构建可运维的后端应用

——从运维的视角看后端开发

关于我

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

DevOps

六点需求

- 保持简单并消除不一致
- 简单地观察和干预内部行为
- 出了问题要有办法发现
- 将一切操作自动化
- 尝试从错误中自动恢复
- 划清组件之间的界限

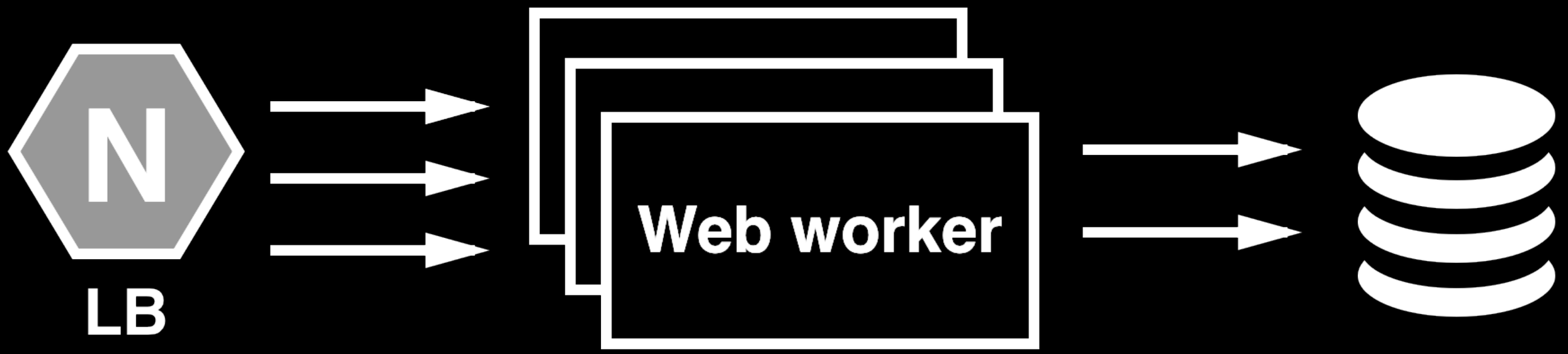
保持简单并消除不一致

“可靠性来自于对「最大程度的简化」的不断追求。”

–C.A.R. Hoare

保持简单并消除不一致

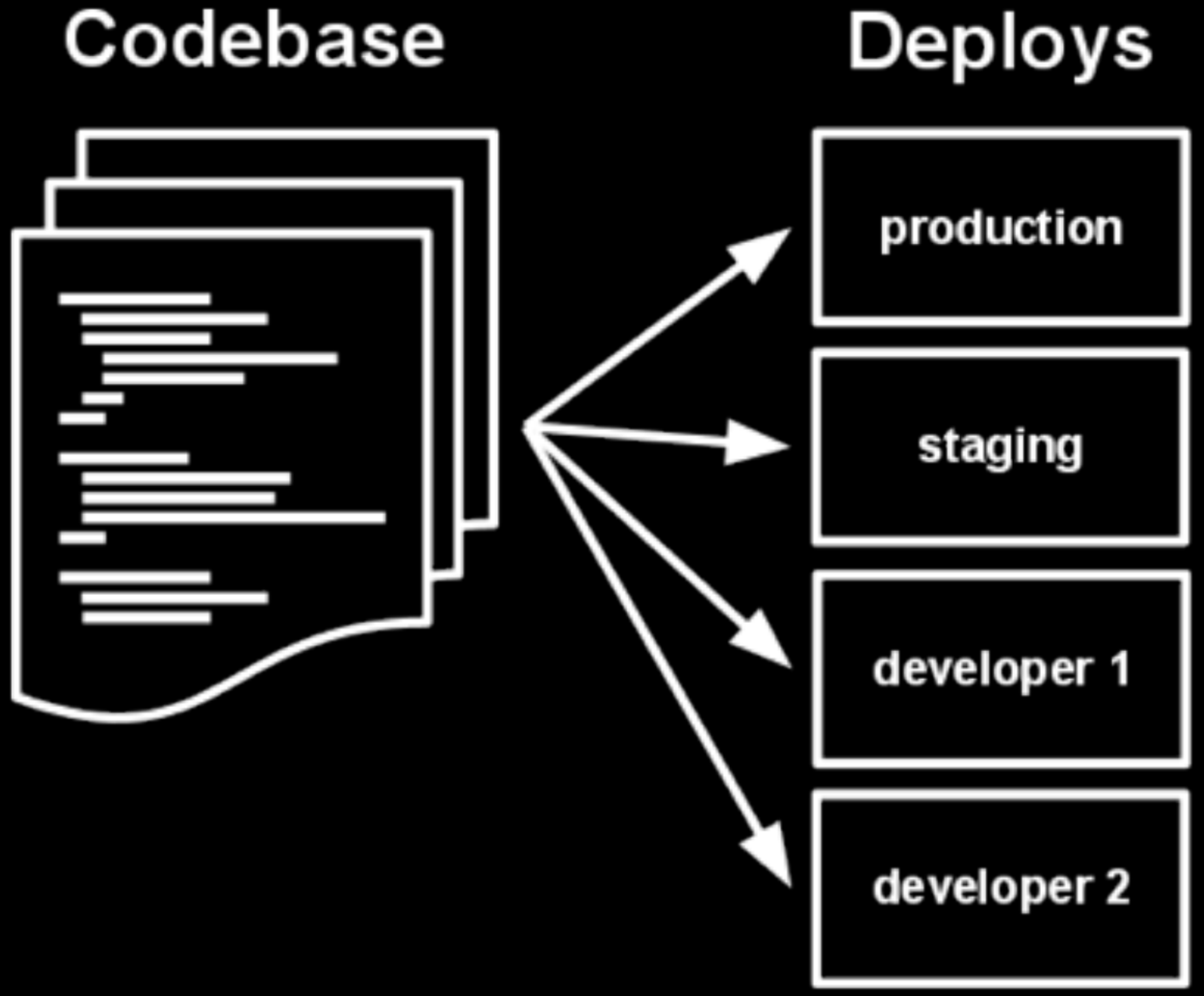
- 无状态的、事务的



stateless = scalable

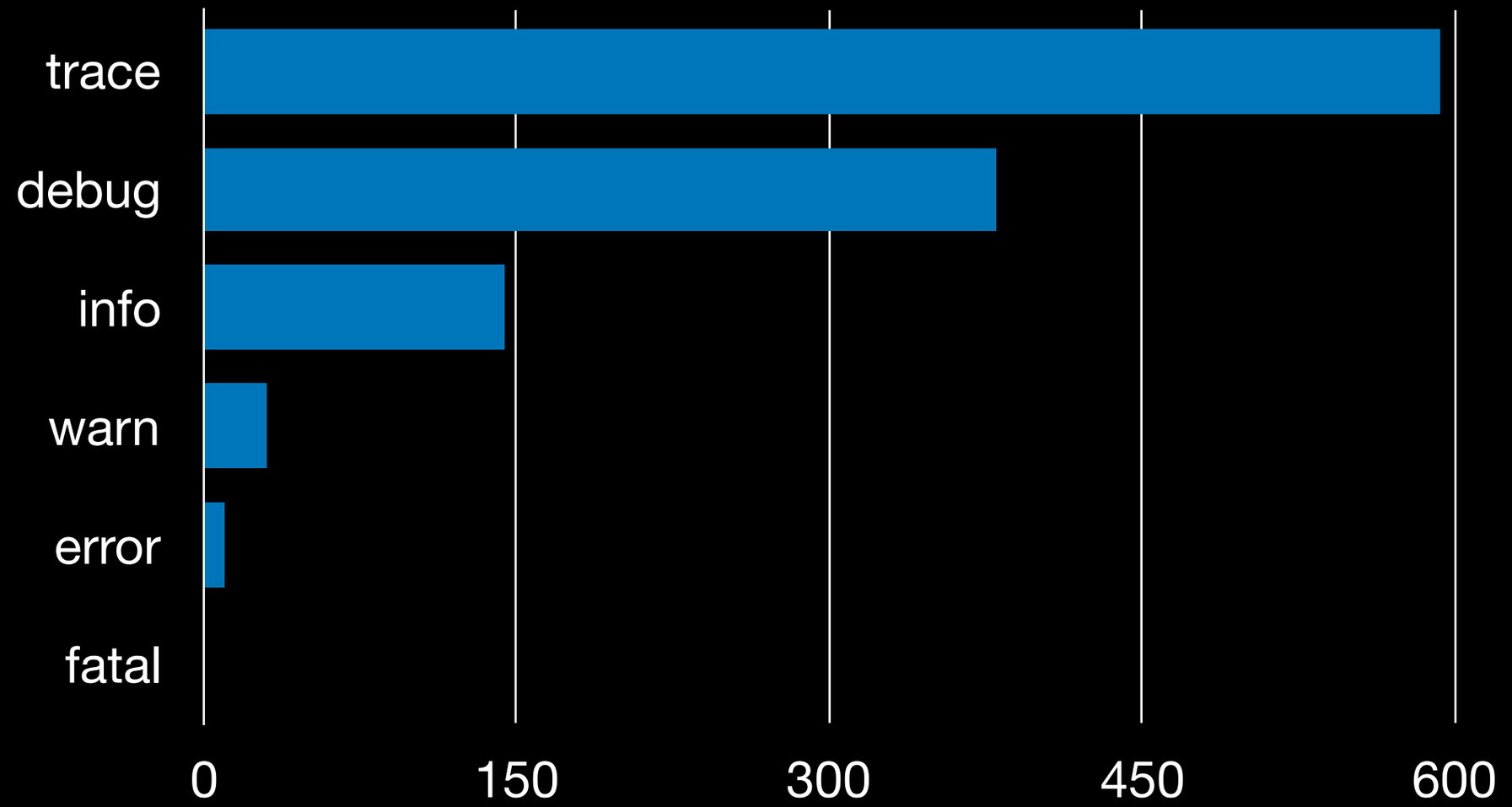
保持简单并消除不一致

- 单一代码库



简单地观察和干预内部行为

- 日志级别



简单地观察和干预内部行为

- 结构化日志（bunyan、winston）

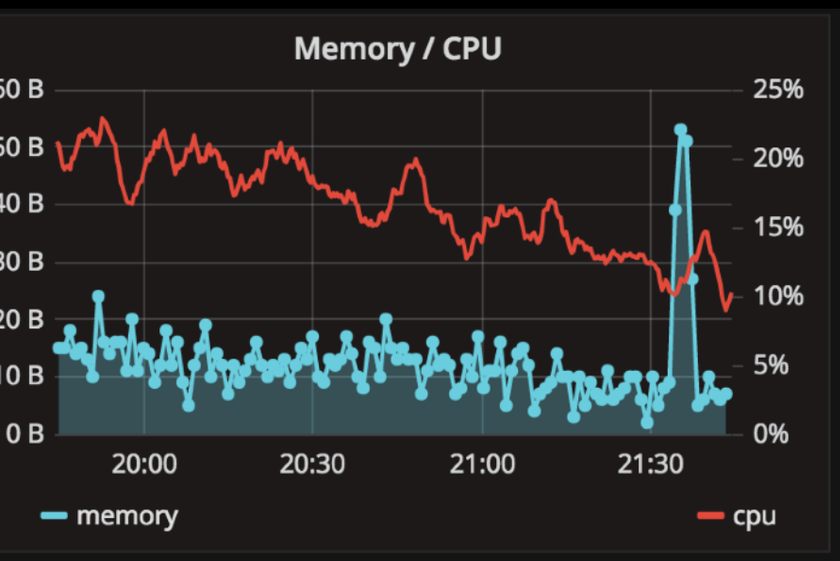
```
40,"username":"jysperm","msg":"login by password","time":"2013-01-04
50,"username":"jysperm","file_id": 123,"msg":"delete a file","time":
40,"username":"someone","file_id": 124,"msg":"upload a file","time":
50,"username":"someone""msg":"update password","time":"2013-01-04T18
```

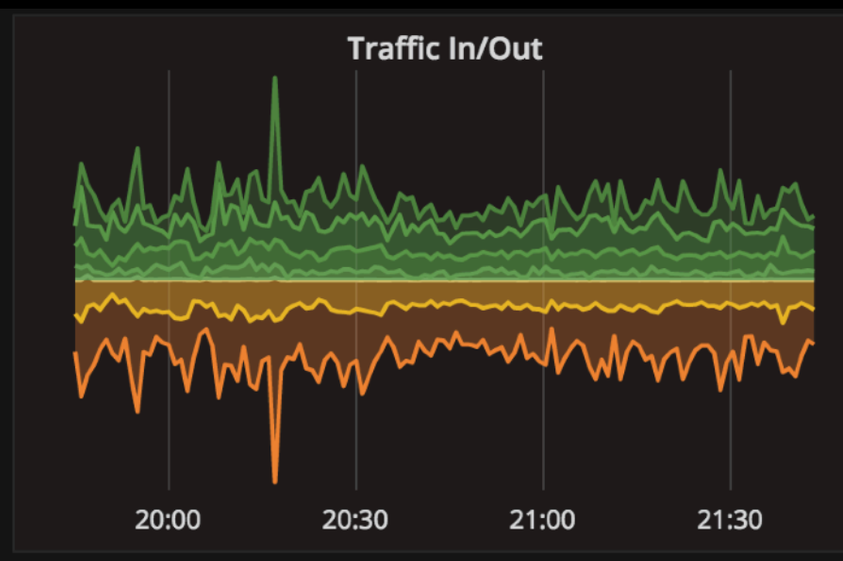
```
cat log | jq -c 'select(.username == "jysperm")'
```

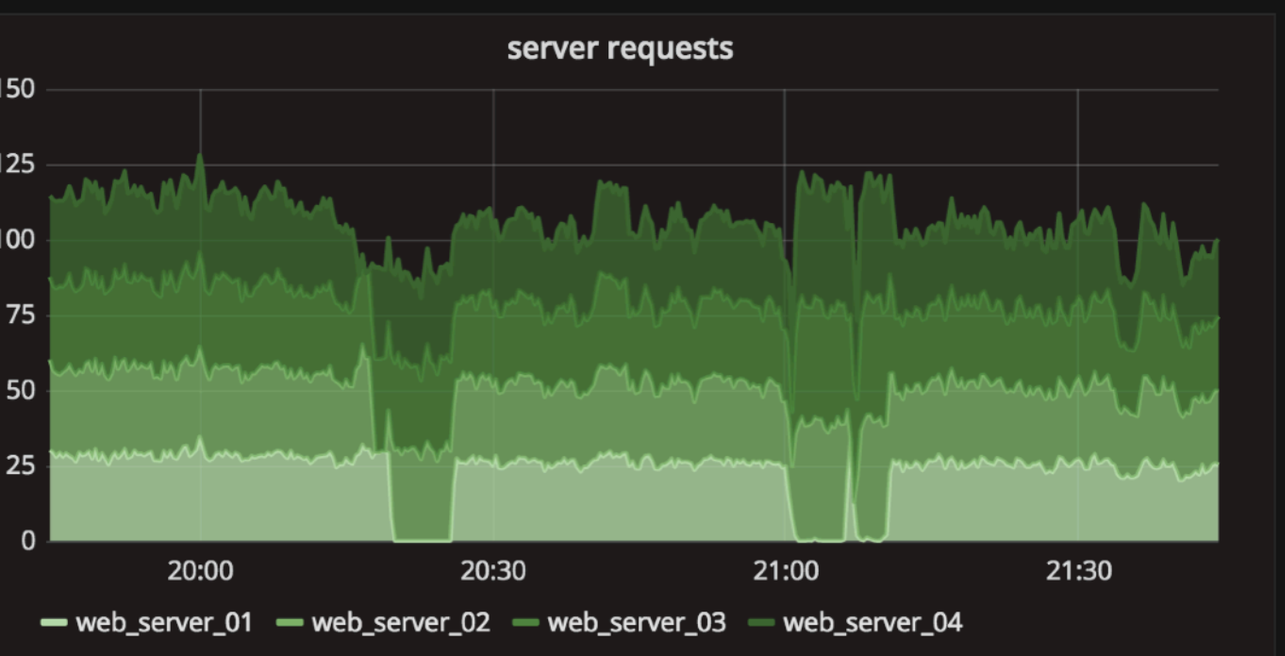
```
30,"username":"jysperm","msg":"login by password","time":"2013-
40,"username":"jysperm","file_id":123,"msg":"delete a file","t
```

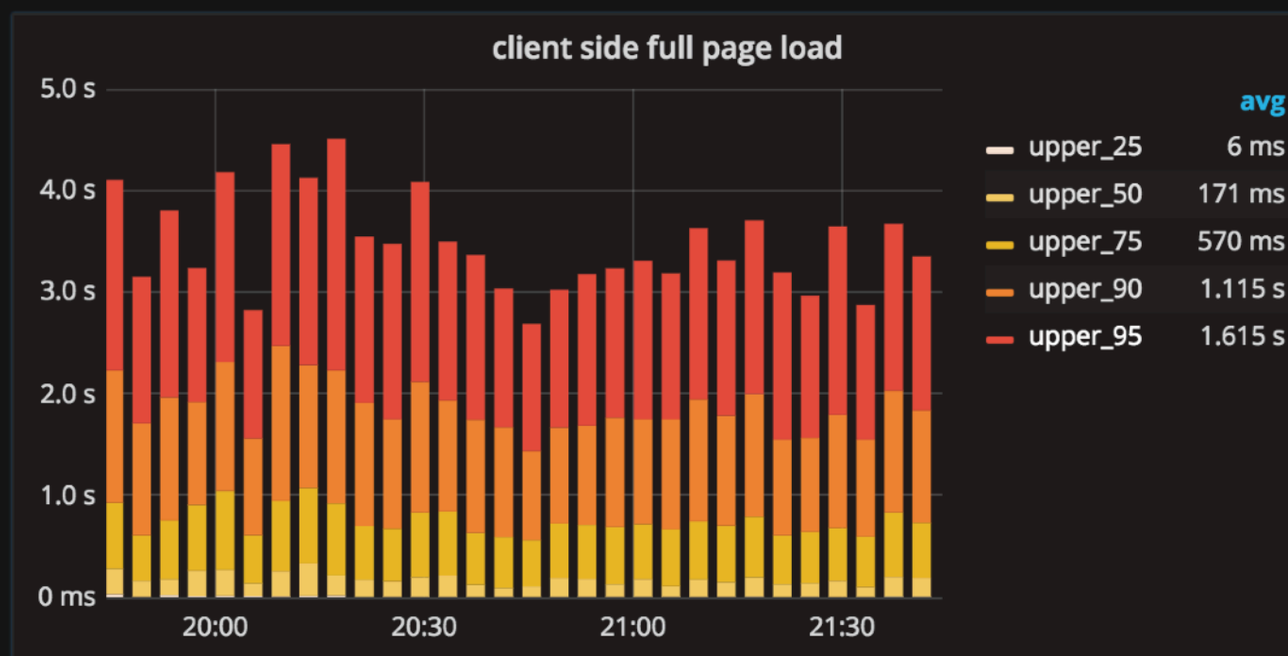
简单地观察和干预内部行为

- 时序数据和图表（Grafana）









简单地观察和干预内部行为

- 时序数据（InfluxDB）

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

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

简单地观察和干预内部行为

- 通过环境变量管理配置

```
// config.js
export.MYSQL_URI = process.env.MYSQL_URI || 'mysql://root@localhost'
export.REDIS_URI = process.env.REDIS_URI || 'redis://localhost:6379'
export.LOG_LEVEL = process.env.LOG_LEVEL || 'debug'
export.LISTEN = process.env.LISTEN || 3000

// server.js
const {LISTEN} = require('./config')

server.listen(LISTEN)
```

简单地观察和干预内部行为

```
// supervisor
[program:myapp]
user = web
command = npm start
directory = /home/web/myapp
autorestart = true
environment =
  MYSQL_URI='mysql://myapp:password@mysql01',
  REDIS_URI='redis://redis-web:6380'

// docker
docker run -e MYSQL_URI='mysql://myapp:password@mysql01' \
           -e REDIS_URI='redis://redis-web:6380' myapp

// shell
export MYSQL_URI='mysql://myapp:password@mysql01'
export REDIS_URI='redis://redis-web:6380'
export LOG_LEVEL='trace'
npm start
```

出了问题要有办法发现

- 健康检查

```
app.get('/self-check', (req, res, next) => {
  Promise.all([
    mysql.query('SELECT * FROM users LIMIT 1'),
    redis.info(),
    fs.readFile('/storage/files/e4bbcf1e'),
    require('http://internal-rpc.service/api'),
    request('https://third-party.service/api')
  ]).then( () => {
    res.send('everything is well')
  }).catch( err => {
    res.status(500).send(err.message)
  })
})
```

出了问题要有办法发现

- 错误收集（Sentry）

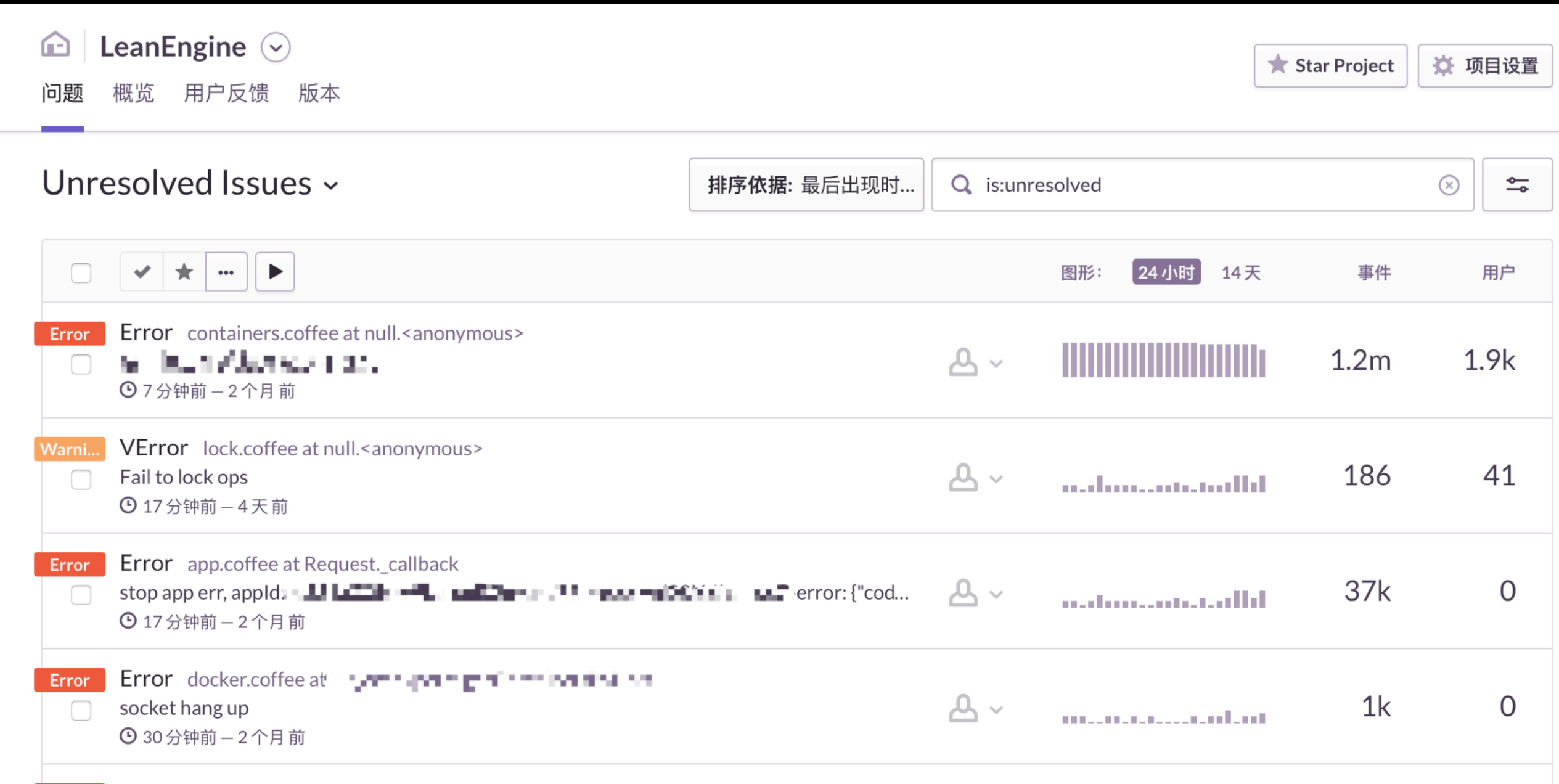
```
var Raven = require('raven');

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'}
});
```

出了问题要有办法发现

- 错误收集（Sentry）



将一切操作自动化

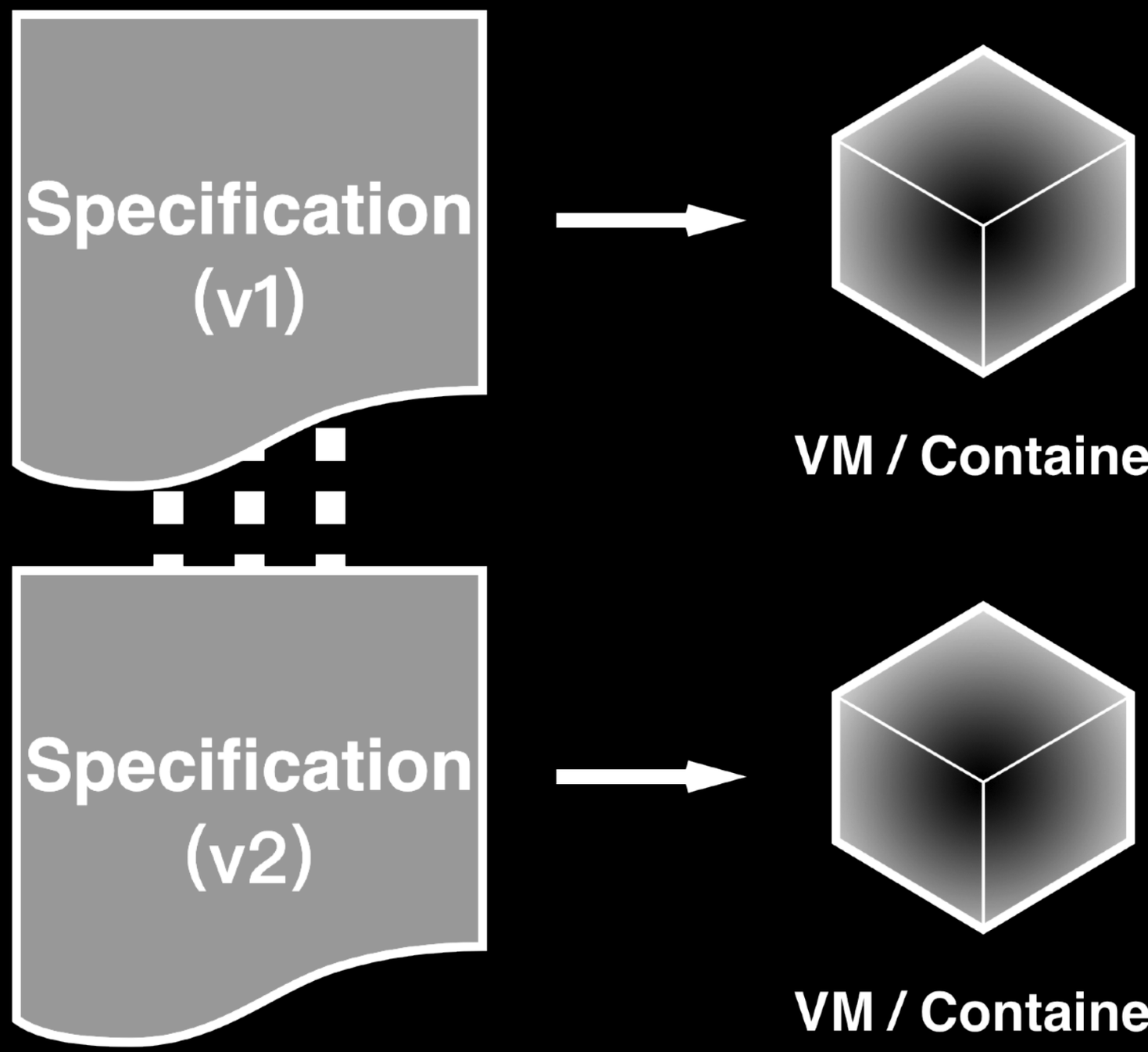
Why

- 保证一致
- 节省时间
- 可重复
- 可复用

How

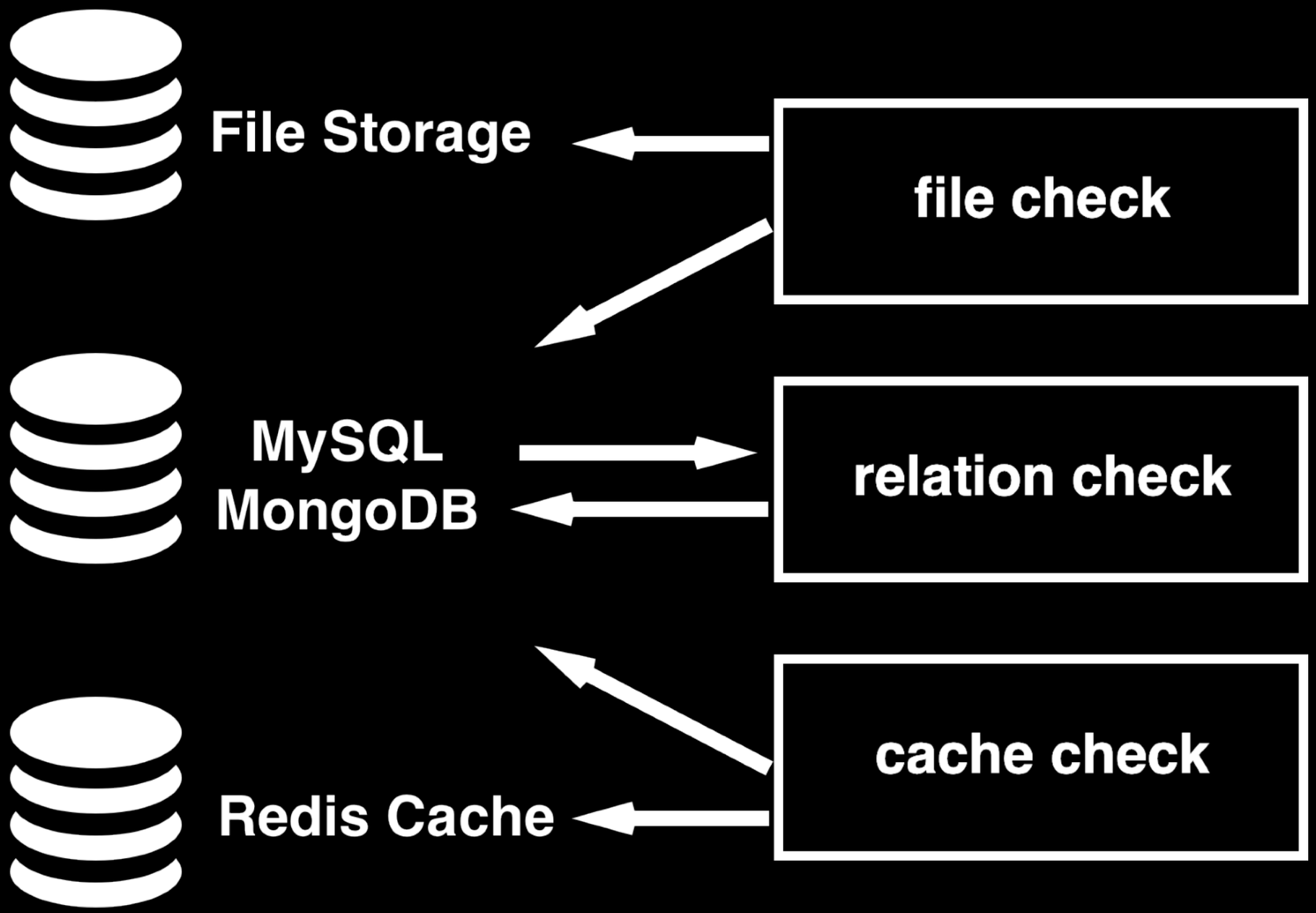
- 依赖（package.json）
- 数据库变更（db-migrate）
- 系统环境（Dockerfile）
- 构建、测试和发布（CI）
- 其他脚本

Immutable infrastructure



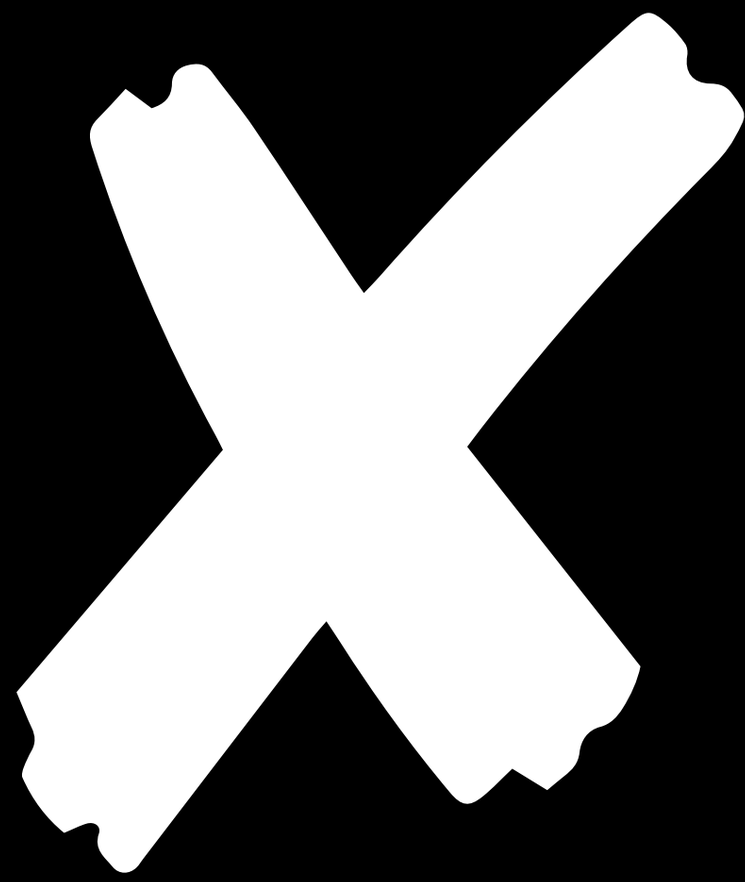
尝试从错误中自动恢复

- 数据一致性修复任务



划清组件之间的界限

- 守护进程（重启）
- 负载均衡（HTTPS、IP 限流）
- 日志（写入文件、轮转和删除）



小结

- 保持简单并消除不一致
- 简单地观察和干预内部行为
- 出了问题要有办法发现
- 将一切操作自动化
- 尝试从错误中自动恢复
- 划清组件之间的界限

Q & A

- SRE：Google 运维解密

- The Twelve-Factor App（Heroku）

LeanCloud

https://leancloud.cn/jobs/