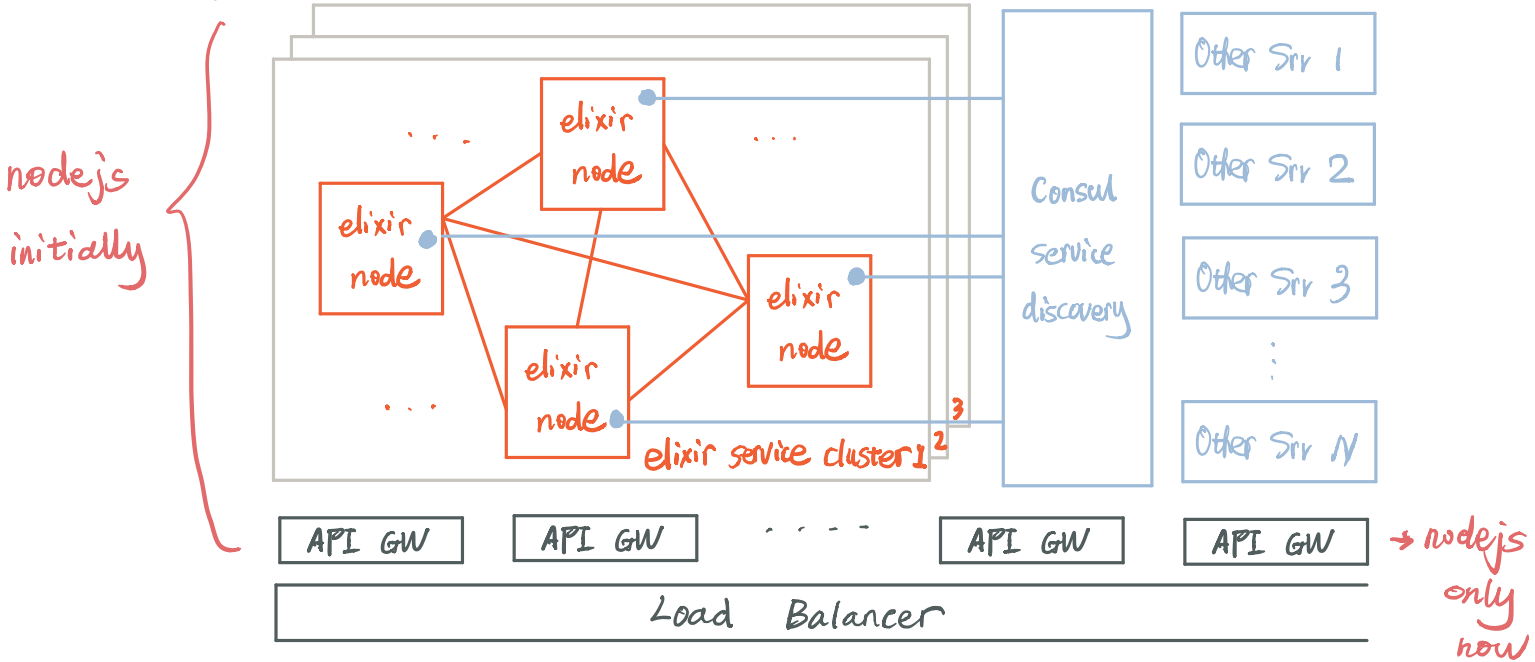
Release Deploy Upgrade Monitor

Elixir Services

in real world

Tyr Chen

Background



Tubi TV infra (simplified)

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
  - reduced latency from 200ms to 50us

    yes ↗ us

    latency → 95%-tile

    200ms → 50us: 1,000 contents, 5,000 - 10,000 rules

The history (and problems)

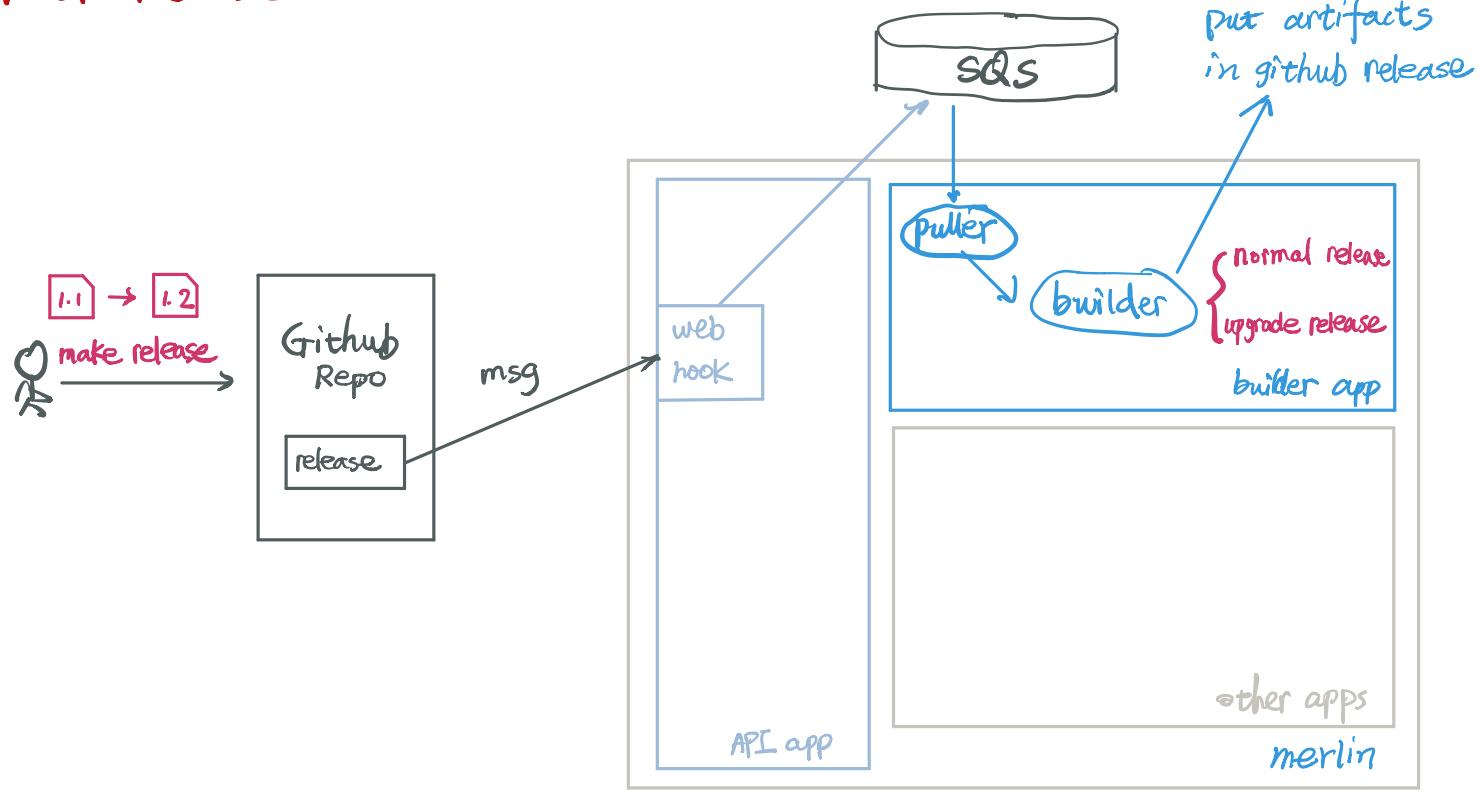
- We built more services
  - stateless (use external data source)
  - stateful (use ets/mnesia)
- We want to be able to release any time
  - simple, automated process
  - without down time
  - engs get notified on bad things

Release with distillery

Challenges:

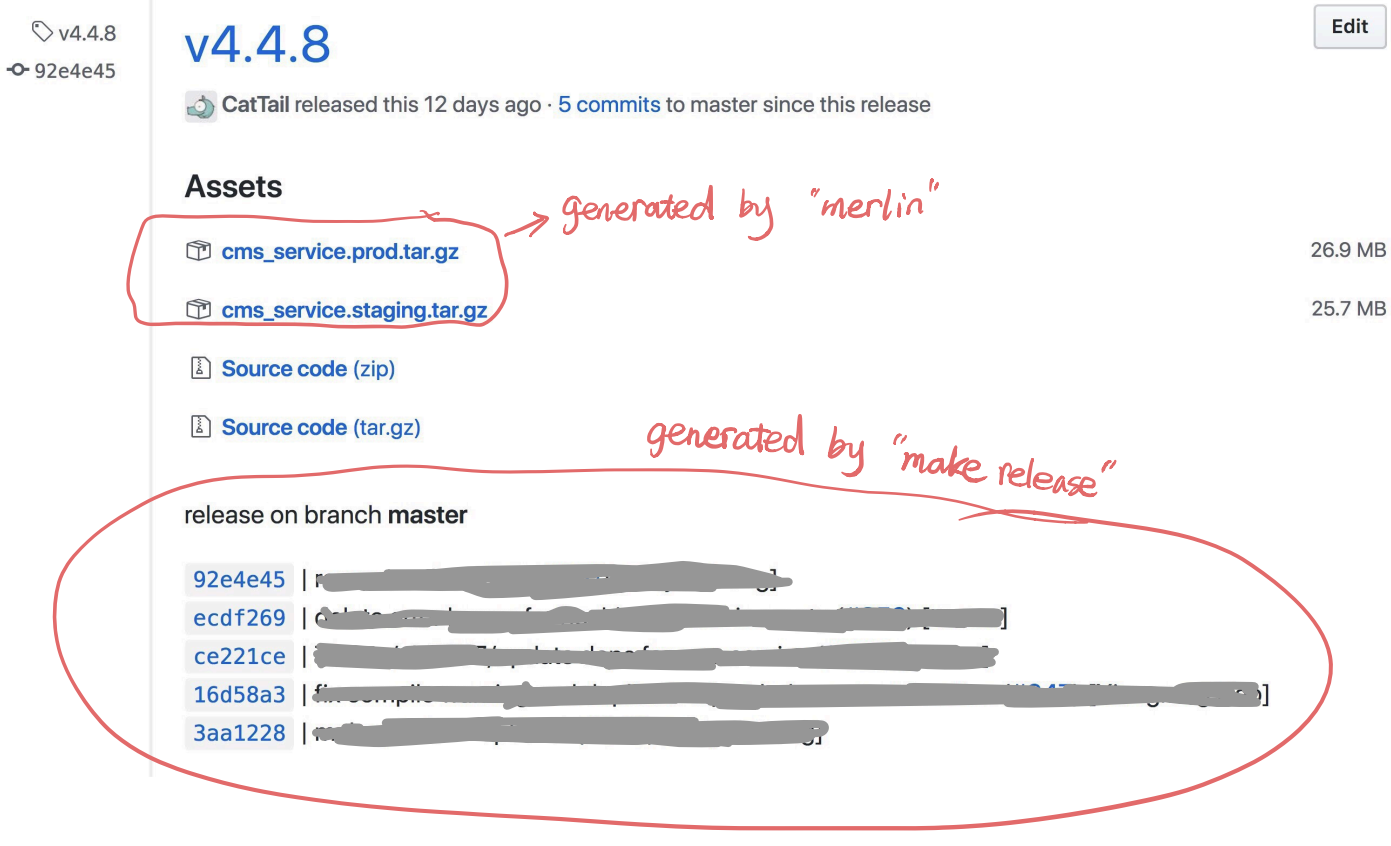
- database access at compile time
- normal release v.s. upgrade release
- vm.args (cookie!)
- source files to copy over

Tubi Release tools - merlin

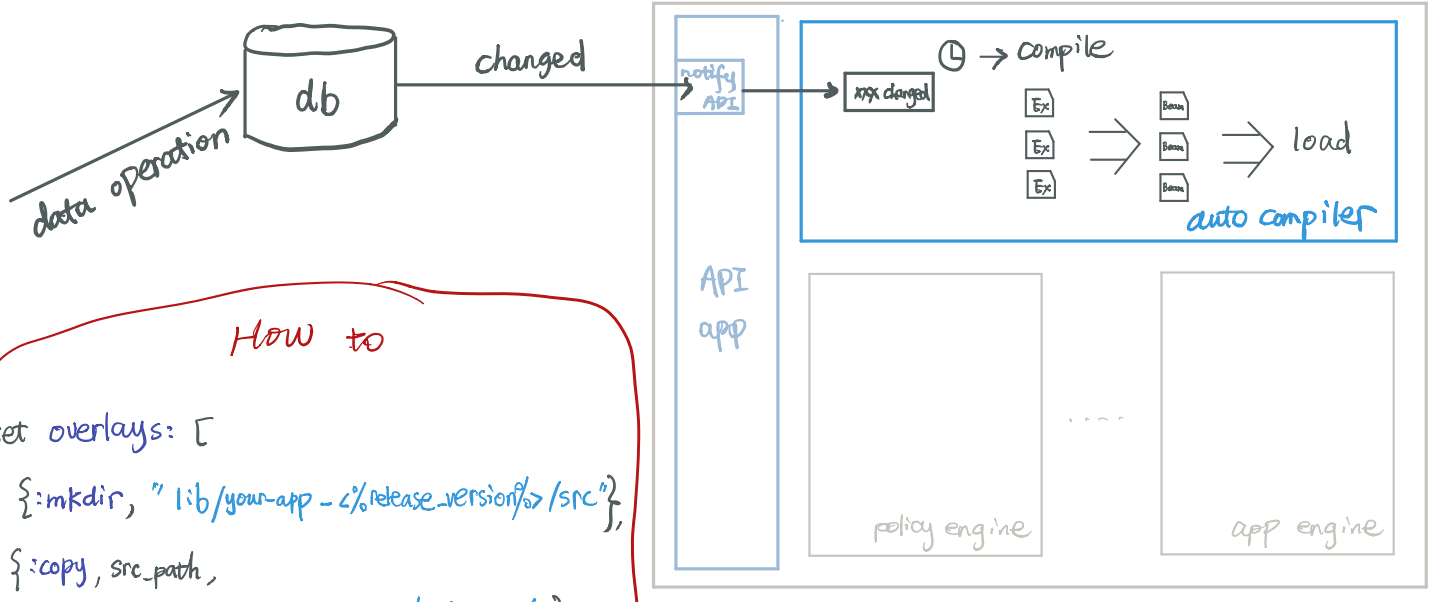


{ mix release
~~mix~~ release --upgrade

(include_erts: true)


v4.4.8
92e4e45
v4.4.8
Edit
CatTail released this 12 days ago · 5 commits to master since this release
Assets
generated by "merlin"
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 "make release"
release on branch master
92e4e45
ecdf269
ce221ce
16d58a3
3aa1228

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)

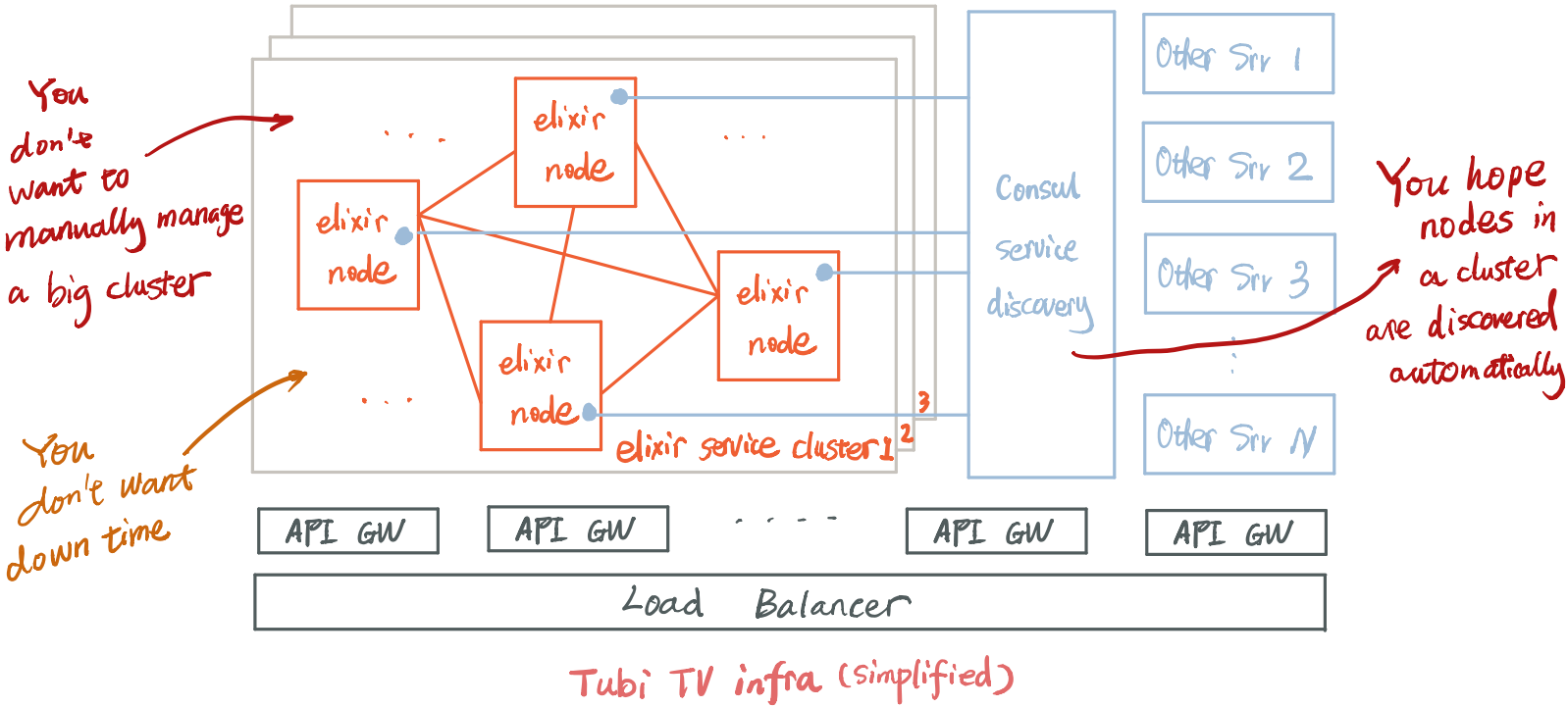
```
1  ## Name of the node
2  -name {{ ansible_hostname }}@{{ ansible_host }}
3
4  ## Cookie for distributed erlang
5  -setcookie {{ tubi_cms_service_cookie }}
6
7  ## ports for cluster
8  -kernel inet_dist_listen_min [redacted] inet_dist_listen_max [redacted]
9
10 ## neighboring nodes
11 -config {{ tubiservice_deploy_to }}/shared/cms_service.config
12
13 ## Mnesia dir
14 -mnesia dir '"{{ tubiservice_deploy_to }}/shared/data/cms_service_mnesia_data/"'
15
```

decided at deployment

We don't use the cookie in the repo (rel/config.exs)

this is generated by ansible

Deploy

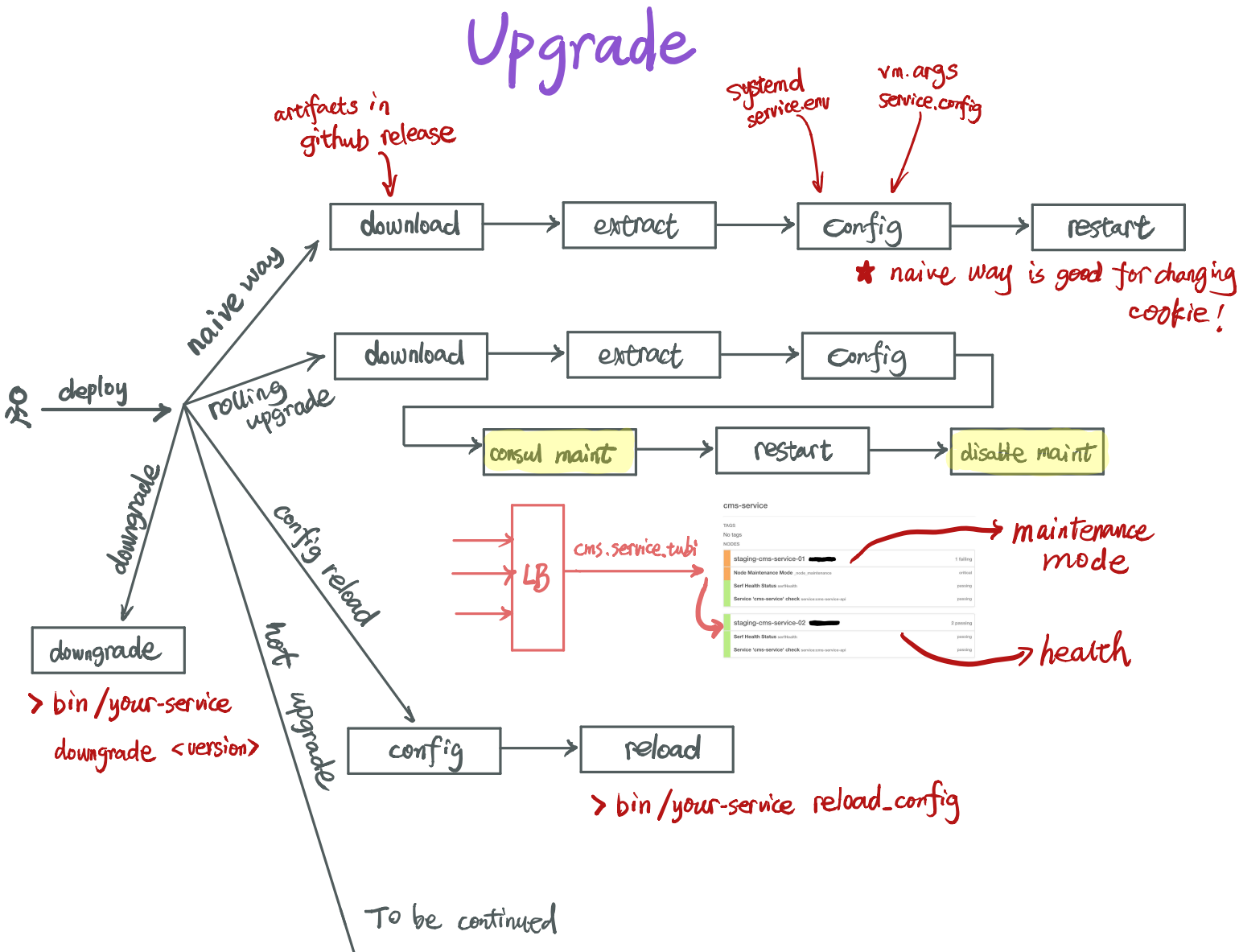


Tubi TV infra (simplified)

- 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
  - release upgrade
  - config upgrade

> stateful service

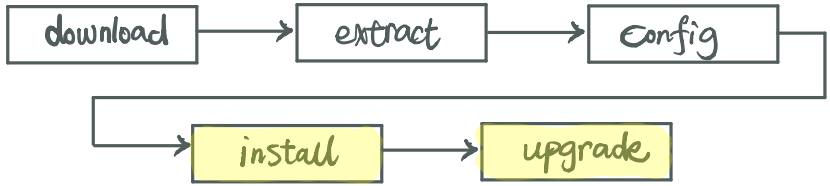
Upgrade


artifacts in github release
systemd service.env
vm.args service.config
download
extract
config
restart
naive way
★ naive way is good for changing cookie!
deploy
rolling upgrade
consul maint
disable maint
downgrade
config reload
hot upgrade
LB
cms.service.tubi
cms-service
staging-cms-service-01
staging-cms-service-02
maintenance mode
health
downgrade
> bin/your-service downgrade <version>
config
reload
> bin/your-service reload_config
To be continued

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

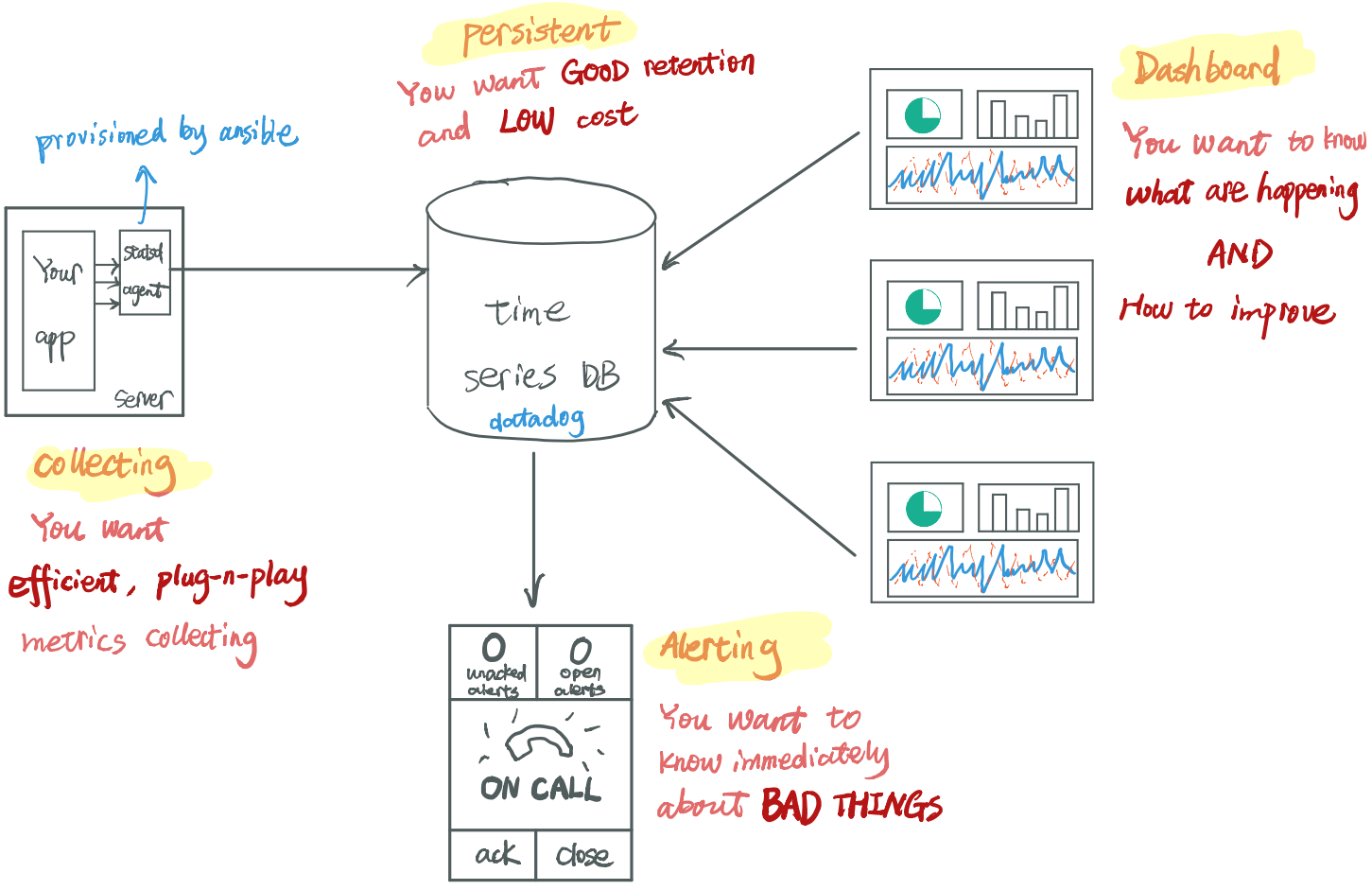
View

lib/quantum/application.ex

```
@@ -10,11 +10,15 @@ defmodule Quantum.Application do
   if @deps do
     def start(_type, _args) do
       Application.ensure_all_started(@deps, :permanent)
-      {:ok, self()}
+      children = []
+      opts = [strategy: :one_for_one, name: Quantum.Supervisor]
+      Supervisor.start_link(children, opts)
     end
   else
     def start(_type, _args) do
-      {:ok, self()}
+      children = []
+      opts = [strategy: :one_for_one, name: Quantum.Supervisor]
+      Supervisor.start_link(children, opts)
     end
   end
 end
```

Note: always try hot upgrade in your staging first!

Monitor


provisioned by ansible
Your app
statsd agent
server
collecting
You want
efficient, plug-n-play
metrics collecting
persistent
You want GOOD retention
and LOW cost
time series DB
datadog
Dashboard
You want to know
what are happening
AND
How to improve
0 unacked alerts
0 open alerts
ON CALL
ack
close
Alerting
You want to
know immediately
about BAD THINGS

What to monitor?

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

ExStatsd

tubitv/ex-datadog-plug

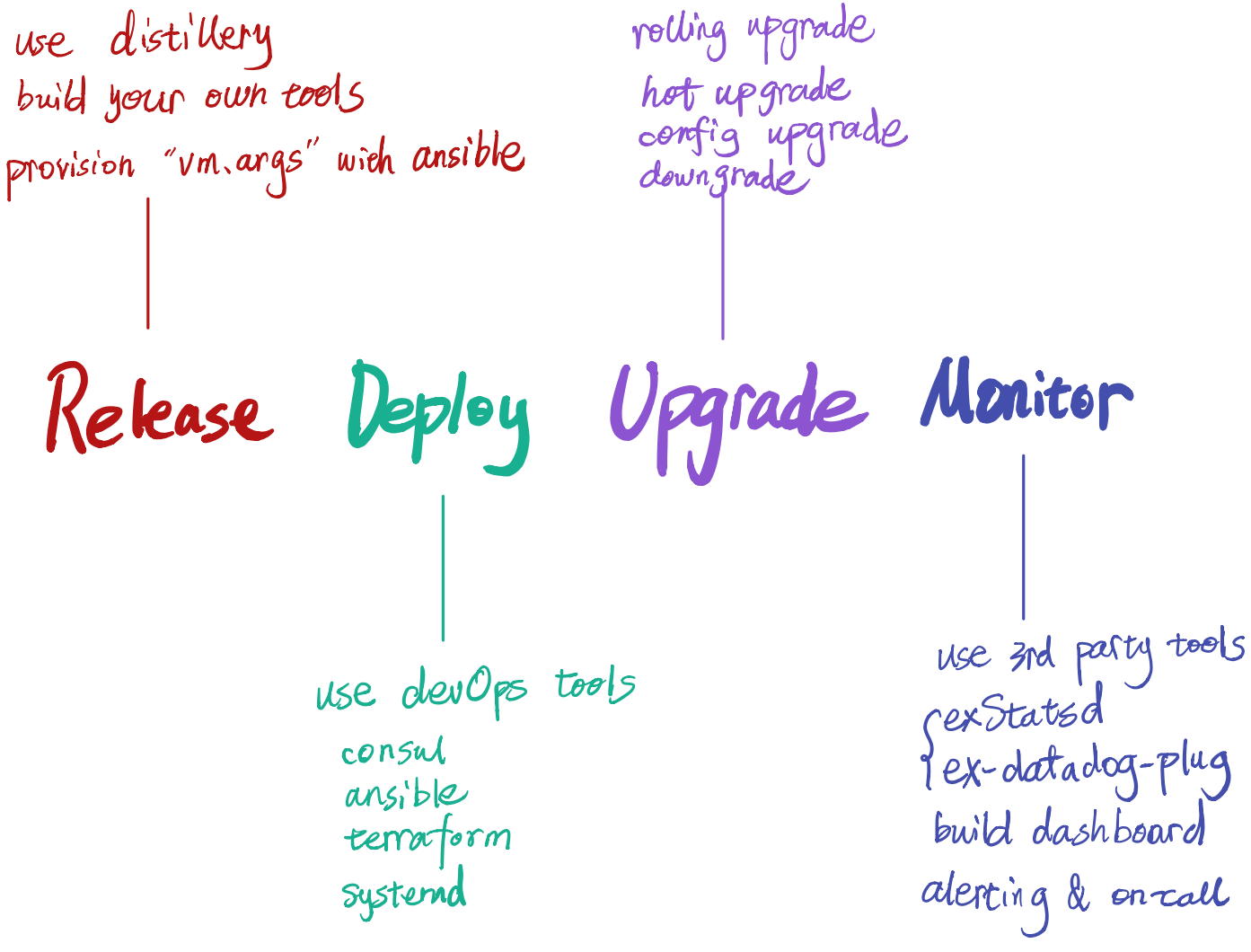
```
plug Plug.RequestId
plug ExDatadog.Plug, prefix: "your-service.public..."
```

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(:wordsize),
    "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
exStatsd
ex-datadog-plug
build dashboard
alerting & on-call

https://github.com/tubitv

My wechat open accounts (in chinese)

https://github.com/tyrchen