Service Pipeline Framework

How We Do Plumbing At eBay

By Dmytro Semenov, Platform Team

About myself

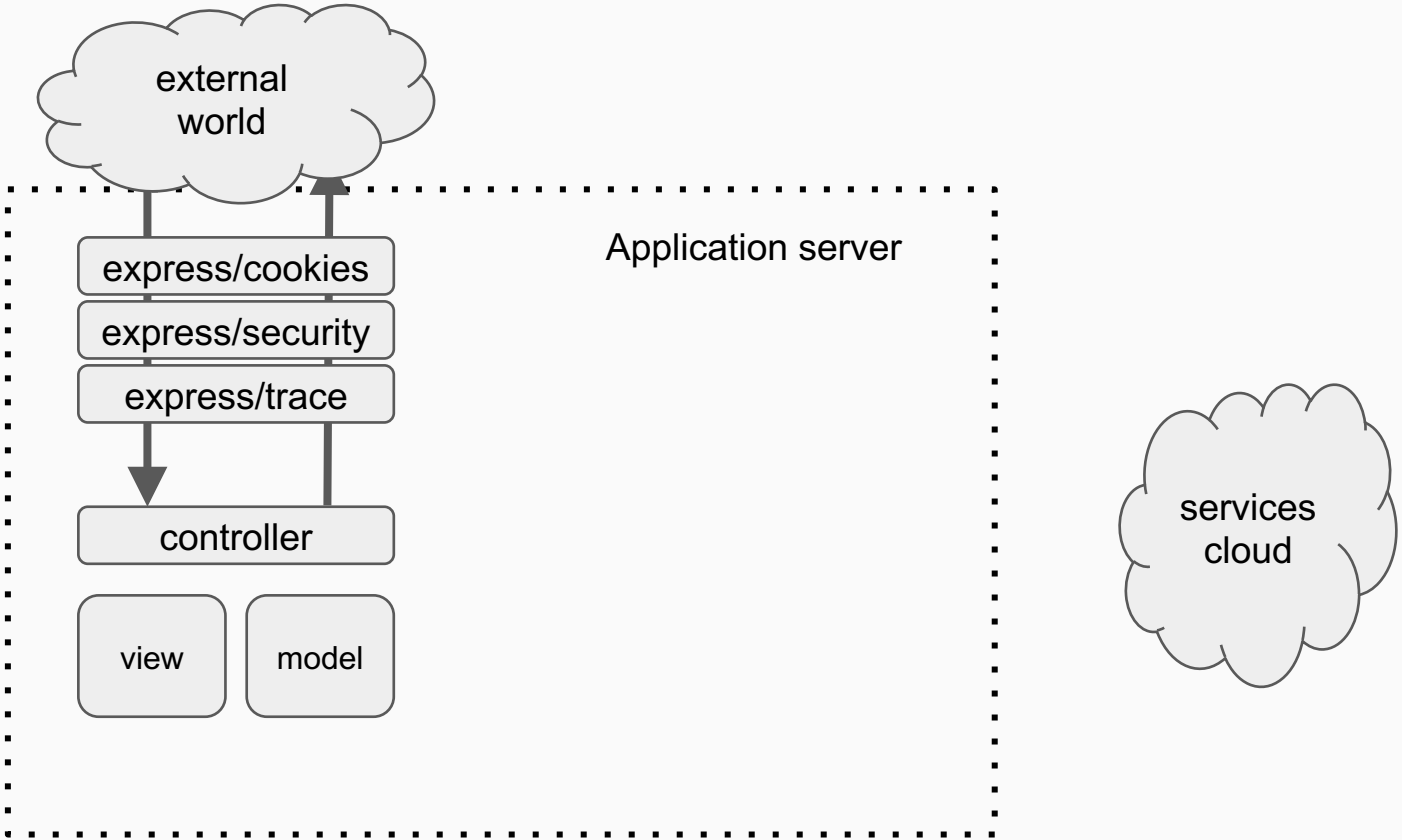
- MTS 2 Engineer at eBay, Inc, Platform team
- Node.JS stack lead developer
- Linkedin: https://www.linkedin.com/in/dmytro-semenov-2b511b2

scale

- > billion of calls a day
- 200+ node applications
- 300 node developers

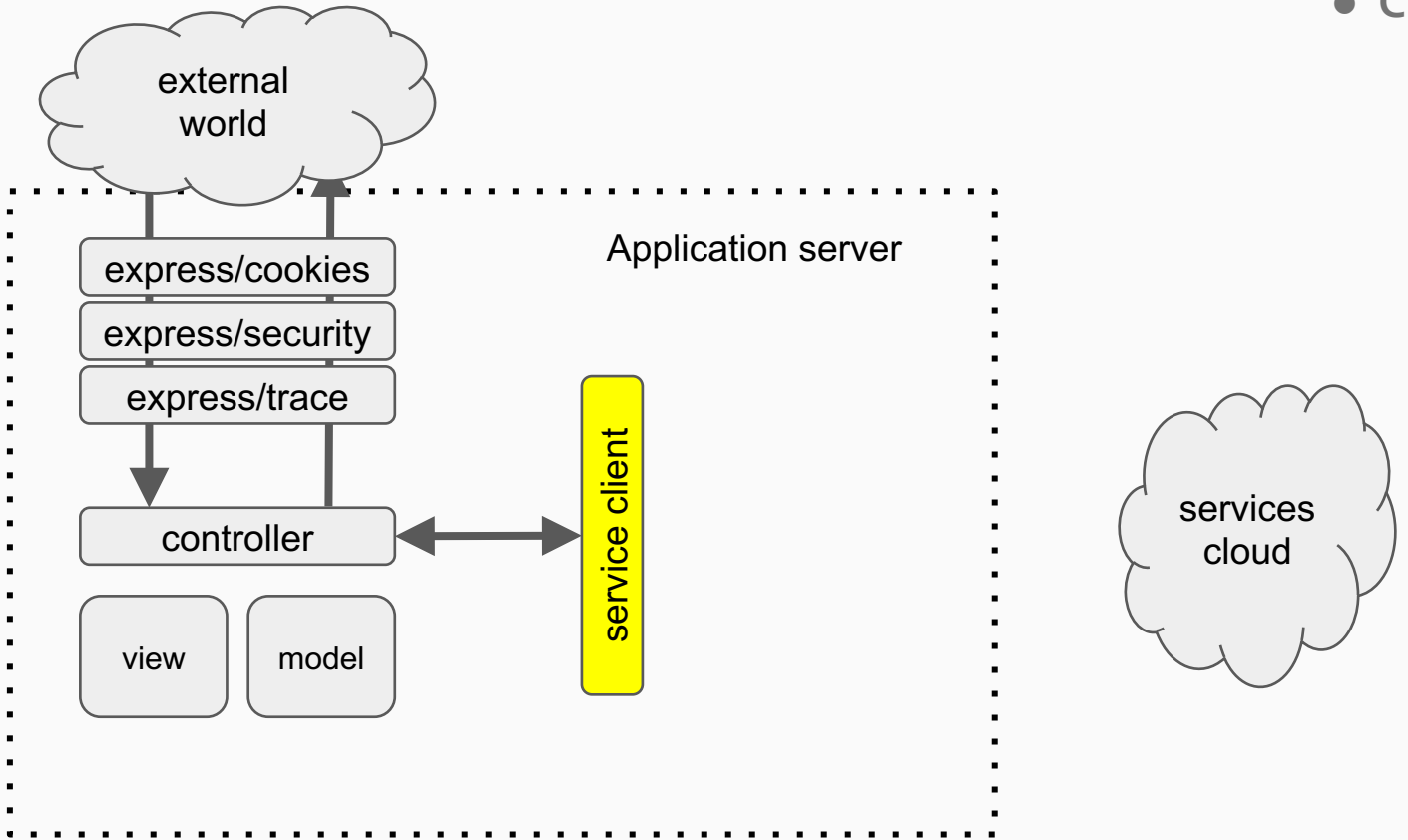
How Can We Scale Service Invocations?

What is takes to make a service call?



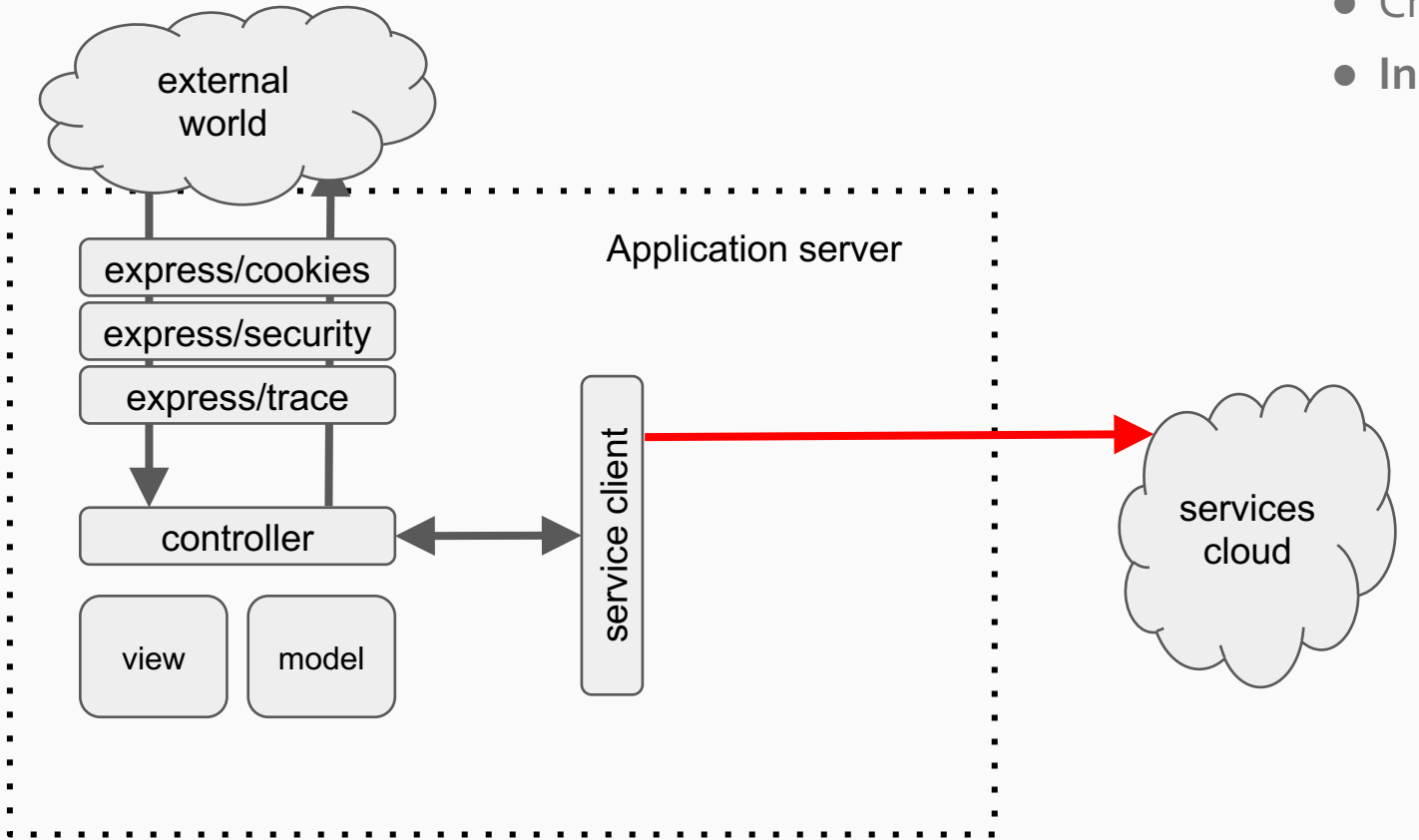
What is takes to make a service call?

- Create an http client

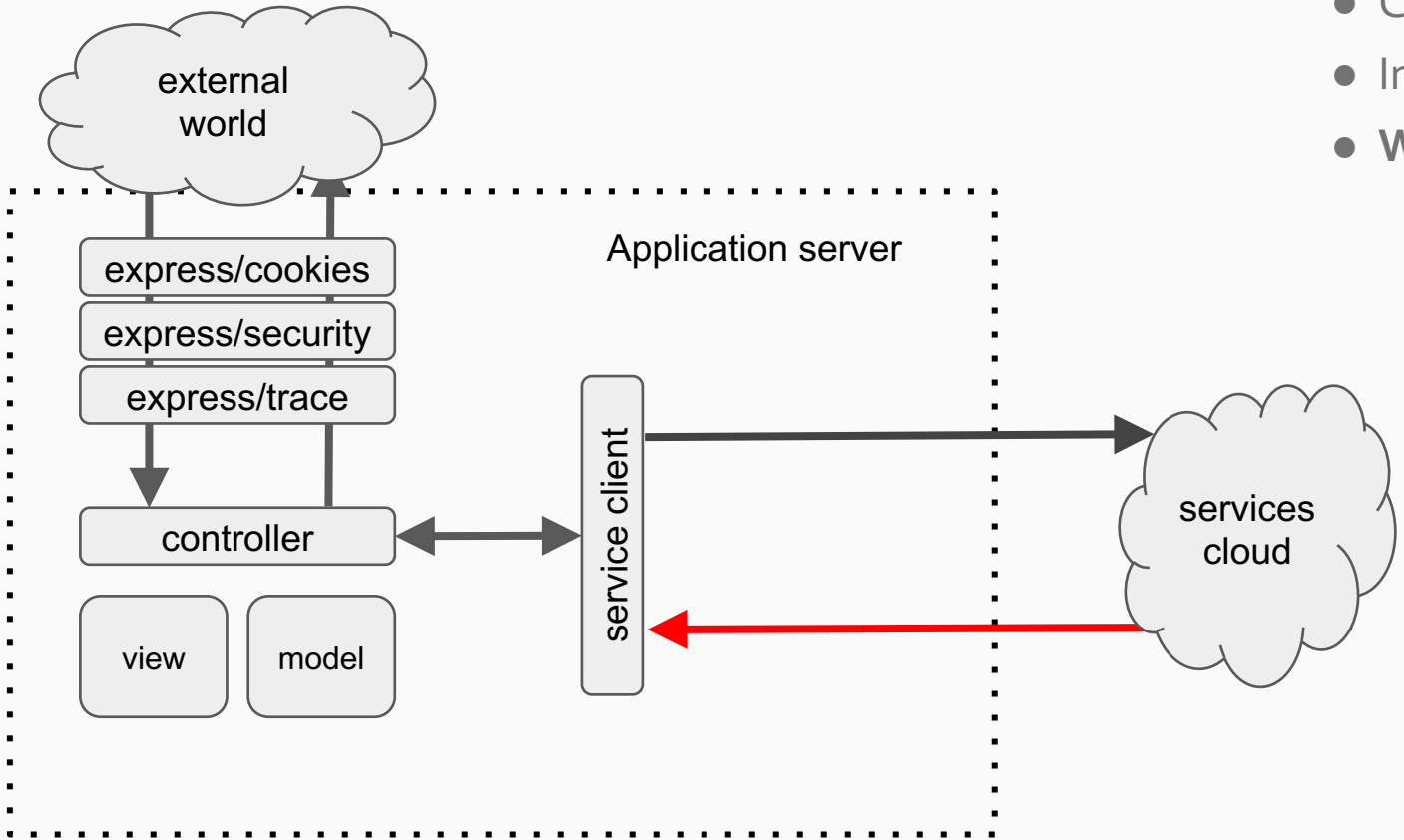


What is takes to make a service call?

- Create an http client
- **Initiate a call**

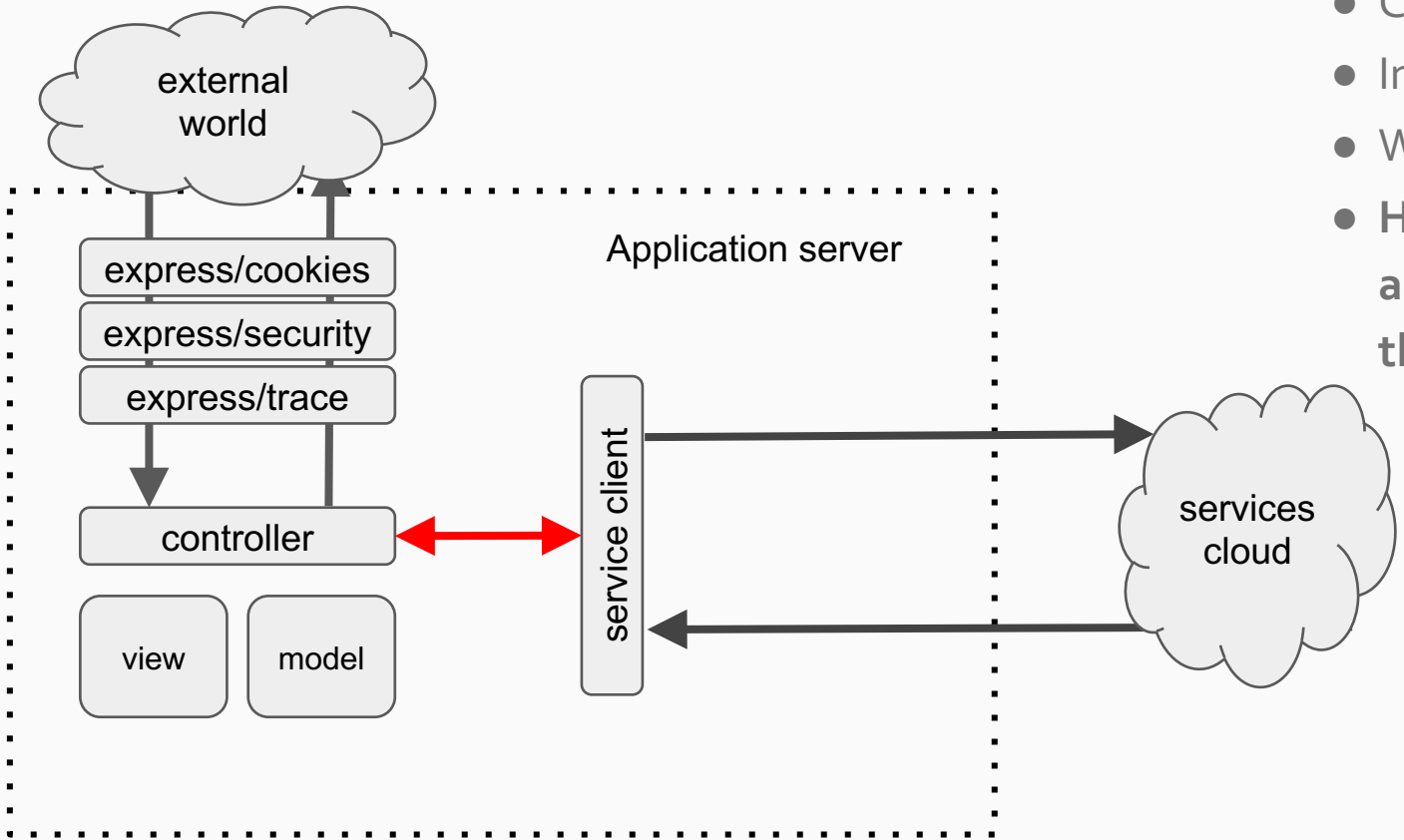


What is takes to make a service call?



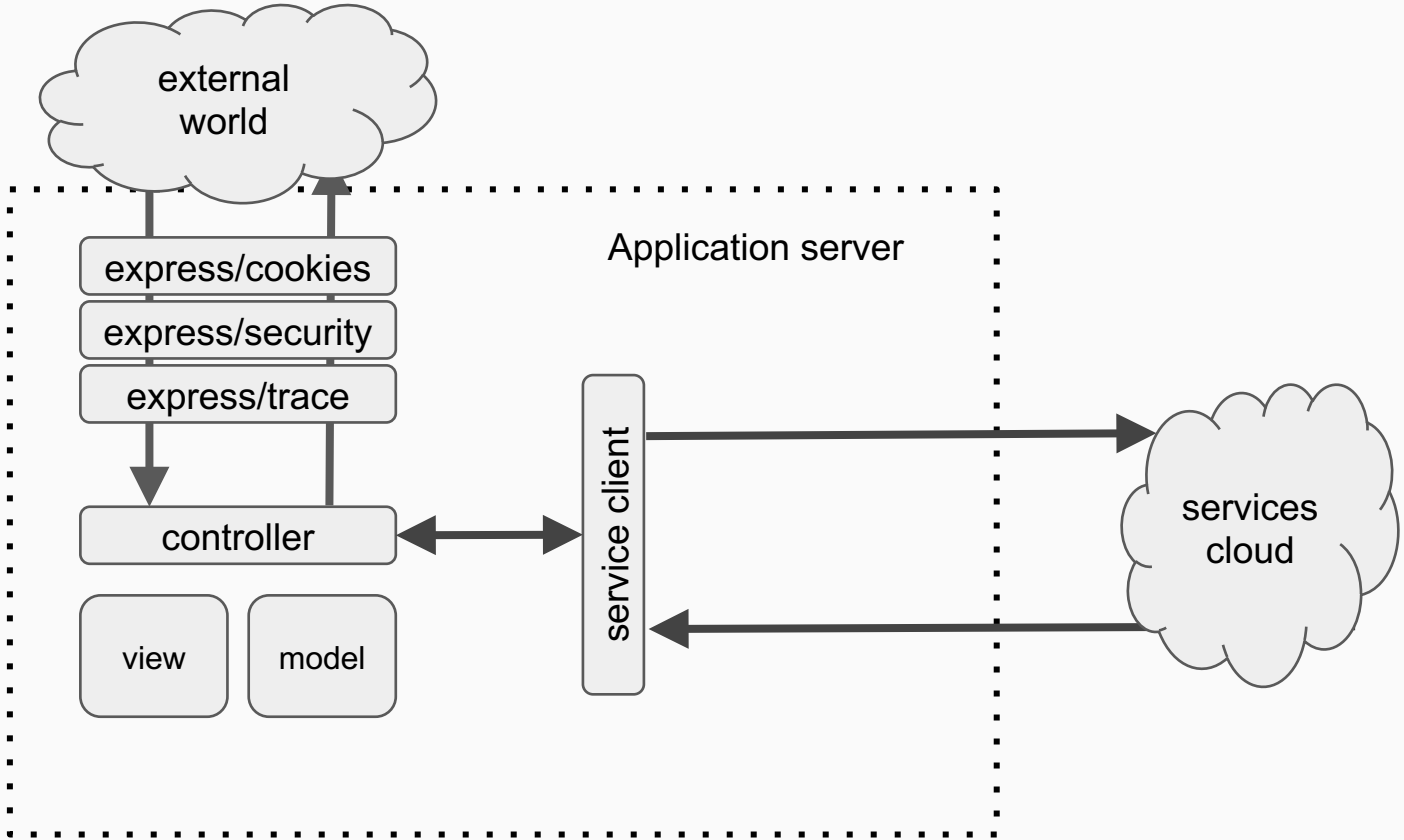
- Create an http client
- Initiate a call
- **Wait for response**

What is takes to make a service call?

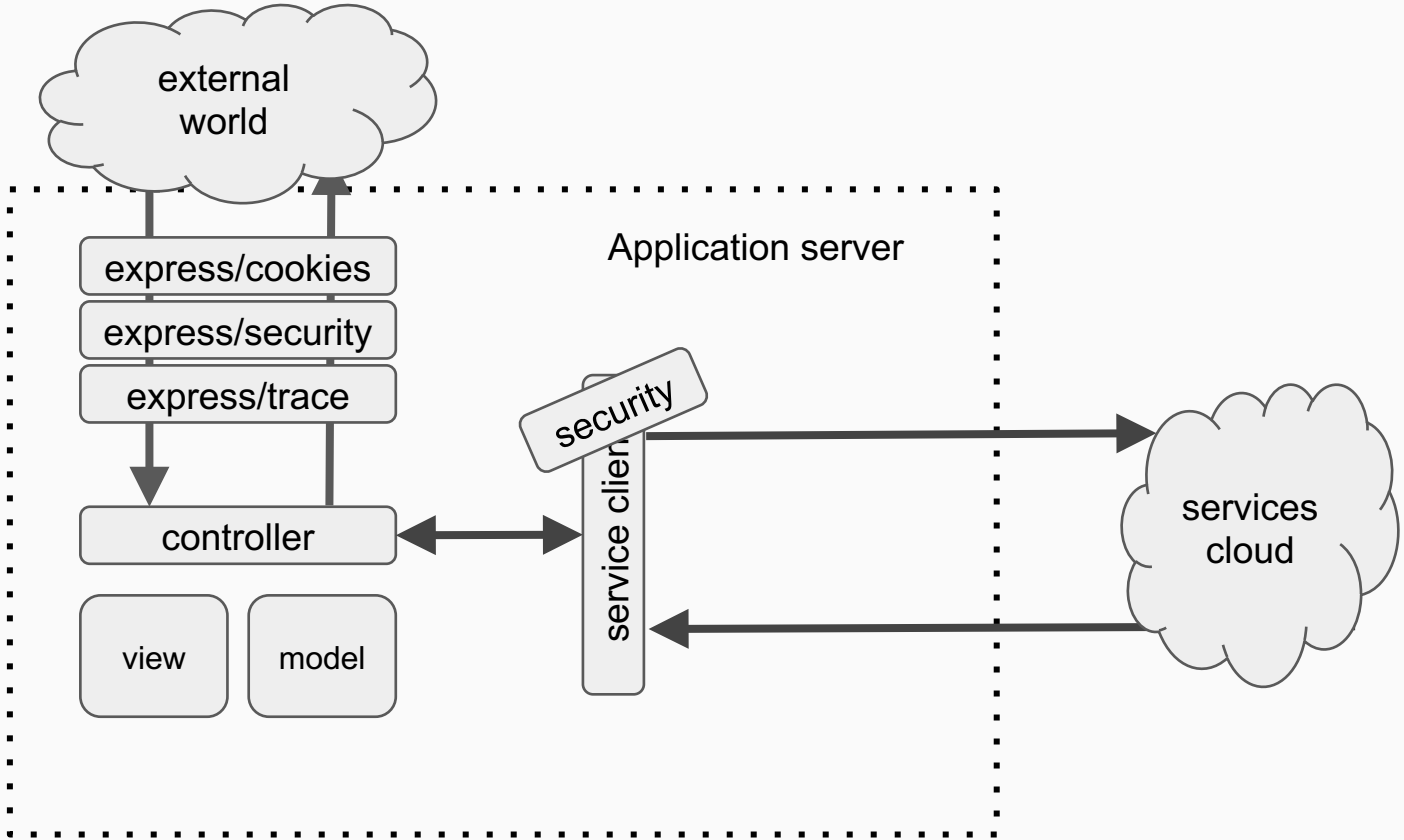


- Create an http client
- Initiate a call
- Wait for response
- **Handle edge cases and return data to the caller**

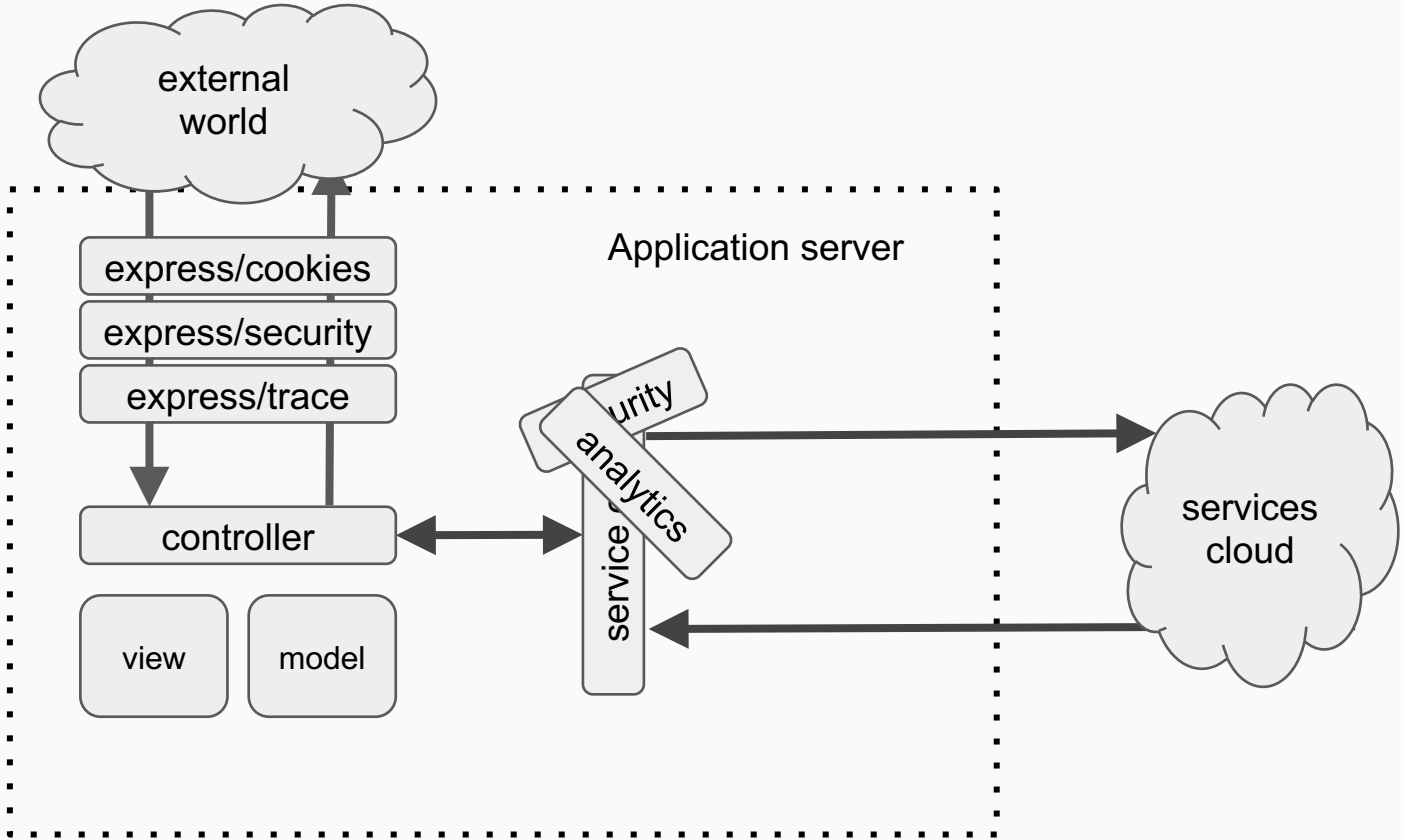
Will it scale? What if you need to ...



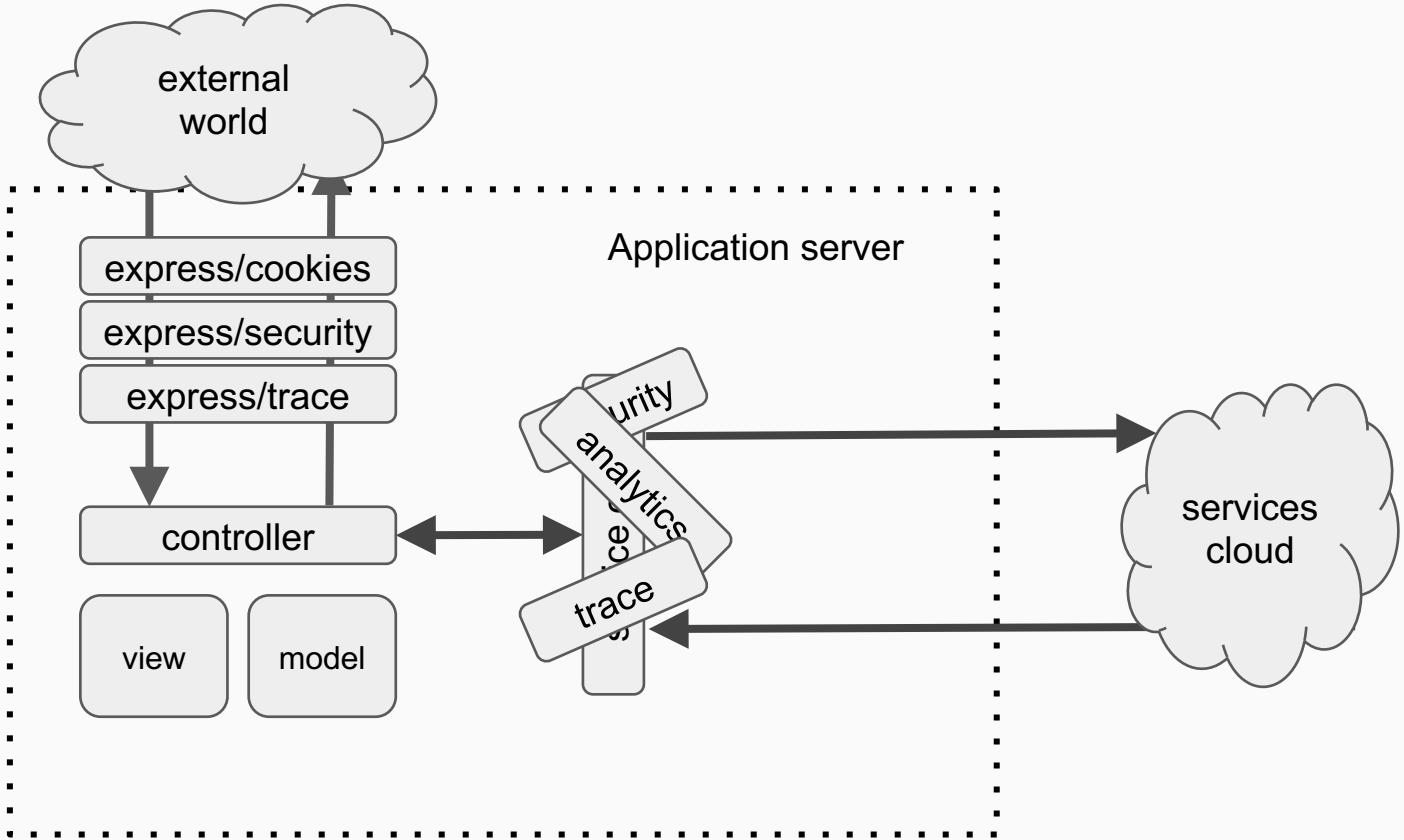
What is takes to make a service call?



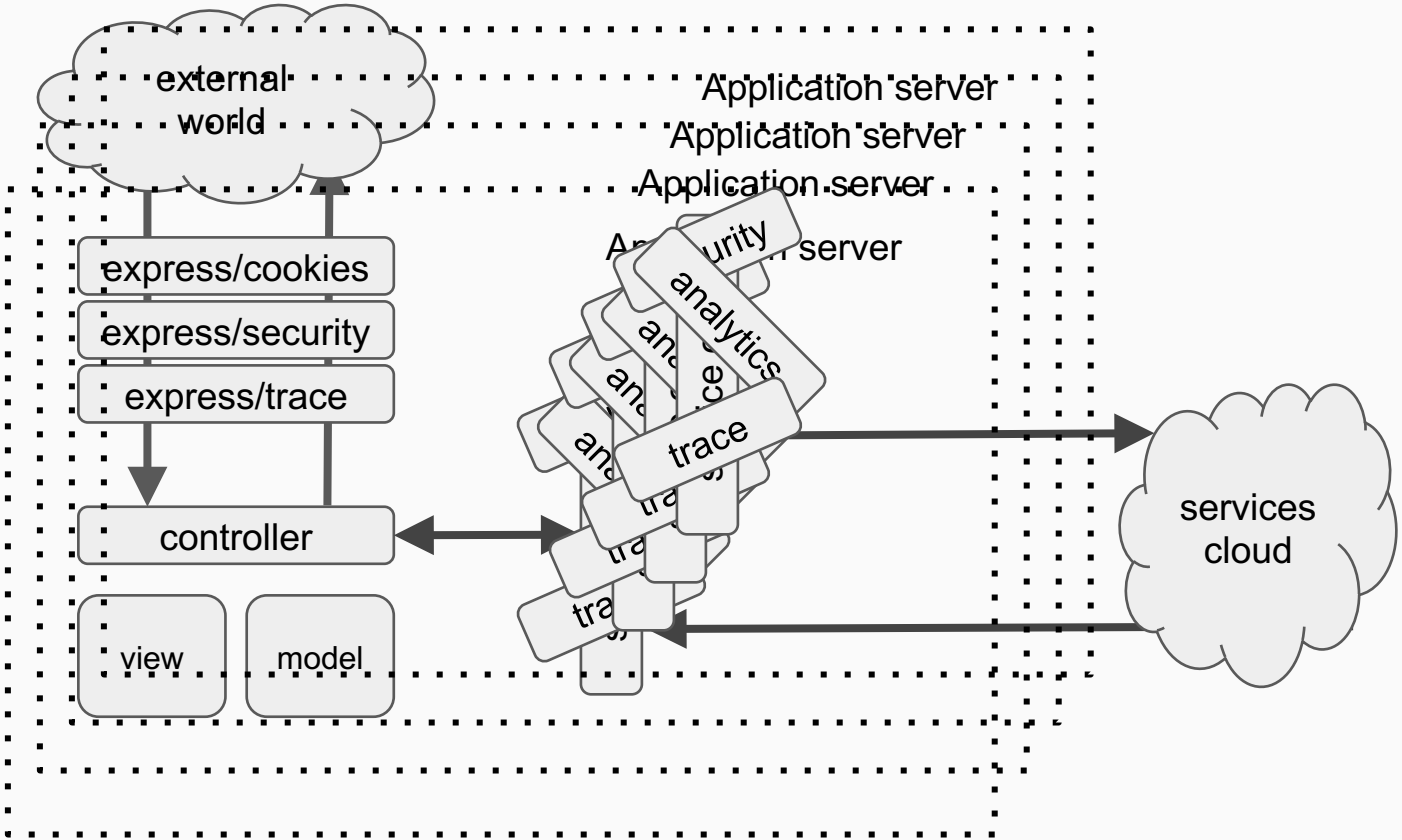
What is takes to make a service call?



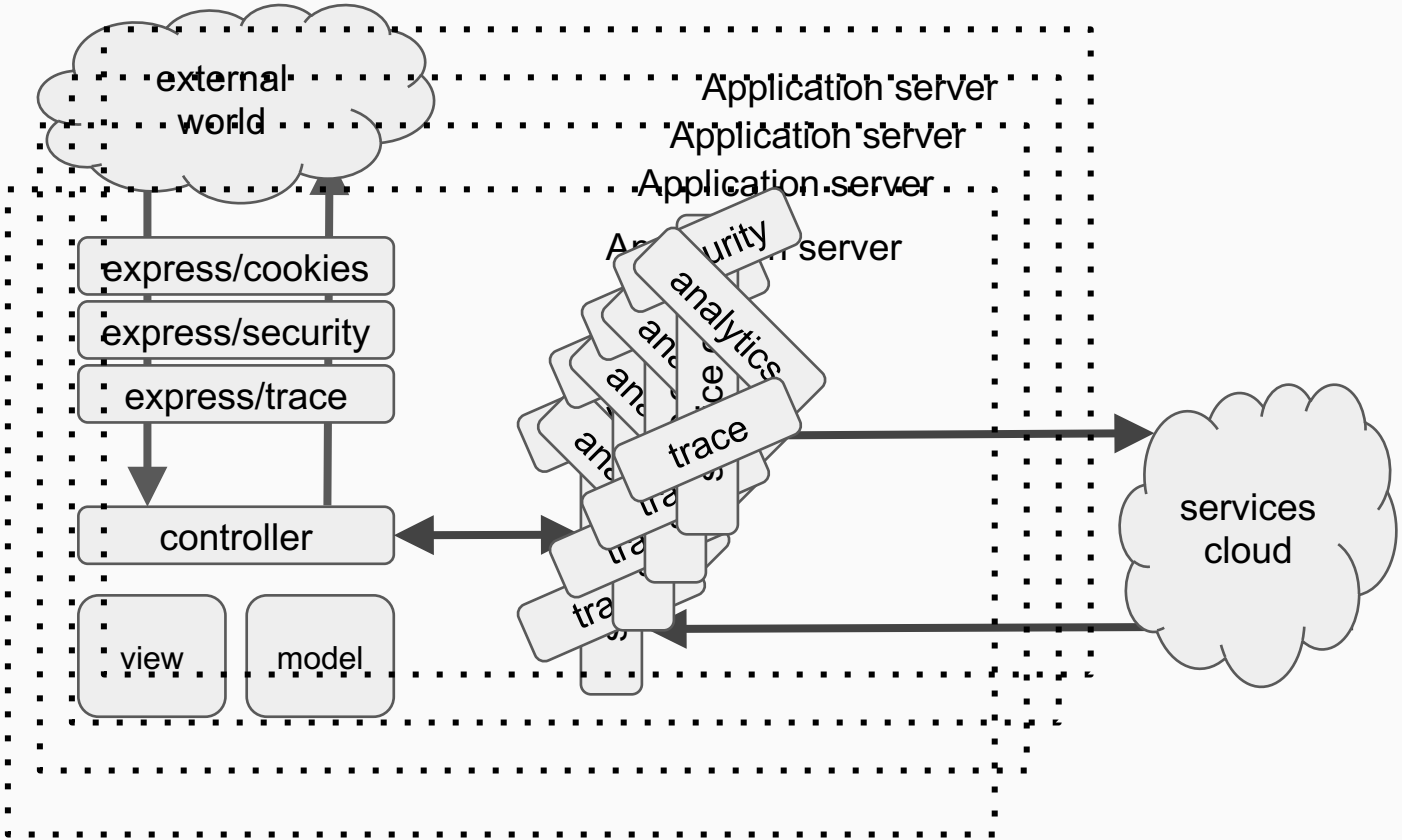
What is takes to make a service call?



How can we upgrade 100+ app?

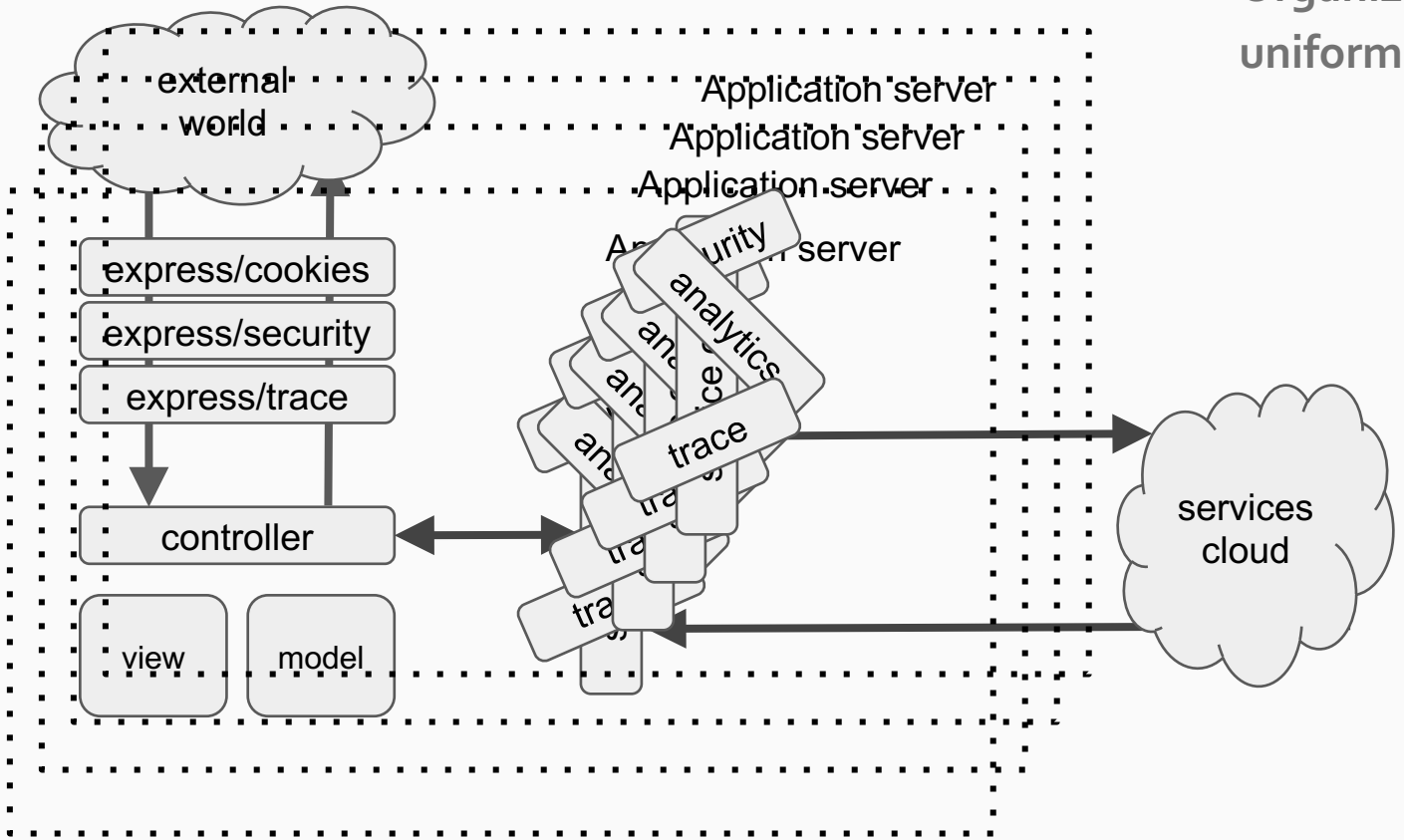


Framework to the rescue!



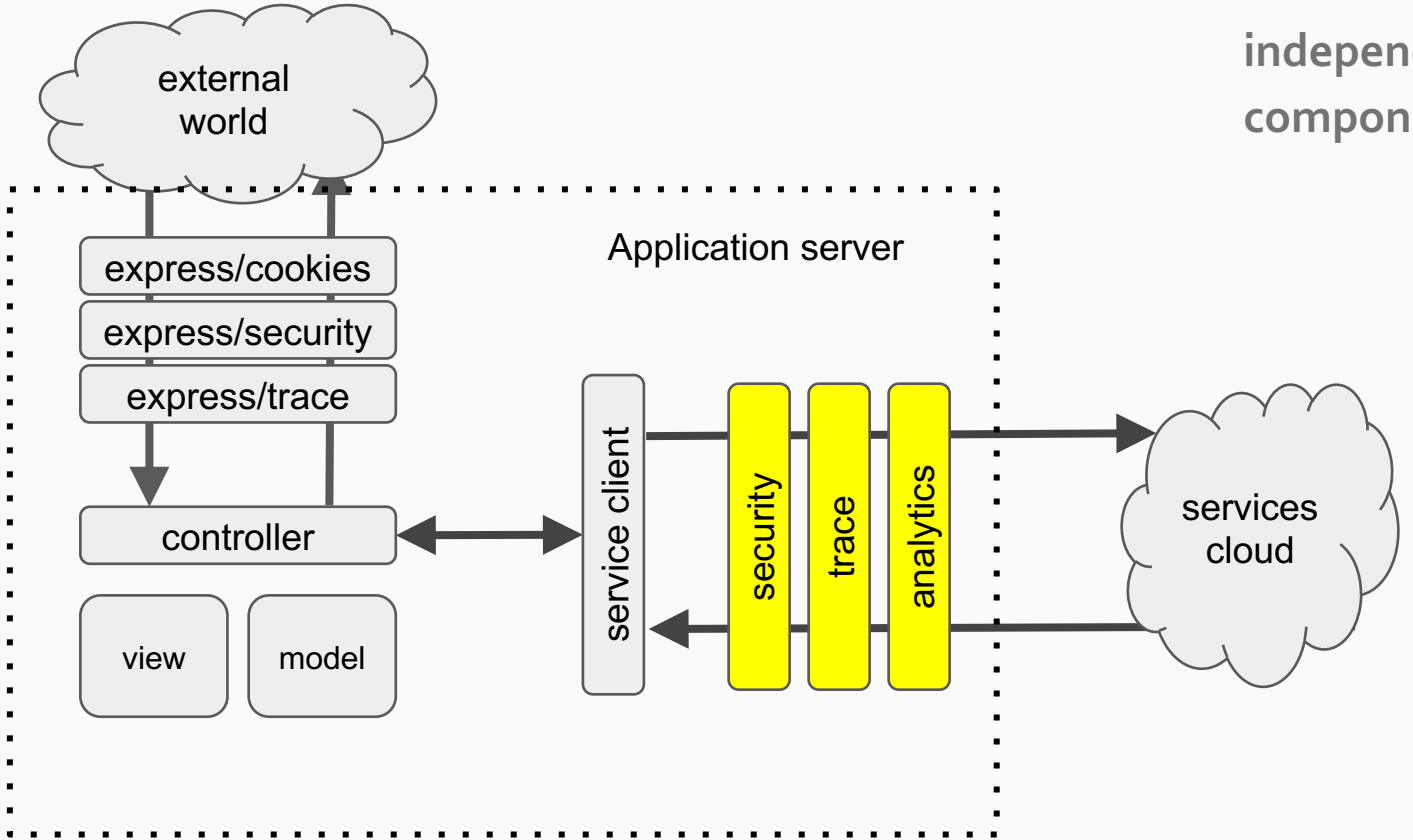
What Is framework good for?

- Organizes service calls with uniform interface

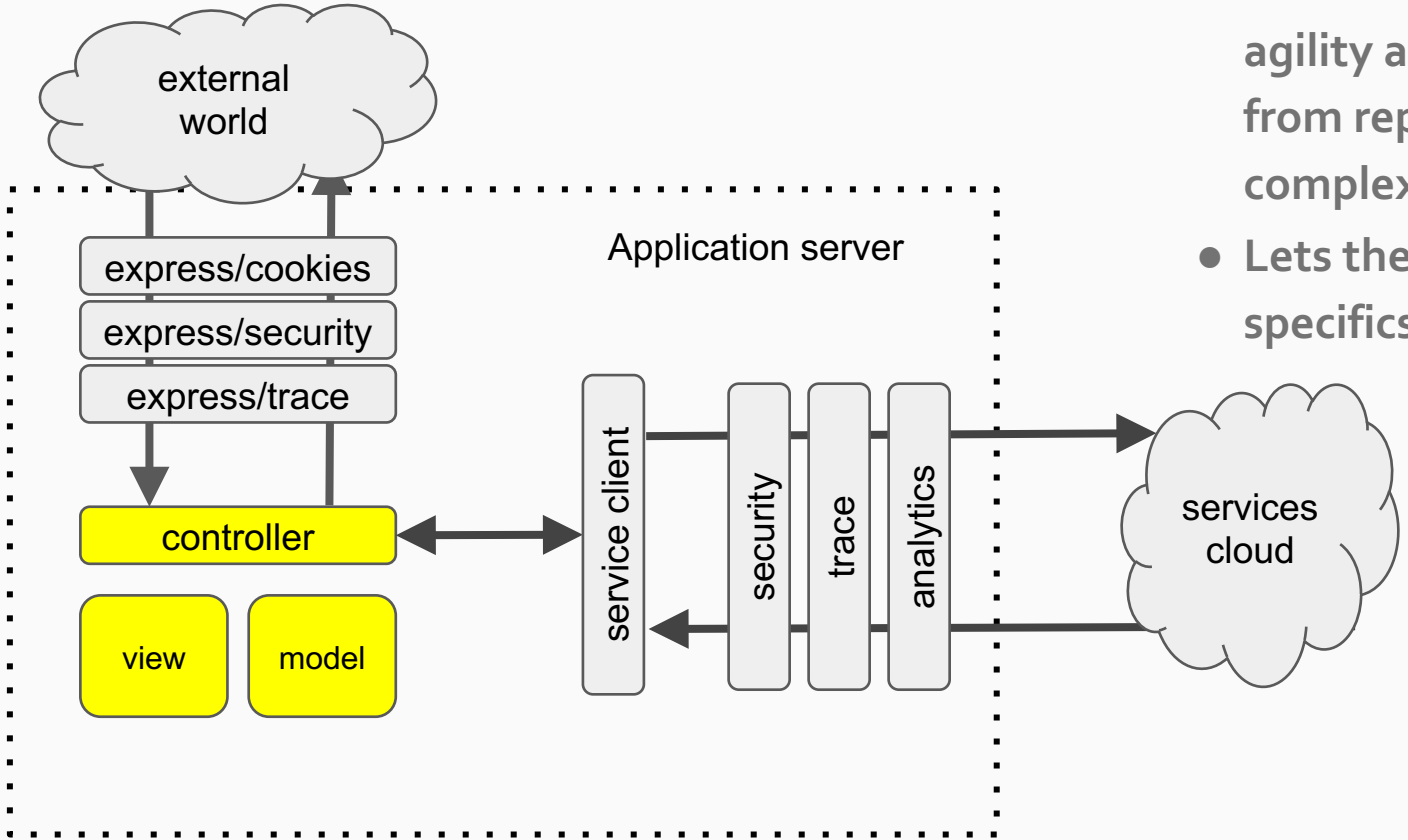


What Is framework good for?

- Structure data flow into independent testable components



What Is framework good for?



- Increases development agility and frees developer from repetitive and complex tasks
- Lets them focus on app specifics

Do we need a new framework? What is out there?

- Request (http protocol, very popular)
- Axios (http protocol, pipeline, isomorphic)
- Seneca (request/response protocol, nodejs)
- Hemera (request/response protocol, multiple languages)
- gRPC (very promising, low latency, multiple languages, schema)

Ideal framework

- Generic
- Isomorphic
- Protocol/transport free
- Minimal API
- **Extension to other frameworks, not always a replacement**

Use-cases

- Pub/Sub
- Request/Response
- Request/Response Stream
- Request Stream/Response
- Request Stream/Response Stream

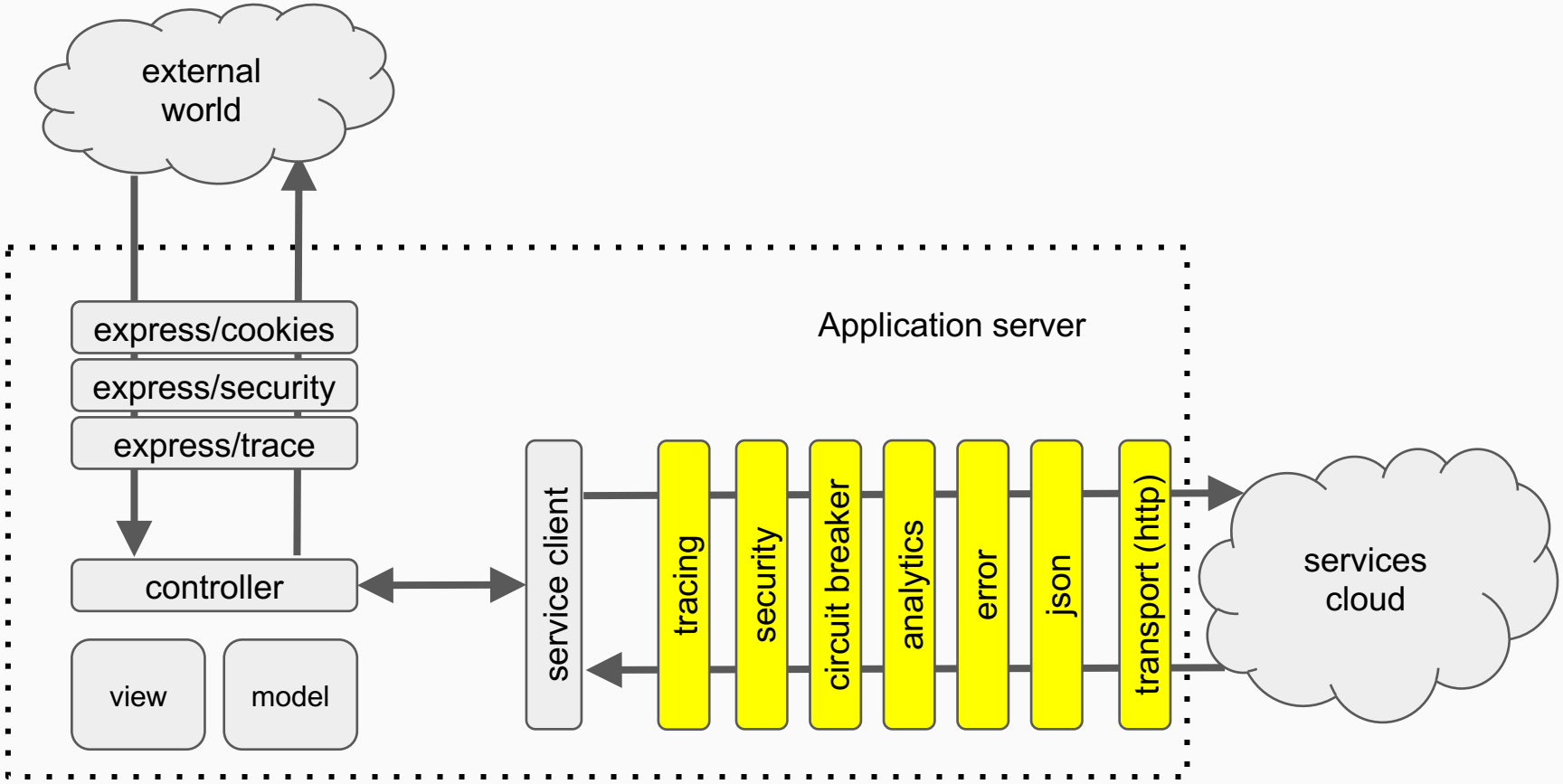
Introducing open-source pipeline framework

trooba

https://trooba.github.io/

"Trooba" [tru:ba'] means "Pipe" in Russian and it is not a pipe.

Structuring data flow



Modules available

- trooba-bootstrap
- trooba-http-transport
- trooba-grpc-transport
- trooba-xhr-transport
- trooba-http-api (superagent)
- trooba-streaming
- trooba-hystrix-handler
- trooba-toobusy-handler

https://github.com/trooba

Assembling pipeline

```
1   // assemble a pipe
2   const pipe = require('trooba')
3     .use('trace')
4     .use('security')
5     .use('http-transport', {
6       hostname: 'localhost',
7       port: 8080
8     })
9     .build();
10  // make a call
11  pipe
12    .create({'some': 'context here'})
13    .request(data, (err, response) => console.log(err, response));
```

Handler example

```
module.exports = function handler(pipe, config) {
  pipe.on('request', (request, next) => { // optional
    // modify request here
    next(); // or next({foo:'bar'})
  });
  pipe.on('error', (err, next) => { // optional
    // do something with error case
    next(); // next(new Error('Boom'))
  });
  pipe.on('response', (response, next) => { // optional
    // modify response here
    next(); // you can provide a new one next({statusCode:404})
  });
};
```

Handler, http-transport example

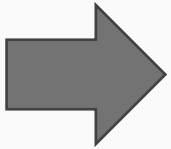
```
1  module.exports = function transport(pipe, config) {
2      pipe.on('request', request => {
3          Wreck.request(request.method,
4            config.url, request, (err, response) => {
5              if (err) return pipe.throw(err);
6              // read response
7              Wreck.read(response, (err, body) => {
8                  response.body = body;
9                  pipe.respond(response);
10             });
11         });
12     });
13 }
```

Is it enough to scale?

- Assembling a pipe with code is still an application team task
- We need to update without affecting the application code

Code vs. Config

```
const pipe = require('trooba')
  .use('trace')
  .use('security')
  .use('http-transport', {
    hostname: 'localhost',
    port: 8080
  })
```

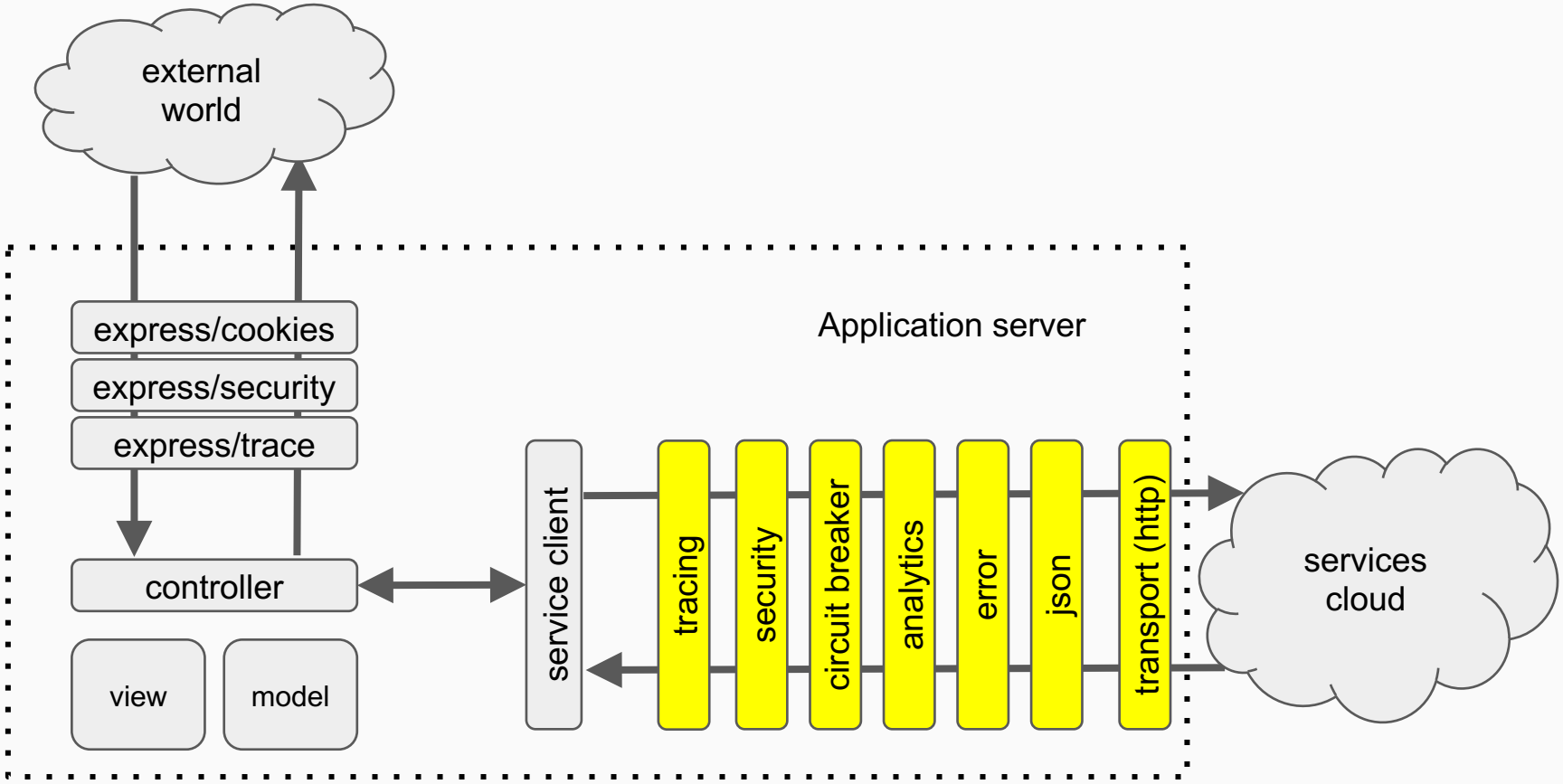


```
const handlersConfig = {
  "trace": {
    "priority": 1,
    "module": "trooba-opentrace"
  },
  "security": {
    "priority": 2,
    "module": "trooba"
  },
  "http-transport": {
    "transport": true,
    "module": "http"
  }
};
```

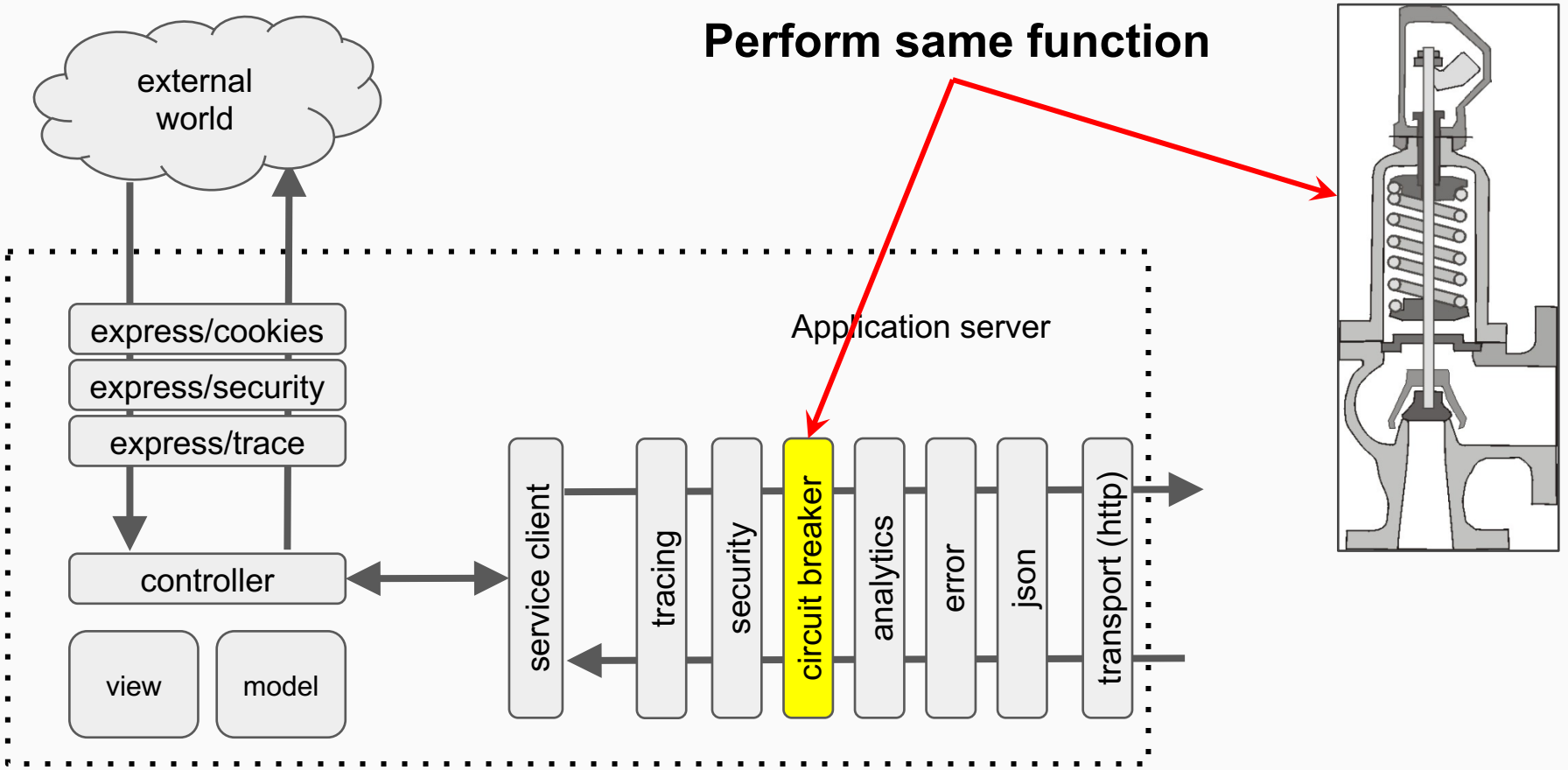
Bootstrapping pipeline from config

```
34  // load initial provider instance
35  const bootstrap = require('trooba-bootstrap');
36  const provider = bootstrap(handlersConfig, clients);
37  // somewhere later in the code
38  const pipe = provider.get('my-rest-service-client');
39  // make a call
40  pipe
41    .create({})
42    .request({foo:'bar'}, (err, response) =>
43              console.log(err, response));
```

How should we order components in the pipeline?



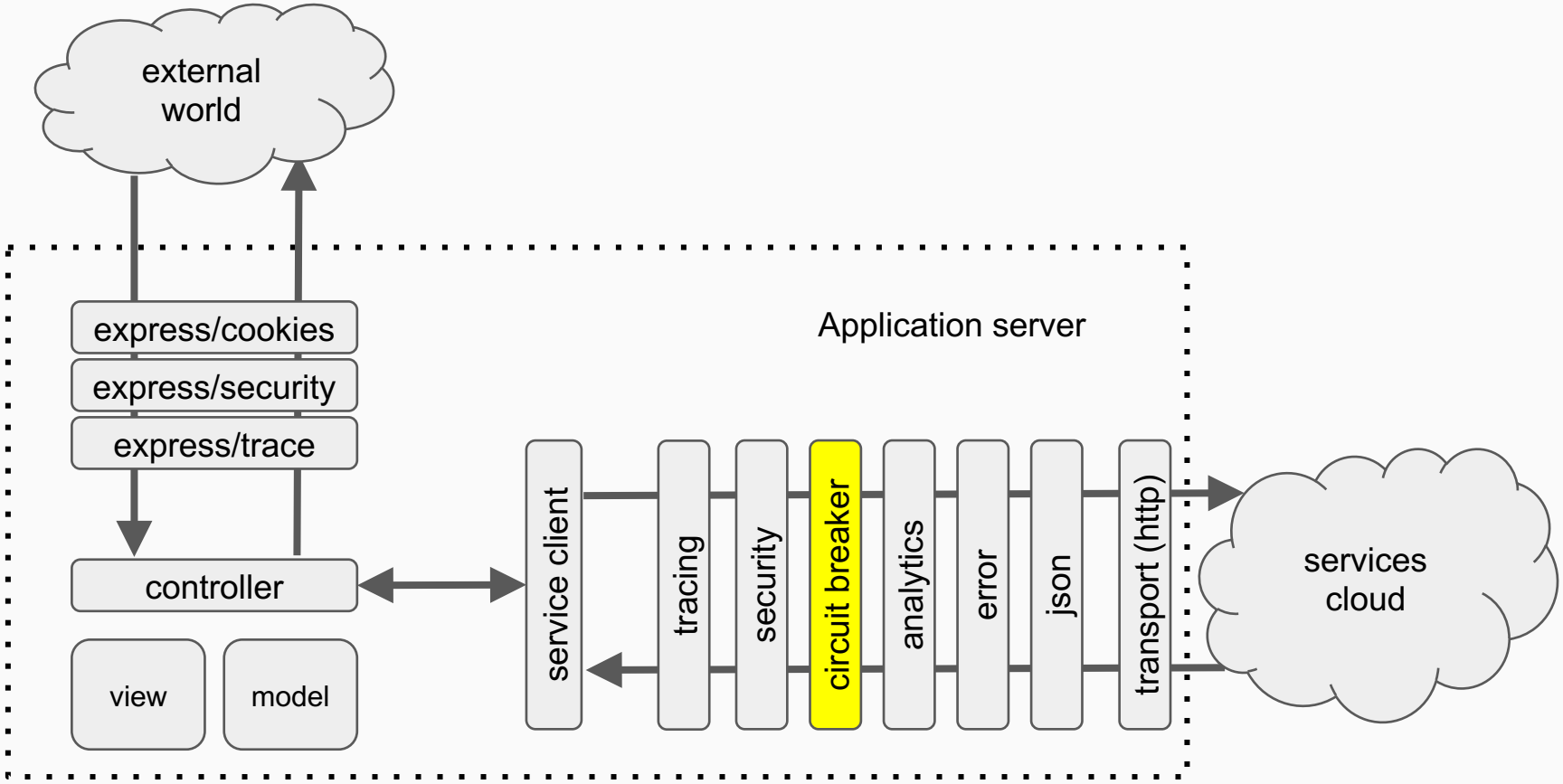
What is common between ***circuit breaker*** and ***pressure relief valve***?



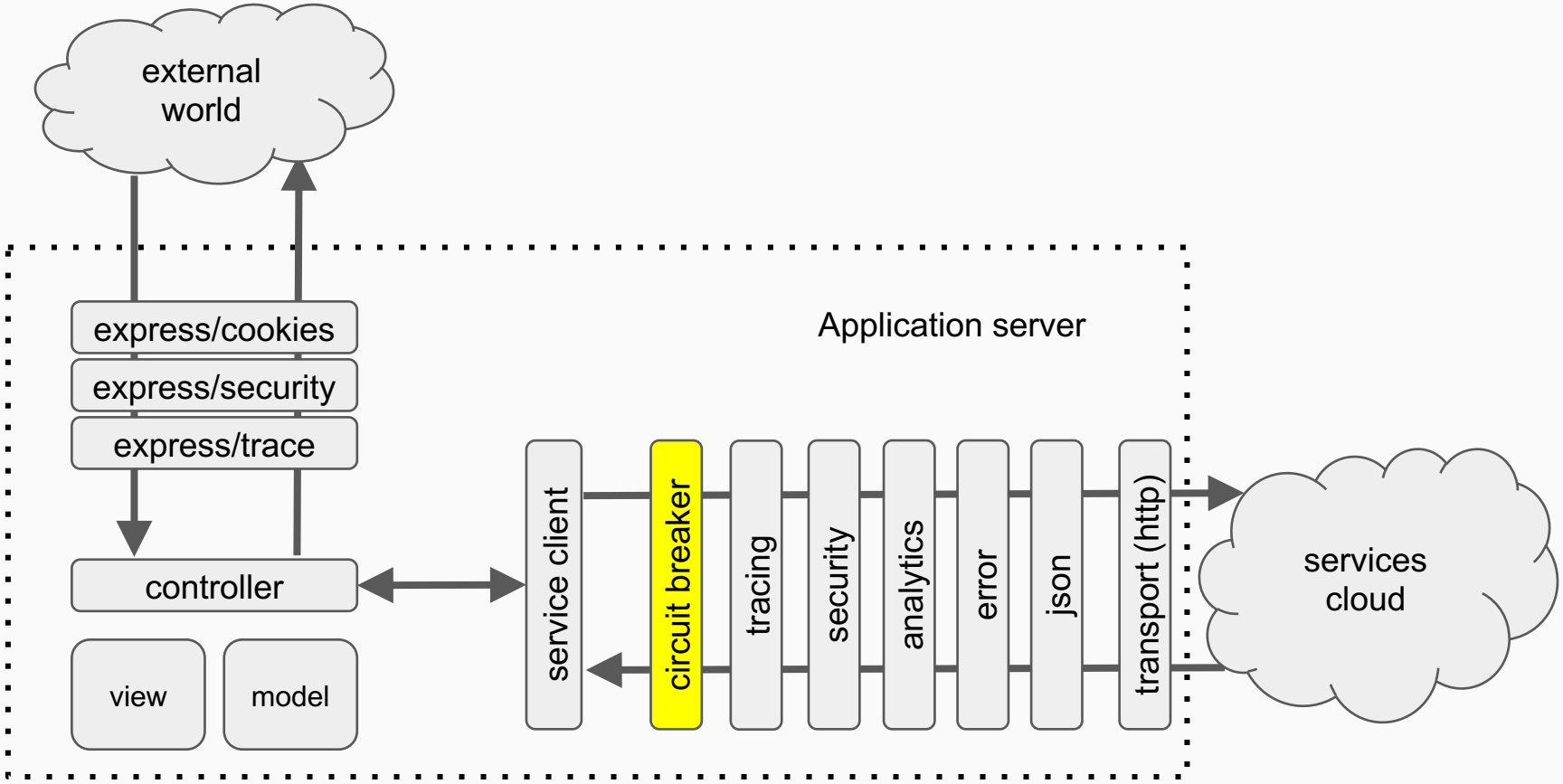
"The pressure relief valve should be located as close as possible to the pressure source or vessel being protected."

Source: http://www.dantevalve.com/faq/best-practices/

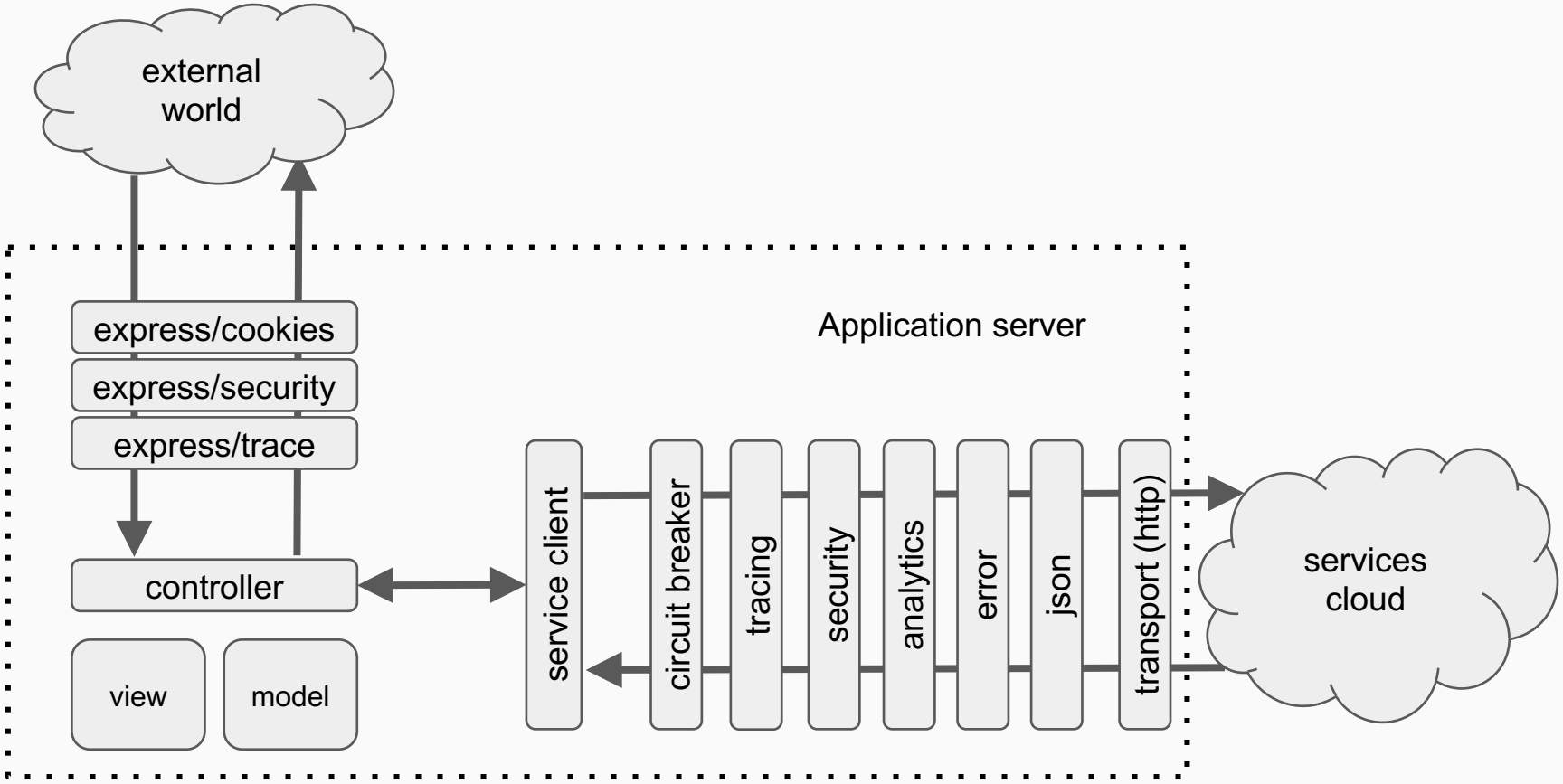
Circuit breaker position: wrong



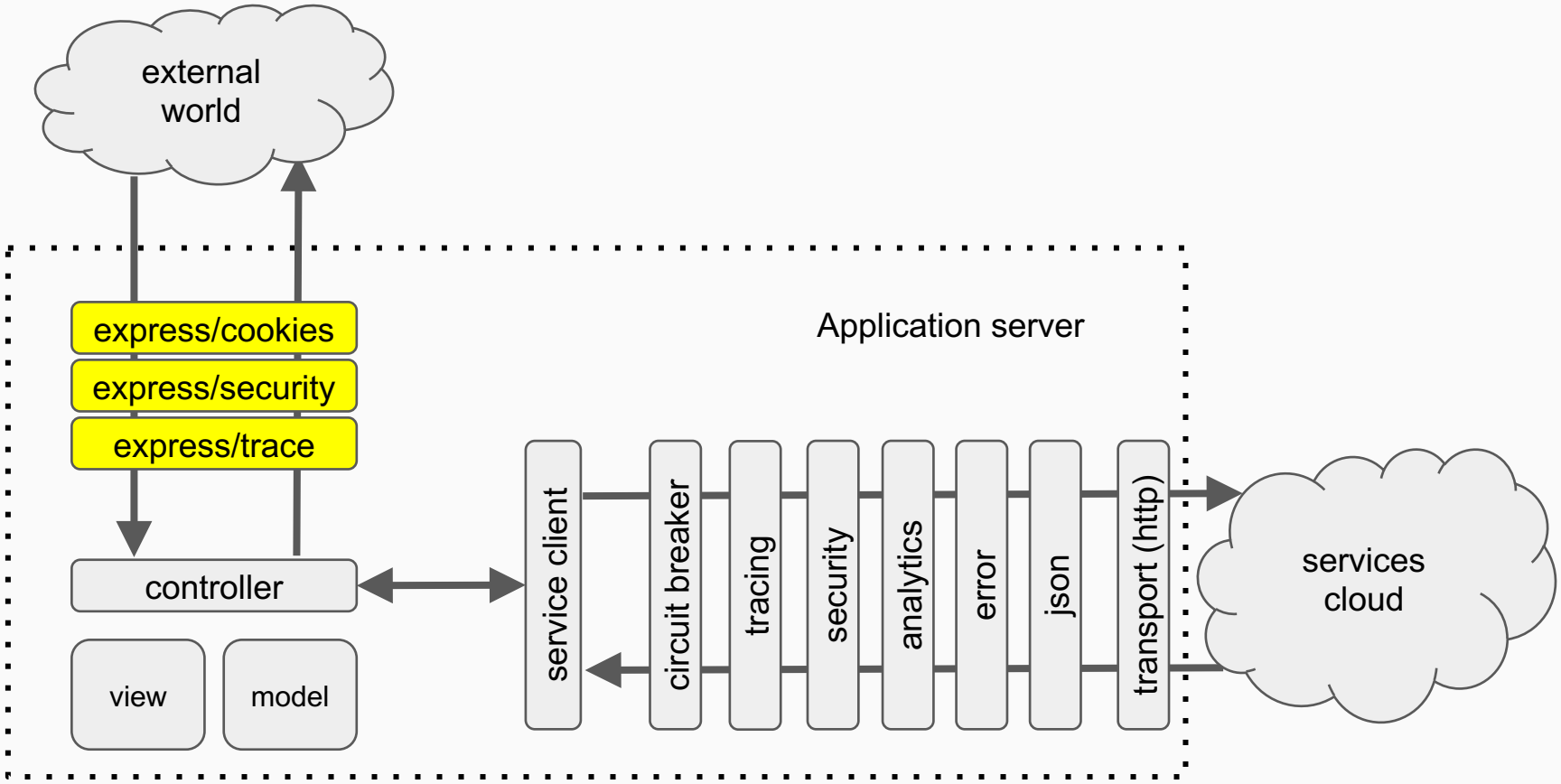
Circuit breaker position: correct

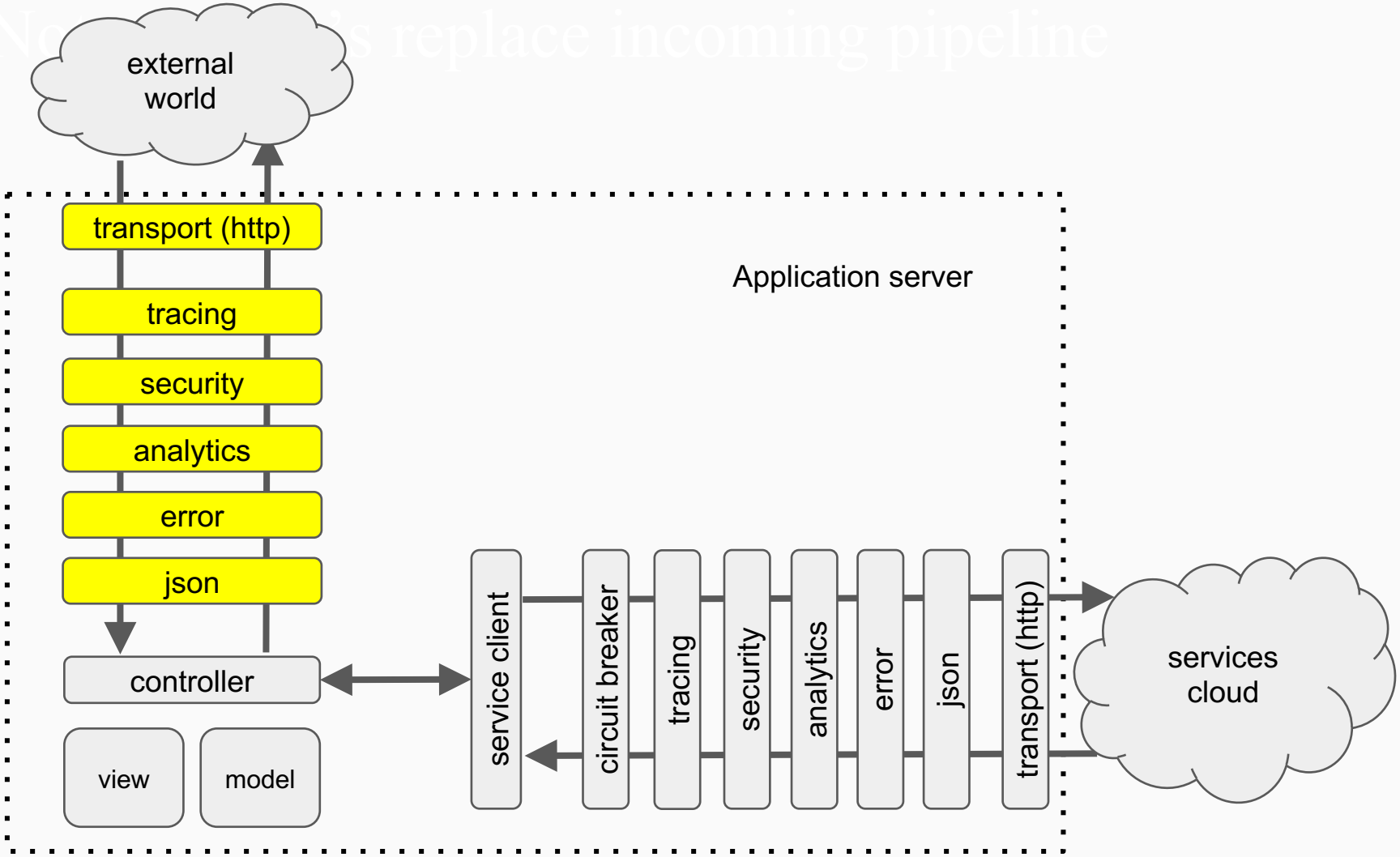


Are we done now?



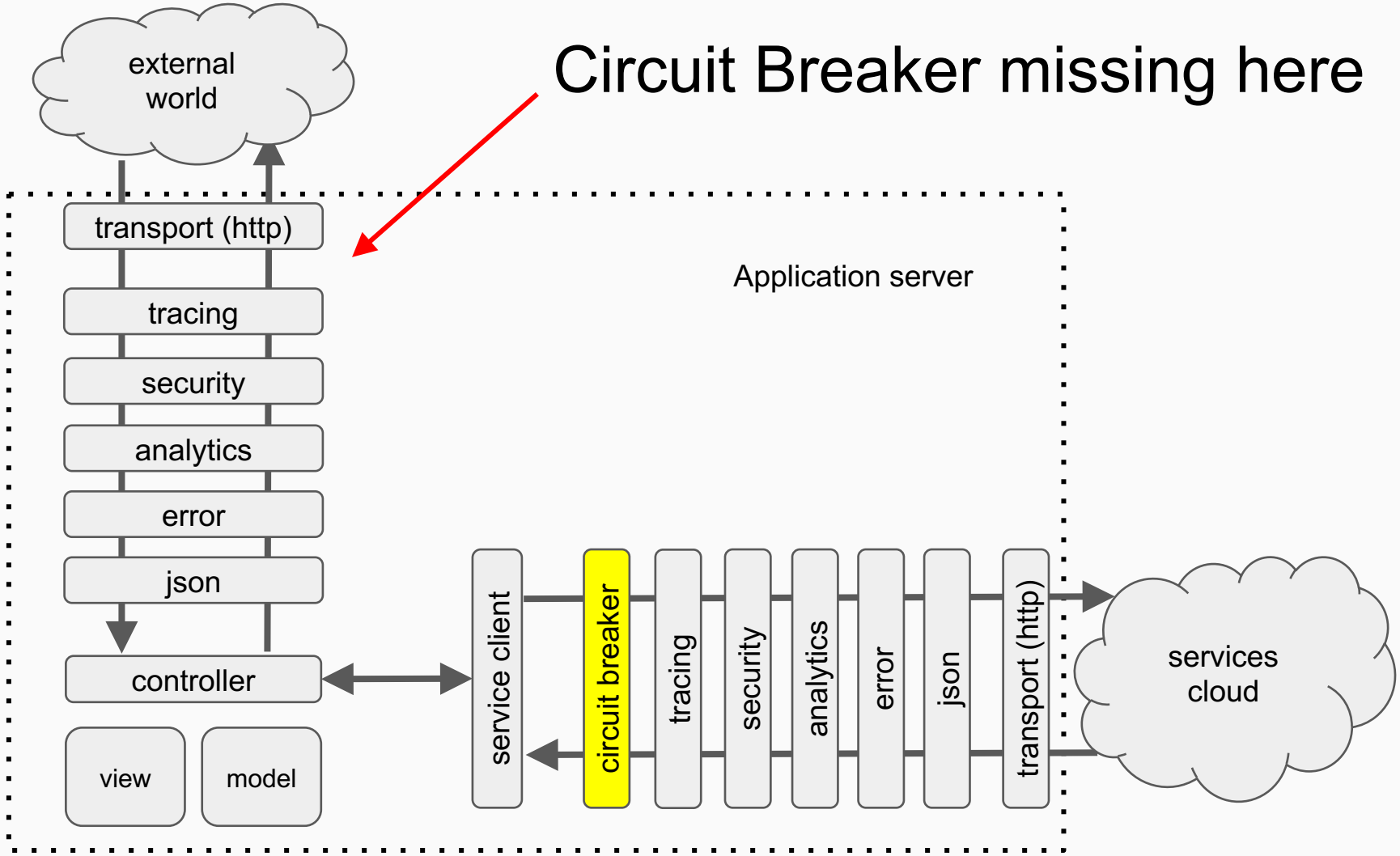
Not yet! Let's replace incoming express pipeline

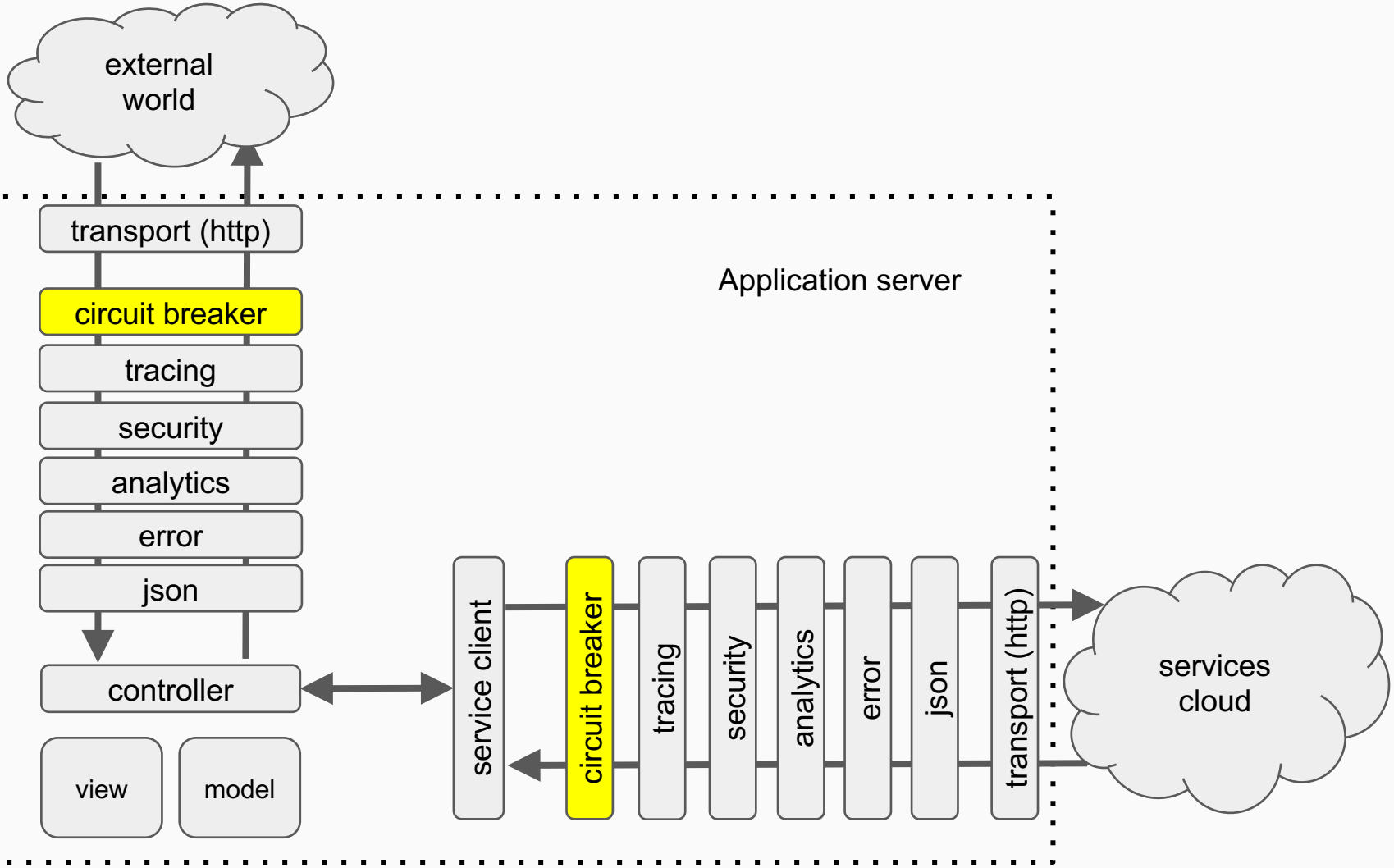



external world
transport (http)
tracing
security
analytics
error
json
Application server
controller
view
model
service client
circuit breaker
tracing
security
analytics
error
json
transport (http)
services cloud

Pipeline pattern helps

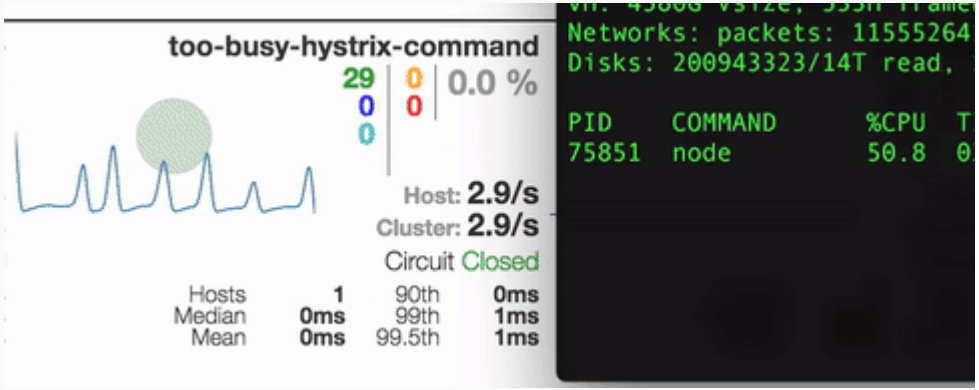
- When it is generic it is easy to spot some gaps?
- Circuit breaker is missing in incoming traffic pipeline. Let’s use it

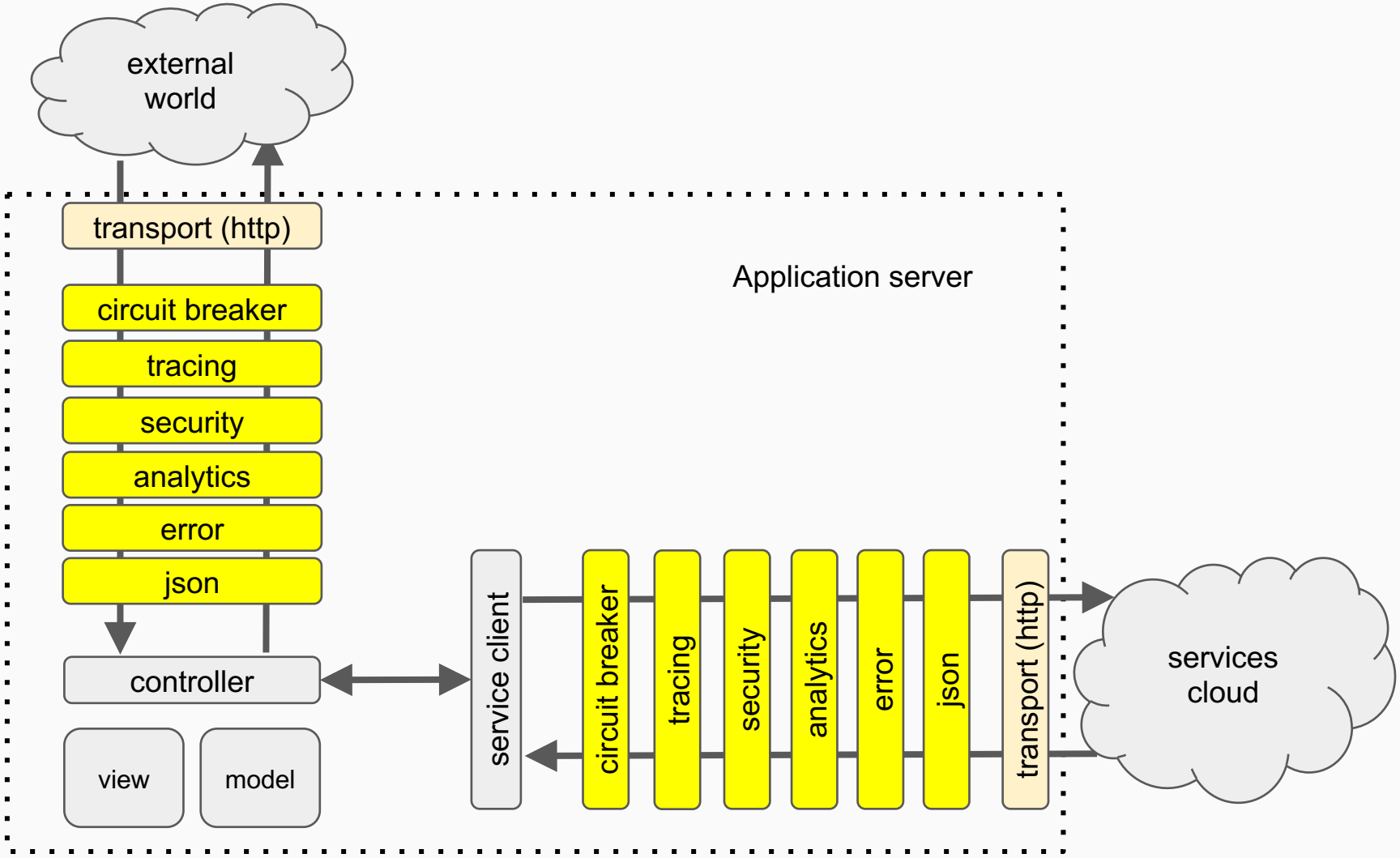

Circuit Breaker missing here
external world
transport (http)
tracing
security
analytics
error
json
controller
view
model
Application server
service client
circuit breaker
tracing
security
analytics
error
json
transport (http)
services cloud


external world
transport (http)
circuit breaker
tracing
security
analytics
error
json
controller
view
model
Application server
service client
circuit breaker
tracing
security
analytics
error
json
transport (http)
services cloud

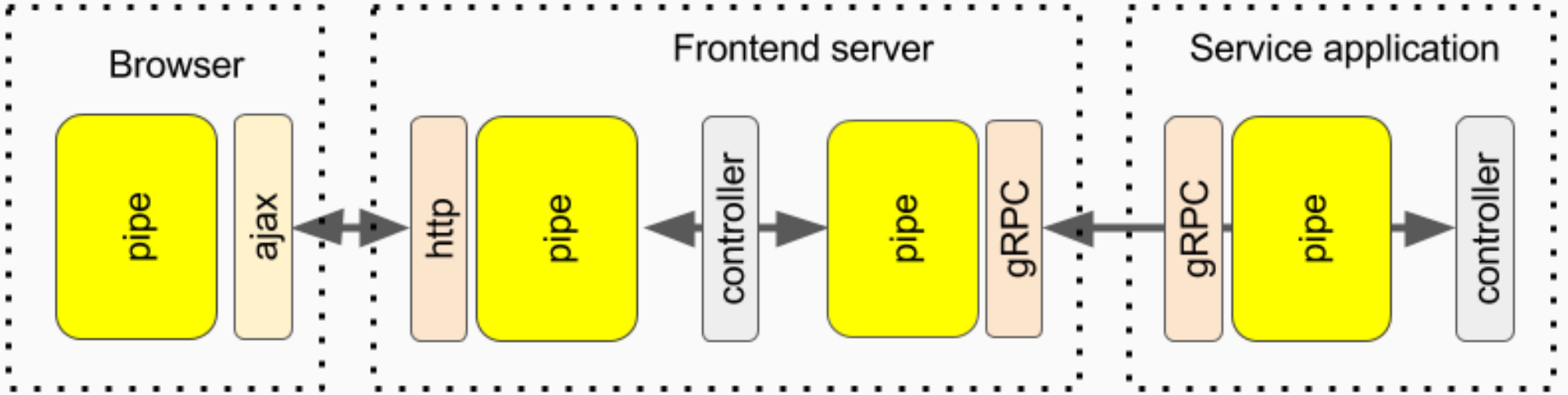
Circuit Breaker on incoming traffic

- Truly completes the application pipeline
- Coupling with too-busy
- Can react to different type of traffic
- Graceful degradation

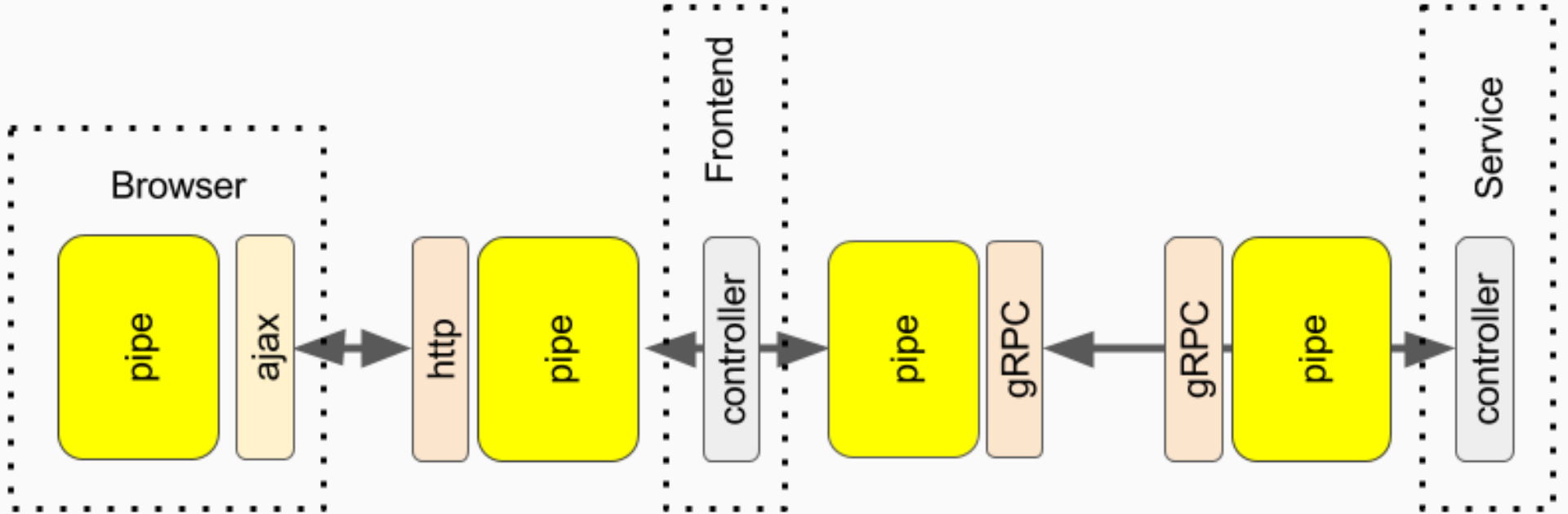



external world
transport (http)
Application server
circuit breaker
tracing
security
analytics
error
json
controller
view
model
service client
circuit breaker
tracing
security
analytics
error
json
transport (http)
services cloud

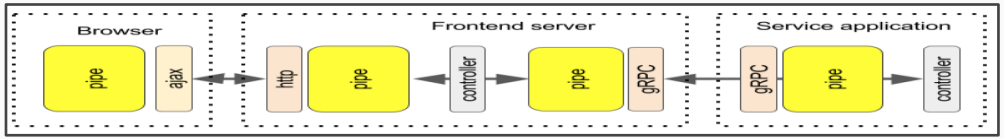
High level view



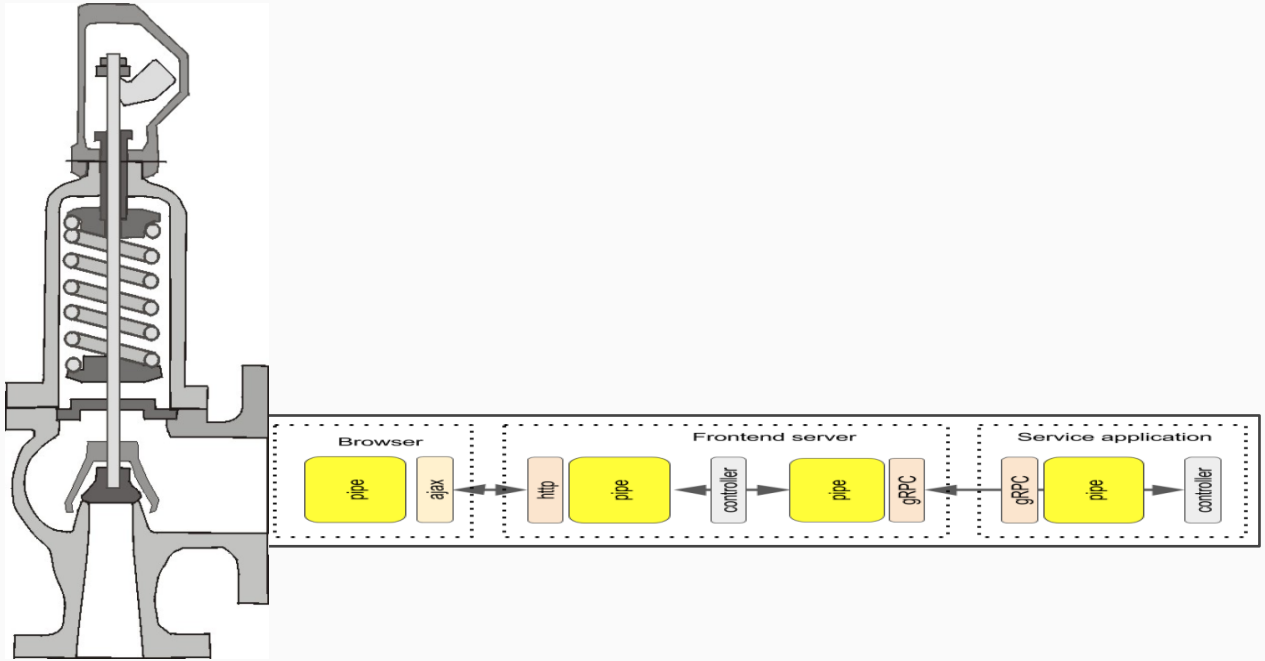
Now: Does it look like a pipe?



Step back further. Does it now look like a pipe?



How about now?



Plumbing at eBay

- All service calls use trooba
- Bootstrap from service client configuration
- Can remotely update pipeline configuration
- Protocols: Soap, REST
- gRPC client (experimental)
- gRPC service (experimental)

Write Your Own Chapter For Trooba

- We have just started a book: https://trooba.github.io/
- Contribute a handler, a transport or an API
- Write a blog explaining your design
- We review it and add it to the book

Questions?