构建后端应用的七点实践

(20:00 开始)

关于我

- 王子亭
- Node.js 开发者
- LeanCloud
- https://jysperm.me
- GitHub: jysperm

HTTP API 文档

自动测试

错误收集

日志

时序数据

负载均衡

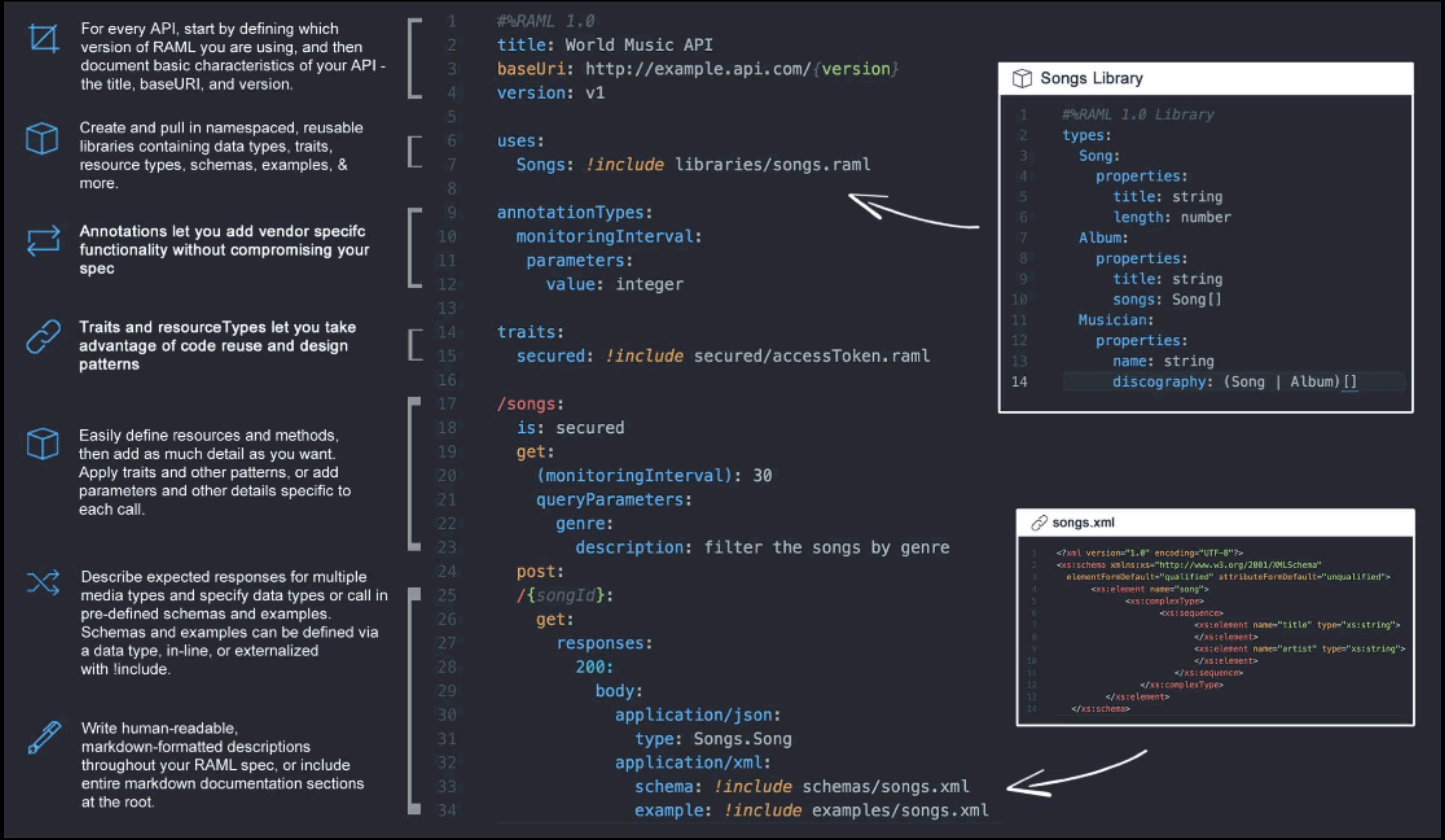
容器化

HTTP API 文档

HTTP API 文档

- Swagger
- RAML

HTTP API 文档



For every API, start by defining which version of RAML you are using, and then document basic characteristics of your API - the title, baseURI, and version.

Create and pull in namespaced, reusable libraries containing data types, traits, resource types, schemas, examples, & more.

Annotations let you add vendor specifc functionality without compromising your spec

Traits and resourceTypes let you take advantage of code reuse and design patterns

Easily define resources and methods, then add as much detail as you want. Apply traits and other patterns, or add parameters and other details specific to each call.

Describe expected responses for multiple media types and specify data types or call in pre-defined schemas and examples. Schemas and examples can be defined via a data type, in-line, or externalized with !include.

Write human-readable, markdown-formatted descriptions throughout your RAML spec, or include entire markdown documentation sections at the root.

```
#%RAML 1.0
title: World Music API
baseUri: http://example.api.com/{version}
version: v1

uses:
  Songs: !include libraries/songs.raml

annotationTypes:
  monitoringInterval:
   parameters:
     value: integer

traits:
  secured: !include secured/accessToken.raml

/songs:
  is: secured
  get:
    (monitoringInterval): 30
    queryParameters:
      genre:
        description: filter the songs by genre
  post:
  /{songId}:
    get:
      responses:
        200:
          body:
            application/json:
              type: Songs.Song
            application/xml:
              schema: !include schemas/songs.xml
              example: !include examples/songs.xml
```

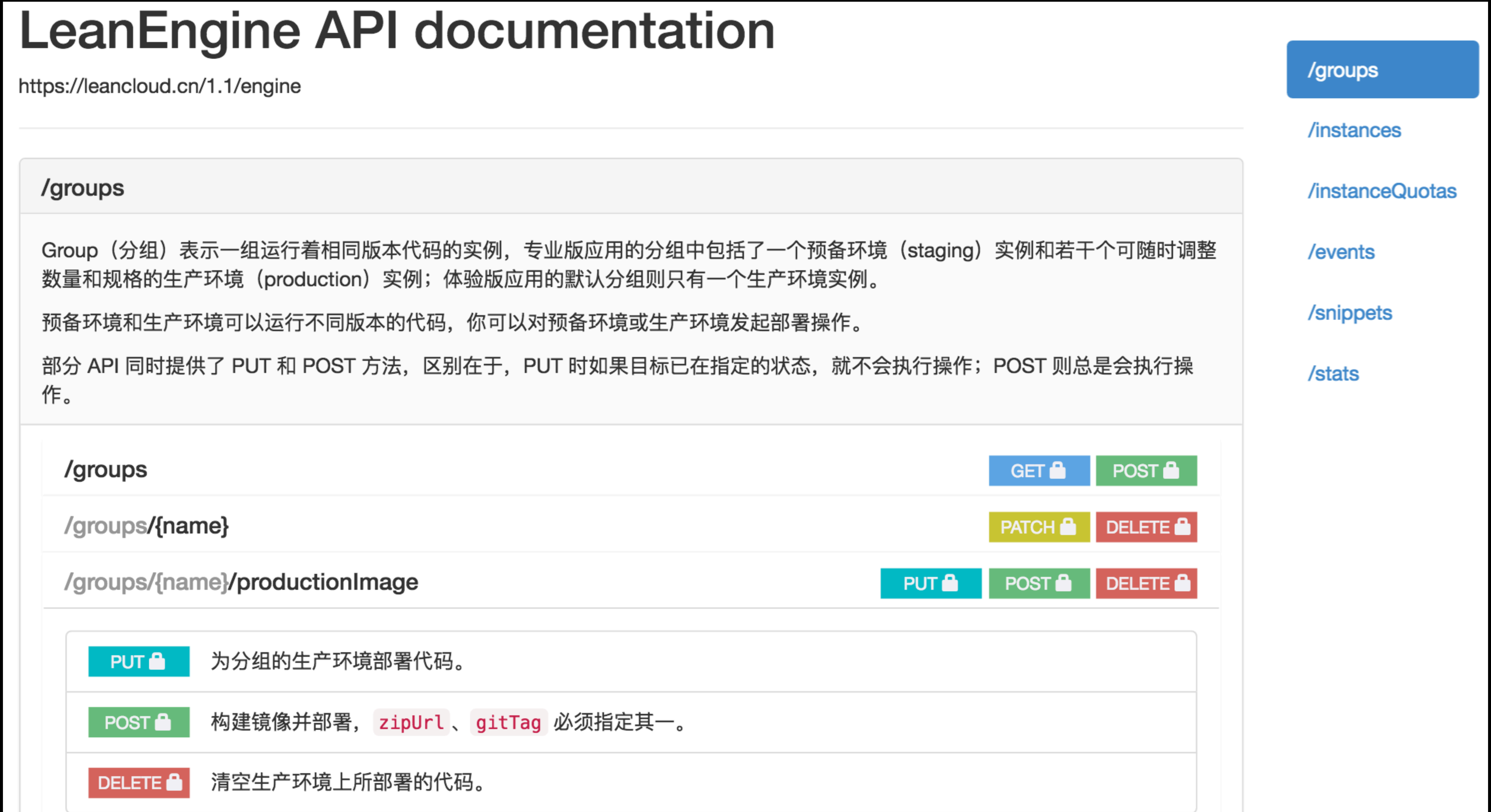
Songs Library

```
#%RAML 1.0 Library
types:
  Song:
    properties:
      title: string
      length: number
  Album:
    properties:
      title: string
      songs: Song[]
  Musician:
    properties:
      name: string
      discography: (Song | Album)[]
```

songs.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="song">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="title" type="xs:string">
                        </xs:element>
                        <xs:element name="artist" type="xs:string">
                        </xs:element>
                     </xs:sequence>
               </xs:complexType>
         </xs:element>
   </xs:schema>
```

HTTP API 文档



LeanEngine API documentation

https://leancloud.cn/1.1/engine

/groups

Group（分组）表示一组运行着相同版本代码的实例，专业版应用的分组中包括了一个预备环境（staging）实例和若干个可随时调整数量和规格的生产环境（production）实例；体验版应用的默认分组则只有一个生产环境实例。

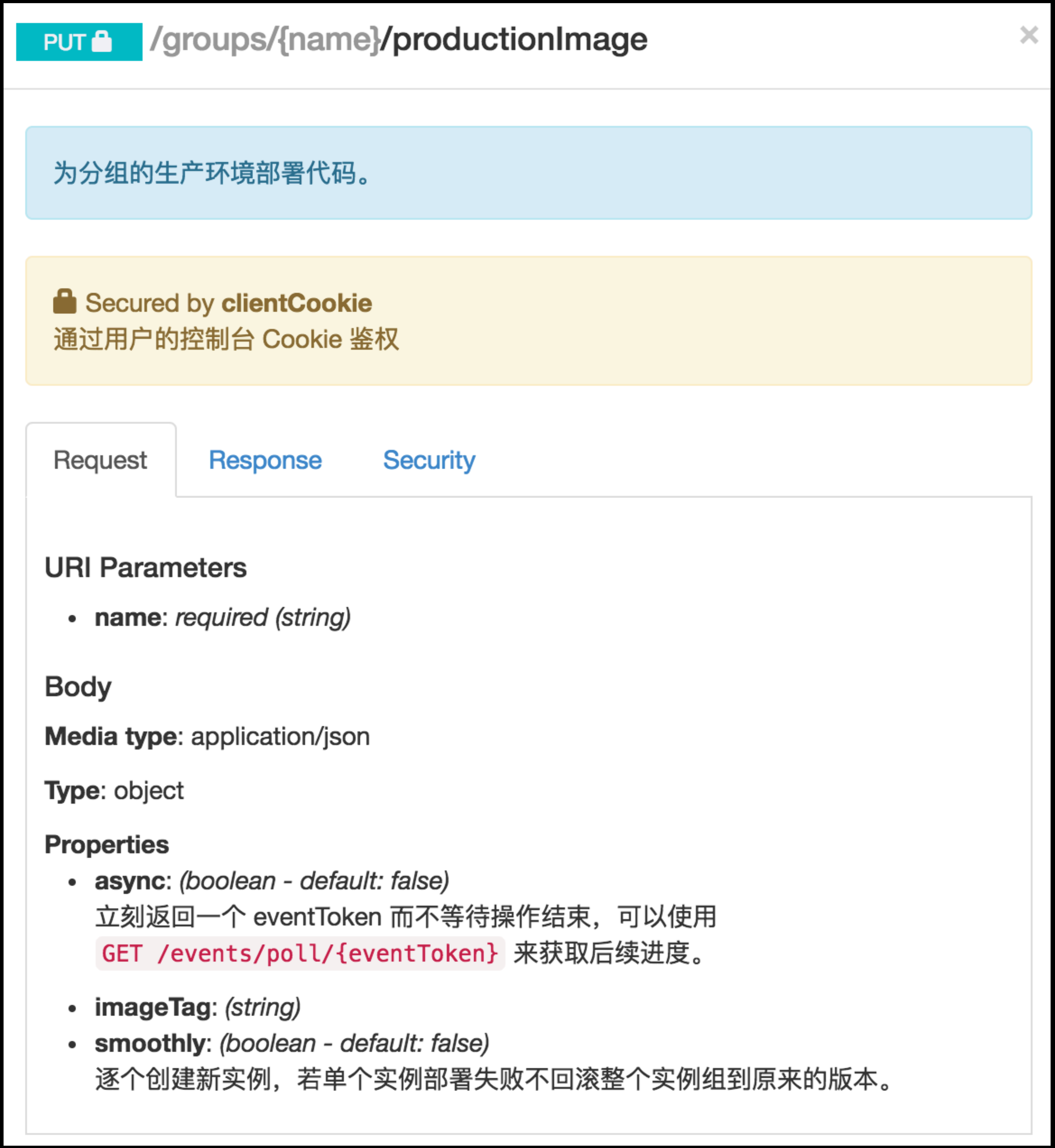
预备环境和生产环境可以运行不同版本的代码，你可以对预备环境或生产环境发起部署操作。

部分 API 同时提供了 PUT 和 POST 方法，区别在于，PUT 时如果目标已在指定的状态，就不会执行操作；POST 则总是会执行操作。

/groups	GET POST
/groups/{name}	PATCH DELETE
/groups/{name}/productionImage	PUT POST DELETE

PUT	为分组的生产环境部署代码。
POST	构建镜像并部署，`zipUrl`、`gitTag` 必须指定其一。
DELETE	清空生产环境上所部署的代码。

HTTP API 文档



PUT /groups/{name}/productionImage

为分组的生产环境部署代码。

Secured by **clientCookie**
通过用户的控制台 Cookie 鉴权

Request | Response | Security

URI Parameters

- **name**: *required (string)*

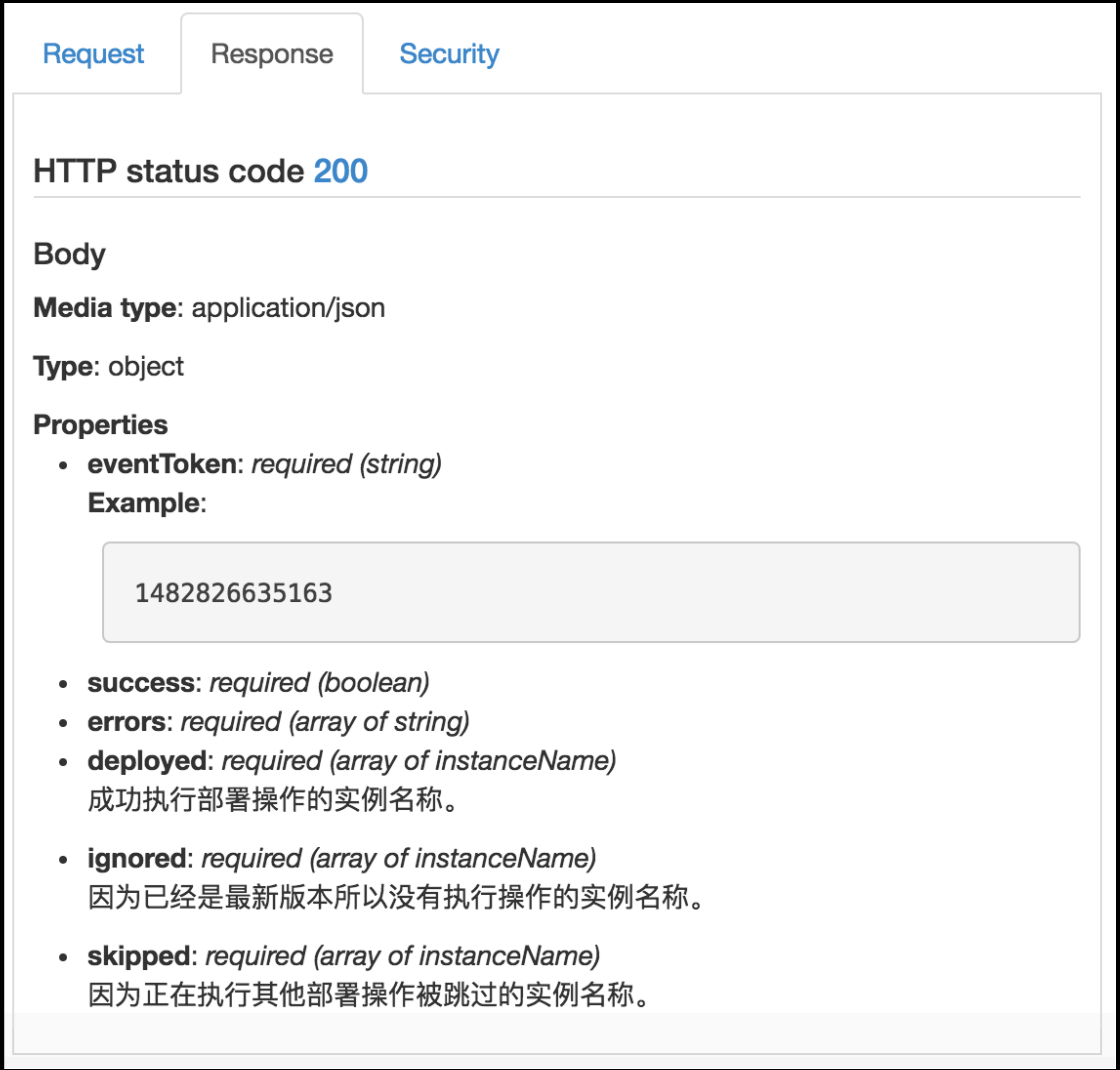
Body

Media type: application/json

Type: object

Properties

- **async**: *(boolean - default: false)*
  立刻返回一个 eventToken 而不等待操作结束，可以使用 `GET /events/poll/{eventToken}` 来获取后续进度。
- **imageTag**: *(string)*
- **smoothly**: *(boolean - default: false)*
  逐个创建新实例，若单个实例部署失败不回滚整个实例组到原来的版本。



Request | Response | Security

HTTP status code 200

Body

Media type: application/json

Type: object

Properties

- **eventToken**: *required (string)*
  Example:

```
1482826635163
```

- **success**: *required (boolean)*
- **errors**: *required (array of string)*
- **deployed**: *required (array of instanceName)*
  成功执行部署操作的实例名称。
- **ignored**: *required (array of instanceName)*
  因为已经是最新版本所以没有执行操作的实例名称。
- **skipped**: *required (array of instanceName)*
  因为正在执行其他部署操作被跳过的实例名称。

自动测试

自动测试 vs. 手动测试

自动测试

- 什么时候写测试
- 优先给哪些部分写测试
- 处理测试的依赖关系
- 自动运行测试（CI）

异常收集

异常收集

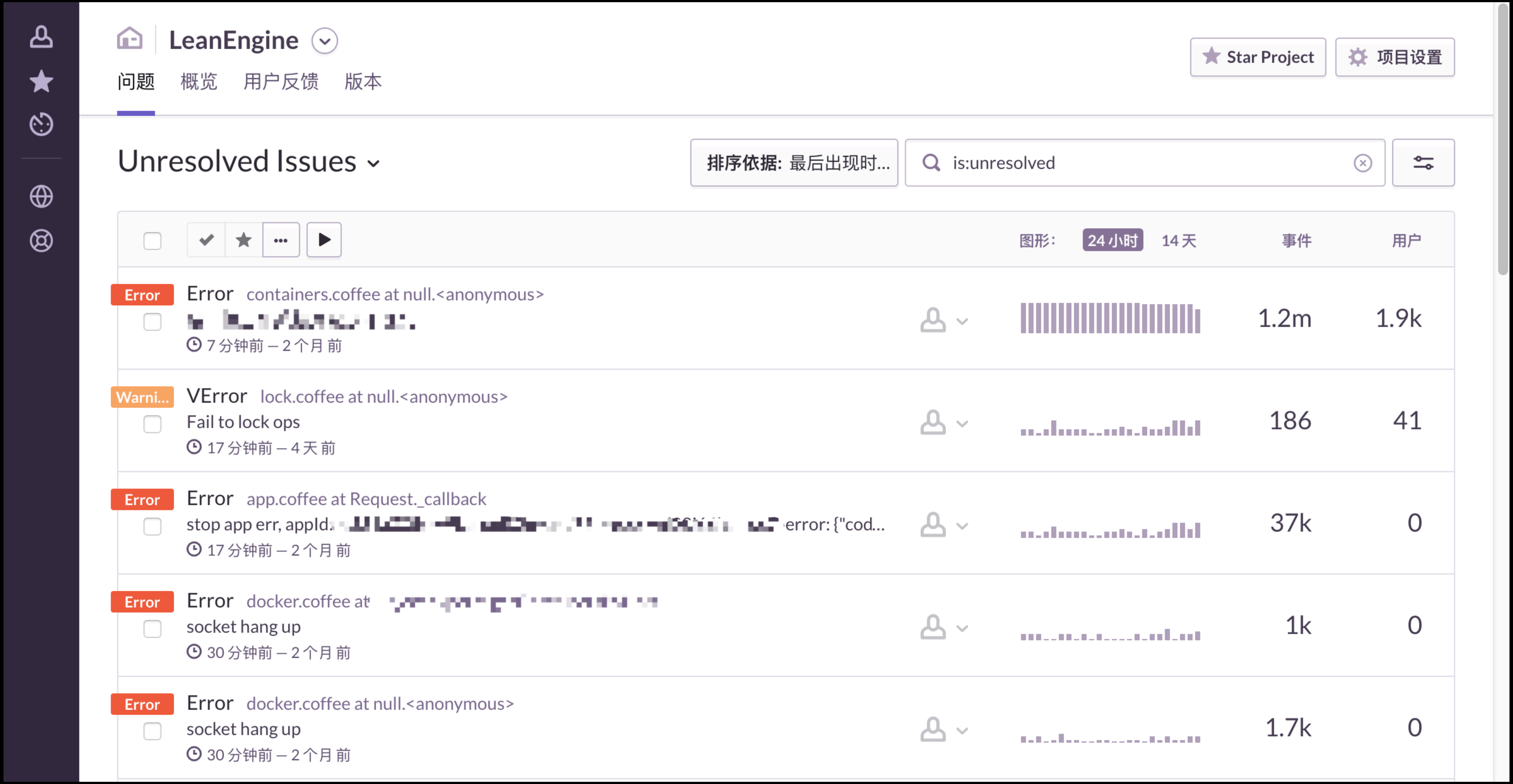
```
function errorAlert(err) {
  AV.Cloud.requestSmsCode({
    mobilePhoneNumber: '1850[redacted]',
    template: 'errorAlert',
    message: err.message
  });
}
```

```
ClassA.create({ ... }, (err, objectA) => {
  if (err) {
    return errorAlert(err);
  }

  ClassB.create({ ... }, (err, objectB) => {
    if (err) {
      return errorAlert(err);
    }

    ClassC.create({objectA, objectB}, (err) => {
      if (err) {
        return errorAlert(err);
      }
    });
  });
});
```

异常收集



LeanEngine

问题 概览 用户反馈 版本

Star Project 项目设置

Unresolved Issues

排序依据: 最后出现时...

is:unresolved

图形: 24 小时 14 天 事件 用户

Error Error containers.coffee at null.<anonymous>
[redacted]
7 分钟前 – 2 个月 前 1.2m 1.9k

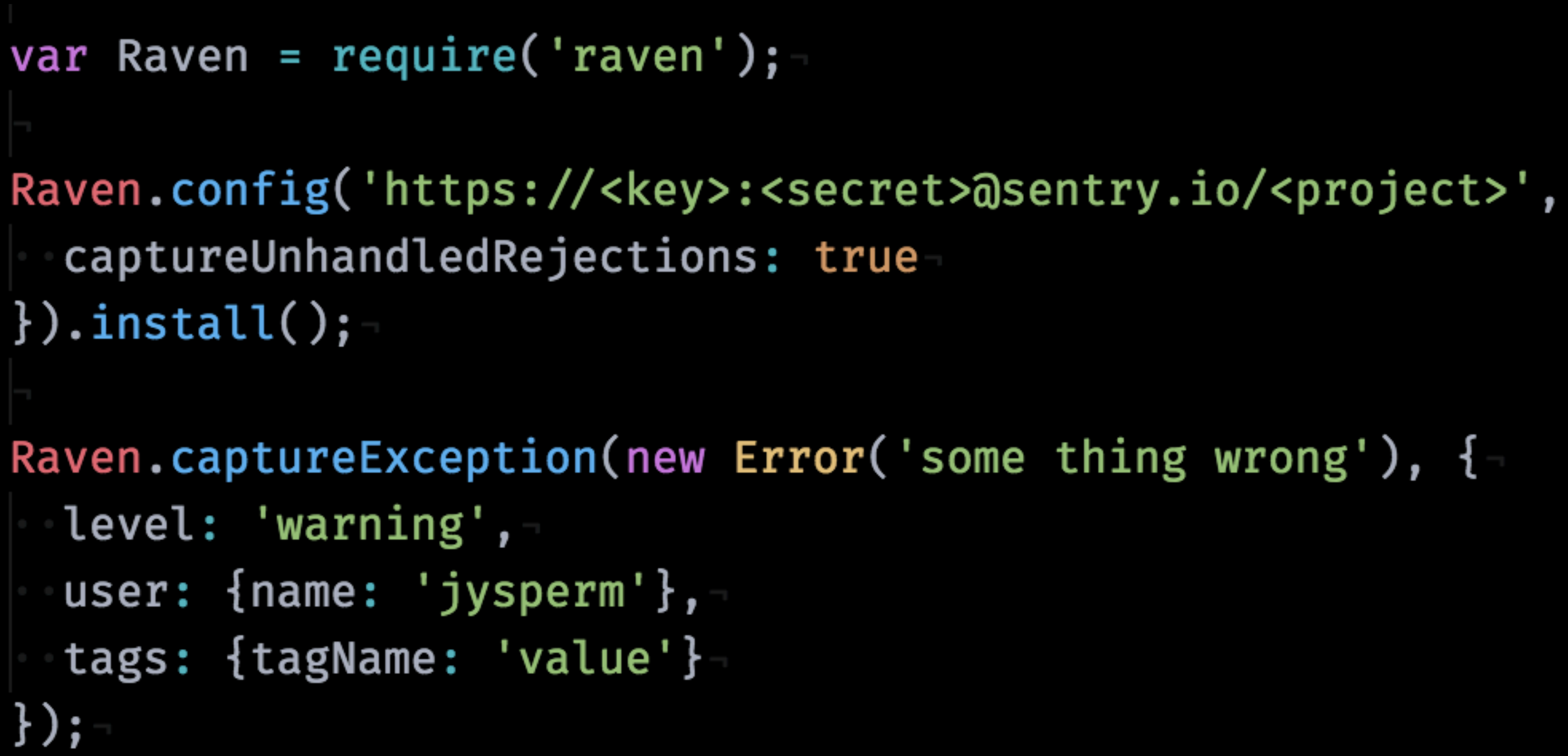
Warni... VError lock.coffee at null.<anonymous>
Fail to lock ops
17 分钟前 – 4 天 前 186 41

Error Error app.coffee at Request._callback
stop app err, appId: [redacted] error: {"cod...
17 分钟前 – 2 个月 前 37k 0

Error Error docker.coffee at [redacted]
socket hang up
30 分钟前 – 2 个月 前 1k 0

Error Error docker.coffee at null.<anonymous>
socket hang up
30 分钟前 – 2 个月 前 1.7k 0

异常收集

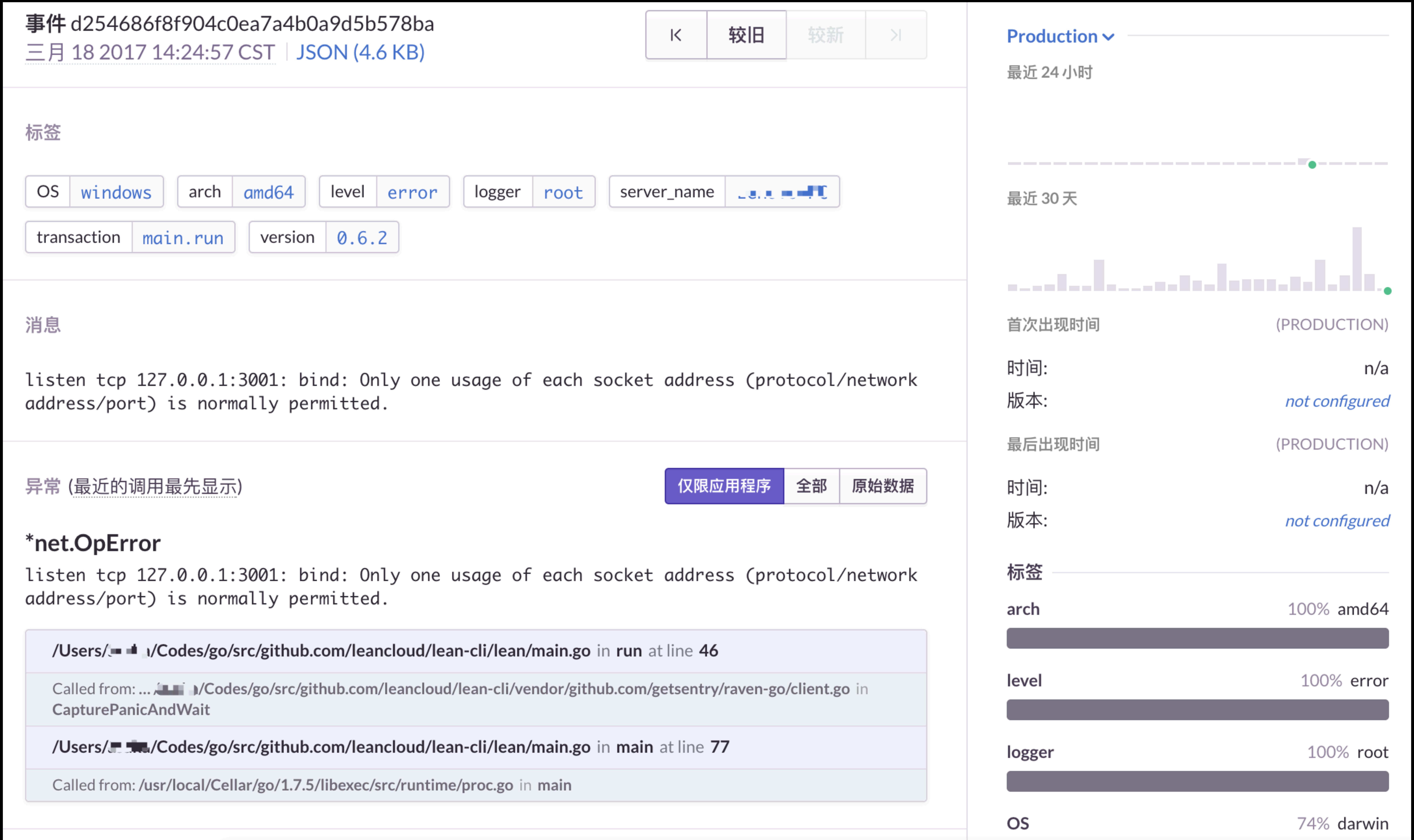


```javascript
var Raven = require('raven');

Raven.config('https://<key>:<secret>@sentry.io/<project>',
  captureUnhandledRejections: true
}).install();

Raven.captureException(new Error('some thing wrong'), {
  level: 'warning',
  user: {name: 'jysperm'},
  tags: {tagName: 'value'}
});
```

异常收集



事件 d254686f8f904c0ea7a4b0a9d5b578ba
三月 18 2017 14:24:57 CST | JSON (4.6 KB)

标签

OS	windows
arch	amd64
level	error
logger	root
server_name	[redacted]
transaction	main.run
version	0.6.2

消息

```
listen tcp 127.0.0.1:3001: bind: Only one usage of each socket address (protocol/network
address/port) is normally permitted.
```

异常 (最近的调用最先显示)

***net.OpError**

```
listen tcp 127.0.0.1:3001: bind: Only one usage of each socket address (protocol/network
address/port) is normally permitted.
```

/Users/[redacted]/Codes/go/src/github.com/leancloud/lean-cli/lean/main.go in **run** at line **46**

Called from: ...[redacted]/Codes/go/src/github.com/leancloud/lean-cli/vendor/github.com/getsentry/raven-go/client.go in CapturePanicAndWait

/Users/[redacted]/Codes/go/src/github.com/leancloud/lean-cli/lean/main.go in **main** at line **77**

Called from: /usr/local/Cellar/go/1.7.5/libexec/src/runtime/proc.go in main

Production

最近 24 小时

最近 30 天

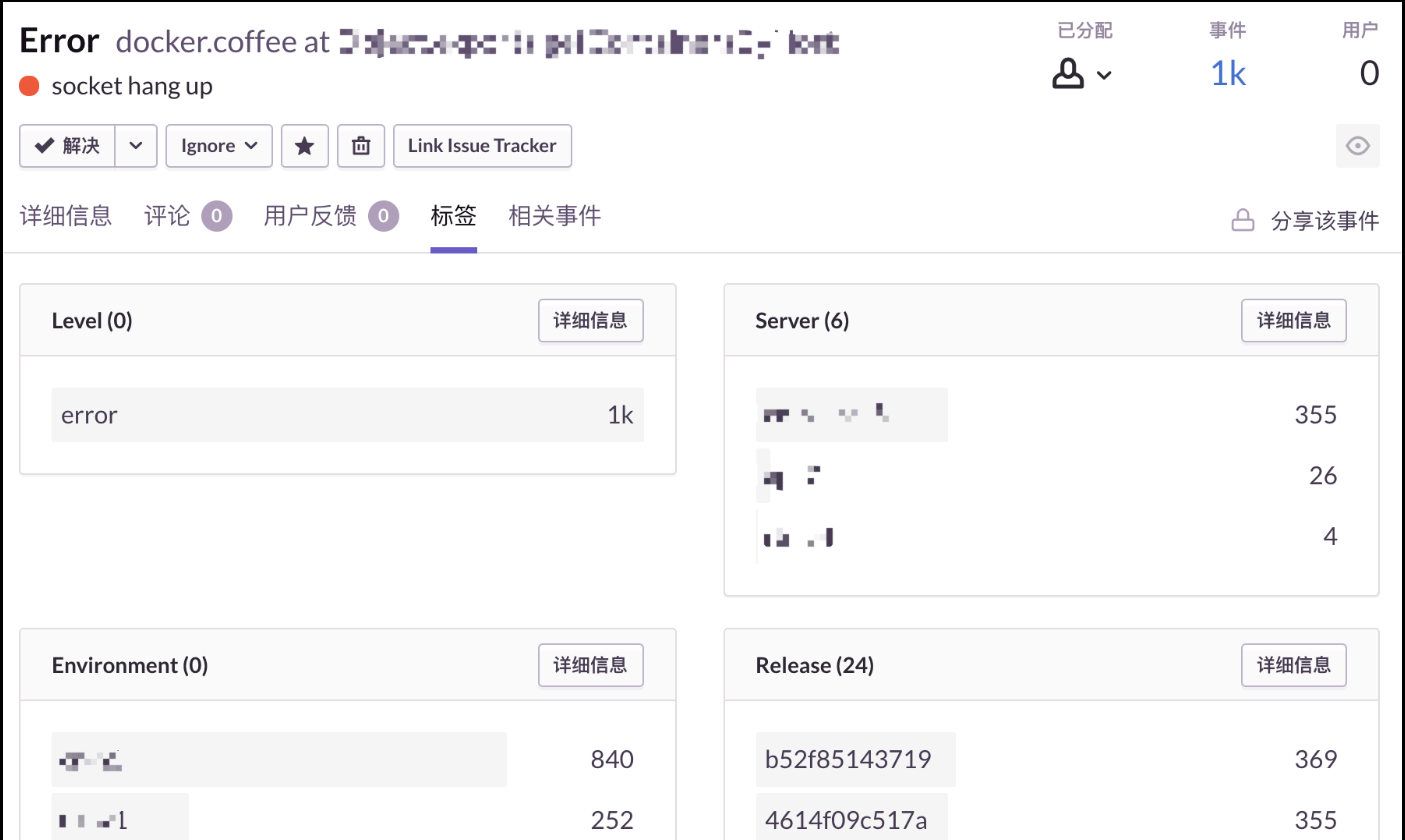
首次出现时间	(PRODUCTION)
时间:	n/a
版本:	*not configured*

最后出现时间	(PRODUCTION)
时间:	n/a
版本:	*not configured*

标签

arch	100%	amd64
level	100%	error
logger	100%	root
OS	74%	darwin

异常收集



Error docker.coffee at [redacted]

● socket hang up

已分配	事件	用户
	1k	0

Level (0) 详细信息

error	1k

Server (6) 详细信息

[redacted]	355
[redacted]	26
[redacted]	4

Environment (0) 详细信息

[redacted]	840
[redacted]	252

Release (24) 详细信息

b52f85143719	369
4614f09c517a	355

异常收集

- 每个错误第一次出现时发送一个报警
- 当被标记解决的错误重新出现时发送一个报警
- 同一个错误一小时内出现十次以上发送一个报警
- 对于最高级别的错误每次都发送报警

日志

ELK

Elasticsearch + Logstash + Kibana

日志收集

```
{"level":40,"username":"jysperm","msg":"login by password","time":"2013-01-04T18:46:23.851Z"}
{"level":50,"username":"jysperm","file_id": 123,"msg":"delete a file","time":"2013-01-04T18:46:23.853Z"}
{"level":40,"username":"someone","file_id": 124,"msg":"upload a file","time":"2013-01-04T18:46:23.853Z"}
{"level":50,"username":"someone""msg":"update password","time":"2013-01-04T18:46:23.853Z"}
 ...
```

```
cat log | jq -c 'select(.username == "jysperm")'
```

```
{"level":30,"username":"jysperm","msg":"login by password","time":"2013-01-04T18:46:23.851Z"}
{"level":40,"username":"jysperm","file_id":123,"msg":"delete a file","time":"2013-01-04T18:46:23.853Z"}
```

时序数据

时序数据库

- 所有数据都与一个时间相关联、所有查询亦与一个时间段相关联
- 能够按照数据的标签（Tag）进行查询和分组聚合
- 能够支持大量的插入和查询操作

时序数据

- OpenTSDB（Java、HBase）
- InfluxDB（Golang）

时序数据

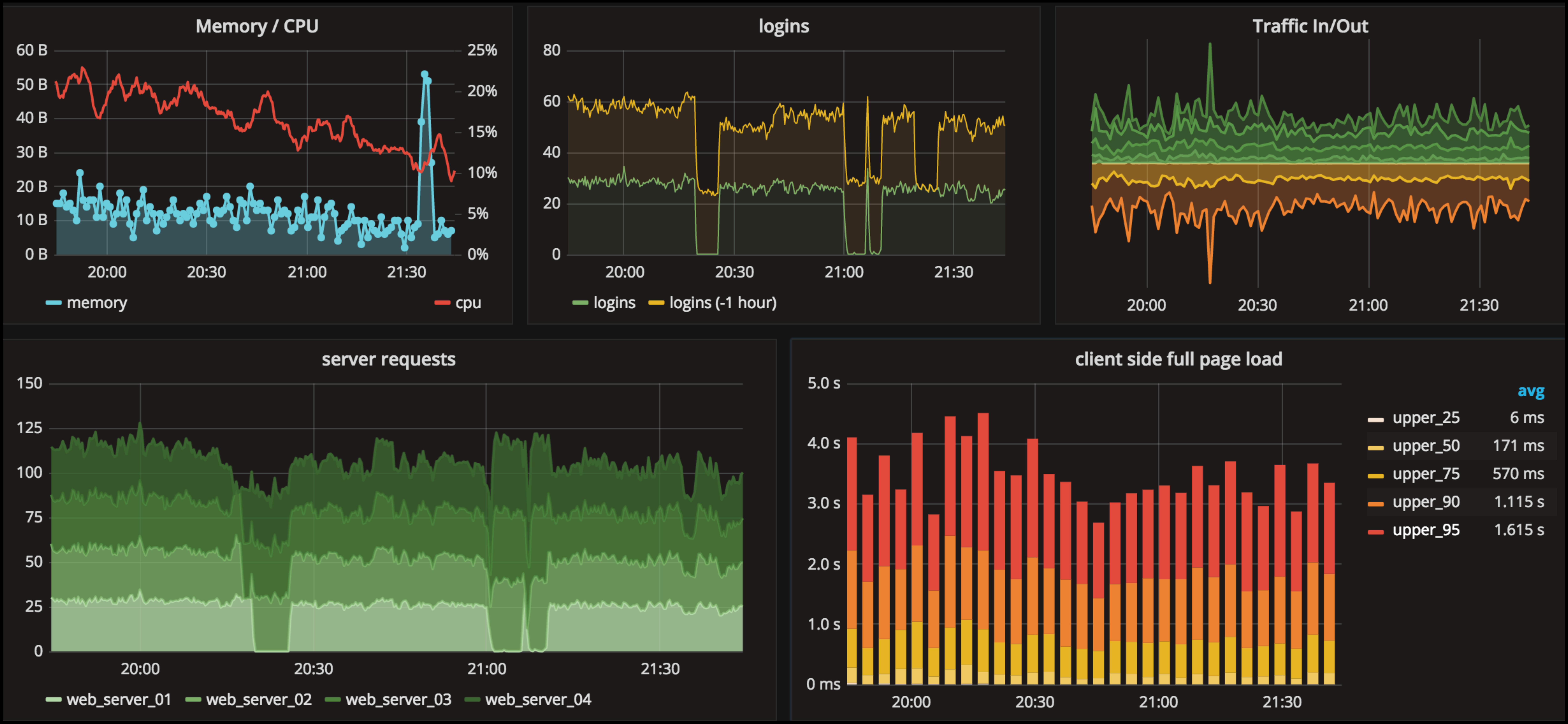
Series	Timestamp	Metric	Value	Tags
(cpu_load,host_name=web1)	'2017-03-28T13:22:01.249Z'	cpu_load	2.1	host_name=web1
(cpu_load,host_name=web1)	'2017-03-28T13:23:01.249Z'	cpu_load	2.2	host_name=web1
(cpu_load,host_name=web2)	'2017-03-28T13:23:01.249Z'	cpu_load	3	host_name=web2
(memory_used,host_name=web2)	'2017-03-28T13:23:01.249Z'	memory_used	1234	host_name=web2

```sql
SELECT mean(value), max(value) FROM cpu_load GROUP BY time(10m);
SELECT value FROM cpu_load WHERE host_name = 'web1';
```

时序数据

- access log
- 关键函数执行次数、耗时、成功率
- 服务器 CPU 内存等负荷信息
- 数据库连接数、缓存容量

时序数据

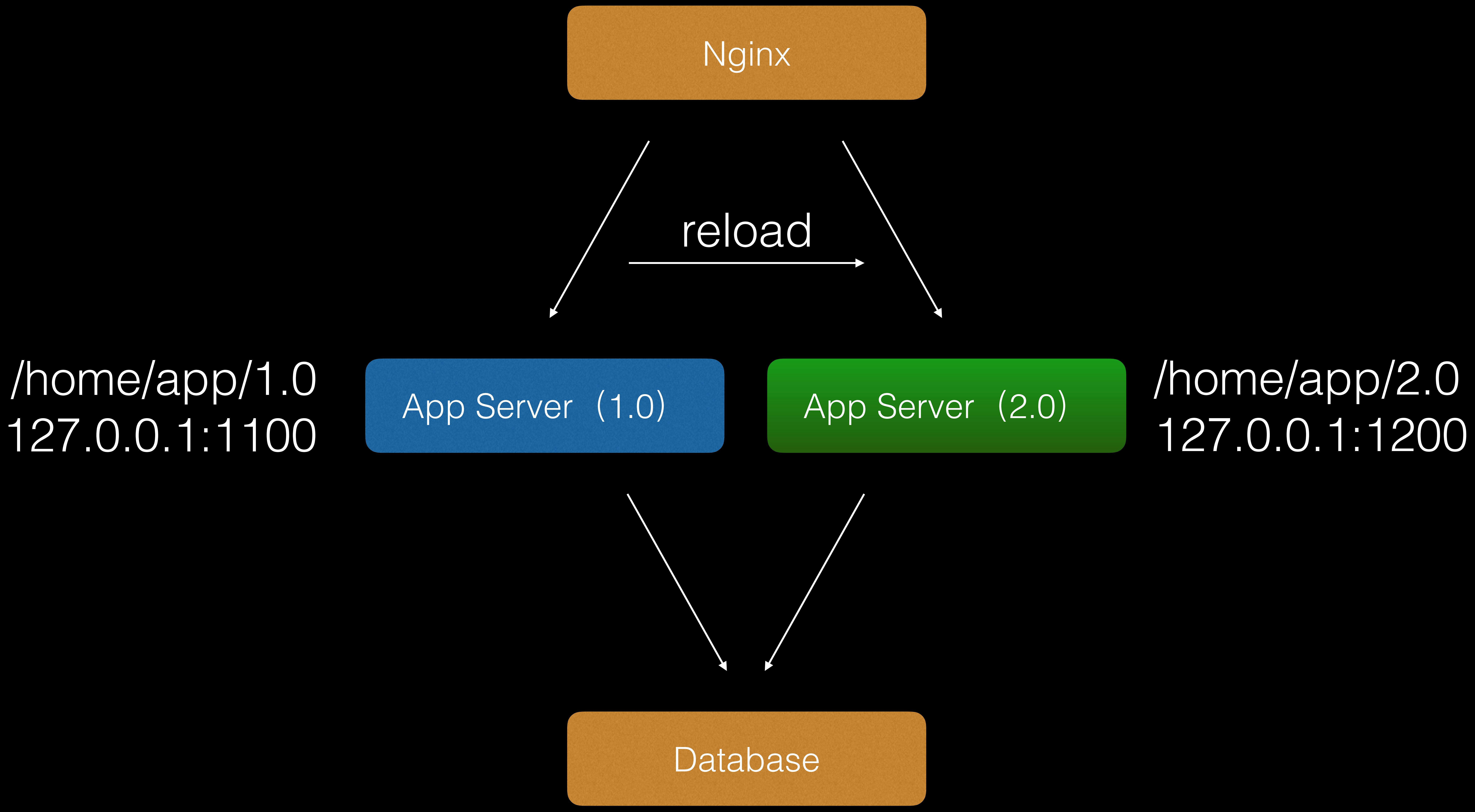

Memory / CPU
60 B
50 B
40 B
30 B
20 B
10 B
0 B
25%
20%
15%
10%
5%
0%
20:00
20:30
21:00
21:30
memory
cpu
logins
80
60
40
20
0
logins
logins (-1 hour)
Traffic In/Out
server requests
150
125
100
75
50
25
0
web_server_01
web_server_02
web_server_03
web_server_04
client side full page load
5.0 s
4.0 s
3.0 s
2.0 s
1.0 s
0 ms
avg
upper_25 6 ms
upper_50 171 ms
upper_75 570 ms
upper_90 1.115 s
upper_95 1.615 s

负载均衡

负载均衡

- 专门的反向代理程序（Nginx）在处理 SSL 和慢请求时会更有效率
- 应用经常要进行版本更新，在更新期间不应中断服务
- 通常一个服务由若干个应用进程来提供（还可能位于不同的服务器）

负载均衡



Nginx
reload
/home/app/1.0
127.0.0.1:1100
App Server （1.0）
App Server （2.0）
/home/app/2.0
127.0.0.1:1200
Database

容器化

容器化

- 将环境、依赖、代码打包为整体
- 更细致地隔离应用、统计和限制资源消耗
- 无差别地调度计算能力

容器化

- 运行在容器中
- 负载均衡（服务发现）
- 日志收集、时序数据收集
- 容器迁移

小结

HTTP API 文档（RAML）

自动测试

错误收集（Sentry）

日志（ELK）

时序数据（InfluxDB、Grafana）

负载均衡（Nginx）

容器化（Docker）