Mid-Air Plane Repair: Debugging in Production

Maxim Fedorov

Performance and Scalability Engineer @ WhatsApp

WhatsApp – What's Up?

#1 messaging app

- Android, iPhone, Windows Phone, JioPhone
- Web and native clients
- More than 1.5B monthly active users
- WhatsApp Business products
- Thousands of servers to maintain

Large Scale Deployment Problems

Testing does not replace troubleshooting

Uncontrolled Environment

Moving pieces

- Hardware
- Operating system
- Kernel version
- Shared libraries
- Running services
- Application components
- Code deployed and executed

Troubleshooting Workflow

Workflow

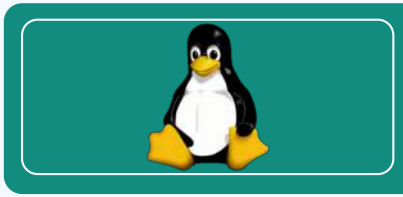
Detection: Monitoring, Alerts, Dashboards

Application & OTP

- Counters & Gauges, per node & aggregates
- Workers -> Factories -> Industries

Virtual Machine

- Internal: schedulers, allocators, queues, processes, ports, microstate accounting
- External: CPU, memory (RSS,VSZ), file descriptors, log file growth



OS, kernel, system applications

- atop data (+log files)
- perf samples (Strobelight)

Hardware Health

- CPU, RAM, NICs, PSU, sensors

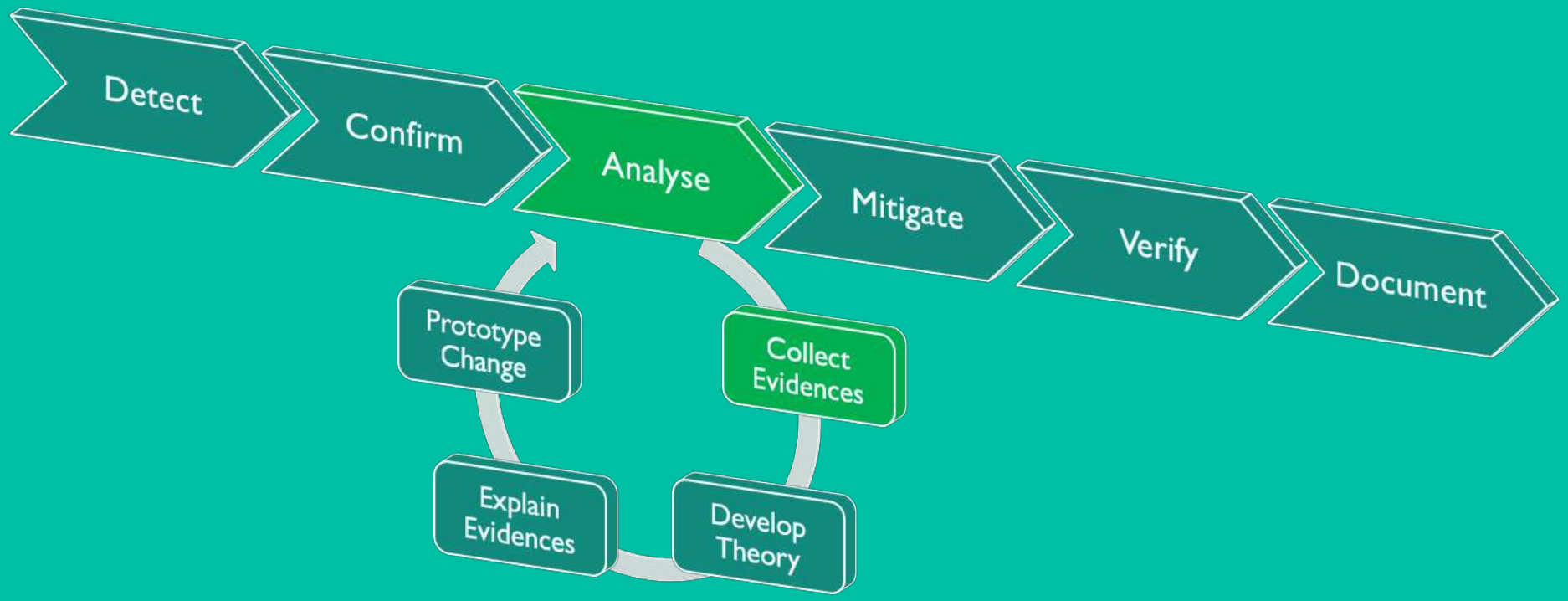
Workflow

Filter Out False Positives

Account for baseline changes

- Soccer matches (login/messages spikes)
- User base growth (640k is enough for everyone)
- Cluster size changes (more users, less servers?)
- BEAM configuration limits (pix table, atom table, etc.)

Workflow


Detect
Confirm
Analyse
Mitigate
Verify
Document
Prototype Change
Collect Evidences
Explain Evidences
Develop Theory

Evidence Collection

Preserve as much as possible, store in a cool dry place

- Crash & core dumps
- Emulator error reports, supervisor reports
- Log files (including host logs, atop logs, kernel messages)
- External change logs and RPC audit

- Additional on-demand data
- High-resolution metrics timelines (counters, gauges)

ectl Framework

Quick look into system state

```
/home/whatsapp# ectl pi 54

Process <0.54.0> (inet_db):
 Current location is gen_server/loop:7 (gen_server.erl:403))
 Current function is gen_server/loop:7
 Message queue length is 112304
 Process size: 23247860 bytes (heap=6664208 stack=80 total=23246872)
 Current stack depth is 2:
  1: gen_server:loop/7 (gen_server.erl:403)
  2: proc_lib:init_p_do_apply/3 (proc_lib.erl:249)
 Dictionary
```

ectl: Connecting Unix Shell and Erlang

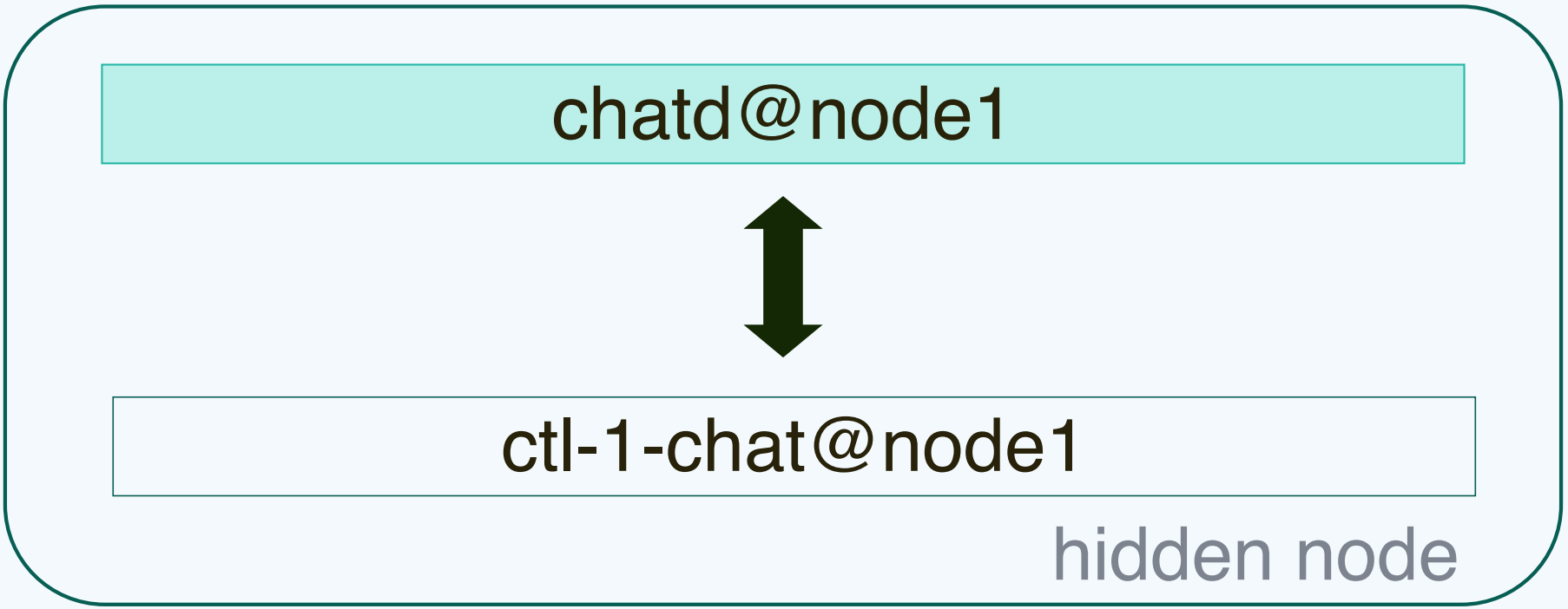
Validators, default values, usage

```
/home/whatsapp# ectl man load

load [MODULES]... [--force] [--background-purge] [--purge purge] [-a] [-d] [-s scope]
                          --  Code load specified module(s), using specific load order and scope
  MODULES string ...              -- Module name to load or force reload
  --force                         -- Forcefully purge modules with processes lingering in old code
  --background-purge              -- Perform asynchronous soft_purge, default: true
  --purge 0 <= purge <= 1000000   -- Load order: wait (ms) for old code to be purged after module
load, default: 30000
  -a, --all                       -- Load all modified modules after specified ones
  -d, --dry                       -- Prepares loading and prints load order, but does not load
  -s, --scope string              -- Perform a scoped operation, see 'ectl scoped_help' for
details
```

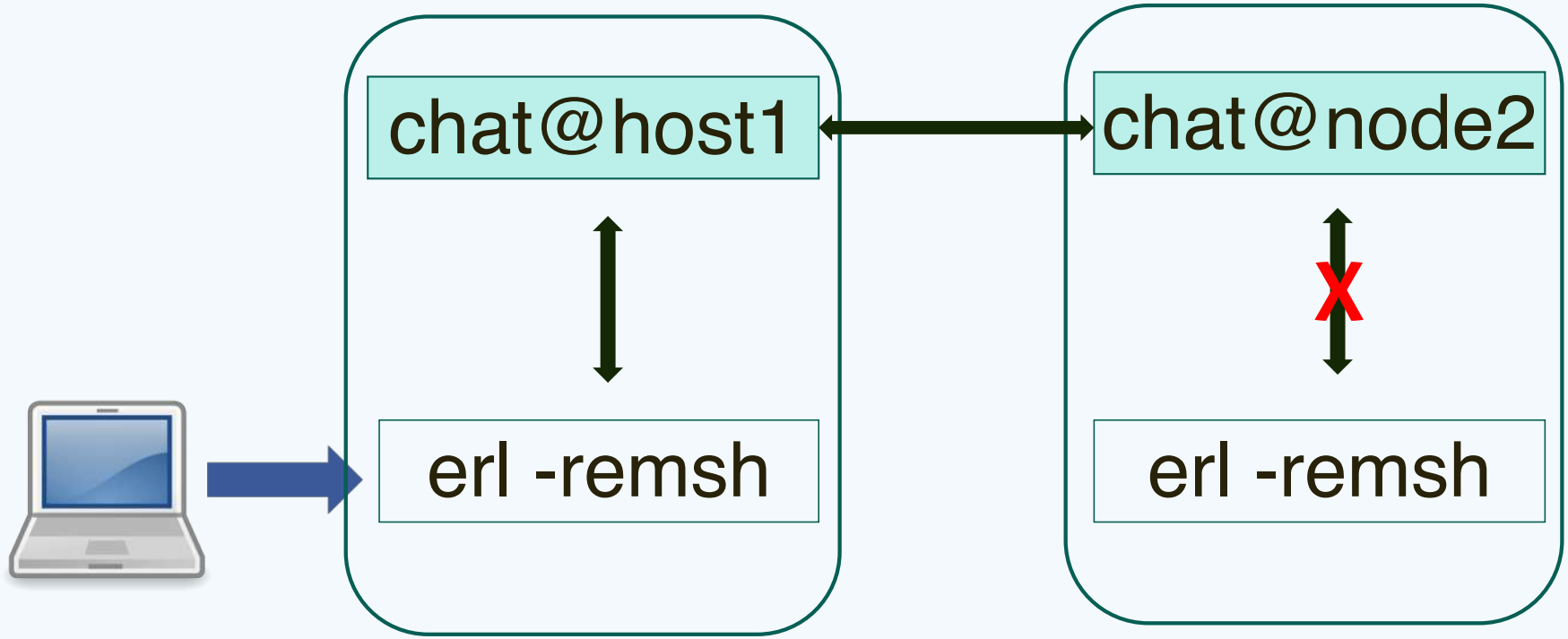
ectl: Just a Hidden Node

ectl, ejabberdctl successor, joins distribution cluster



Remote Shell

When deeper introspection is required



Erlang power_shell

Available on GitHub

```
Eshell V10.2  (abort with ^G)

1> dist_util:gen_challenge().
** exception error: undefined function dist_util:gen_challenge/0

2> eval(dist_util, gen_challenge, []).
558236439
```

OTP - Patching Shell

Syntax sugar makes a candy

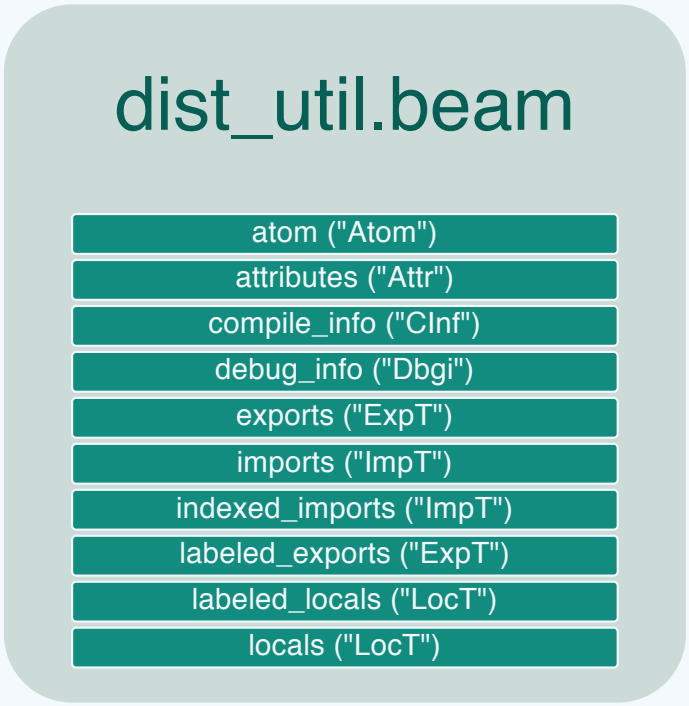
```
4> dist_util:gen_challenge().
** exception error: undefined function dist_util:gen_challenge/0

5> application:set_env(stdlib, shell_debug_unexported, true).
ok

6> dist_util:gen_challenge().
558236439
```

Calling Non-Exported Functions

Executing interpreted code from embedded debug_info

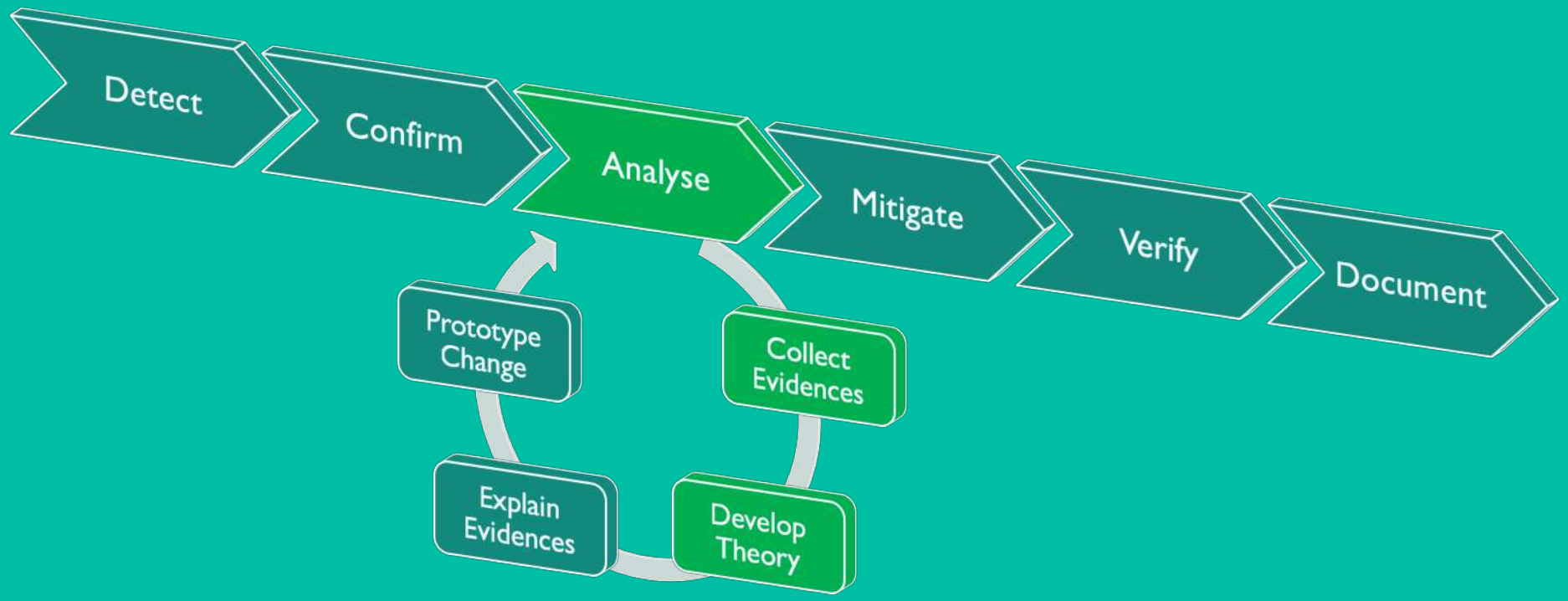



```
 {function,166,handshake_other_started,1,
           [{clause,166,[{match,...}],[],[...]},
            {clause,198,[{...}],[...],...}]},
 {function,202,log_disallowed_node_attempt,1,
           [{clause,202,[{...}],[],...}]},
 {function,213,check_dflags,2,[{clause,213,[...],...}]},

< ... >

 {function,458,gen_challenge,0,
           [{clause,458,[],[],
                    [{match,459,
                            {var,459,'A'
```

Workflow


Detect
Confirm
Analyse
Mitigate
Verify
Document
Prototype Change
Collect Evidences
Explain Evidences
Develop Theory

Realtime Monitoring Data

High resolution metrics to confirm an ongoing issue

```
washared528    | ERL   msg------------ dist--------- wan----------  sched mem---
time           | nodes qlen qmax  recv  msgin msgout  wanin wanout  %util tot Mb
01/24 20:39:10   1200    0    0 37043   2623   2631   4525  10074   92.3  21328
01/24 20:39:11   1200    1    1 40571   2734   2699   4620  10600   92.7  21305
01/24 20:39:12   1200    1    1 56928   2582   2678   5507  12572   95.7  21316
```

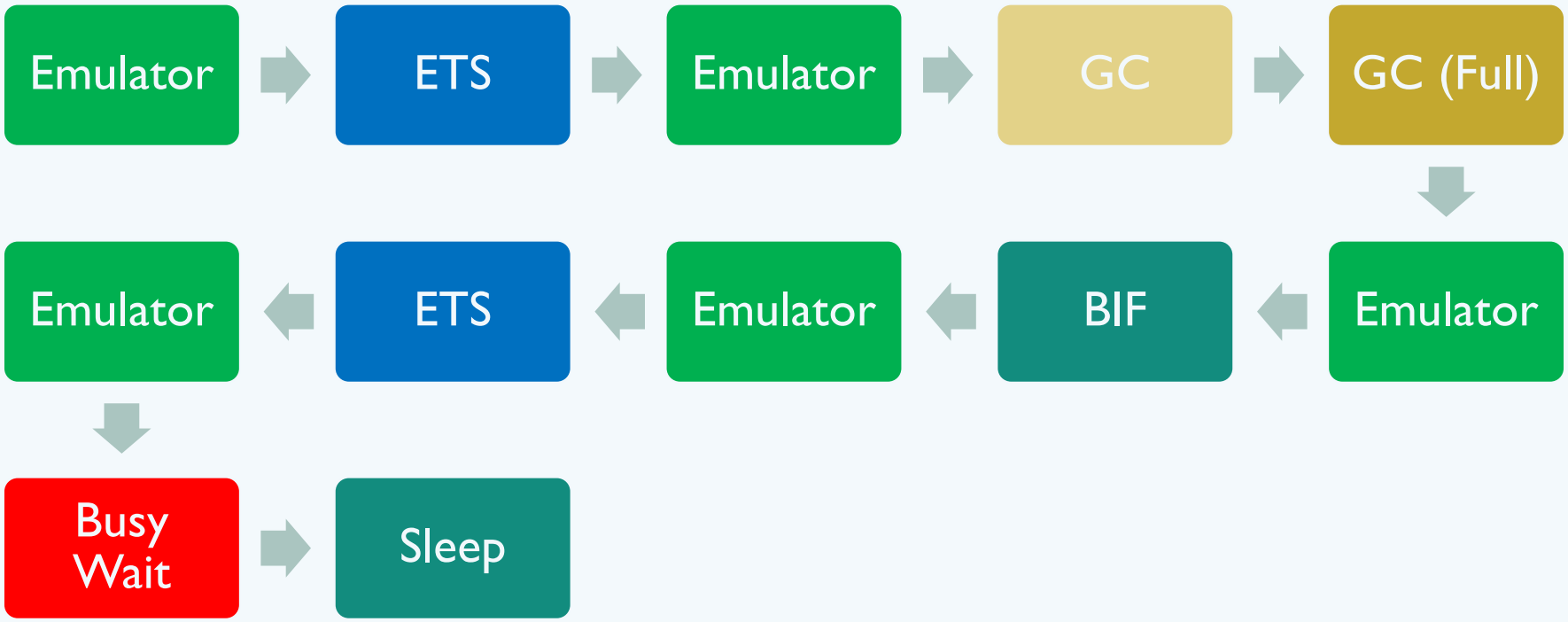
Resource Classes

CPU, memory, network

- Normal schedulers (emulator threads)
- Dirty CPU schedulers
- Dirty I/O schedulers
- I/O polling thread(s)
- Auxiliary threads

Scheduler Is New CPU Core

Life of a perfectly normal scheduler



Microstate Accounting

With some CPU to spare

Thread	alloc	bif	emulator	ets	gc	nif	sleep
Stats per type:							
async	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
aux	0.15%	0.00%	0.00%	0.00%	0.00%	0.00%	98.12%
dirty_cpu_sche	0.13%	0.00%	0.00%	0.00%	0.34%	0.00%	99.38%
dirty_io_sched	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
poll	0.07%	0.00%	0.00%	0.00%	0.00%	0.00%	99.27%
scheduler	**6.06%**	**10.14%**	**29.70%**	**9.00%**	**3.42%**	**5.94%**	**23.17%**

Microstate Accounting

ETS lock contention

Thread	alloc	bif	emulator	ets	gc	nif	sleep
dirty_cpu_sche	4.13%	0.00%	0.00%	0.00%	0.34%	0.00%	99.38%
scheduler	0.41%	0.82%	4.11%	82.02%	1.01%	1.01%	7.11%

Microstate Accounting

Misbehaving NIFs (possible lock contention)

Thread	alloc	bif	emulator	ets	gc	nif	sleep
dirty_cpu_sche	4.13%	0.00%	0.00%	0.00%	0.34%	0.00%	99.38%
scheduler	2.66%	2.14%	10.11%	2.02%	1.01%	78.94%	2.81%

Microstate Accounting

Too much garbage produced

Thread	alloc	bif	emulator	ets	gc_full	nif	sleep
dirty_cpu_sche	7.13%	0.00%	0.00%	0.00%	82.61%	0.00%	10.71%
scheduler	5.96%	11.04%	19.50%	9.00%	0.42%	5.94%	45.17%

Memory

RAM consumption – inspired by recon library by Fred Hebert

- Allocators statistics
- Memory leaks analysis
- Supervision tree aggregates

Allocation Statistics

Readily available via ectl framework

```
dev1214.atn /home/whatsapp# ectl erl_alloc
=================================================================================
Allocator       % util   Allocated       Used       Free     Blocks   Avg Size  Strategy
=================================================================================
temp_alloc        0.00      4.13MB         0B      4.13MB          0          -     gf,af
sl_alloc          0.05      1.03MB       592B      1.03MB          7         84B   aoffcbf
std_alloc         5.87     67.03MB      3.94MB     63.10MB      20823        198B   aoffcbf
ll_alloc         34.95    196.50MB     68.67MB    127.83MB      73507        979B   aoffcbf
eheap_alloc      44.56      1.22GB    555.28MB    690.84MB      24444     23.26KB   aoffcbf
ets_alloc        17.59      18.38GB     3.23GB     15.15GB   21004902        165B   aoffcbf
fix_alloc        18.89     65.03MB     12.29MB     52.75MB      62698        205B   aoffcbf
literal_alloc    80.23      4.00MB      3.21MB    809.65KB        300     10.95KB      aobf
binary_alloc     49.99      1.10GB    565.45MB    565.58MB     137795      4.20KB   aoffcbf
driver_alloc     15.01     65.03MB      9.76MB     55.27MB      16157        633B   aoffcbf
=================================================================================
```

Allocation Statistics

Changed allocation strategy

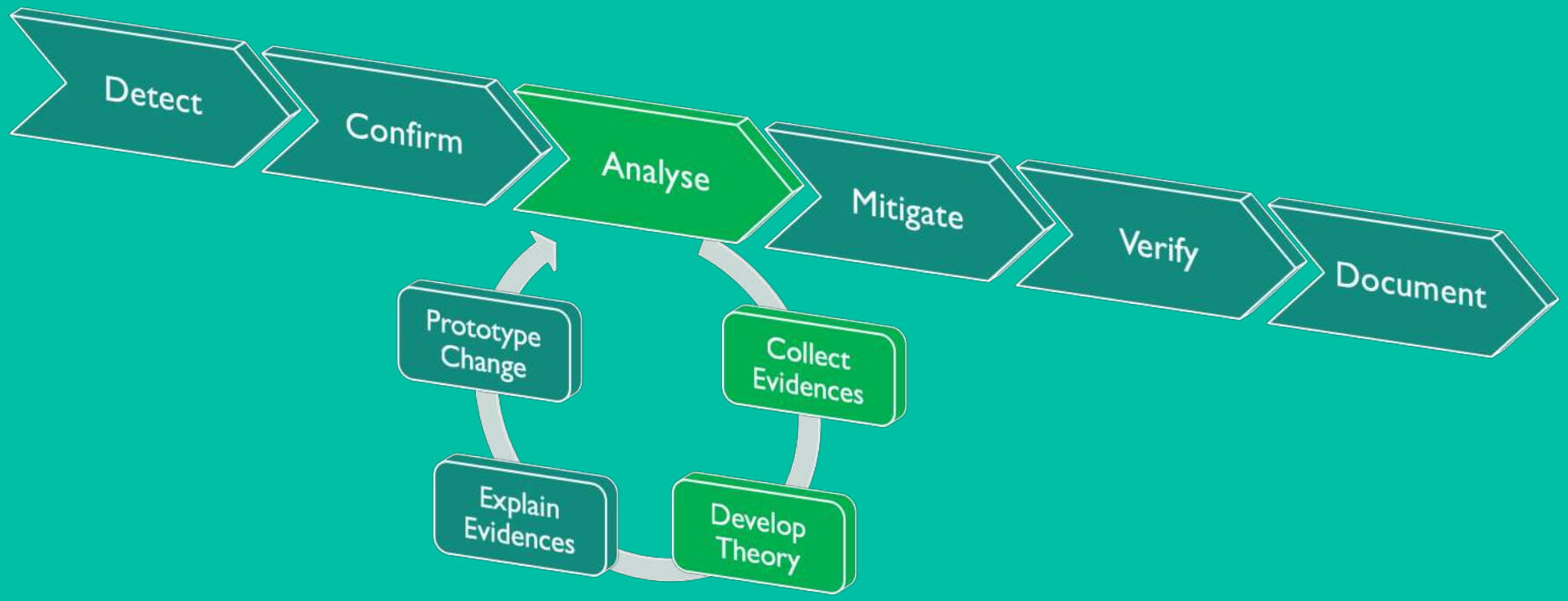
```
dev1214.atn /home/whatsapp# ectl erl_alloc
=================================================================================
Allocator       % util   Allocated       Used       Free     Blocks   Avg Size  Strategy
=================================================================================
temp_alloc        0.00      4.13MB         0B      4.13MB          0          -     gf,af
sl_alloc          0.02      1.03MB       200B      1.03MB          2       100B   aoffcbf
std_alloc         5.30     67.03MB      3.55MB     63.48MB      20727       179B   aoffcbf
ll_alloc         50.04    136.50MB     68.30MB     68.20MB      72750       984B   aoffcbf
eheap_alloc      50.33      1.03GB    532.54MB    525.59MB      24208    22.53KB   aoffcbf
ets_alloc        72.79      3.09GB      2.25GB    862.42MB   16335118       148B      aobf
fix_alloc        18.58     65.03MB     12.08MB     52.95MB      60052       210B   aoffcbf
literal_alloc    79.17      4.00MB      3.17MB    853.28KB        287    11.30KB      aobf
binary_alloc     45.94      1.07GB    503.09MB    591.94MB     132180     3.90KB   aoffcbf
driver_alloc     18.23     65.03MB     11.86MB     53.18MB      28038       443B   aoffcbf
=================================================================================
```

Memory Leaks Analysis

Forcing garbage collection to detect dirty processes

```
dev1214.atn /home/whatsapp# ectl erl_leaks
=================================================================================================
Pid            Name          binary_refs            binary_size    binary_count              heap_size
=================================================================================================
<0.6410.0>    <...>    1392547 (1392547)    230.61MB (230.61MB)     1264 (1264)    224.14KB (211.66KB)
<0.3424.6>    <...>      700592 (700591)    163.29MB (163.29MB)      860 (859)    362.67KB (277.06KB)
<0.5564.0>    <...>      107753 (107753)    155.94MB (155.94MB)      444 (444)    586.82KB (574.34KB)
<0.175.0>    wa_wan      284156 (284156)    126.50MB (126.50MB)      588 (588)     85.61KB (73.13KB)
<0.54.0>    inet_db      175151 (175150)    123.72MB (123.72MB)      913 (912)    586.82KB (501.21KB)
<0.23406.5>   <...>      950628 (950628)    114.72MB (114.72MB)      978 (978)    224.14KB (216.43KB)
<0.23438.5>   <...>      345748 (345748)     69.19MB (69.19MB)       592 (592)    224.14KB (216.43KB)
<0.3482.6>    <...>              152 (0)          56.50MB (0B)          8 (0)     12.48KB (-20696B)
<0.70.0>        pg2               45 (0)          54.27MB (0B)          5 (0)     12.48KB (-20696B)
<0.340.0> wa_shell                43 (0)          54.27MB (0B)          5 (0)     12.48KB (-20696B)
=================================================================================================
```

Workflow


Detect
Confirm
Analyse
Mitigate
Verify
Document
Prototype Change
Collect Evidences
Explain Evidences
Develop Theory

gdb: Open Heart Surgery

Non-stop mode + background execution

```
(gdb) file /usr/lib64/erlang/erts-10.2/bin/beam.smp
(gdb) source /usr/lib64/erlang/etc/etp-commands
(gdb) set target-async 1
(gdb) set pagination off
(gdb) set non-stop on
(gdb) attach 97806 &
```

etp-commands

BEAM is able to survive short interruptions

```
(gdb) thread 73
(gdb) interrupt 73
(gdb) etp-schedulers
--- Scheduler 1 ---
 IX: 0
 CPU Binding: 0
  - Run Queue -
  Length: total=1123, max=0, high=0, normal=1122,
low=1, port=0
```

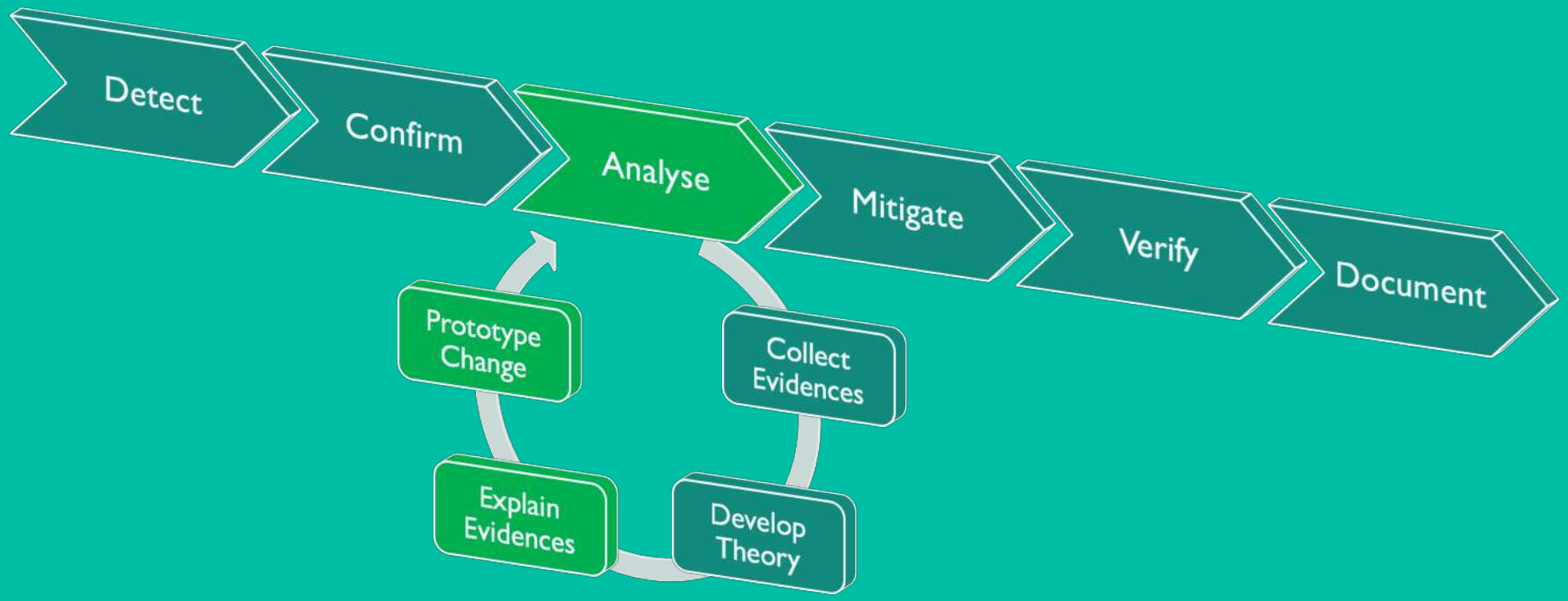
Hack The Source

```
            for (i=nlocks-1; i>=0; --i) {
-               lock_vec[i] = enif_rwlock_create("crypto_stat");
+               snprintf(lock_name, sizeof(lock_name), "crypto_%s", CRYPTO_get_lock_name(i));
+               lock_vec[i] = enif_rwlock_create(lock_name);
                if (lock_vec[i]==NULL) return NULL;
```

Meaningful lock name

```
(chatd@cdev0015.frc)3> lcnt:conflicts().
            lock    id   #tries  #collisions  collisions [%]  time [us]  duration [%]
           -----   ---  ------- ------------ --------------- ---------- -------------
      crypto_rsa     1   280851       190678         67.8929  899670045      617.4626
         db_tab  9282 13009375       132871          1.0213    5075031        3.4831
      run_queue    34 51182832       771052          1.5065     987565        0.6778
```

Workflow


Detect
Confirm
Analyse
Mitigate
Verify
Document
Prototype Change
Collect Evidences
Explain Evidences
Develop Theory

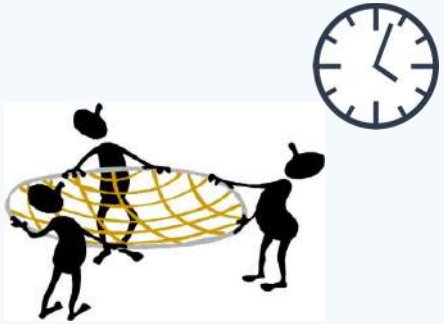
Prototyping In Production

Trying a change in production – fastest turnaround time

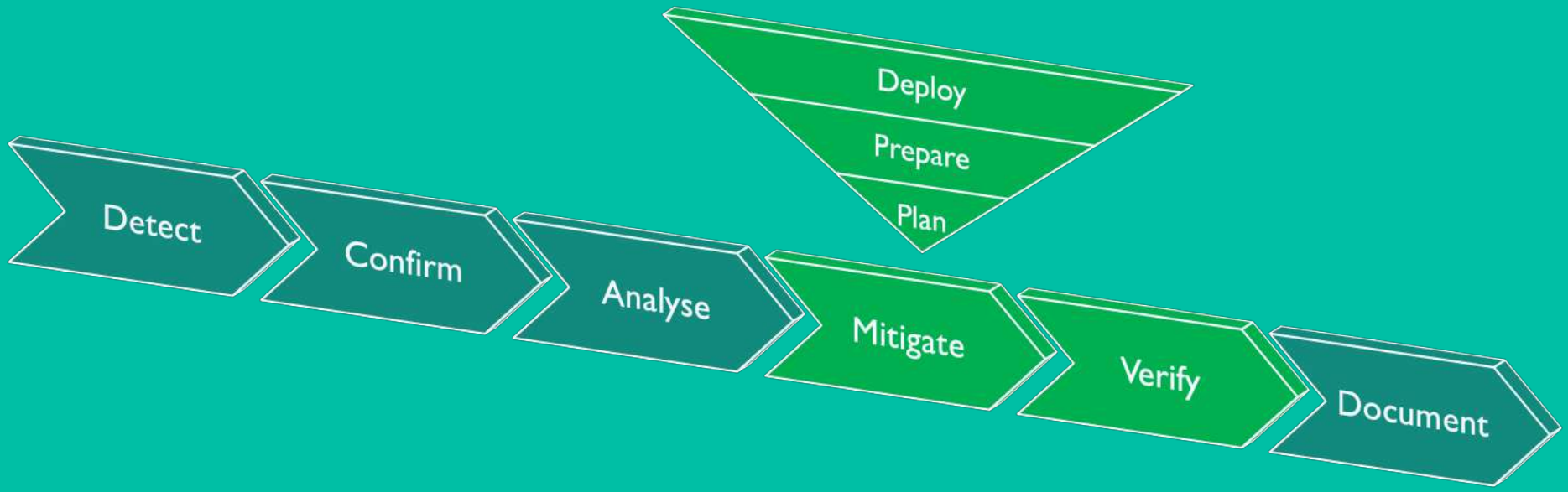
- Toolchain is installed on every production node
- Using code hot-load
- Immediate effect – for good or bad
- Does not replace testing

Reliability Layer: Peace of Mind

Messaging service is a closed loop



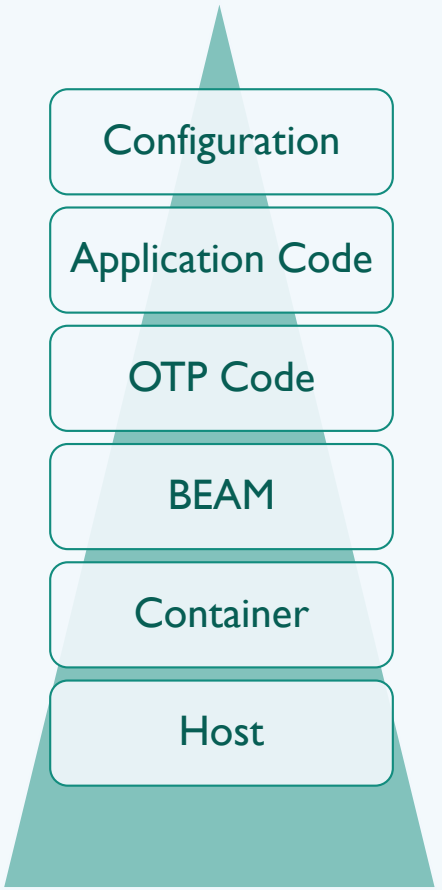
Workflow


Deploy
Prepare
Plan
Detect
Confirm
Analyse
Mitigate
Verify
Document

Plan, Prepare, Deploy

Plan always comes first

- Define change level
- Announce/notify interested parties
- Execute mitigation plan
- Test the whole stack:
  - hot code load/live update
  - restart of an underlying layer

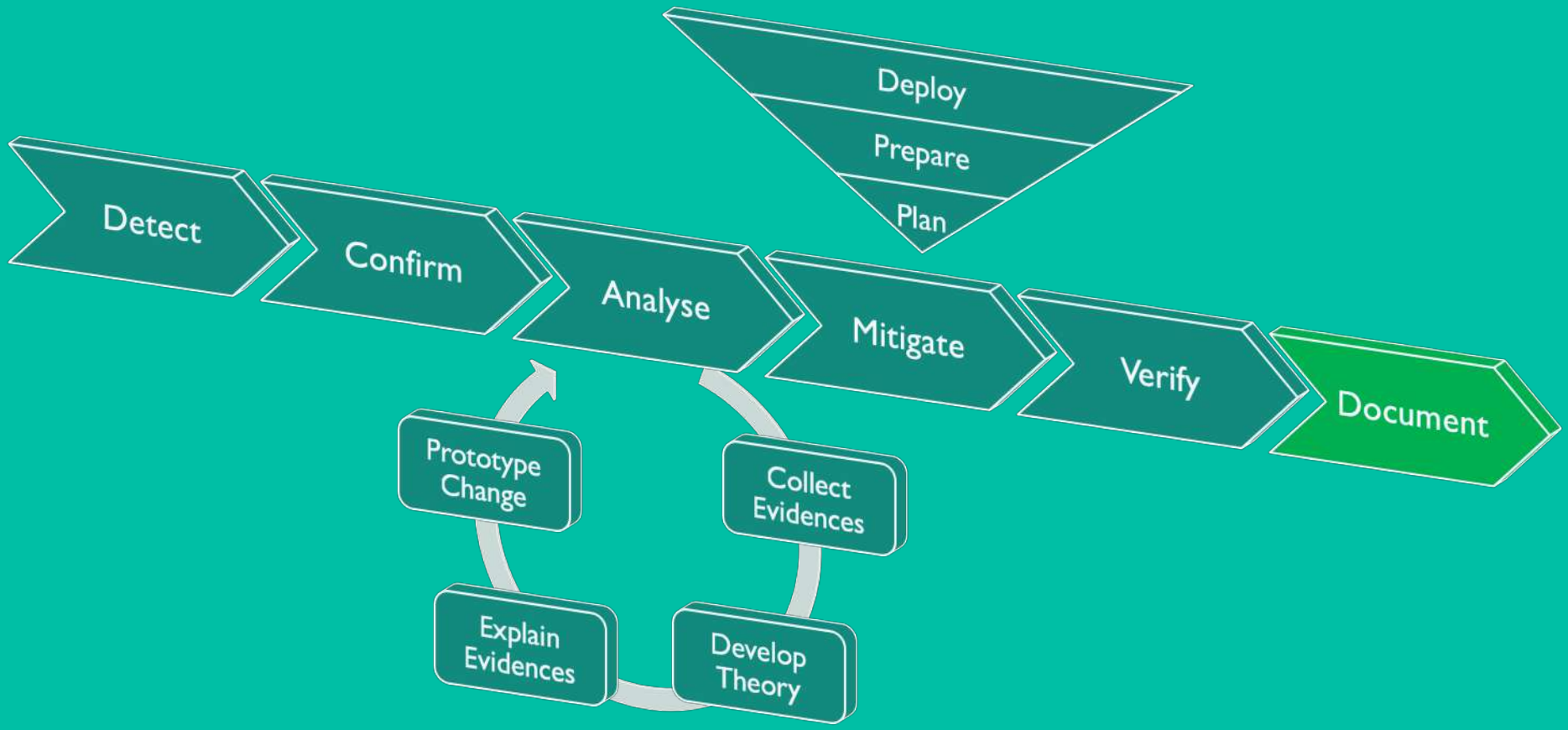


Slow Roll

Keep failover option as long as possible

- Deploy to a single node
- Deploy to a range of nodes in a single DC
- [Optional] Deploy to a range of nodes in multiple DCs
- Deploy to a single DC
- Full-scale deployment

Workflow


Deploy
Prepare
Plan
Detect
Confirm
Analyse
Mitigate
Verify
Document
Prototype Change
Collect Evidences
Explain Evidences
Develop Theory

Test Case As Documentation

It's better if it's automated

- Stays up to date
- Serves as a contract (interface and use cases)
- Catches regressions
- Not always possible
- Not always feasible
- Does not replace **monitoring and evidence collection**!

Questions?

Maxim Fedorov
dane@whatsapp.com
GitHub: max-au, whatsapp