My simple reverse proxy in Elixir

@tony612

What to cover

- What does my simple reverse proxy looks like?
- How to implement it in Elixir
- Mox - Mocks and explicit contracts in Elixir
- How to update map(especially JSON) in Elixir
- Diff of map

What not to cover

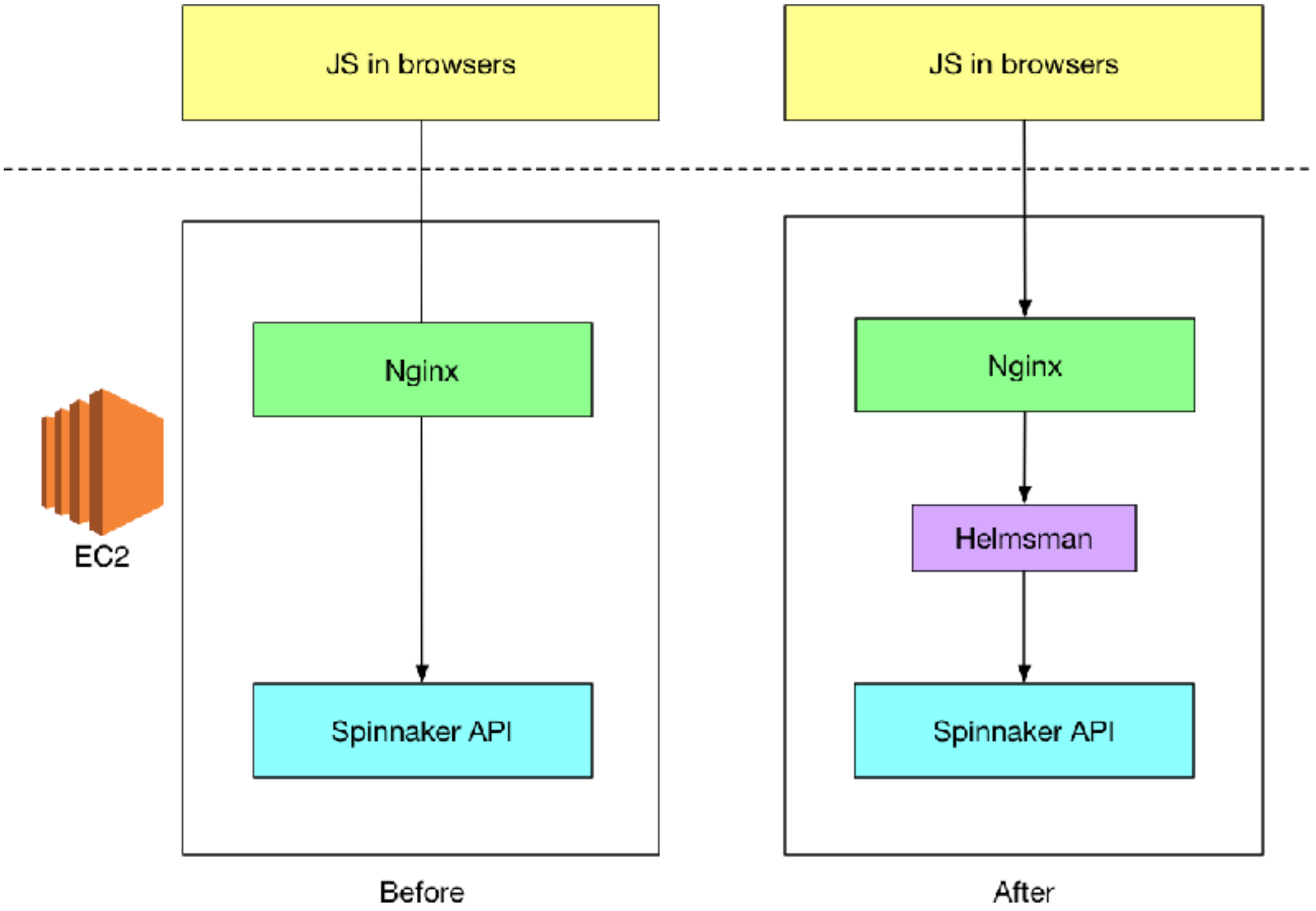
- How to implement a high performance reverse proxy like Nginx
- How to implement features like SSL, upstream health check, load balancing...

What is it for?

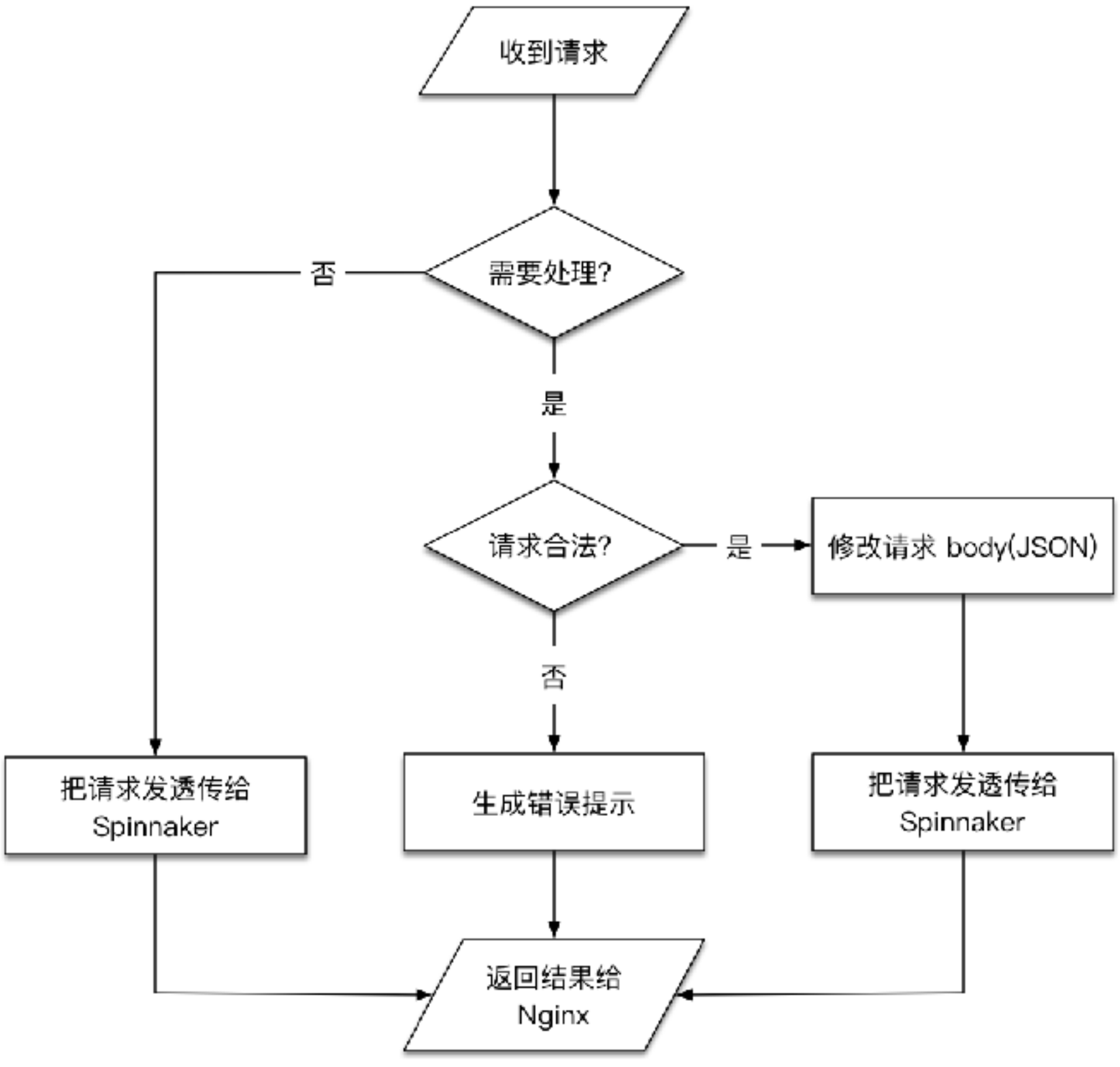
A hack for Spinnaker

- Liulishuo uses Spinnaker for Continuous Delivery(to k8s)
- We need to customize Spinnaker based on our needs, like we need k8s pod resources is required.
- We don’t want to change Spinnaker directly

Structure


JS in browsers
Nginx
Spinnaker API
EC2
Before
JS in browsers
Nginx
Helmsman
Spinnaker API
After

Logic


收到请求
需要处理?
否
是
请求合法?
是
修改请求 body(JSON)
否
把请求发透传给 Spinnaker
生成错误提示
把请求发透传给 Spinnaker
返回结果给 Nginx

Why Elixir?

- I like it! 😄
- Dynamic language(JSON handling)
- Performance
- Hot reloading
- Simple

Reverse proxy

- Based on https://github.com/slogsdon/elixir-reverse-proxy
  - delete Transfer-Encoding(hop-by-hop header)
- Fix x-forwarded-for
- Delete cache

Tests

- 96 tests, 0 failures
- [TOTAL] 72.7%

Mox

Mox

Mox is a library for defining concurrent mocks in Elixir.

The library follows the principles outlined in "Mocks and explicit contracts", summarized below:

1. No ad-hoc mocks. You can only create mocks based on behaviours
2. No dynamic generation of modules during tests. Mocks are preferably defined in your `test_helper.exs` or in a `setup_all` block and not per test
3. Concurrency support. Tests using the same mock can still use `async: true`
4. Rely on pattern matching and function clauses for asserting on the input instead of complex expectation rules

https://github.com/plataformatec/mox

https://hexdocs.pm/mox/Mox.html

- Live coding
- How is Mox implemented?

Mox summary

High

- No ad-hoc mocks. No magic, more clear.
- Pattern match. Return a mock based on the input pattern in every test case. Easier to write mocks.

```ruby
# ruby
expect(Client).to receive(:latest).with('lls', 123).and_return({})
```

Low

- Function calls in nested processes needs global mode, which is conflict with :async

Updating maps in Elixir

- update nested map
- some maps don’t need to be updated
- need to store some intermediate state
- Live coding

Diff of two items

```
iex(22)> m1
%{a: %{b: 1}}
iex(23)> m2
%{a: %{b: 2}}
iex(24)> {left, right} = Helmsman.Diff.diff(m1, m2, %{diff_type: :cli, enabled: true})
{"%{a: %{b: \e[31m1\e[0m}}", "%{a: %{b: \e[32m2\e[0m}}"}
iex(25)> IO.puts left
%{a: %{b: 1}}
:ok
iex(26)> IO.puts right
%{a: %{b: 2}}
:ok
```

Diff

- ExUnit.Diff has no public API
- But we can extract the code from

https://github.com/elixir-lang/elixir/blob/14156f2003b55b41040cc2809e71c0ff54334839/lib/ex_unit/lib/ex_unit/formatter.ex#L309

mine is at

https://gist.github.com/tony612/686e5a050f5254e0df2ab8ed109bbe1e