropy_build_limit", [
  {default, "1h"},
  {datatype, {duration, ms}},
  hidden
]}.

{translation,
 "riak_kv.anti_entropy_build_limit",
 fun(Conf) ->
    {cuttlefish:conf_get("anti_entropy.tree.build_limit.number", Conf),
     cuttlefish:conf_get("anti_entropy.tree.build_limit.per_timespan", Conf)}
 end}.
```

```
anti_entropy.tree.build_limit.number = 1
anti_entropy.tree.build_limit.per_timespan = 1h
```

```
[
    {riak_kv, [
        {anti_entropy_build_limit, {1, 3600000}}
    ]}
].
```

```
%% @doc By default, Bitcask keeps all of your data around. If your
%% data has limited time-value, or if for space reasons you need to
%% purge data, you can set the `expiry` option. If you needed to
%% purge data automatically after 1 day, set the value to `1d`.
%%
%% Default is: `off` which disables automatic expiration
{mapping, "bitcask.expiry", "bitcask.expiry_secs", [
  {datatype, [{atom, off}, {duration, s}]},
  hidden,
  {default, off}
]}.

{translation, "bitcask.expiry_secs",
 fun(Conf) ->
   case cuttlefish:conf_get("bitcask.expiry", Conf) of
     off -> -1;
     I when is_integer(I) -> I;
     _ -> cuttlefish:invalid("bad value for bitcask expiry")
   end
 end
}.
```

`bitcask.expiry = off`

```
[
  {bitcask, [
    {expiry_secs, -1}
  ]}
].
```

`bitcask.expiry = 1m30s`

```
[
  {bitcask, [
    {expiry_secs, 90}
  ]}
].
```

$name

```
{mapping,
  "listener.http.$name",
  "riak_api.http", [
  {datatype, ip}
]}.

{translation,
 "riak_api.http",
  fun(Conf) ->
      HTTP =
          cuttlefish_variable:filter_by_prefix(
              "listener.http",
              Conf),
      [ IP || {_, IP} <- HTTP]
  end
}.
```

```
listener.http.internal = 127.0.0.1:8098
listener.http.external = 10.10.1.12:8098
```

```
[
     {riak_api, [
          {http, [ {"127.0.0.1", 8098},
                   {"10.10.1.12", 8098}]}
     ]}
].
```

How would a default even work here?

```
{include_default, Substitution}
```

- Mappings can have a variable in the key.
- This represents a default value for that variable

new mapping property!

```
{mapping,
  "listener.http.$name",
  "riak_api.http", [
  {default, {"127.0.0.1", 8098}},
  {datatype, ip},
  {include_default, "internal"}
]}.

{translation,
 "riak_api.http",
  fun(Conf) ->
      HTTP =
          cuttlefish_variable:filter_by_prefix(
              "listener.http",
              Conf),
      [ IP || {_, IP} <- HTTP]
  end
}.
```

```
## listener.http.<name> is an IP address and TCP port that the Riak
## HTTP interface will bind.
##
## Default: 127.0.0.1:8098
##
## Acceptable values:
##   - an IP/port pair, e.g. 127.0.0.1:10011
listener.http.internal = 127.0.0.1:8098
```

```
{mapping,
  "riak_control.auth.user.$username.password",
  "riak_control.userlist", [
  {commented, "pass"},
  {include_default, "name"}
]}.

{translation,
"riak_control.userlist",
fun(Conf) ->
  UserList = cuttlefish_variable:filter_by_prefix(
                "riak_control.auth.user", Conf),
  Users = [
    {Username, Password}
    || {[_, _, _, Username, _], Password} <- UserList ],
  case Users of
      [] ->
          cuttlefish:unset();
      _ -> Users
  end
end}.
```

```
riak_control.auth.user.joe.password = 1234
riak_control.auth.user.miki.password = 5678
```

```
[
    {riak_control, [
        {userlist, [ {"joe", "1234"},
                     {"miki", "5678"} ]}
    ]}
].
```

```
riak_control.auth.user.joe.password = 1234
##riak_control.auth.user.miki.password = 5678
```

```
[
    {riak_control, [
        {userlist, [ {"joe", "1234"},
                     {"miki", "5678"} ]}
    ]}
].
```

```
## If riak control's authentication mode (riak_control.auth.mode)
## is set to 'userlist' then this is the list of usernames and
## passwords for access to the admin panel.
##
## Acceptable values:
##   - text
## riak_control.auth.user.name.password = pass
```

Support Functions

- `cuttlefish:conf_get/2 & 3`
- `cuttlefish:unset/0`
- `cuttlefish:invalid/1`
- `cuttlefish_variable:tokenize/1`
- `cuttlefish_variable:fuzzy_matches/2`
- `cuttlefish_variable:filter_by_prefix/2`

And Many More!!

Validators

```
{validator, "name", "message when fails",
   fun(Value) ->
       %% Returns true if valid
       %% false if not
       Value > 10
end}.
```

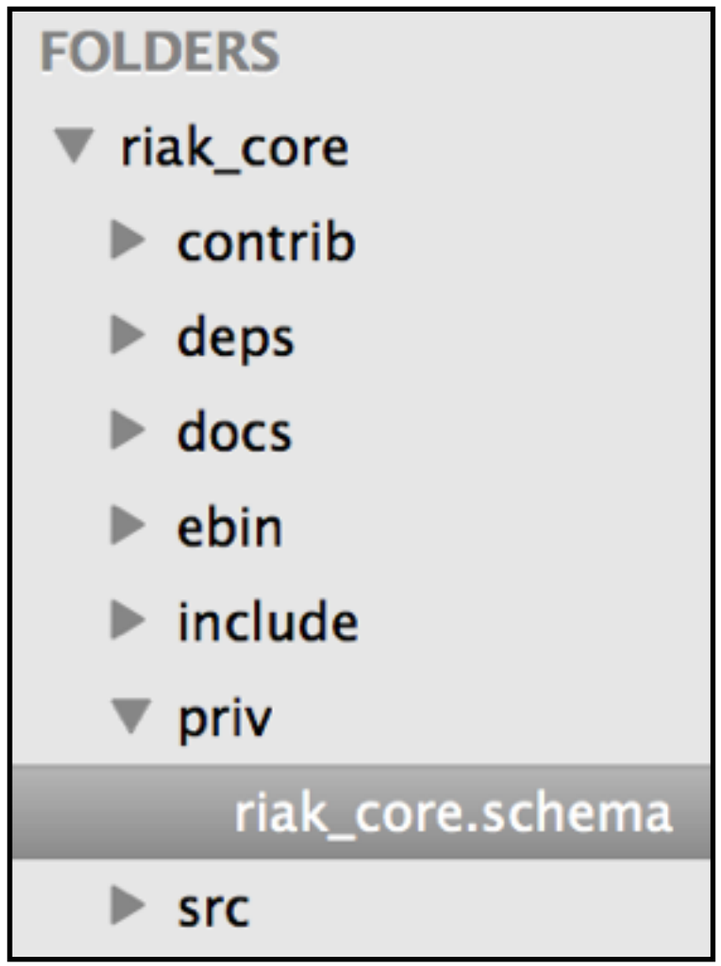
1. ‘validator’
2. Name
3. Failure Message
4. fun((term()) -> boolean())

```
{validators, ListOfValidators}
```

- Which validators to run on this value
- we’ll get more to that later

```
%% @doc Number of partitions in the cluster (only valid when first
%% creating the cluster). Must be a power of 2, minimum 8 and maximum
%% 1024.
{mapping, "ring_size", "riak_core.ring_creation_size", [
  {datatype, integer},
  {default, 64},
  {validators, ["ring_size^2", "ring_size_max", "ring_size_min"]},
  {commented, 64}
]}.

%% ring_size validators
{validator, "ring_size_max", "2048 and larger are supported, but
considered advanced config",
 fun(Size) ->
  Size =< 1024
 end}.

{validator, "ring_size_min", "must be at least 8",
 fun(Size) ->
  Size >= 8
 end}.

{validator, "ring_size^2", "not a power of 2",
 fun(Size) ->
  (Size band (Size-1) =:= 0)
 end}.
```

What even do you do with these Schemas?



Drop it like it's ./priv

Twist your mustache

{{bwahahahaha}}

```
%% @see platform_bin_dir
{mapping,
  "platform_data_dir",
  "riak_core.platform_data_dir", [
  {datatype, directory},
  {default, "{{platform_data_dir}}"}
]}.
```

Write an EUnit Test!

Defaults Test

```
Config = cuttlefish_unit:generate_templated_config(
                "../priv/riak_core.schema",
                [], context()),

cuttlefish_unit:assert_config(
      Config, "riak_core.ring_creation_size", 64),
cuttlefish_unit:assert_not_configured(
      Config, "riak_core.ssl.certfile"),
```

context()

```
%% this context() represents the substitution variables that rebar
%% will use during the build process.  riak_core's schema file is
%% written with some {{mustache_vars}} for substitution during
%% packaging cuttlefish doesn't have a great time parsing those, so we
%% perform the substitutions first, because that's how it would work
%% in real life.
context() ->
    [
    [
        {handoff_port, "8099"},
        {platform_bin_dir , "./bin"},
        {platform_data_dir, "./data"},
        {platform_etc_dir , "./etc"},
        {platform_lib_dir , "./lib"},
        {platform_log_dir , "./log"}
    ].
```

Custom Override Test

```
Conf = [
  {["ring_size"], 8},
  {["ssl", "certfile"], "/absolute/etc/cert.pem"}
],

Config = cuttlefish_unit:generate_templated_config(
               "../priv/riak_core.schema",
               Conf, context()),

cuttlefish_unit:assert_config(
     Config, "riak_core.ring_creation_size", 8),
cuttlefish_unit:assert_config(
     Config,
     "riak_core.ssl.certfile",
     "/absolute/etc/cert.pem"),
```

How about a nice cup o’ reltool?

reltool.config

```
{overlay, [
   {copy, "../deps/cuttlefish/cuttlefish",
          "{{erts_vsn}}/bin/cuttlefish"},
   %% Cuttlefish Schema Files have a priority order.
   %% Anything in a file prefixed with 00- will override
   %% anything in a file with a higher numbered prefix.

   %% Please only use 0[0-9]-*.schema for development purposes
   %% NOTHING PERMANENT
   {template, "files/riak.schema", "lib/10-riak.schema"},
   {template, "../deps/cuttlefish/priv/erlang_vm.schema",
              "lib/11-erlang_vm.schema"},

   {template, "../deps/riak_core/priv/riak_core.schema",
              "lib/12-riak_core.schema"},
   {template, "../deps/riak_api/priv/riak_api.schema",
              "lib/13-riak_api.schema"},
   {template, "../deps/riak_kv/priv/riak_kv.schema",
              “lib/14-riak_kv.schema"},
...
```

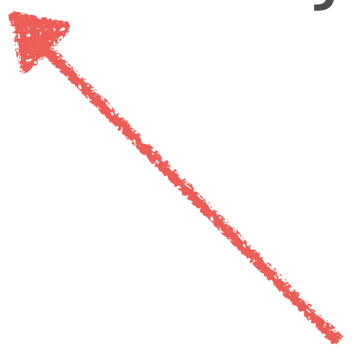
`merge`

- You can redefine mappings.
- this allows you to only partially override the mapping, instead of blowing it away entirely

new mapping property!

rel/rebar.config

```
{lib_dirs, ["../deps"]}.
{plugin_dir, "../deps/cuttlefish/src"}.
{plugins, [cuttlefish_rebar_plugin]}.
{cuttlefish_filename, "riak.conf"}.
```



Your Name Here!

Do you even node_package?

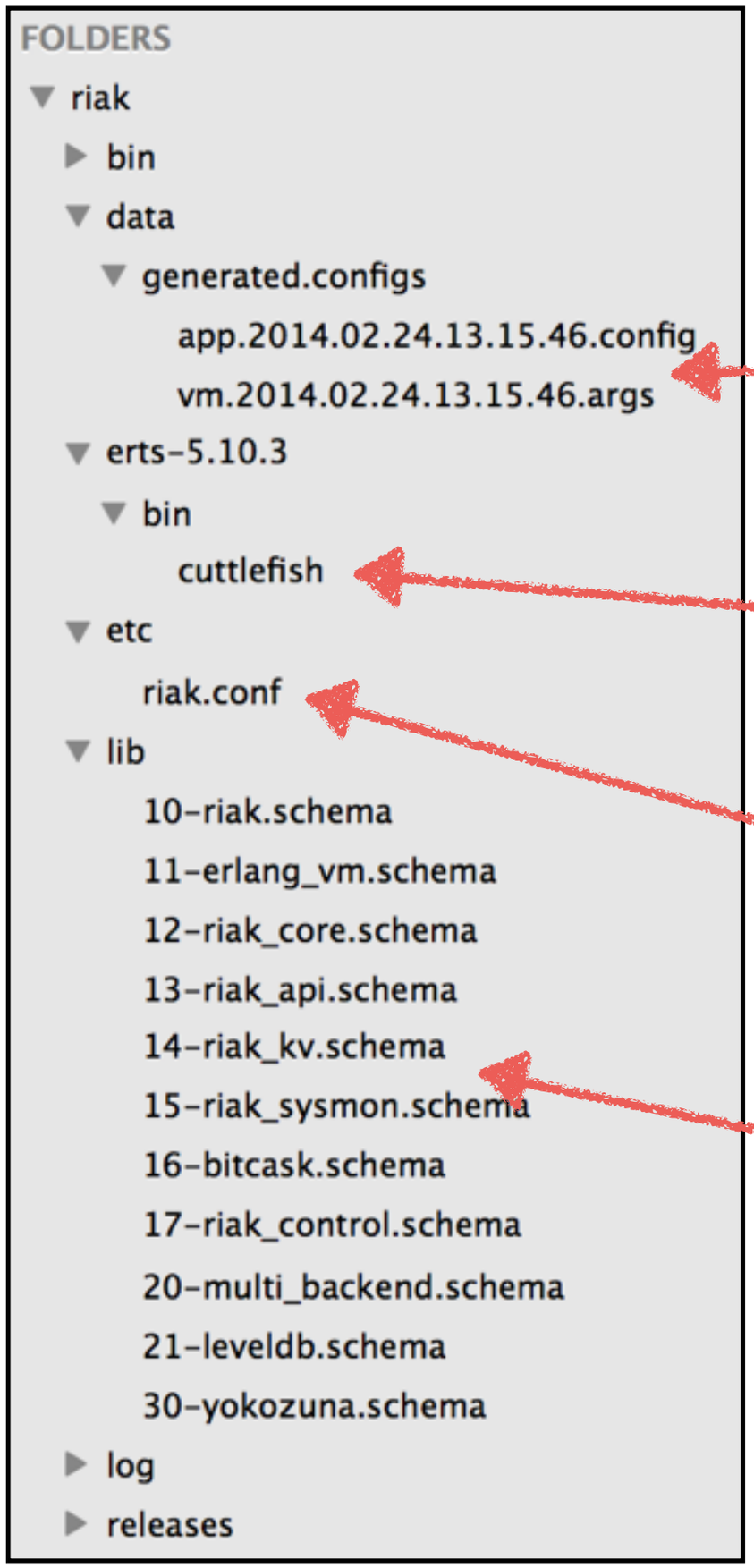
```
%% cuttlefish (rel/vars.config)
{cuttlefish,              "on"}.
{cuttlefish_conf,     "riak.conf"}.
```

node_package's gift to you

```
./cuttlefish -e ~/dev/basho/riak_ee/rel/riak/etc
             -s ~/dev/basho/riak_ee/rel/riak/lib
             -d ~/dev/basho/riak_ee/rel/riak/data/generated.configs
             -c ~/dev/basho/riak_ee/rel/riak/etc/riak.conf
```

```
➜  cuttlefish git:(develop) ./cuttlefish --help
Usage: ./cuttlefish [-h] [-e <etc_dir>] [-d <dest_dir>]
[-f <dest_file>] [-s <schema_dir>] [-i <schema_file>]
[-c <conf_file>] [-a <app_config>] [-l <log_level>] [-p]

  -h, --help        Print this usage page
  -e, --etc_dir     etc dir
  -d, --dest_dir     specifies the directory to write the config
                    file to
  -f, --dest_file    the file name to write
  -s, --schema_dir  a directory containing .schema files
  -i, --schema_file individual schema file, will be processed in
                    command line order, after -s
  -c, --conf_file   a cuttlefish conf file, multiple files allowed
  -a, --app_config  the advanced erlangy app.config
  -l, --log_level   log level for cuttlefish output
  -p, --print       prints schema mappings on stderr
```

Where's it all end up?



- Generated Config Files
- escript
- riak.conf
- Schemas

a generated app.config

```
            {pr,0}]}]},
 {riak_api,
     [{disable_pb_nagle,true},
      {pb_backlog,128},
      {http,[{"127.0.0.1",8098}]},
      {pb,[{"127.0.0.1",8087}]},
      {honor_cipher_order,true},
      {tls_protocols,['tlsv1.2']},
      {check_crl,true}]},
 {riak_kv,
     [{dvv_enabled,true},
      {max_siblings,100},
      {warn_siblings,25},
      {max_object_size,52428800},
      {warn_object_size,5242880},
      {secure_referer_check,true},
      {retry_put_coordinator_failure,true},
      {hook_js_vm_count,2},
      {reduce_js_vm_count,6},
      {map_js_vm_count,8},
      {anti_entropy_leveldb_opts,
          [{use_bloomfilter,true},
```

a generated vm.args

```
-setcookie riak
+P 256000
+e 256000
-env ERL_CRASH_DUMP ./log/erl_crash.dump
-env ERL_FULLSWEEP_AFTER 0
+Q 65536
+A 64
-name riak@127.0.0.1
+K true
+W w
-smp enable
```

How'd that happen?

```
{mapping, "erlang.smp", "vm_args.-smp", [
  {default, enable},
  {datatype, {enum, [enable, auto, disable]}},
  hidden
]}.
```

```
erlang.smp = enable
```

```
[
    {vm_args, [
        {"-smp", enable}
    ]}
].
```

vm.args magic!

```
➜  riak git:(develop) ✗ ./bin/riak config generate

 -config /Users/joe/dev/basho/riak_ee/rel/riak/data/
generated.configs/app.2014.03.07.11.13.09.config

-args_file /Users/joe/dev/basho/riak_ee/rel/riak/
data/generated.configs/vm.2014.03.07.11.13.09.args

-vm_args /Users/joe/dev/basho/riak_ee/rel/riak/data/
generated.configs/vm.2014.03.07.11.13.09.args
```

github.com/basho/cuttlefish