Инфраструктура распределенных приложений на Node.js

Stanislav Gumeniuk,
SEMrush

Инфраструктура распределенных приложений

1. Инструменты для запуска, мониторинга, etc;
2. Архитектурные решения: от монолита к микросервисам;
3. Инкапсуляция микросервисов;
4. Связывание микросервисов и сервисов;

HelloWorld / Node.js

```
"use strict";
var Koa = require('koa');
const app = new Koa();
app.use(ctx => {
    ctx.body = 'Hello Koa@2';
});

app.listen(3000);
```

Что мы хотим?

1. Разделение логов на потоки, управление потоками логов;
2. Мониторинг состояния приложения (загрузка CPU, память, статус, etc..);
3. Автоматический перезапуск при падении.

Запускаем HelloWorld

Базовый инструмент

```
$ node app.js
```

Запускаем HelloWorld

Базовый инструмент

```
$ node app.js
```

Умный инструмент

```
$ nodemon app.js
```

Запускаем HelloWorld

Базовый инструмент

```
$ node app.js
```

Умный инструмент

```
$ nodemon app.js
```

Продвинутый инструмент

```
$ pm2 start app.js
```


EPM2

pm2 start app.js

```
$ pm2 start app.js
```

```
[PM2] Starting app.js in fork_mode (1 instance)
[PM2] Done.
```

App name	id	mode	pid	status	restart	uptime	memory	watching
app	16	fork	86349	online	0	0s	15.738 MB	disabled

HelloWorld / Node.js / 2

```
var Koa = require('koa');
const app = new Koa();

app.use(ctx => {
        console.log('send responce');
        ctx.body = 'Hello Koa@2';
});
console.error('fake error');
app.listen(5000);
```

pm2 start app.js / pm2 logs app

```
$ pm2 logs app
[PM2] Tailing last 20 lines for [app] process

app-14 (err): fake error

app-14 (out): send responce
app-14 (out): send responce

[PM2] Streaming realtime logs for [app] process
```

pm2 info <app_name>

```
Describing process with id 2 - name index
┌──────────────────┬──────────────────────────────────────────────┐
│ status           │ online                                       │
│ name             │ index                                        │
│ restarts         │ 1                                            │
│ uptime           │ 3s                                           │
│ script path      │ /Users/vigo5190/proj/koajs-hello-world/index.js │
│ script args      │ N/A                                          │
│ error log path   │ /Users/vigo5190/.pm2/logs/index-error-2.log  │
│ out log path     │ /Users/vigo5190/.pm2/logs/index-out-2.log    │
│ pid path         │ /Users/vigo5190/.pm2/pids/index-2.pid        │
│ interpreter      │ node                                         │
│ interpreter args │ N/A                                          │
│ script id        │ 2                                            │
│ exec cwd         │ /Users/vigo5190/proj/koajs-hello-world       │
│ exec mode        │ fork_mode                                    │
│ node.js version  │ 6.1.0                                        │
│ watch & reload   │ ✘                                            │
│ unstable restarts │ 0                                           │
│ created at       │ 2016-05-17T18:35:14.936Z                     │
└──────────────────┴──────────────────────────────────────────────┘
```

Ещё возможности pm2

- Запуск несколько приложений одной командой;
- Deploy одной командой;
- Встроенная кластеризация(!)

ecosystem.json

```
{
  "apps": [
    {
      "name": "app",
      "script": "app.js",
      "env": {
        "NODE_ENV": "production"
      }
    }
  ]
}
```

pm2 startOrRestart ecosystem.json

```
$ pm2 startOrRestart ecosystem.json
```

```
[PM2][WARN] Applications app not running, starting...
[PM2] App [app] launched (1 instances)
```

App name	id	mode	pid	status	restart	uptime	memory	watching
app	15	fork	57722	online	0	0s	16.609 MB	disabled

ecosystem.json

```
{
  "apps": [
    {
      "name": "app",
      "script": "app.js",
      "env": {
        "NODE_ENV": "production"
      },
      "instances": "2"
    }
  ]
}
```

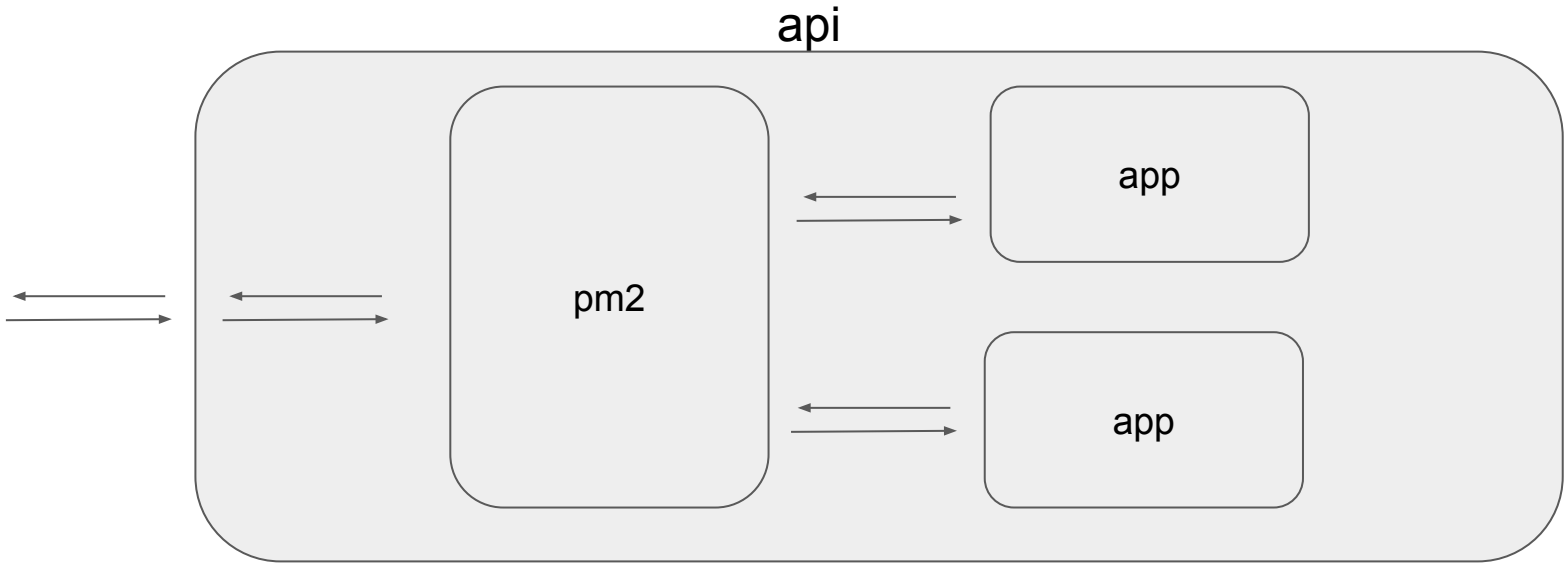
pm2 startOrRestart ecosystem.json

```
$ pm2 startOrRestart ecosystem.json
```

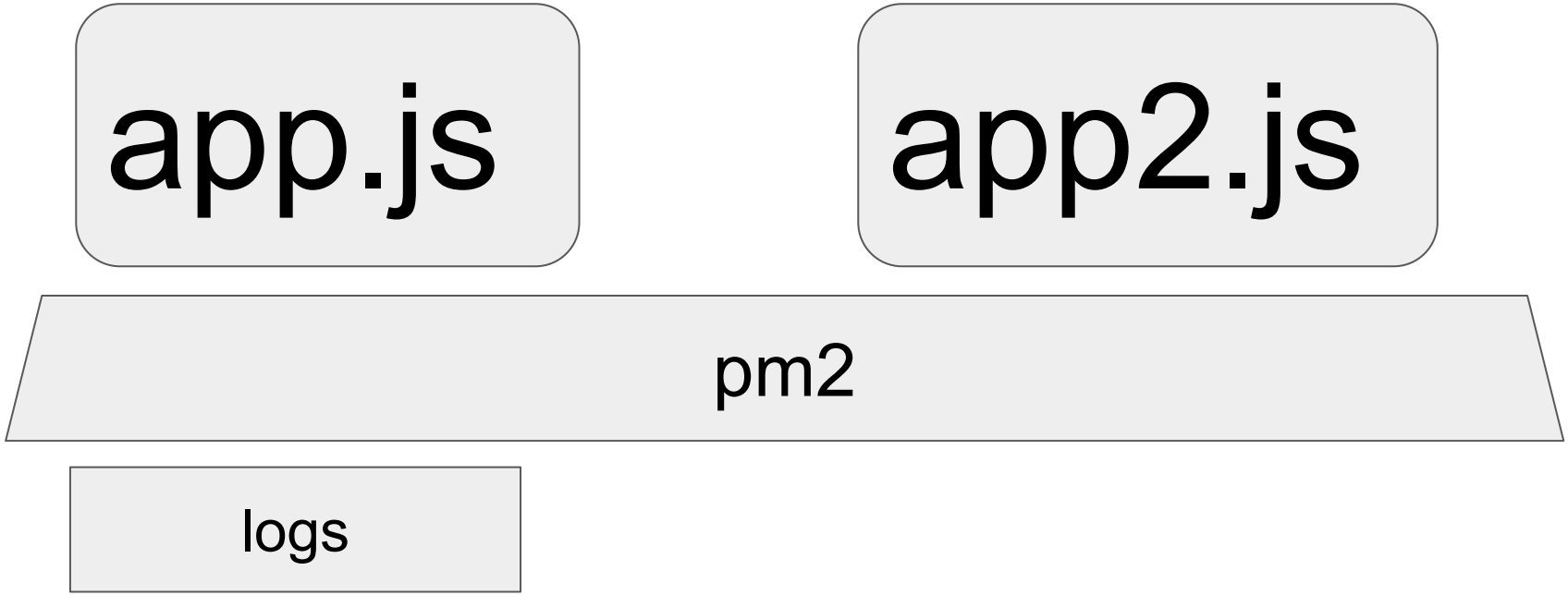
```
[PM2][WARN] Applications app not running, starting...
[PM2] App [app] launched (2 instances)
```

App name	id	mode	pid	status	restart	uptime	memory	watching
app	17	cluster	18900	online	0	0s	25.418 MB	disabled
app	18	cluster	18904	online	0	0s	19.484 MB	disabled

pm2 cluster



Инфраструктура с гипервизором



pm2:

1. Логи: разбиты по потокам и пишутся в файлы
2. Мониторинг: memory, restarts, etc..
3. Перезапуск при падении
4. Кластеризация из коробки
5. Deploy

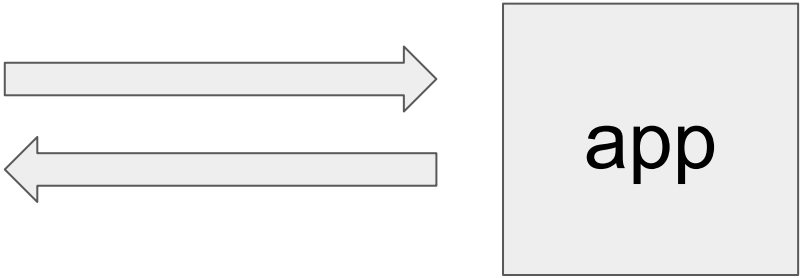
Архитектура

от монолита к микросервисам

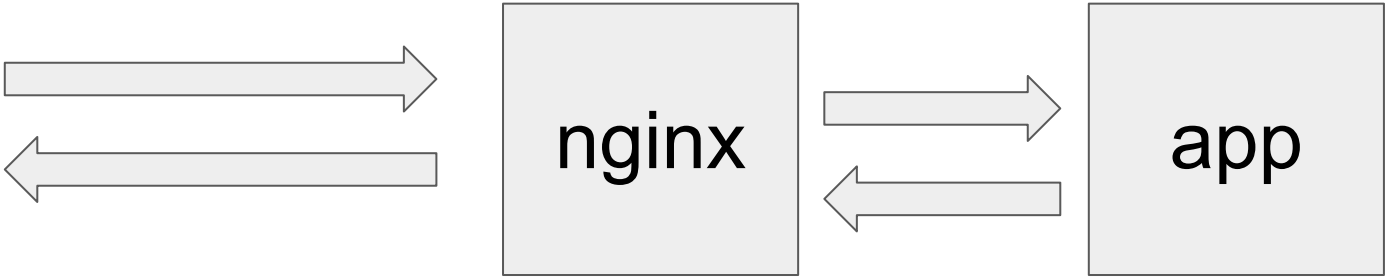
От монолита к микросервисам

1. Падение монолита - падение всей системы;
2. Падение микросервиса - падение элемента системы, при сохранении рабочей системы;
3. Распределение нагрузок;
4. Проще внедрять и заменять новые элементы в систему;
5. Мониторинг состояний каждого элемента системы.

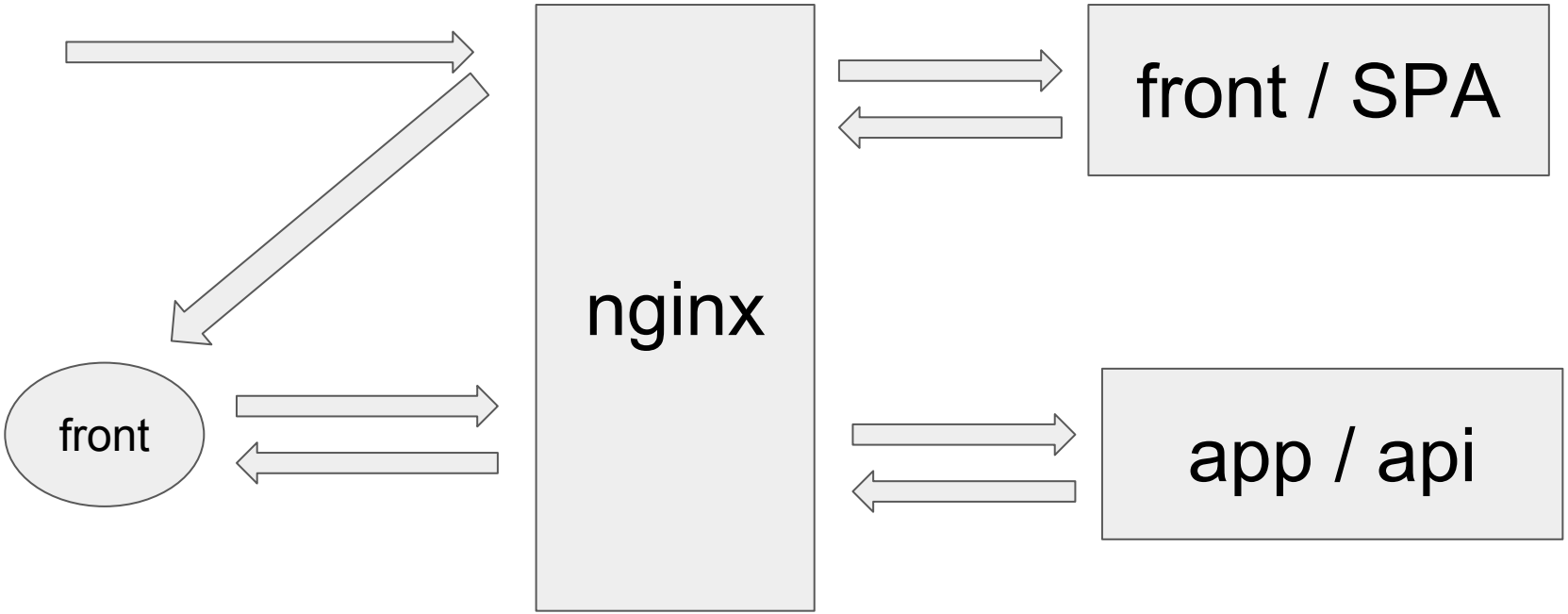
Взаимодействие с внешним миром



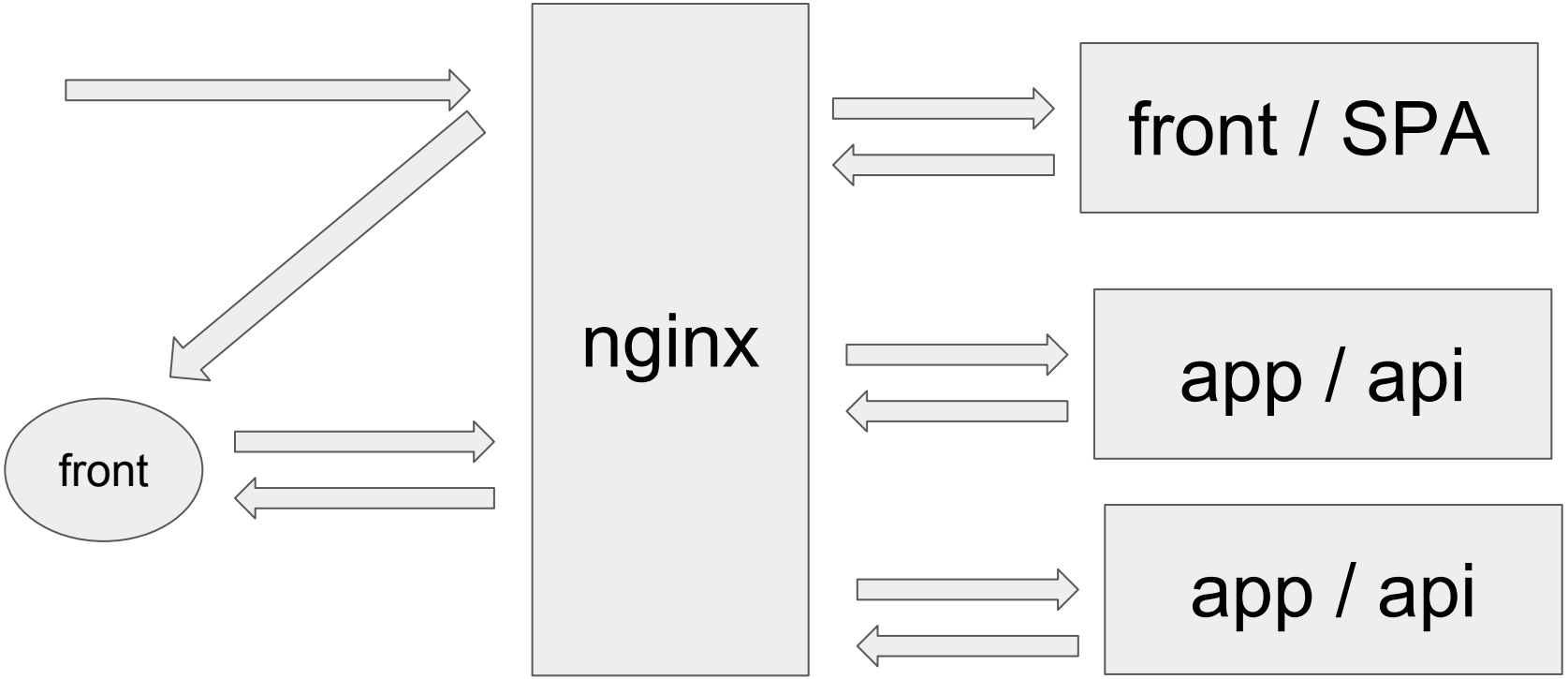
Взаимодействие с внешним миром



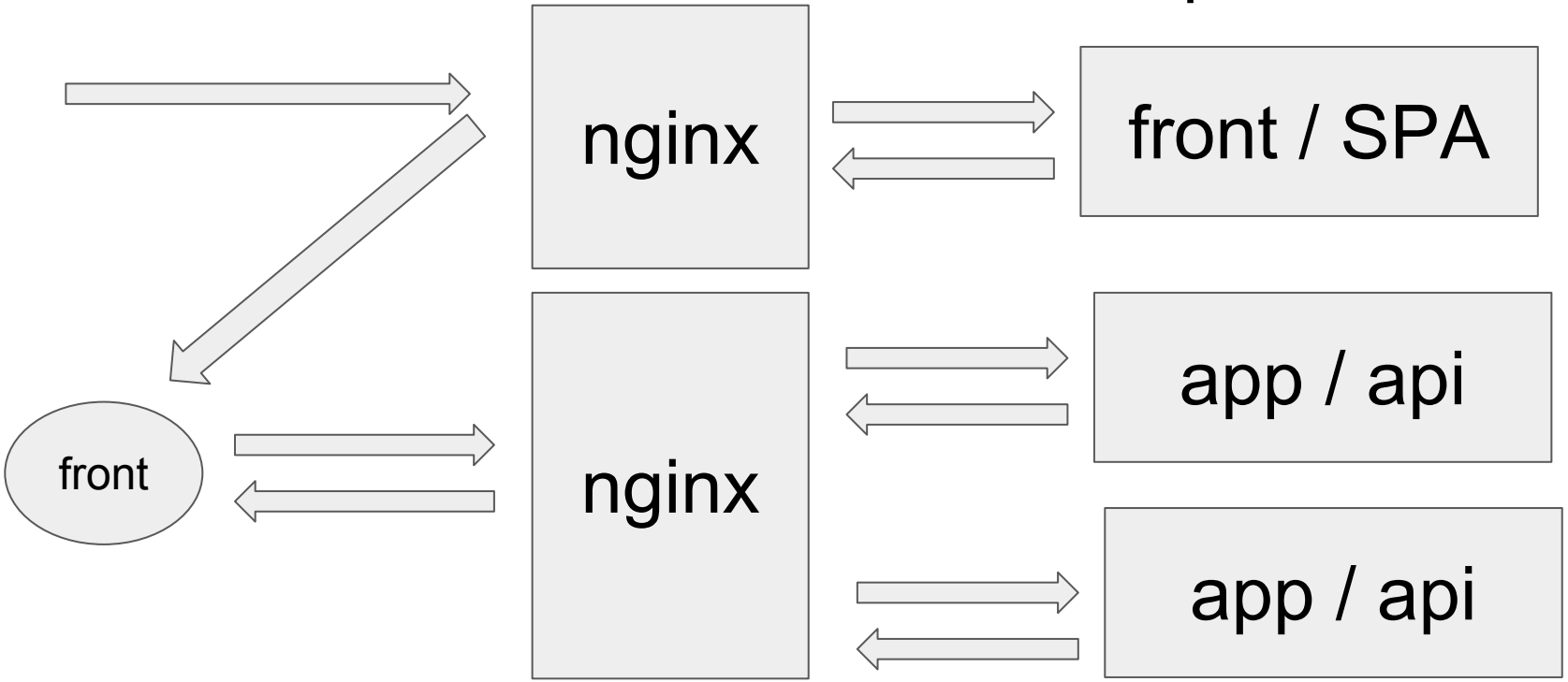
Взаимодействие с внешним миром



Взаимодействие с внешним миром

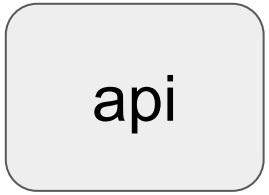


Взаимодействие с внешним миром

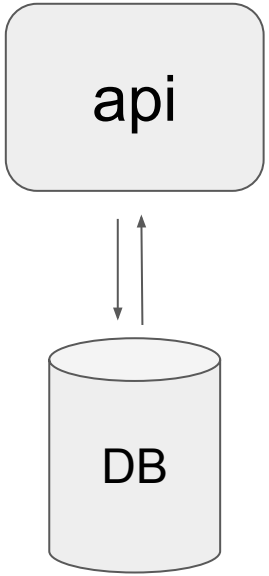


Архитектура

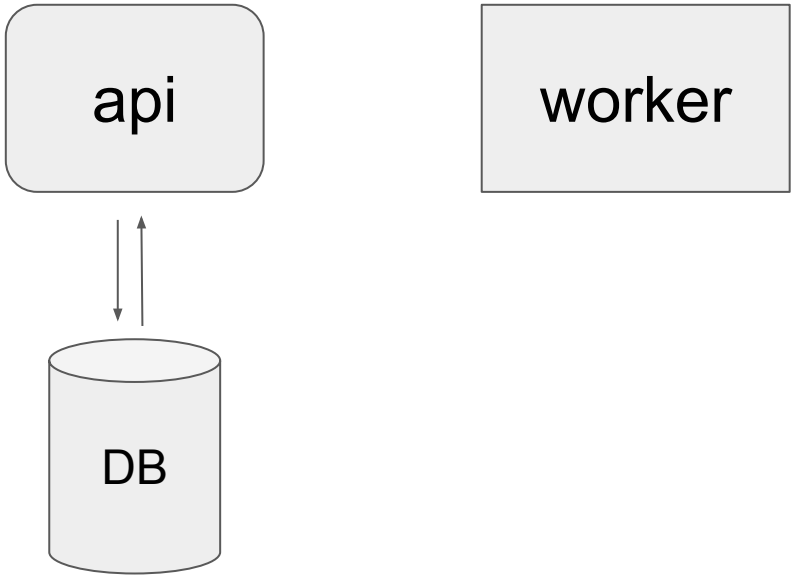
Архитектура



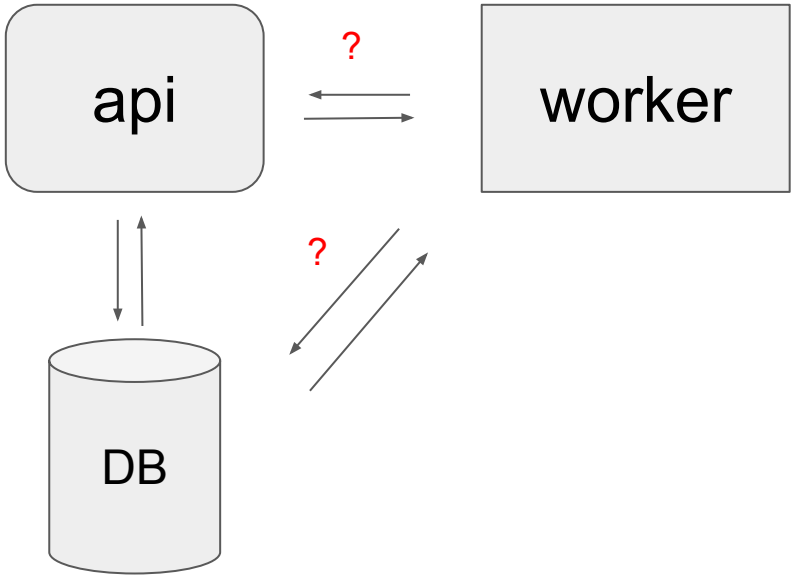
Архитектура



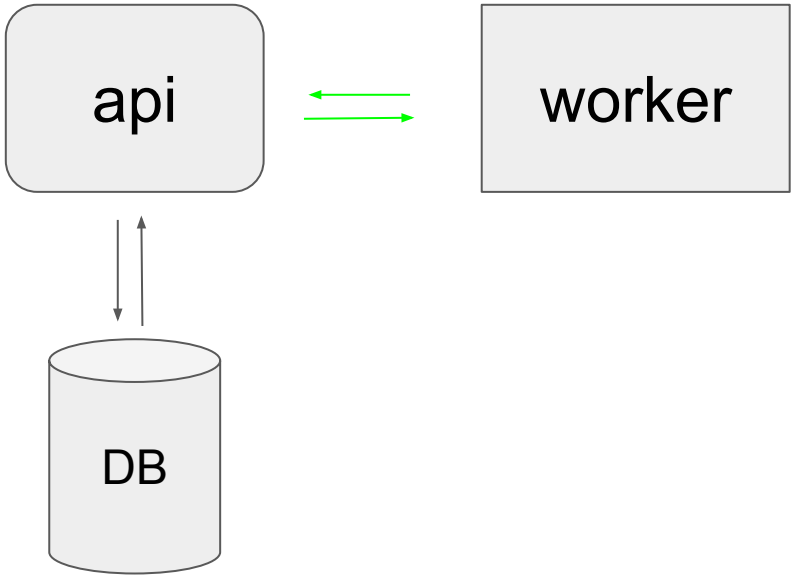
Архитектура



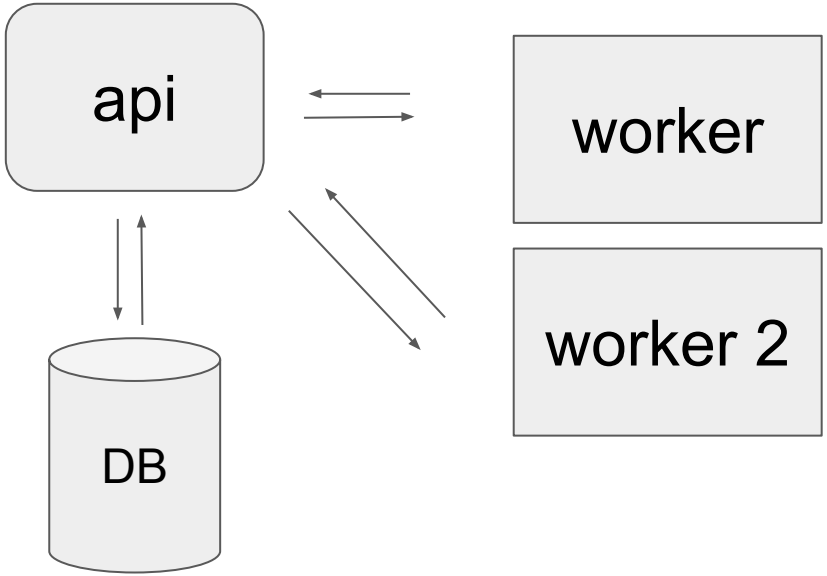
Архитектура



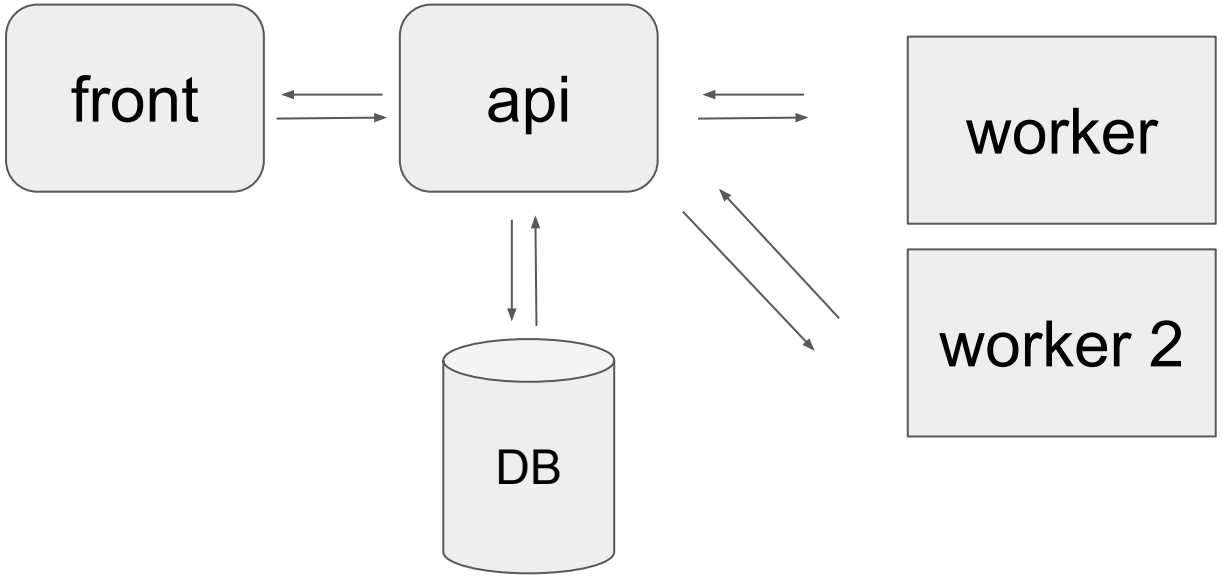
Архитектура



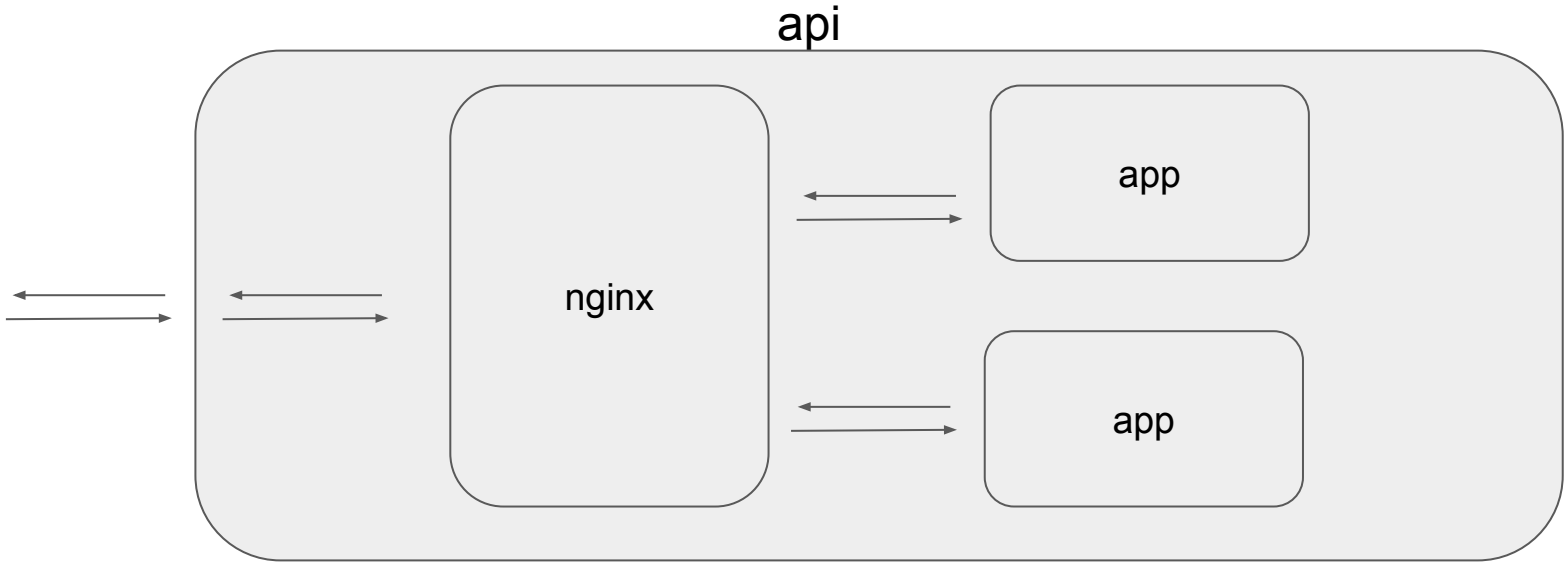
Архитектура



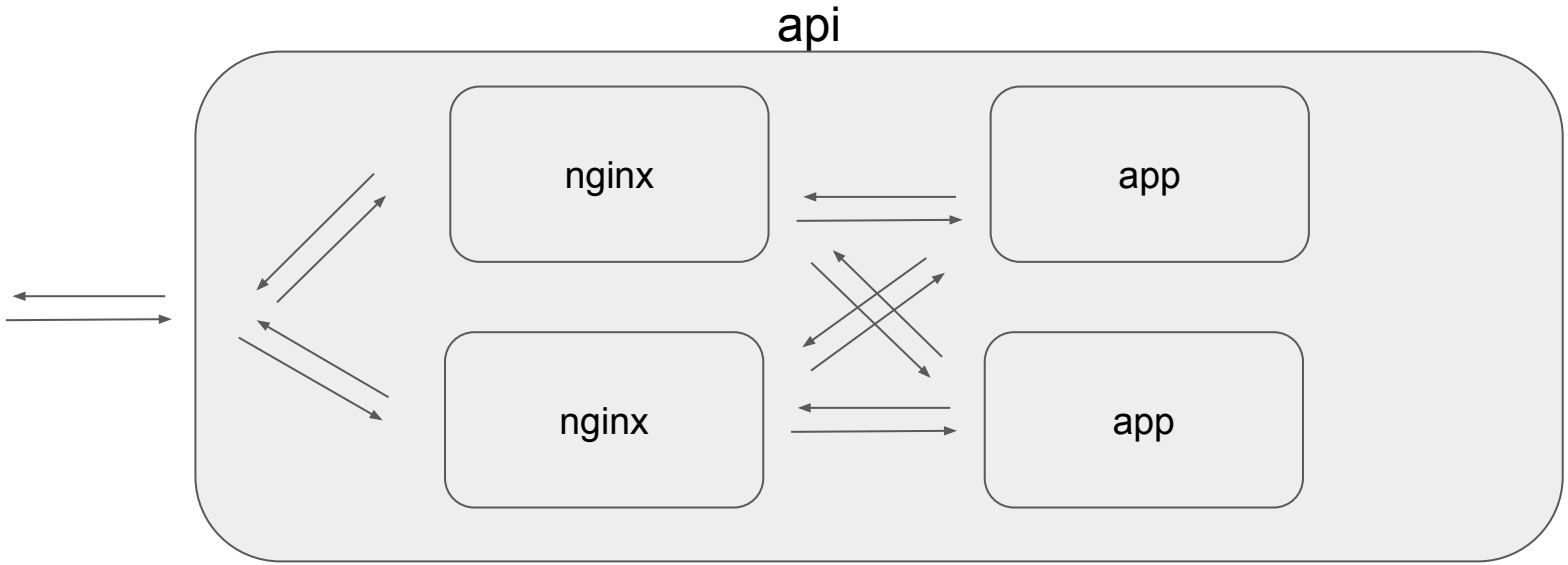
Архитектура



Архитектура API



Архитектура API




docker

Инкапсуляция микросервисов - Docker

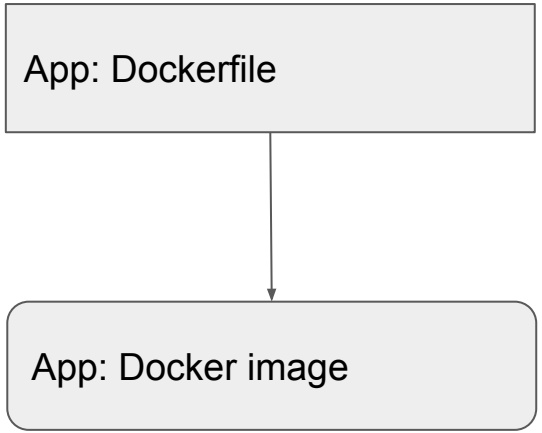
1. Все зависимости внутри;
2. Единое хранилище образов приложений;
3. Быстрая поставка приложения.

Docker lifeline : Dockerfile

App: Dockerfile

```
FROM node:4.2.3

ADD . /code

WORKDIR /code

RUN npm i

CMD node app.js

EXPOSE 5000
```

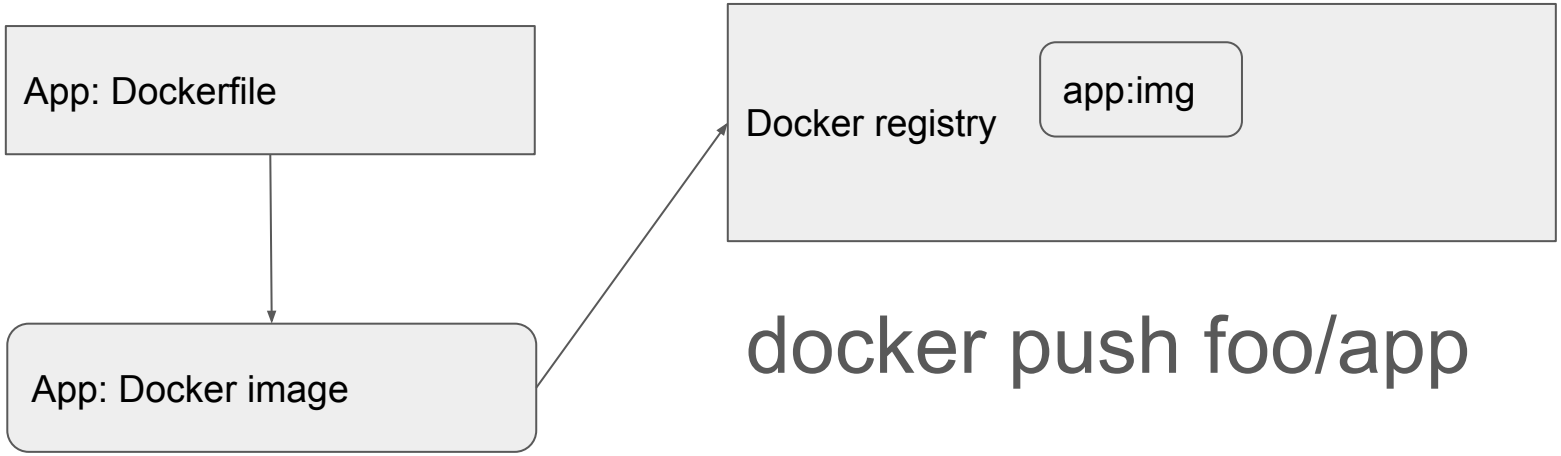
Docker lifeline : docker build img



docker build -t foo/app .

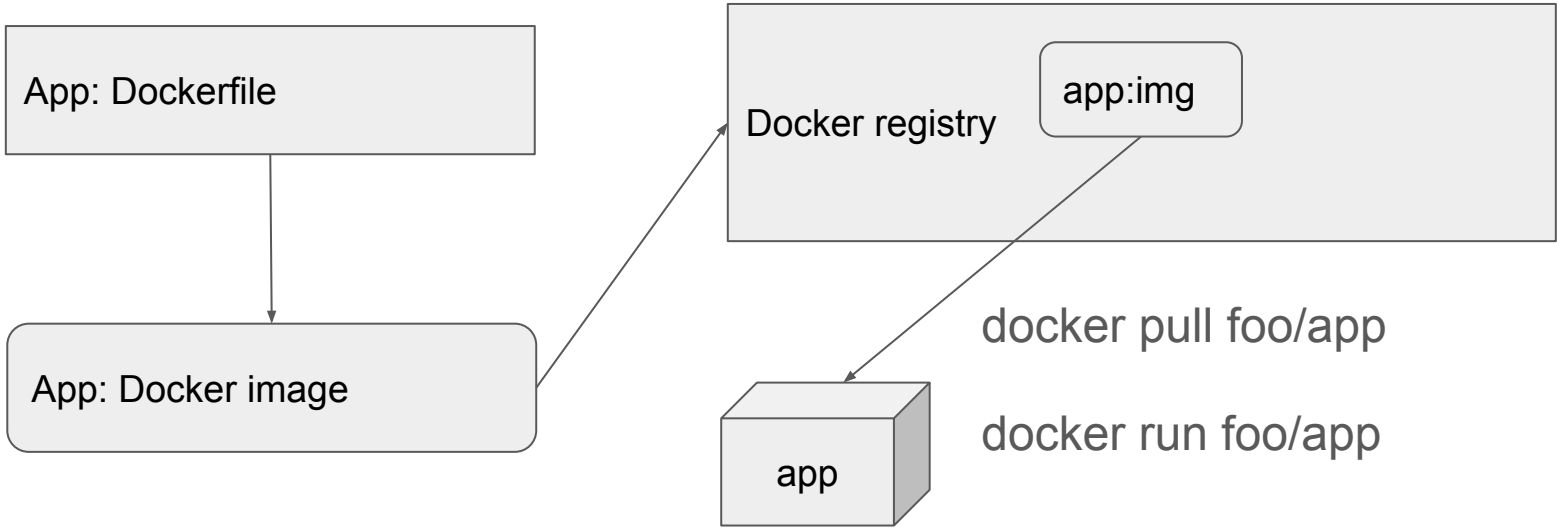
docker build -t foo/app:1.0 .

Docker lifeline : push image

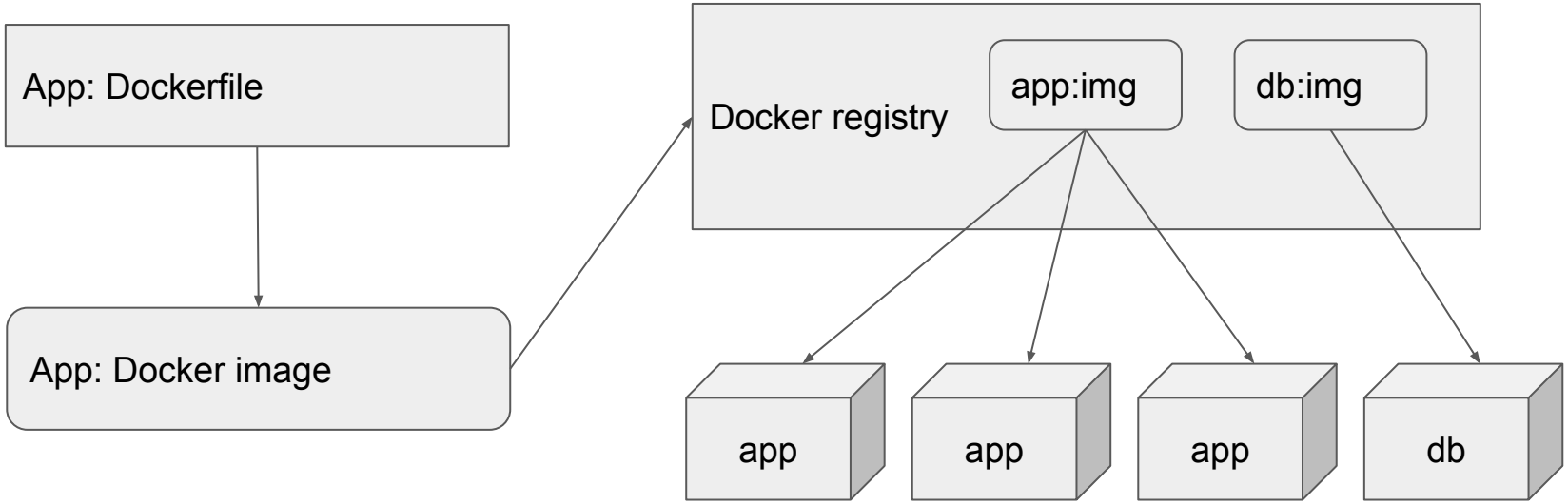


docker push foo/app

Docker lifeline



Docker lifeline



Dockerfile

```
FROM node:4.2.3
ADD . /code
WORKDIR /code
RUN npm i
CMD node app.js
EXPOSE 5000
```

Dockerfile

```
FROM node:4.2.3
ADD . /code
WORKDIR /code
RUN npm i
CMD node app.js
EXPOSE 5000
```

Dockerfile

```
FROM node:4.2.3
ADD . /code
WORKDIR /code
RUN npm i
CMD node app.js
EXPOSE 5000
```

Dockerfile

```
FROM node:4.2.3
ADD . /code
WORKDIR /code
RUN npm i
CMD node app.js
EXPOSE 5000
```

Dockerfile

```
FROM node:4.2.3
ADD . /code
WORKDIR /code
RUN npm i
CMD node app.js
EXPOSE 5000
```

Dockerfile + pm2

```
FROM node:4.2.3
ADD . /code
WORKDIR /code
RUN npm i && npm i -g pm2
CMD pm2 startOrRestart ecosystem.json --env docker --no-daemon
EXPOSE 5000
```

Что мы получаем от Docker

- Удобная поставка приложения:
  - запуск приложения одной командой;
  - возможность запуска множества копий приложения;
  - экономия времени системных администраторов;
- Инкапсуляция зависимостей:
  - все зависимости внутри
  - отсутствие конфликтов версионирования зависимостей
  - не оставляет лишних следов в хостовой системе
- Docker + pm2 = стабильность при “случайных” падениях приложения

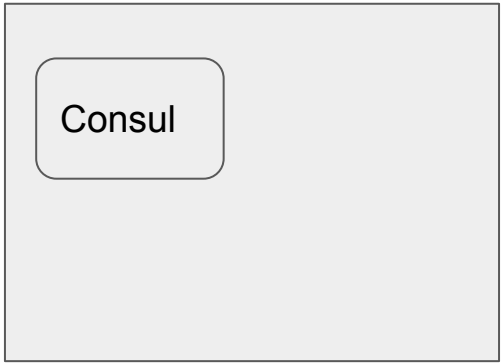
Consul

Связывание микросервисов

Решение проблем с помощью Consul

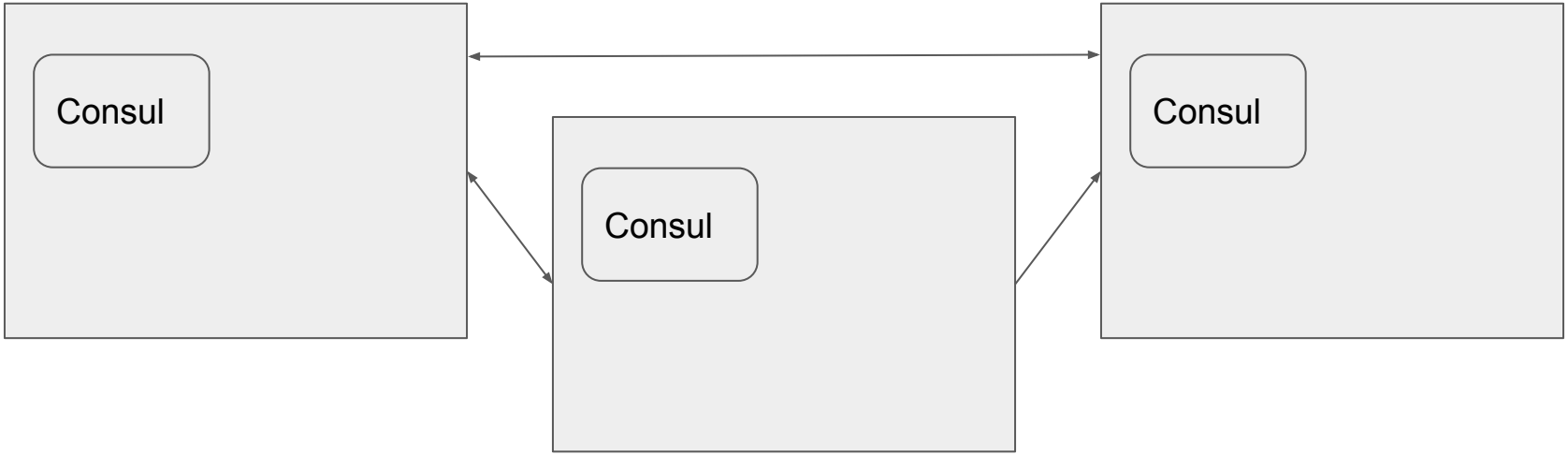
1. Мониторинг статуса сервисов;
2. Реестр всех запущенных сервисов;
3. Поиск сервисов;
4. Доставка конфигов “налету”;

Consul

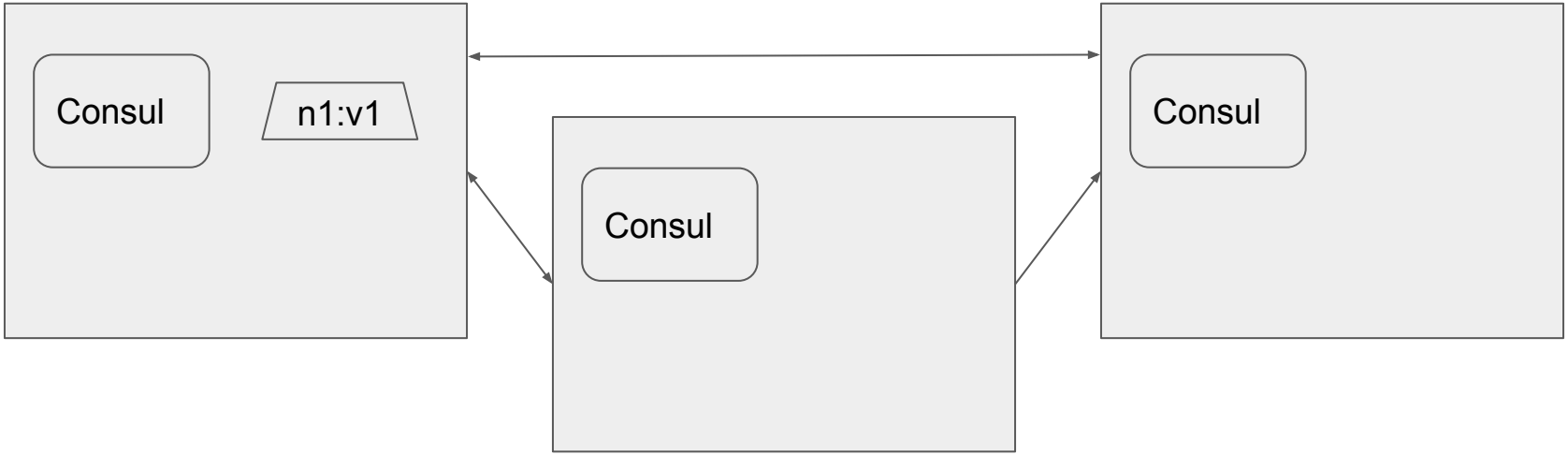


localhost:8300 - server rpc
localhost:8301 - serf lan
localhost:8302 - serf wan
localhost:8400 - cli rpc
localhost:8500 - http api
localhost:8600 - dns

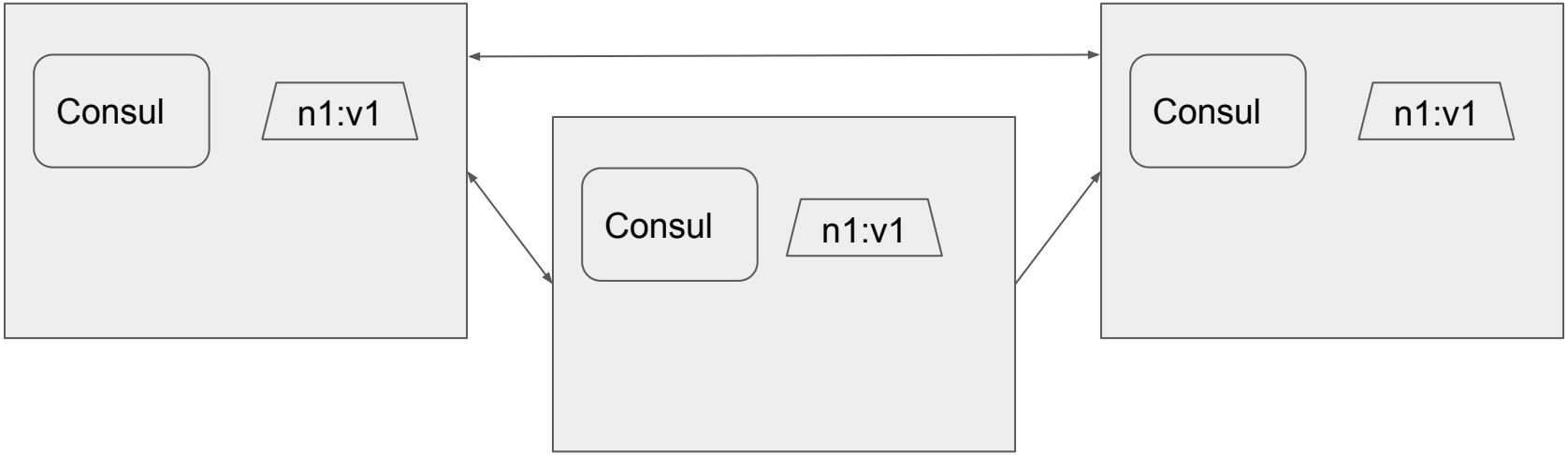
Consul



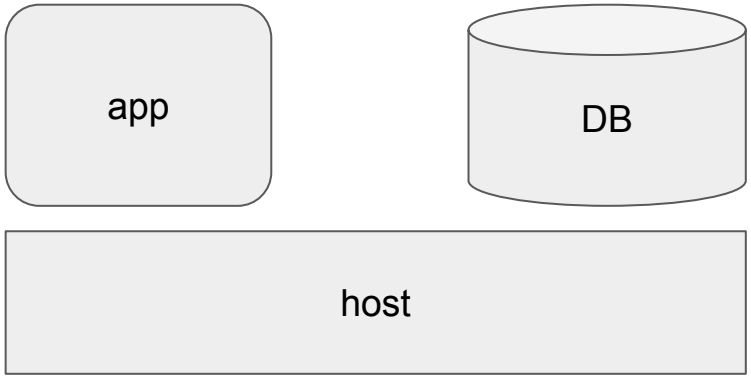
Consul : Key-Value



Consul : Key-Value

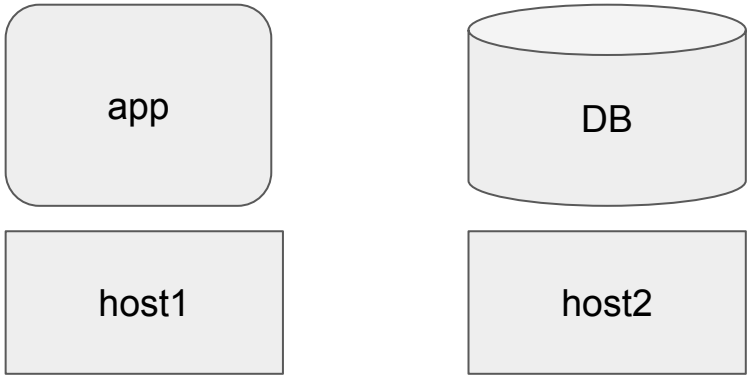


Consul : Service Discovery



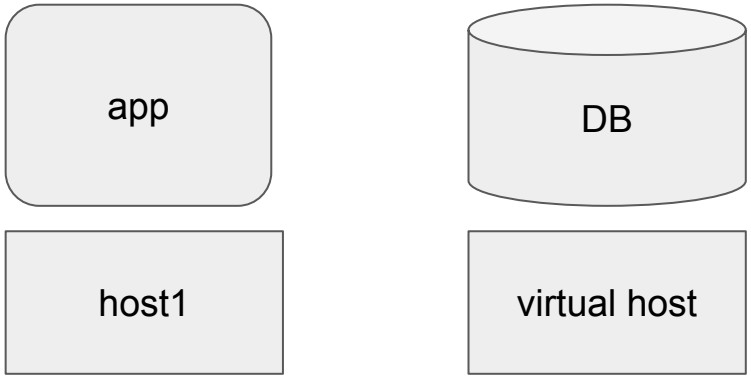
database: ‘localhost:3306’

Consul : Service Discovery



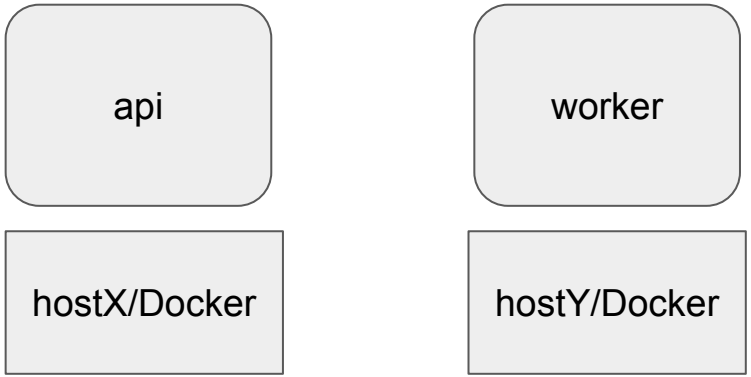
database: ‘host2:3306’

Consul : Service Discovery



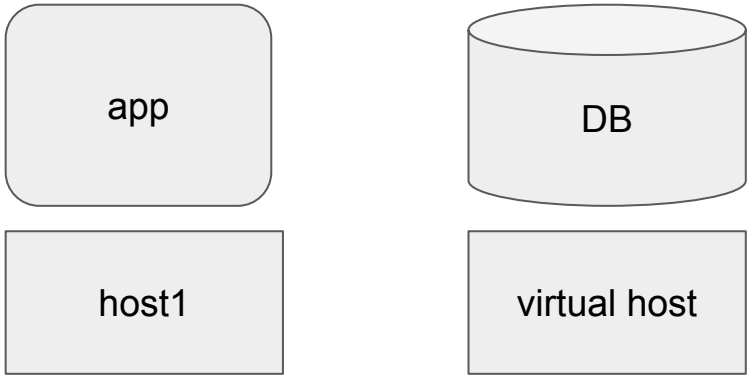
database: ‘???:3306’

Consul : Service Discovery



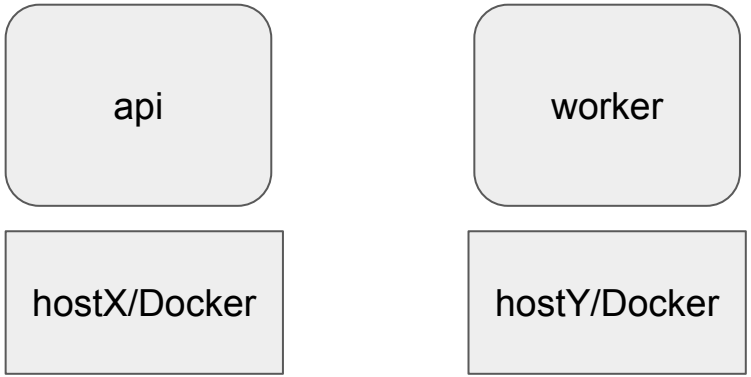
api: '???:???'

Consul : Service Discovery



database: ‘db.service.sd:3306’

Consul : Service Discovery



curl http://sd/service/api

```
{
    host: ‘api.service.sd’,
    port: ‘8888’
}
```

Установка Consul

1. Установить на каждый сервер

Установка Consul

1. Установить на каждый сервер
2. Использовать Docker!

Установка Consul

1. Установить на каждый сервер
2. Использовать Docker!

```
$ docker run --name consul -h $HOSTNAME  \
    -p 10.0.1.1:8300:8300 \
    -p 10.0.1.1:8301:8301 \
    -p 10.0.1.1:8301:8301/udp \
    -p 10.0.1.1:8302:8302 \
    -p 10.0.1.1:8302:8302/udp \
    -p 10.0.1.1:8400:8400 \a
    -p 10.0.1.1:8500:8500 \
    -p 172.17.42.1:53:53/udp \
    -d -v /mnt:/data \
    progrium/consul -server -advertise 10.0.1.1 -join 10.0.1.2
```

Добавляем поддержку Consul для pm2

$ pm2 install pm2-consul

Добавляем поддержку Consul для pm2

$ pm2 install pm2-consul

```
Module activated
┌─────────────┬─────────┬────────────┬────────┬─────────┬─────┬──────────┐
│ Module      │ version │ target PID │ status │ restart │ cpu │ memory   │
├─────────────┼─────────┼────────────┼────────┼─────────┼─────┼──────────┤
│ pm2-consul  │ 0.1.1   │ 79857      │ online │ 0       │ 0%  │ 50.172 MB│
└─────────────┴─────────┴────────────┴────────┴─────────┴─────┴──────────┘
```

Добавляем поддержку Consul для pm2

Добавляем поддержку Consul в Docker

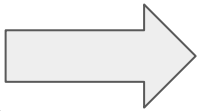
```
$ docker pull gliderlabs/registrator:latest

$ docker run -d \
    --name=registrator \
    --net=host \
    --volume=/var/run/docker.sock:/tmp/docker.sock \
    gliderlabs/registrator:latest \
      consul://localhost:8500
```

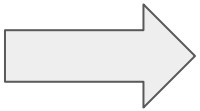
Добавляем поддержку Consul в Docker

- Dnsmasq 53->8600
- docker run … -dns $HOSTNAME

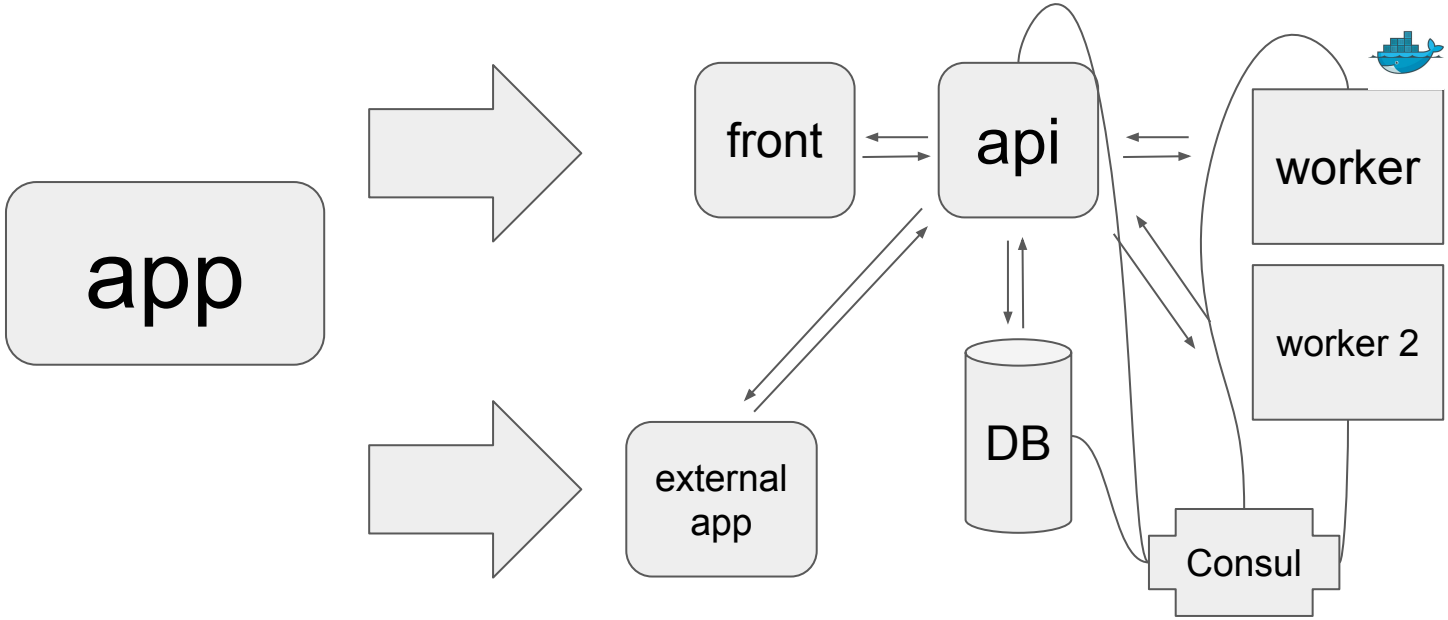
Что мы получаем от Consul?

- Распределенное key-value хранилище
- Service Discovery
- Monitoring





- Полное понимание статуса сервисов
- Доступность сервисов
- Легкое управление конфигами и быстрая доставка конфигов сервисам


app
front
api
worker
worker 2
DB
external app
Consul

Спасибо!

Ссылки и контакты

Stanislav Gumeniuk
https://gumeniuk.com/
mailto: i@vigo.su
@vigo5190

https://hub.docker.com/r/progrium/consul/
https://www.npmjs.com/package/pm2-consul