

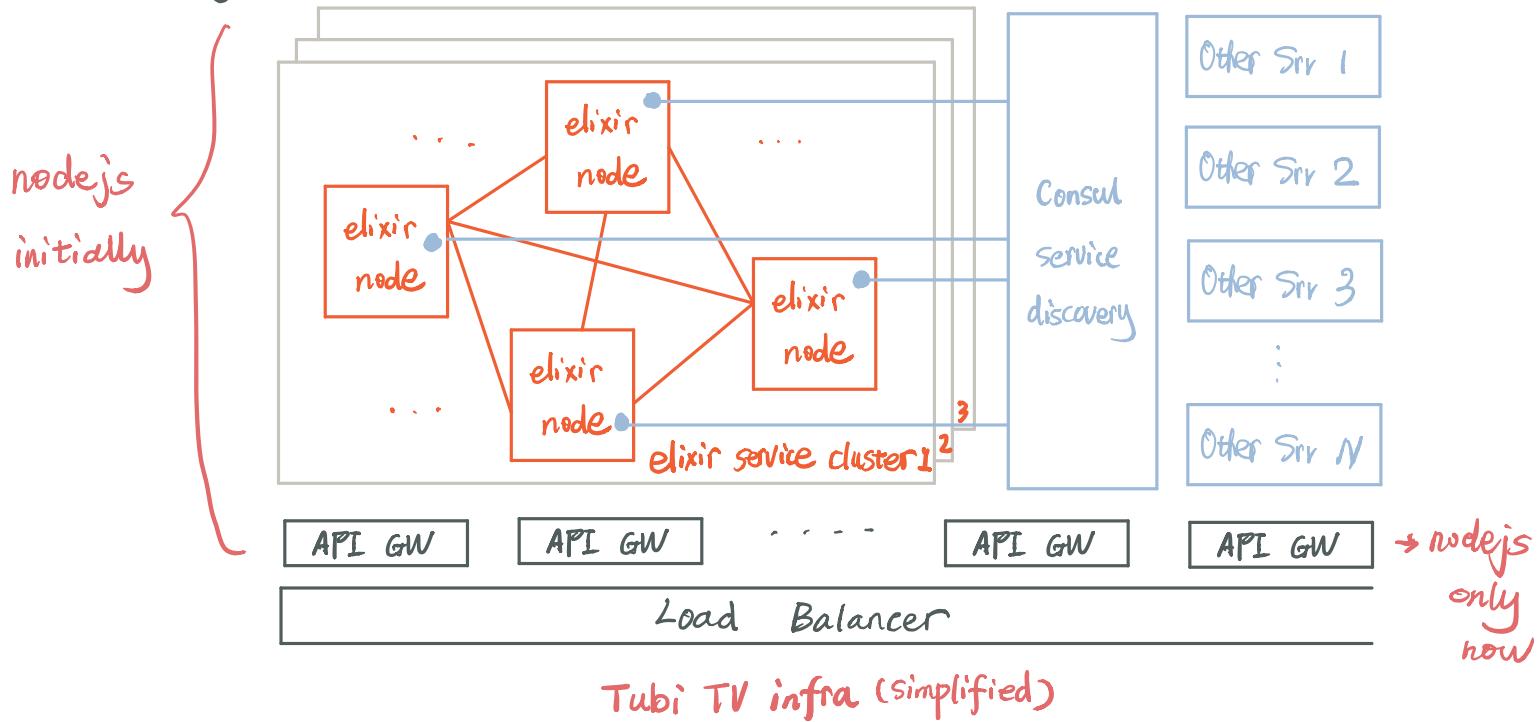
Release Deploy Upgrade Monitor

Elixir Services

in real world

Tyr Chen

# Background



# The history (and problems)

- Core problem: which video is allowed to play?
  - (`["us", "CA"]`, `{1/1/2018, 3/31/2018}`, not `["roku", "firetv"]`)  
⇒ allowed
- compiles business rules (in db) into fn  
yes  
us
- reduced latency from 200ms to 50us
  - 95%-tile
  - 1,000 contents
  - 5,000 - 10,000 rules

# The history (and problems)

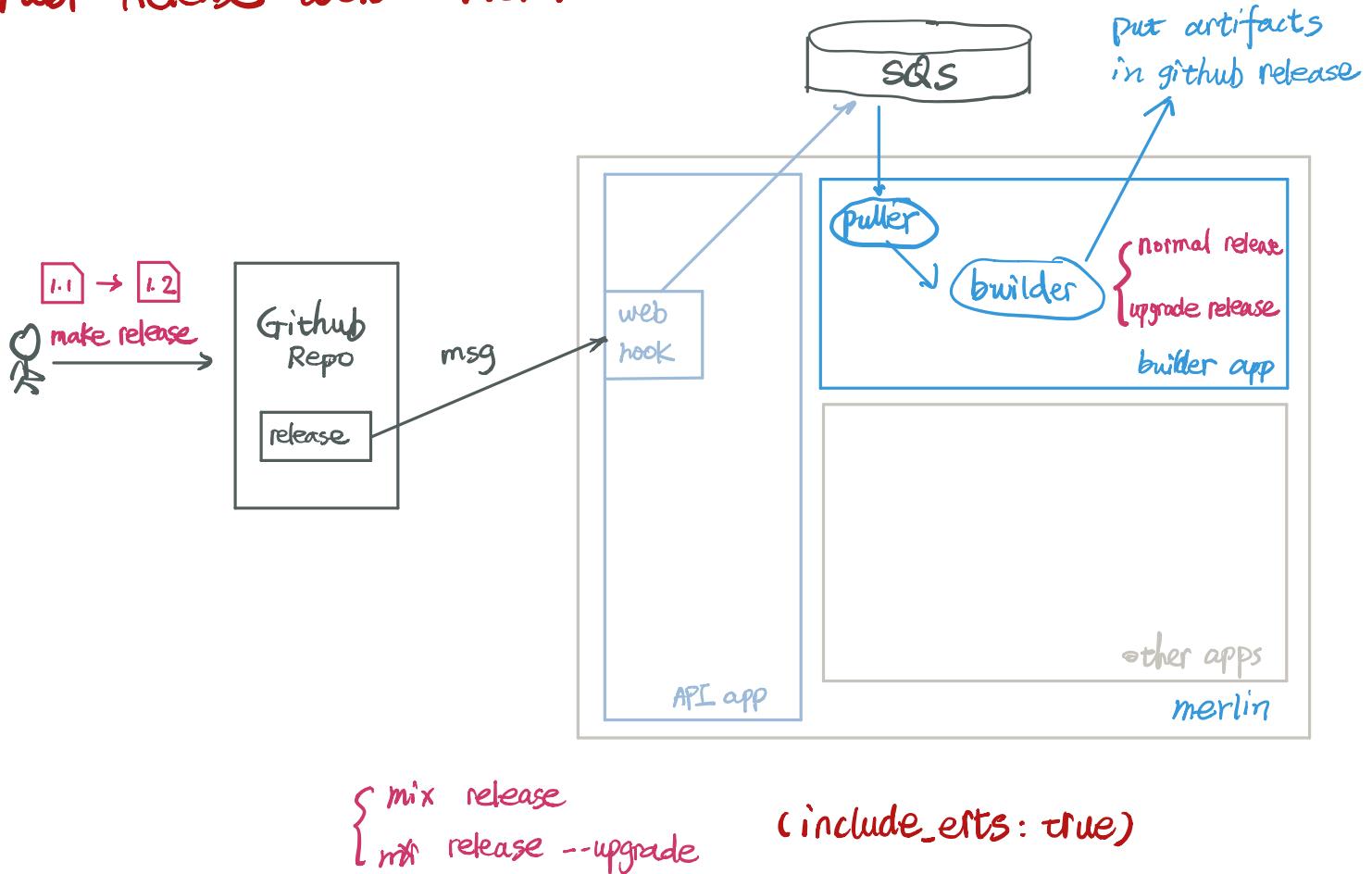
- We built more services
  - stateless (use external data source)
  - stateful (use ets / mnesia)
- We want to be able to release **any time**
  - simple, automated process
  - without down time
  - engs get notified on bad things

# Release with distillery

Challenges:

- database access at compile time
- normal release v.s. upgrade release
- vm.args (cookie!)
- source files to copy over

## Tubi Release tools - merlin



v4.4.8

92e4e45

Edit

# v4.4.8

 CatTail released this 12 days ago · 5 commits to master since this release

## Assets

 [cms\\_service.prod.tar.gz](#)

26.9 MB

 [cms\\_service.staging.tar.gz](#)

25.7 MB

 [Source code \(zip\)](#)

 [Source code \(tar.gz\)](#)

generated by "merlin"

release on branch master

generated by "make release"

92e4e45 | [REDACTED]

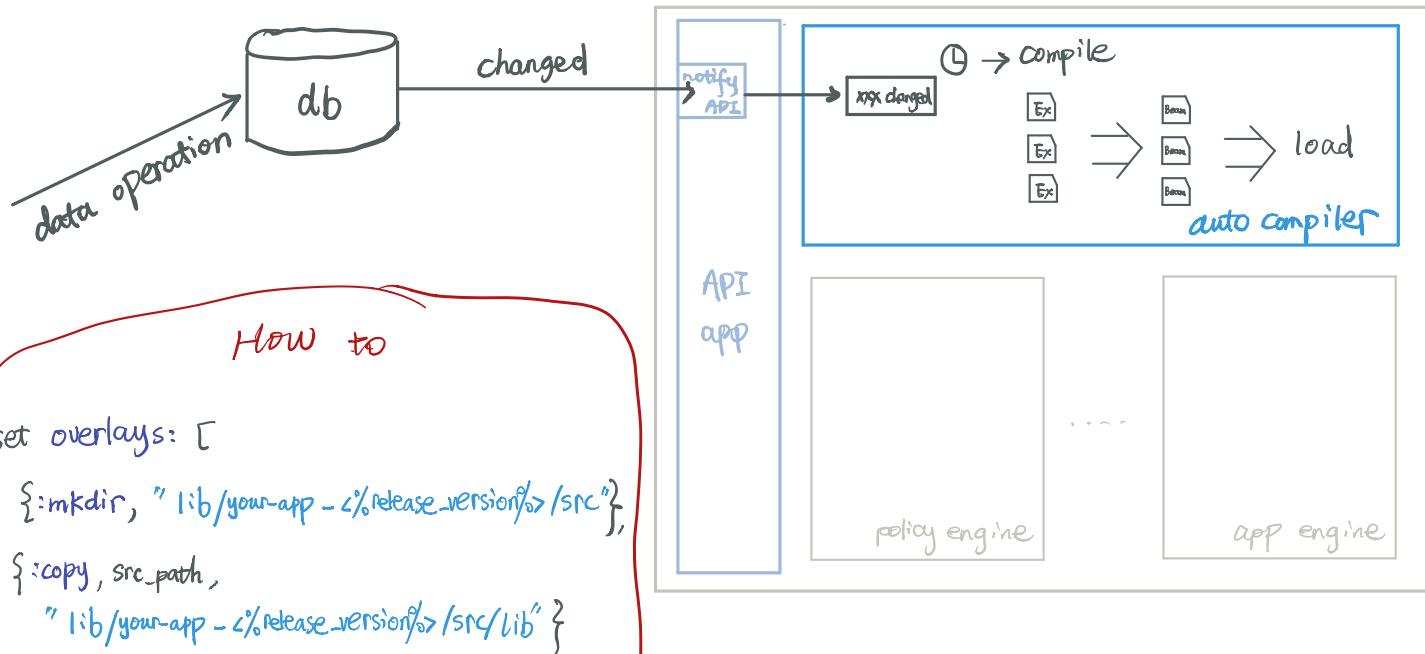
ecdf269 | [REDACTED]

ce221ce | [REDACTED]

16d58a3 | [REDACTED]

3aa1228 | [REDACTED]

# Src files



How to

set overlays: [

```
{:mkdir, "lib/your-app-#{%release_version%}/src"},  
{:copy, src_path,  
 "lib/your-app-#{%release_version%}/src/lib"}
```

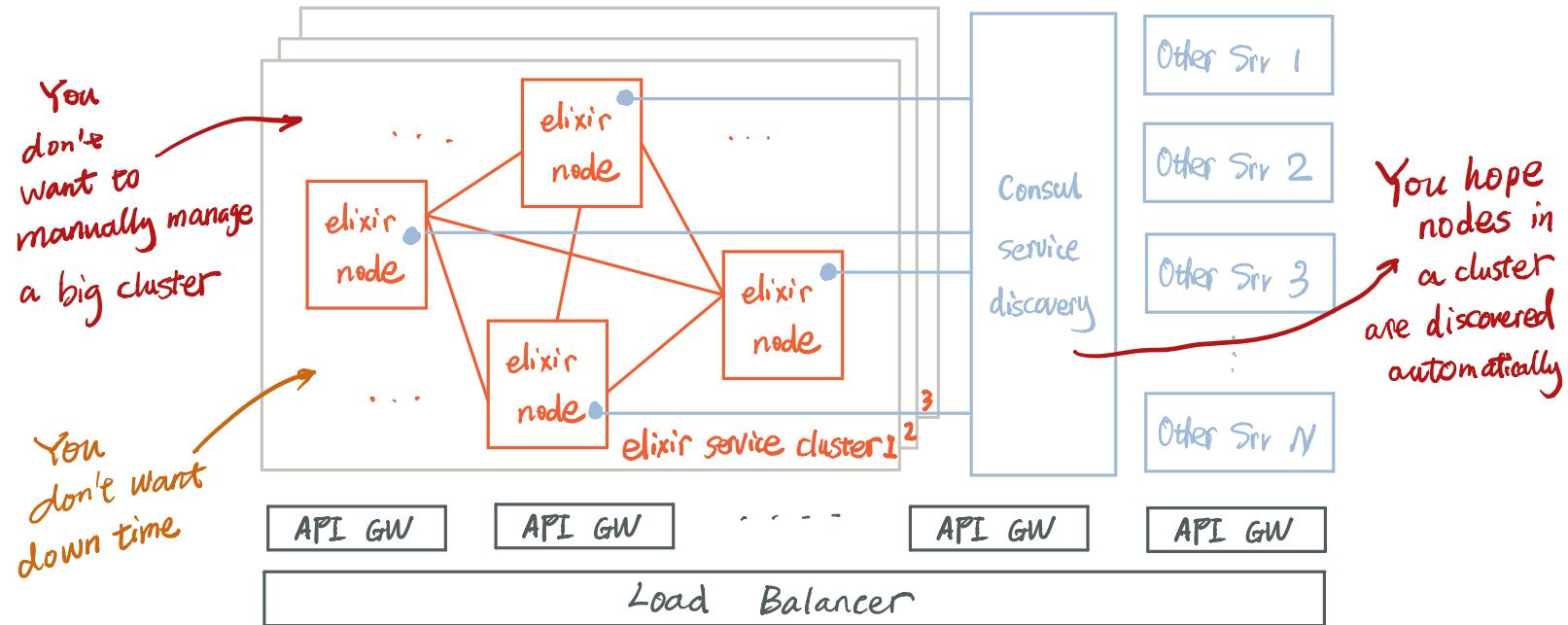
]

rel/config.exs

## vm.args (do not use the one distillery generated)

```
1 ## Name of the node
2 -name {{ ansible_hostname }}@{{ ansible_host }} decided at deployment
3
4 ## Cookie for distributed erlang
5 -setcookie {{ tubi_cms_service_cookie }} We don't use the cookie in  
the repo (rel/config.exs)
6
7 ## ports for cluster
8 -kernel inet_dist_listen_min [REDACTED] inet_dist_listen_max [REDACTED]
9
10 ## neighboring nodes
11 -config {{ tubiservice_deploy_to }}/shared/cms_service.config this is generated by  
ansible
12
13 ## Mnesia dir
14 -mnesia dir "'{{ tubiservice_deploy_to }}/shared/data/cms_service_mnesia_data/'"
15
```

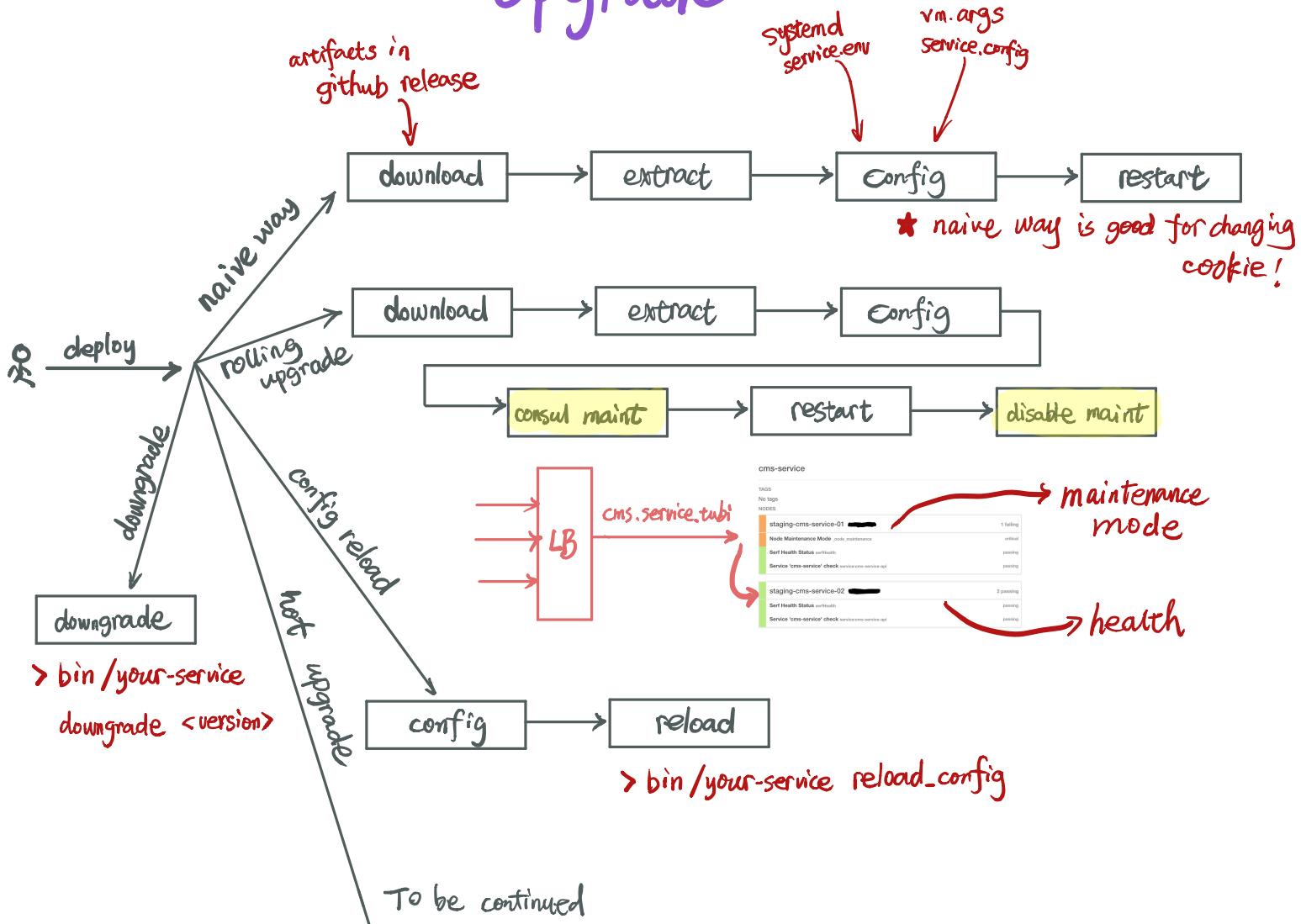
# Deploy



Tubi TV infra (simplified)

- use devOps tool (tubi loves terraform & ansible)
  - example: tubitv/overseer/tools
- use service discovery tool (tubi loves consul)
- use service manager (tubi loves systemd)
- In-Service Software Upgrade (ISSU)
  - blue / green (for stateless service)
  - rolling upgrade
    - > stateful service
  - release upgrade
  - config upgrade

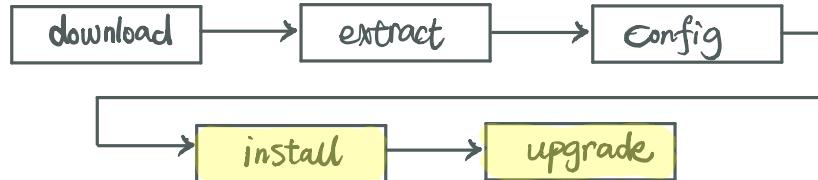
# Upgrade



Myth:

Do not use  
hot upgrade

why not?



> bin/your-service install <release\_version>

> bin/your-service upgrade <release\_version>

if you met error on hot upgrade:

iex > :release\_handler.install\_release('1.0.1', [[:update\_paths, true]])

then follow error info and solve it.

quantum

app doesn't  
have top supervisor

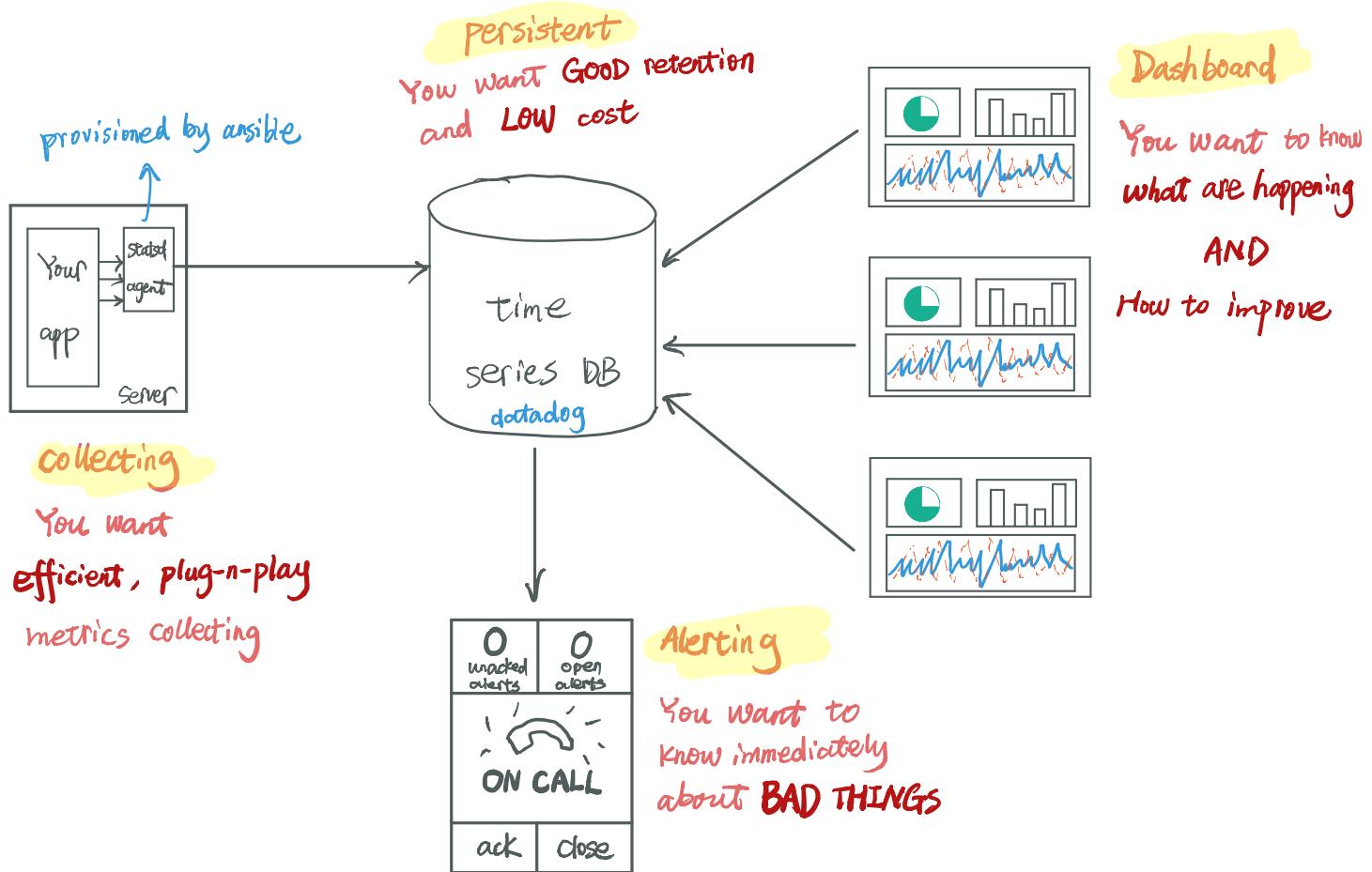
c-rack/quantum-elixir/pull/277

```
diff --git a/lib/quantum/application.ex b/lib/quantum/application.ex
@@ -10,11 +10,15 @@ defmodule Quantum.Application do
 10   if @deps do
 11     def start(_type, _args) do
 12       Application.ensure_all_started(@deps, :permanent)
 13     -{:ok, self()}
 14 
 15   else
 16     def start(_type, _args) do
 17     -{:ok, self()}
 18 
 19   end
 20 end
 21 end

@@ -10,11 +10,15 @@ defmodule Quantum.Application do
 10   if @deps do
 11     def start(_type, _args) do
 12       Application.ensure_all_started(@deps, :permanent)
 13     + children = []
 14     + opts = [strategy: :one_for_one, name: Quantum.Supervisor]
 15     + Supervisor.start_link(children, opts)
 16   end
 17   else
 18     def start(_type, _args) do
 19     + children = []
 20     + opts = [strategy: :one_for_one, name: Quantum.Supervisor]
 21     + Supervisor.start_link(children, opts)
 22   end
 23 end
 24 end
```

Note: always try hot upgrade in your staging first!

# Monitor



# What to monitor?

- 95/99%-tile response time
- QPS
- node healthiness
- error rates
- :erlang.statistics()
- Your own business logic

ExStatsd



tubitv/ex-datadog-plug

plug Plug.RequestId

plug ExDatadog.Plug, prefix: "your-service.public.."

→ erlang statistics metric

```
defp update do(:context_switches, {value, _}) do
  ExStatsD.gauge(value, "statistics.context_switches", tags: @datadog_tag)
end

defp update do(:garbage_collection, {num_of_gcs, words_reclaimed, _}) do
  ExStatsD.gauge(num_of_gcs, "statistics.num_of_gcs", tags: @datadog_tag)
  ExStatsD.gauge(words_reclaimed, "statistics.words_reclaimed", tags: @datadog_tag)

  ExStatsD.gauge(
    words_reclaimed * :erlang.system_info(:wordsizes),
    "statistics.bytes_reclaimed",
    tags: @datadog_tag
  )
end
```

use distillery  
build your own tools  
provision "vm.args" with ansible

rolling upgrade  
hot upgrade  
config upgrade  
downgrade

## Release Deploy Upgrade Monitor

use devOps tools  
consul  
ansible  
terraform  
systemd

use 3rd party tools  
rexStatsd  
flex-datadog-plug  
build dashboard  
alerting & on-call



<https://github.com/tubityv>



*My wechat open accounts  
(in chinese)*

<https://github.com/tyrchen>

