Логирование и Docker. Distributed tracing.

Не забудь включить запись!

План

- Сбор логов с Docker контейнеров
- Distributed tracing
- Zipkin

Как собрать логи приложения в Docker?

- Docker logging drivers
- Sidecar контейнеры
- Писать из приложений в систему логирования

Logging drivers

- Собирают STDOUT, STDERR выводы контейнеров
- Добавляют метаинформацию
- Формируют сообщения для отправки в системы логирования
- Есть несколько встроенных, также подключаются в качестве плагинов
- По умолчанию используется Json-File driver

Logging drivers

Драйвер по умолчанию задается в файле:

/etc/docker/daemon.json

```
{
  "log-driver": "json-file"
}
```

Определение драйвера для отдельного контейнера **--log-driver**:

```
$ docker run --log-driver json-file alpine echo hello world
```

Json-file

- Являет драйвером по умолчанию
- Хранит логи о каждого контейнера в отдельном JSON файле
- Логи удаляются вместе с контейнером

```
$ tail -f $(docker inspect -f {{.LogPath}} dockerpuma_ui_1)

{"log":"* Min threads: 0, max threads: 16\n","stream":"stdout","time":"2017-11-
06T12:24:39.094400172Z"}
{"log":"* Environment: development\n","stream":"stdout","time":"2017-11-
06T12:24:39.094402562Z"}
{"log":"* Listening on tcp://0.0.0.0:9292\n","stream":"stdout","time":"2017-11-
06T12:24:43.311598353Z"}
{"log":"Use Ctrl-C to stop\n","stream":"stdout","time":"2017-11-
06T12:24:43.311883143Z"}
{"log":"89.106.198.71 - - [06/Nov/2017:12:25:02 +0000] \"GET / HTTP/1.1\" 200 -
0.0323\n”,"stream":"stderr","time":"2017-11-06T12:25:02.162951447Z"}
{"log":"89.106.198.71 - - [06/Nov/2017:12:25:02 +0000] \"GET /favicon.ico
HTTP/1.1\" 404 471 0.0013\n","stream":"stderr","time":"2017-11-
06T12:25:02.613096563Z"}
```

Journald-драйвер

- Пишет в общий системный журнал
- Если логов много - journald.conf требует тюнинга

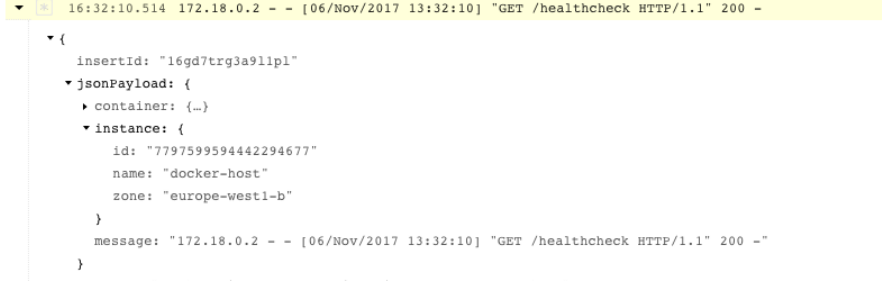
```
$ journalctl CONTAINER_NAME=dockerpuma_ui_1

Nov 06 13:14:41 docker-host dockerd[1264]: * Min threads: 0, max threads: 16
Nov 06 13:14:41 docker-host dockerd[1264]: * Environment: development
Nov 06 13:14:45 docker-host dockerd[1264]: * Listening on tcp://0.0.0.0:9292
Nov 06 13:14:45 docker-host dockerd[1264]: Use Ctrl-C to stop
Nov 06 13:15:23 docker-host dockerd[1264]: 89.106.198.71 - -
[06/Nov/2017:13:15:23 +0000] "GET / HTTP/1.1" 200 - 0.0378
```

GCP-драйвер

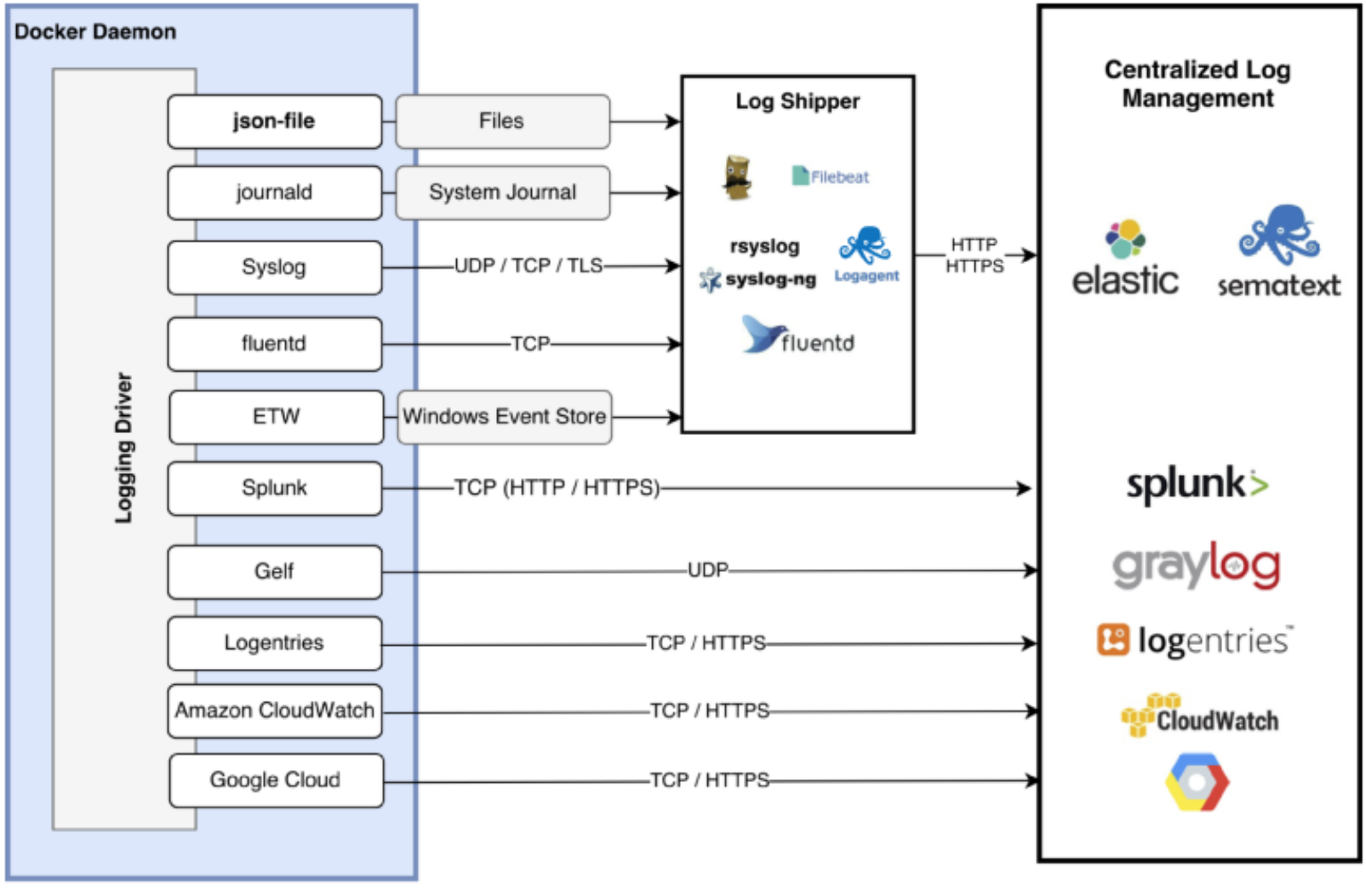
- Работает только в GCP
- Автоматически пишет логи в Google Stackdriver
- Не совместим с чтением с помощью docker logs

/etc/docker/daemon.json

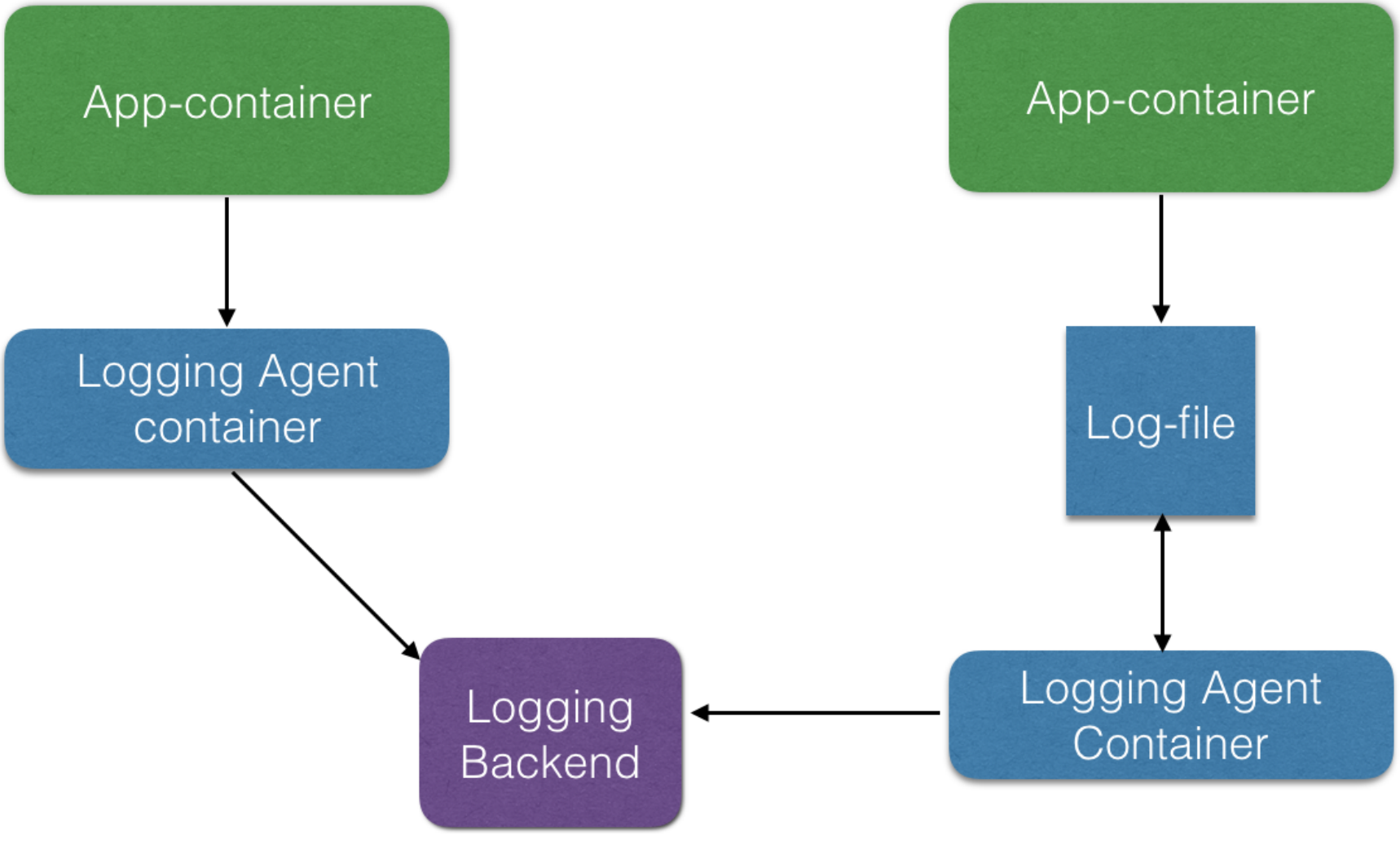
```
{
  "log-driver": "gcplogs",
  "log-opts": {
    "gcp-meta-name": "docker-project-name"
  }
}
```



Logging driver



Sidecar контейнеры



Логирование приложения

1. STDOUT, STDERR
2. Если приложение не умеет писать в STDERR, STDOUT:

```
 ln -sf /dev/stdout /var/log/nginx/access.log \ &&
ln -sf /dev/stderr /var/log/nginx/error.log
```

3. Напрямую в систему логирования (ELK, Graylog, Splunk, etc)
4. в Bind-Mount Volume (осторожно, логов может быть много):

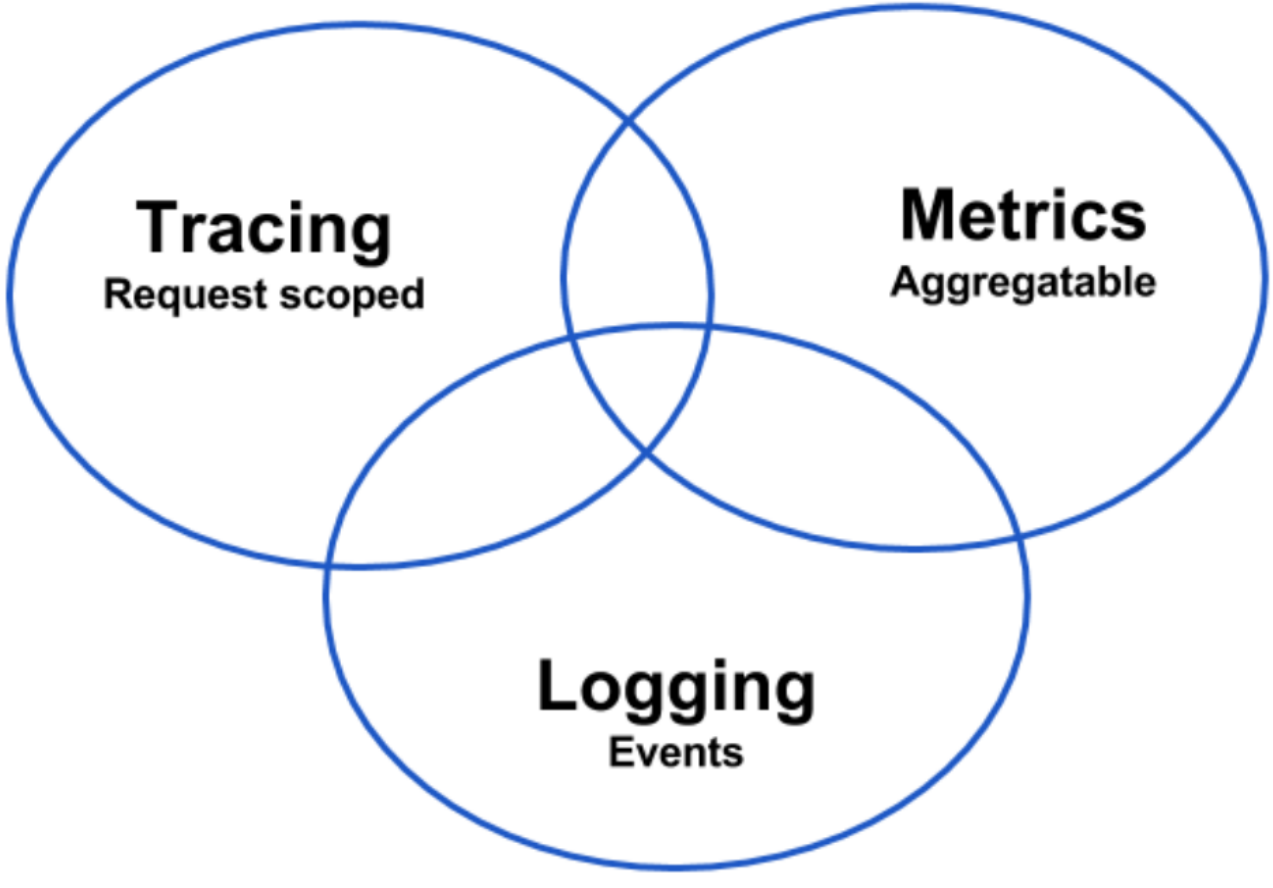
   docker run -v /var/log/nginx.log:/var/log/nginx.log
5. Не пишем логи в файлы внутри контейнера!

Distributed tracing

Three pillars of observability

- Metrics - количественные показатели по событиям, формирующие тренды
- Logging - журнал событий сервисов
- Distributed tracing - журнал событий с причино-следственной связью

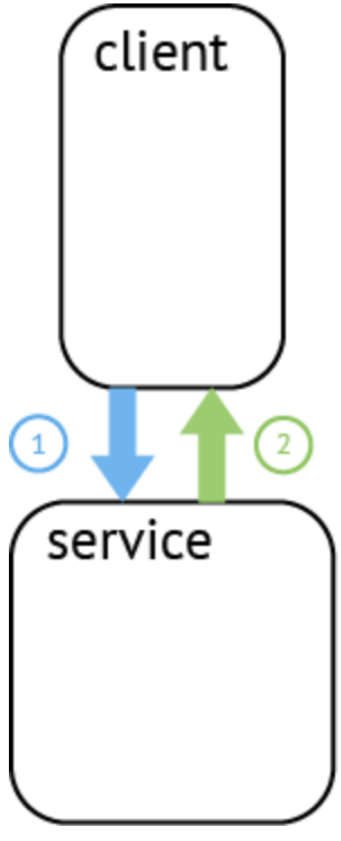
Разные задачи, разные инструменты



Источник:Peter Bourgon's post

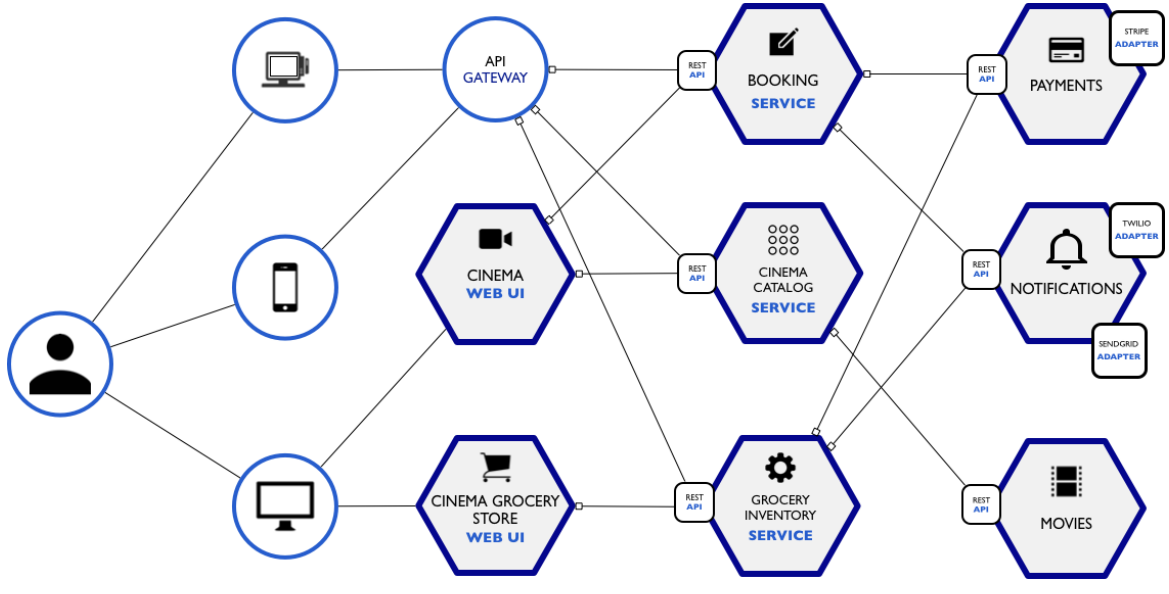
Trace

Описывает историю одного события (в распределенной системе)



Проблемы микросервисов

- Запрос от клиента проходит через несколько микросервисов
- Нет видимости, как работает система в целом
- Трудности дебага Latency



Distributed tracing

- Позволяет представить графы задержки запросов (traces) в реальном времени
- нализ графов помогает найти причины долгих запросов

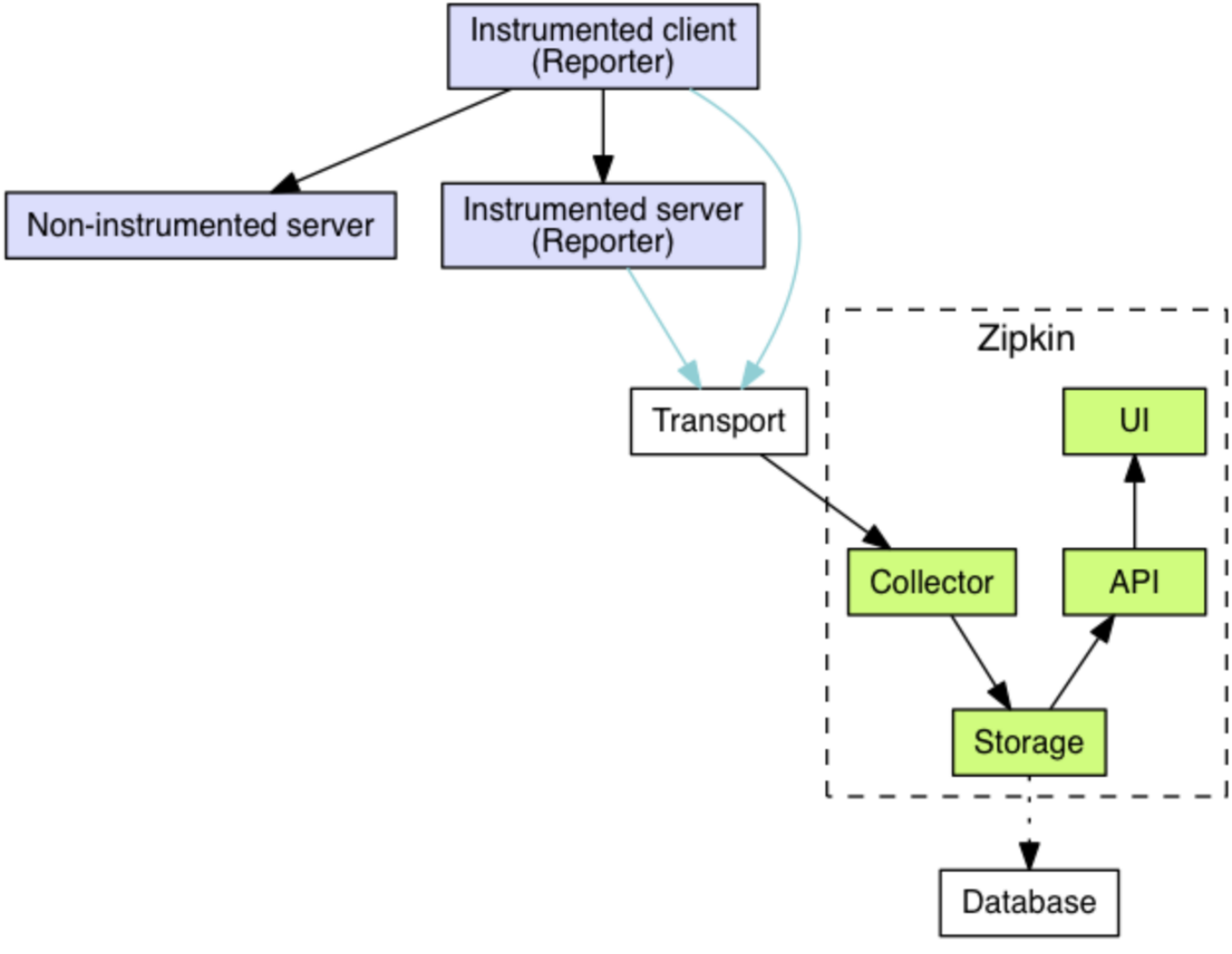
Системы трасировки

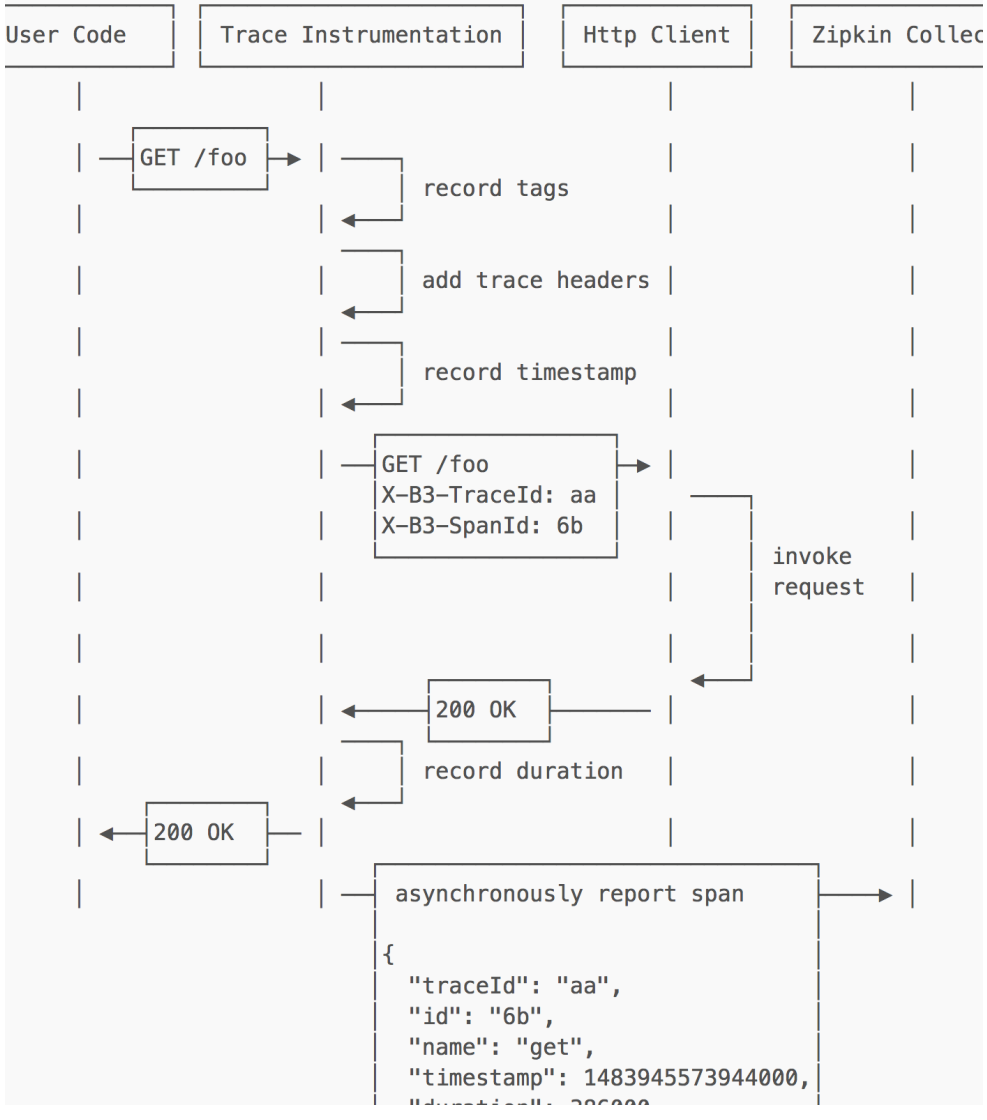
- Jaeger
- Appdash
- Zipkin
- Apache SkyWalking

Zipkin

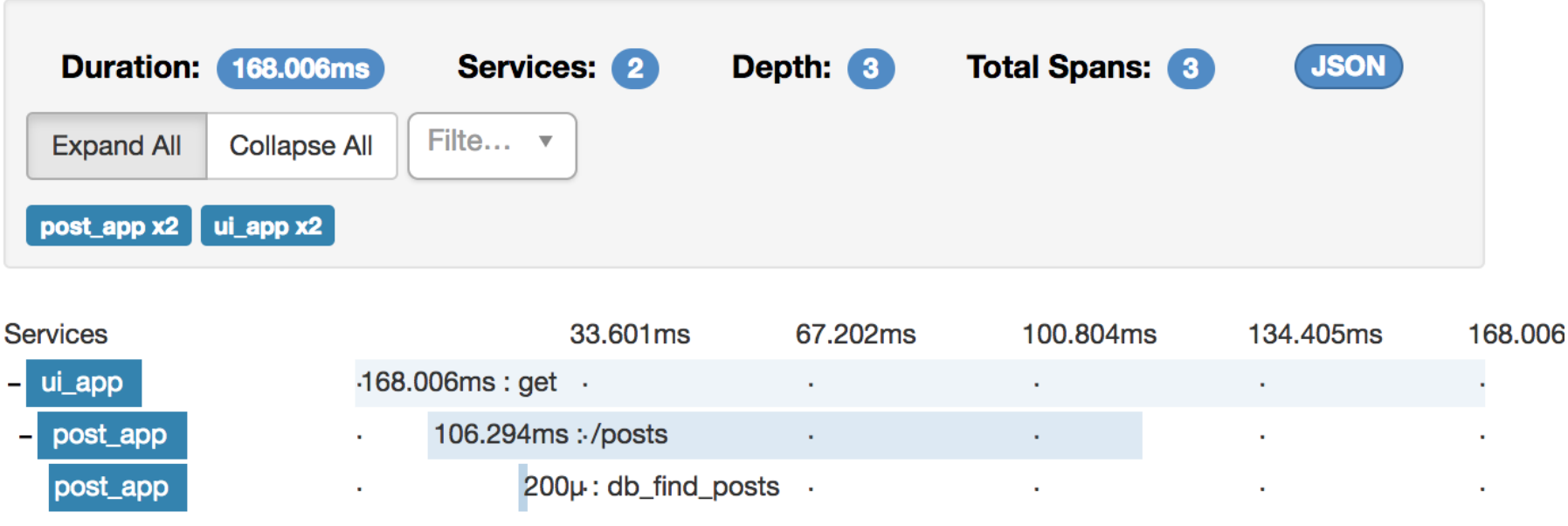
- 2012 год
- Разработан компанией Twitter
- Open source (https://github.com/openzipkin/zipkin)

Архитектура



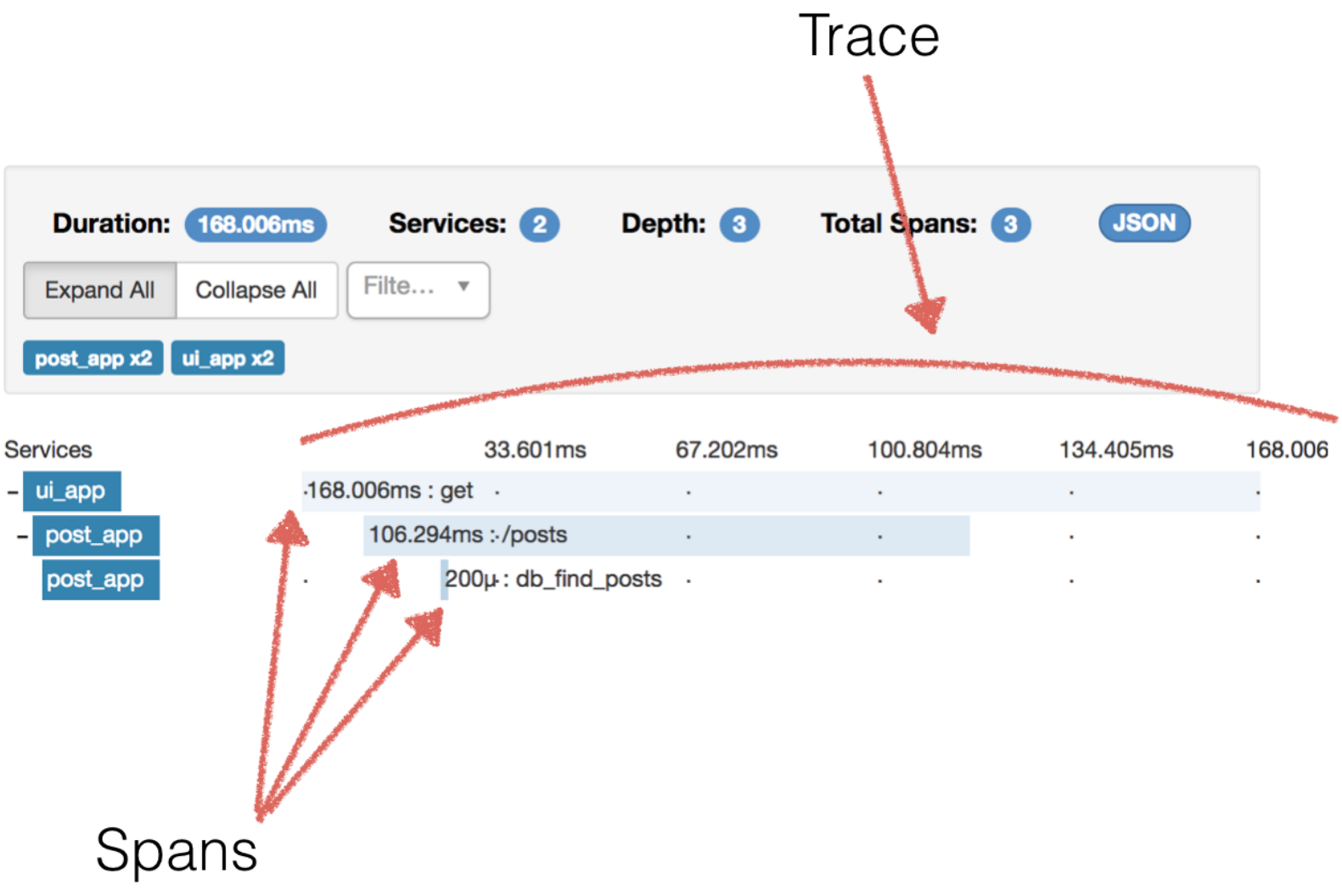

User Code
Trace Instrumentation
Http Client
Zipkin Collec
GET /foo
record tags
add trace headers
record timestamp
GET /foo
X-B3-TraceId: aa
X-B3-SpanId: 6b
invoke
request
200 OK
record duration
200 OK
asynchronously report span
{
"traceId": "aa",
"id": "6b",
"name": "get",
"timestamp": 1483945573944000,

Пример Zipkin



Основные понятия

- Span - одна завершившаяся операция в рамках запроса, содержит события и тэги
- Trace - граф задержки всего запроса, состоит из span-ов
- Tracer - библиотека в коде приложения, которые позволяют собирать и отправлять информацию о span-ах


Trace
Duration: 168.006ms
Services: 2
Depth: 3
Total Spans: 3
JSON
Expand All
Collapse All
Filte…
post_app x2
ui_app x2
Services
33.601ms
67.202ms
100.804ms
134.405ms
168.006
ui_app
168.006ms : get
post_app
106.294ms :/posts
post_app
200µ : db_find_posts
Spans