

Tinkoff Python

Лекция 9

Python

Микросервисы



Ахтаров Данил

# Про что говорим

1. Как работает интерпретатор
2. Логирование
3. Микросервисы, сервисы, макросервисы
4. Архитектура web приложений
5. Мониторинг
6. CI/CD
7. ...

# Зачем?

Проще диагностировать проблемы

Меньше магии

Всегда знаем куда пойти посмотреть

Что такое  
интерпретатор?

# Достоинство?

Исходный текст  
(test.py)

Байт код  
(test.pyc)

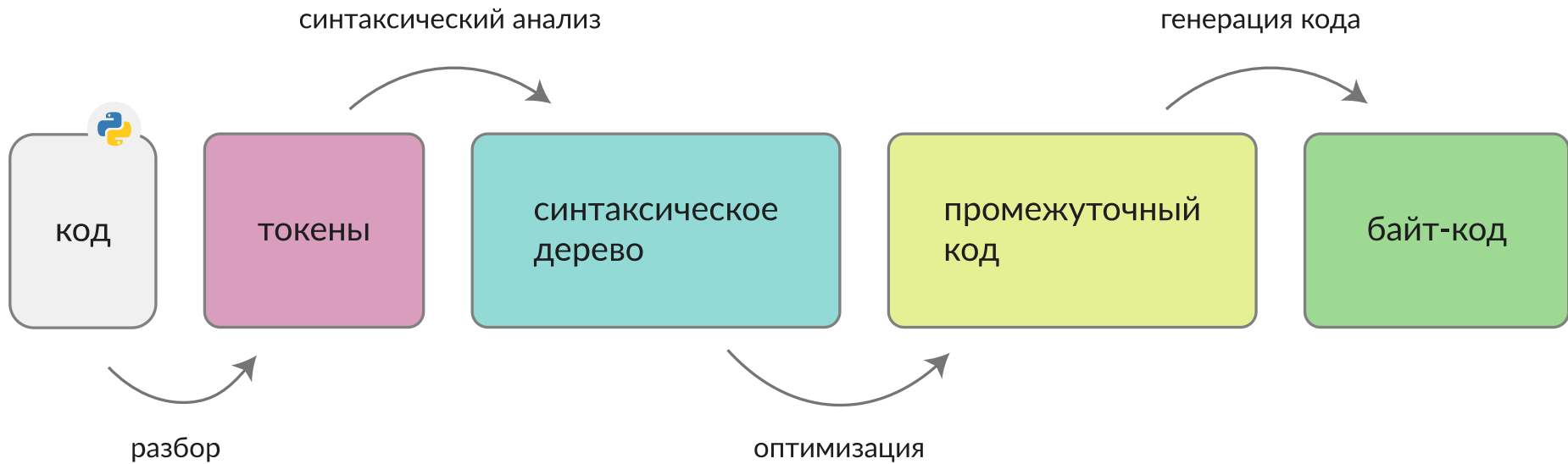
Выполнение  
(PVM)

# Зачем?

Компактное представление

Ограниченный набор команд

Можно сразу интерпретировать без дополнительных шагов





# Tokenizer

```
1 def say_hello():  
2     print("Hello, World!")  
3  
4 say_hello()
```

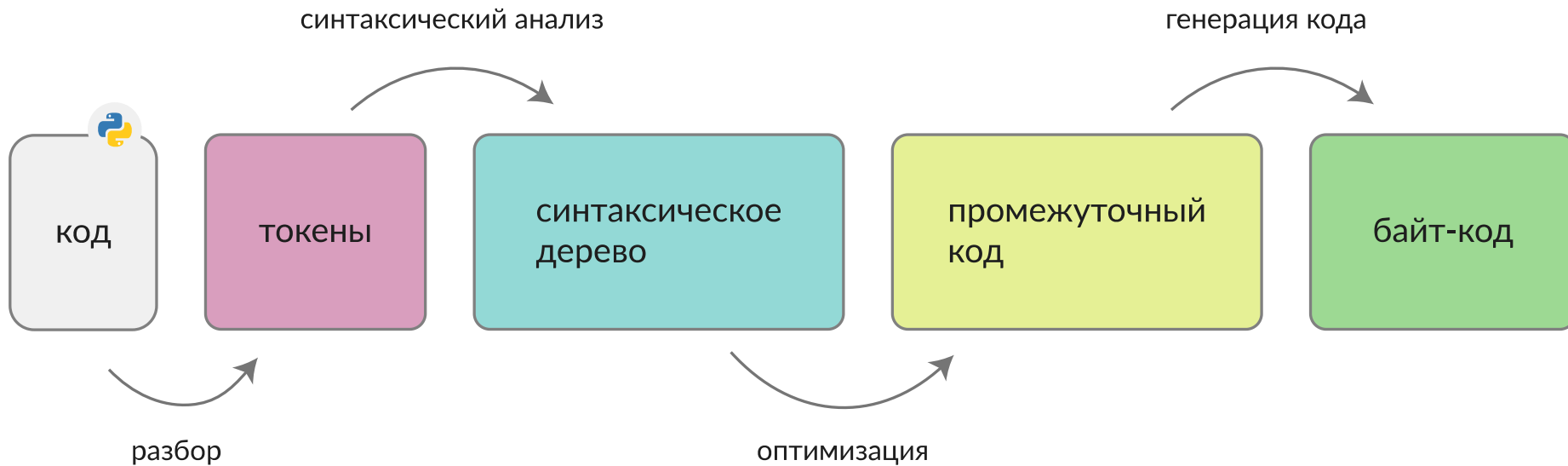
```
1 $ python -m tokenize hello.py  
2 0,0-0,0:          ENCODING          'utf-8'  
3 1,0-1,3:          NAME              'def'  
4 1,4-1,13:         NAME              'say_hello'  
5 1,13-1,14:        OP                '('  
6 1,14-1,15:        OP                ')'  
7 1,15-1,16:        OP                ':'  
8 1,16-1,17:        NEWLINE           '\n'  
9 2,0-2,4:          INDENT             '    '  
10 2,4-2,9:          NAME              'print'  
11 2,9-2,10:         OP                '('  
12 2,10-2,25:        STRING            '"Hello, World!'"'  
13 2,25-2,26:        OP                ')'  
14 2,26-2,27:        NEWLINE           '\n'  
15 3,0-3,1:          NL                '\n'  
16 4,0-4,0:          DEDENT             ''  
17 4,0-4,9:          NAME              'say_hello'  
18 4,9-4,10:         OP                '('  
19 4,10-4,11:        OP                ')'  
20 4,11-4,12:        NEWLINE           '\n'  
21 5,0-5,0:          ENDMARKER         ''
```

Можно получить ошибки  
несоответствия грамматике

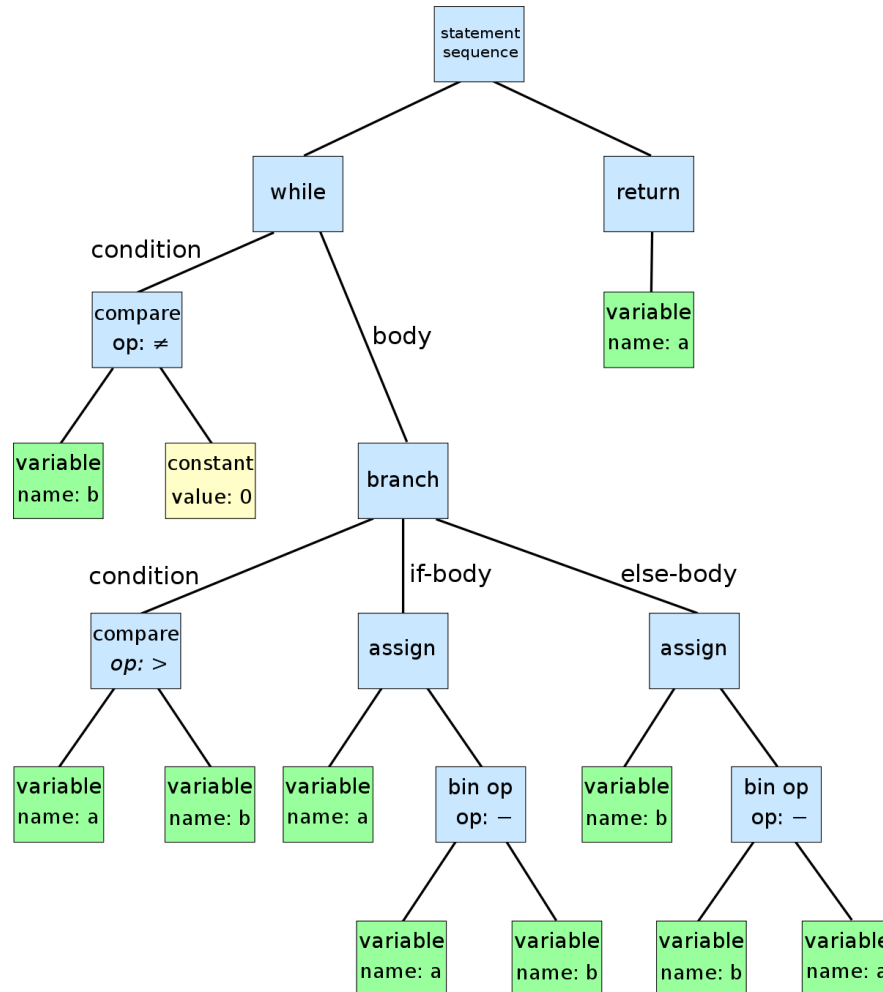
# Грамматика языка

```
1 ...
2 comp_op: '<' | '>' | '==' | '>=' | '<=' | '<>' | '!=' | 'in' | 'not' 'in' | 'is' | 'is' 'not'
3 star_expr: '*' expr
4 expr: xor_expr ('|' xor_expr)*
5 xor_expr: and_expr ('^' and_expr)*
6 and_expr: shift_expr ('&' shift_expr)*
7 shift_expr: arith_expr (('<<' | '>>') arith_expr)*
8 arith_expr: term (('+' | '-') term)*
9 ...
```

<https://github.com/python/cpython/blob/master/Grammar/Grammar>



# Abstract Syntax Tree



Можно получить  
синтаксические ошибки

# AST

```
1 import ast
2
3 tree = ast.parse('print("Hello world")')
4
5 tree
6 # <_ast.Module at 0x10a3ba0b8>
```

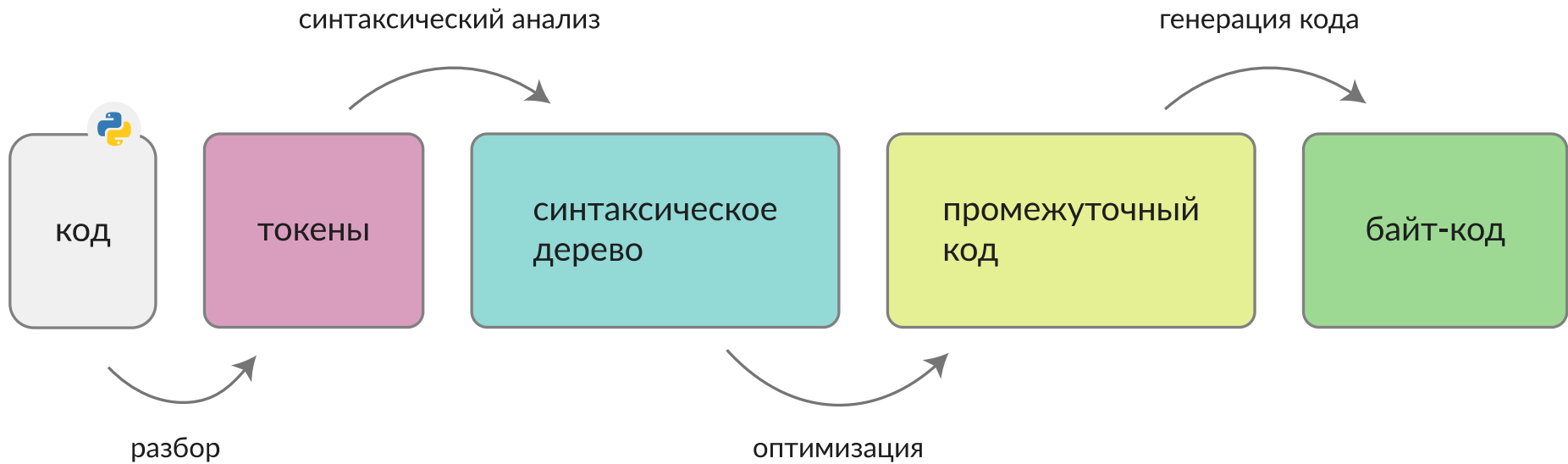
<https://docs.python.org/3/library/ast.html>

# AST

```
1 import ast
2
3 node = ast.UnaryOp()
4 node.op = ast.USub()
5 node.operand = ast.Constant()
6 node.operand.value = 5
7 node.operand.lineno = 0
8 node.operand.col_offset = 0
9 node.lineno = 0
10 node.col_offset = 0
```

<https://docs.python.org/3/library/ast.html>





# Дизассемблирование байткода

```
1  import dis
2
3  def foo(y):
4      x = 1
5      return x + y
6
7  dis.dis(foo)
```

1	4	0	LOAD_CONST	1	(1)
2		2	STORE_FAST	1	(x)
3					
4	5	4	LOAD_FAST	1	(x)
5		6	LOAD_FAST	0	(y)
6		8	BINARY_ADD		
7		10	RETURN_VALUE		

<https://docs.python.org/3/library/dis.html>

# Дизассемблирование байткода

```
1 def f(num):  
2     if num == 42:  
3         return True  
4     return False
```

1	(1)	(2)	(3)	(4)	(5)	(6)	(7)
2	---	---	---	---	-----	---	-----
3	2			0	LOAD_FAST	0	(num)
4		-->		2	LOAD_CONST	1	(42)
5				4	COMPARE_OP	2	(==)
6				6	POP_JUMP_IF_FALSE	12	
7							
8	3			8	LOAD_CONST	2	(True)
9				10	RETURN_VALUE		
10							
11	4		>>	12	LOAD_CONST	3	(False)
12				14	RETURN_VALUE		

```

1 // ceval.c
2 for (;;) {
3     switch (opcode) {
4         case TARGET(NOP): {
5             FAST_DISPATCH();
6         }
7         case TARGET(LOAD_FAST): {
8             ...
9         }
10        case TARGET(LOAD_CONST): {
11            PREDICTED(LOAD_CONST);
12            PyObject *value = GETITEM(consts, oparg);
13            Py_INCREF(value);
14            PUSH(value);
15            FAST_DISPATCH();
16        }
17        case TARGET(STORE_FAST): {
18            PREDICTED(STORE_FAST);
19            PyObject *value = POP();
20            SETLOCAL(oparg, value);
21            FAST_DISPATCH();
22        }
23        case TARGET(POP_TOP): {
24            ...
25        }
26        case TARGET(ROT_TWO): {
27            ...
28        }
29        case TARGET(ROT_THREE): {
30            ...
31        }
32        case TARGET(ROT_FOUR): {
33            ...
34        }
35        case TARGET(DUP_TOP): {
36            ...
37        }
38        case TARGET(DUP_TOP_TWO): {

```

# CPython

The screenshot shows the GitHub interface for the 'python/cpython' repository. At the top, the repository name is displayed with a sub-header 'python / cpython'. To the right are interaction buttons: 'Sponsor', 'Watch' (with a dropdown arrow and '1.1k' watchers), 'Unstar' (with '30.2k' stars), and 'Fork' (with '14.1k' forks). Below these is a navigation bar with tabs for 'Code', 'Pull requests' (showing '1,103'), 'Actions', 'Security', and 'Insights'. The main description reads 'The Python programming language' followed by the URL 'https://www.python.org/'. A statistics bar shows '106,802 commits', '13 branches', '0 packages', '422 releases', and '1,195 contributors', with a 'View license' link. At the bottom, there's a 'Branch: master' dropdown, a 'New pull request' button, and a row of buttons: 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button with a dropdown arrow.

python / cpython

Sponsor Watch 1.1k Unstar 30.2k Fork 14.1k

Code Pull requests 1,103 Actions Security Insights

The Python programming language <https://www.python.org/>

106,802 commits 13 branches 0 packages 422 releases 1,195 contributors View license

Branch: master New pull request Create new file Upload files Find file Clone or download

<https://github.com/python/cpython>

# Что есть что в репо?

- Grammar - грамматика питона
- Include - .h файлы для C кода (здесь же объявляются почти все основные структуры)
- Lib - стандартная библиотека на питоне
- Modules - стандартная библиотека на C
- Objects - объектная система питона, встроенные структуры данных
- Parser - парсинг исходного кода (до ast)
- Python - интерпретатор, компилятор байт кода

```
1  /* Minimal main program -- everything is loaded from the library */
2
3  #include "Python.h"
4  #include "pycore_pylifecycle.h"
5
6  #ifdef MS_WINDOWS
7  int
8  wmain(int argc, wchar_t **argv)
9  {
10     return Py_Main(argc, argv);
11 }
12 #else
13 int
14 main(int argc, char **argv)
15 {
16     return Py_BytesMain(argc, argv);
17 }
18 #endif
```

# Parser/

- token.c
- tokenizer.c
- parser.c



# Python/

- ast.c
- ceval.c

```

1 // ceval.c
2 for (;;) {
3     switch (opcode) {
4         case TARGET(NOP): {
5             FAST_DISPATCH();
6         }
7         case TARGET(LOAD_FAST): {
8             ...
9         }
10        case TARGET(LOAD_CONST): {
11            PREDICTED(LOAD_CONST);
12            PyObject *value = GETITEM(consts, oparg);
13            Py_INCREF(value);
14            PUSH(value);
15            FAST_DISPATCH();
16        }
17        case TARGET(STORE_FAST): {
18            PREDICTED(STORE_FAST);
19            PyObject *value = POP();
20            SETLOCAL(oparg, value);
21            FAST_DISPATCH();
22        }
23        case TARGET(POP_TOP): {
24            ...
25        }
26        case TARGET(ROT_TWO): {
27            ...
28        }
29        case TARGET(ROT_THREE): {
30            ...
31        }
32        case TARGET(ROT_FOUR): {
33            ...
34        }
35        case TARGET(DUP_TOP): {
36            ...
37        }
38        case TARGET(DUP_TOP_TWO): {

```

# Фреймы

Модуль

code, args, kwargs,  
const, etc.

main

code, args, kwargs,  
const, etc.

sort

code, args, kwargs,  
const, etc.

sort\_file

**current frame**

# Получение текущего фрейма

```
1 import inspect
2
3 current_frame = inspect.currentframe()
4
5 current_frame
6 # <frame at 0x1068357a8, file '<...>', line 1, code <module>>
7
8 current_frame.f_back
9 # <frame at 0x7fa91b874df8, file './.py', line 2981, code run_code>
```

```
>>> import requests
>>> requests.get(42)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python3/dist-packages/requests/api.py", line 55, in get
    return request('get', url, **kwargs)
  File "/usr/lib/python3/dist-packages/requests/api.py", line 44, in request
    return session.request(method=method, url=url, **kwargs)
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 421, in request
    prep = self.prepare_request(req)
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 359, in prepare_request
    hooks=merge_hooks(request.hooks, self.hooks),
  File "/usr/lib/python3/dist-packages/requests/models.py", line 287, in prepare
    self.prepare_url(url, params)
  File "/usr/lib/python3/dist-packages/requests/models.py", line 338, in prepare_url
    "Perhaps you meant http://{0}?".format(url))
requests.exceptions.MissingSchema: Invalid URL '42': No schema supplied. Perhaps you meant http://42?
```

```

1 // ceval.c
2 for (;;) {
3     switch (opcode) {
4         case TARGET(NOP): {
5             FAST_DISPATCH();
6         }
7         case TARGET(LOAD_FAST): {
8             ...
9         }
10        case TARGET(LOAD_CONST): {
11            PREDICTED(LOAD_CONST);
12            PyObject *value = GETITEM(consts, oparg);
13            Py_INCREF(value);
14            PUSH(value);
15            FAST_DISPATCH();
16        }
17        case TARGET(STORE_FAST): {
18            PREDICTED(STORE_FAST);
19            PyObject *value = POP();
20            SETLOCAL(oparg, value);
21            FAST_DISPATCH();
22        }
23        case TARGET(POP_TOP): {
24            ...
25        }
26        case TARGET(ROT_TWO): {
27            ...
28        }
29        case TARGET(ROT_THREE): {
30            ...
31        }
32        case TARGET(ROT_FOUR): {
33            ...
34        }
35        case TARGET(DUP_TOP): {
36            ...
37        }
38        case TARGET(DUP_TOP_TWO): {

```

# Выводы

- Python - компилируемый и интерпретируемый язык
- Интерпретатор - стековый
- Байт код кэшируется
- Фреймы создаются всегда при вызове функции
- Оптимизаций мало

# Logging

<https://docs.python.org/3/library/logging.html>



# Зачем?

А что с print-ом не так?

# Использование

```
1 import logging
2
3
4 logger = logging.getLogger(__name__)
5
6
7 def foo():
8     logger.info('Some log text')
```

# Конфигурирование

```
1 import logging
2
3
4 logging.basicConfig(filename="file.log", level=logging.INFO)
```



```

1  [loggers]
2  keys=root,exampleApp
3
4  [handlers]
5  keys=fileHandler, consoleHandler
6
7  [formatters]
8  keys=myFormatter
9
10 [logger_root]
11 level=CRITICAL
12 handlers=consoleHandler
13
14 [logger_exampleApp]
15 level=INFO
16 handlers=fileHandler
17 qualname=exampleApp
18
19 [handler_consoleHandler]
20 class=StreamHandler
21 level=DEBUG
22 formatter=myFormatter
23 args=(sys.stdout,)
24
25 [handler_fileHandler]
26 class=FileHandler
27 formatter=myFormatter
28 args=("config.log",)
29
30 [formatter_myFormatter]
31 format=%(asctime)s - %(name)s - %(levelname)s - %(message)s
32 datefmt=

```

```

1  import logging
2
3
4  logging.config.fileConfig('logging.conf')

```

# Уровни логирования

1. DEBUG
2. INFO
3. WARNING
4. ERROR
5. CRITICAL

# Exception

```
1 import logging
2
3 logging.basicConfig(filename="filename.log", level=logging.INFO)
4 logger = logging.getLogger(__name__)
5
6 try:
7     raise RuntimeError
8 except RuntimeError:
9     logger.exception("Error!")
```

# Форматирование

```
1 FORMAT = '%(asctime)-15s %(clientip)s %(user)-8s %(message)s'
2 logging.basicConfig(format=FORMAT)
3
4 d = {'clientip': '192.168.0.1', 'user': 'fbloggs'}
5 logging.warning('Protocol problem: %s', 'connection reset', extra=d)
```

```
1 2006-02-08 22:20:02,165 192.168.0.1 fbloggs Protocol problem: connection reset
```



# Handlers

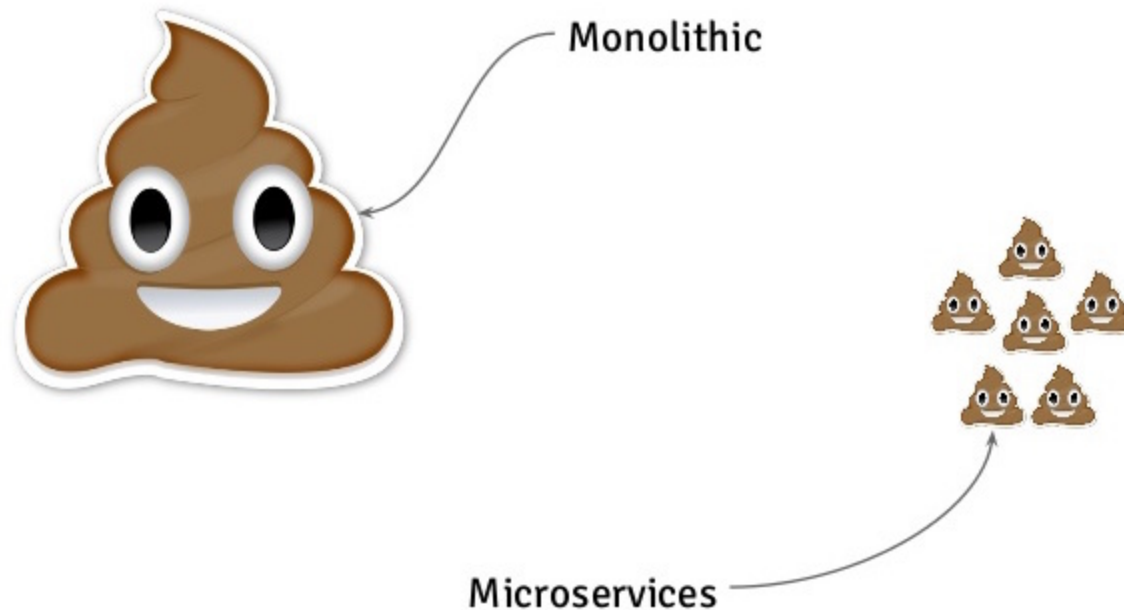
```
1 logger = logging.getLogger(__name__)
2 logger.setLevel(logging.INFO)
3
4 # create the logging file handler
5 fh = logging.FileHandler("filename.log")
6
7 formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
8 fh.setFormatter(formatter)
9
10 # add handler to logger object
11 logger.addHandler(fh)
12
13 logger.info("Program started")
```

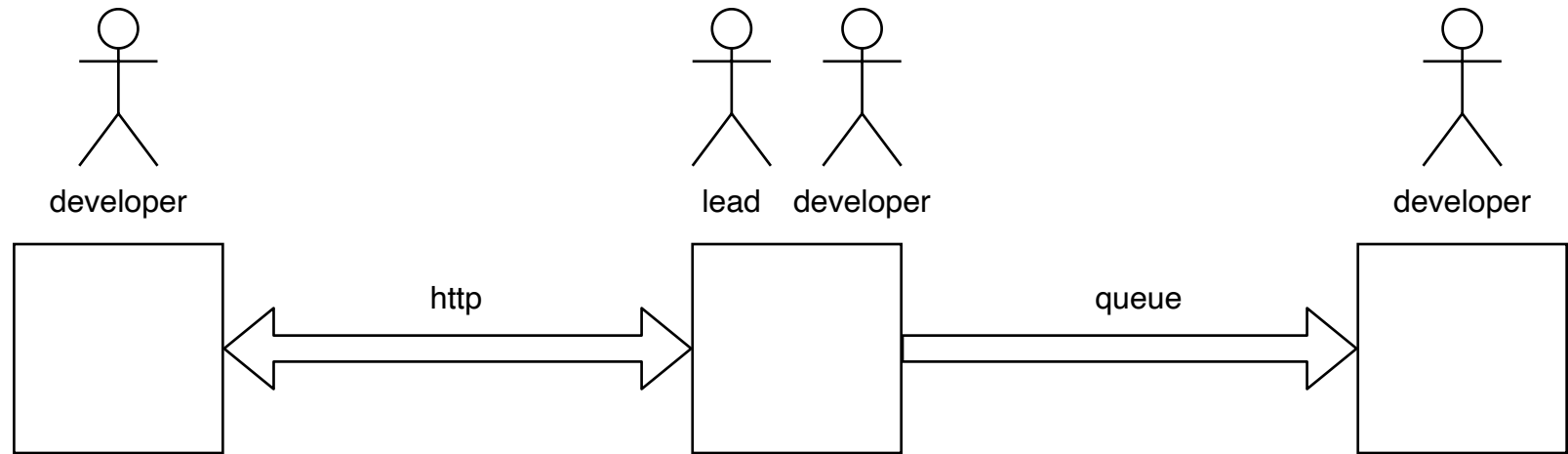
Всегда используйте  
logging

# Микросервисы

# Зачем?

## Monolithic vs Microservices





# Сервисы

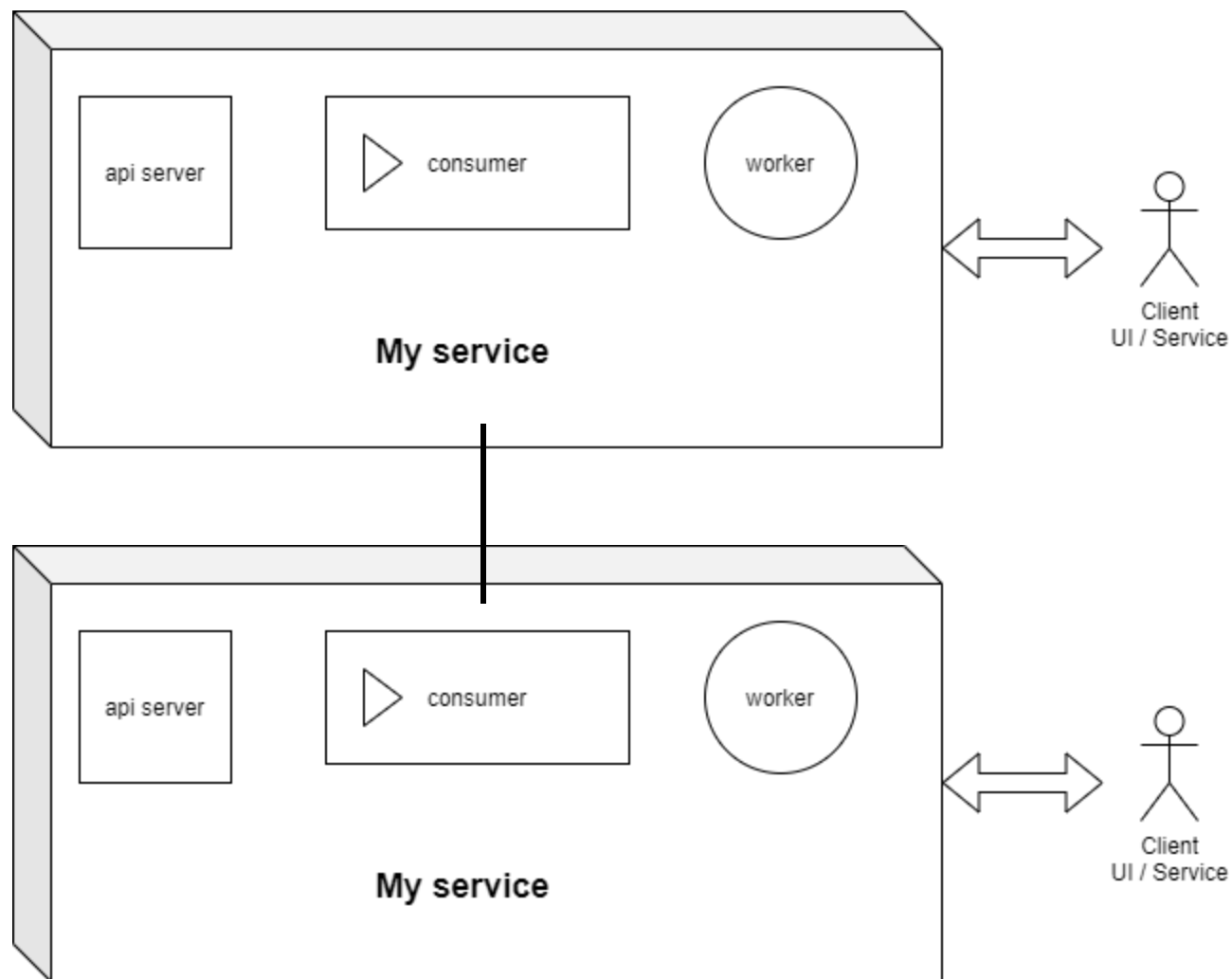


# Микросервисы

Worker, Consumer, REST API Server,  
Connector, ...

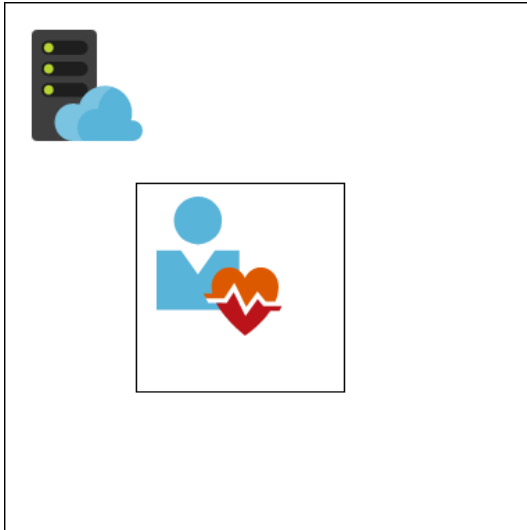


# Макросервисы

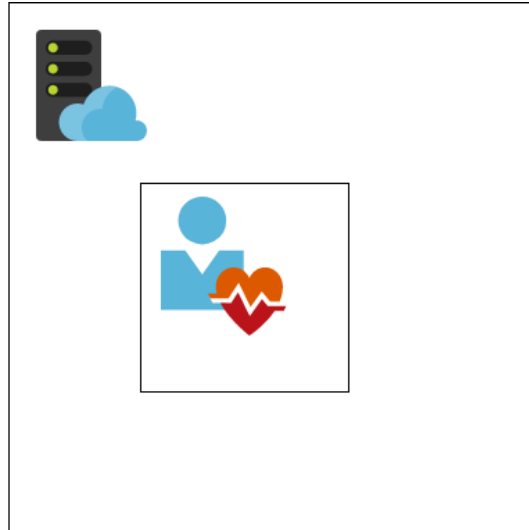


# Stateless

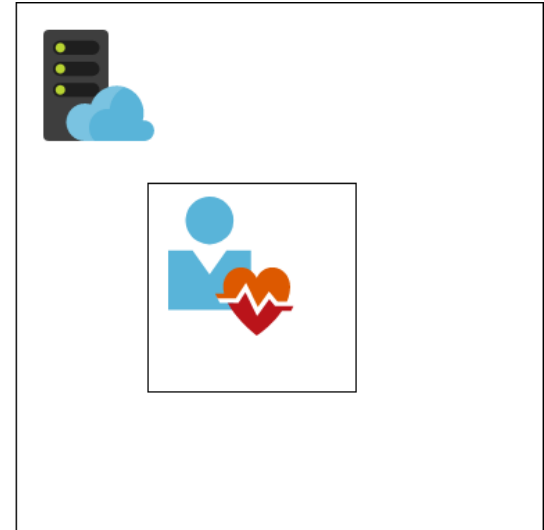
Host 1



Host 2

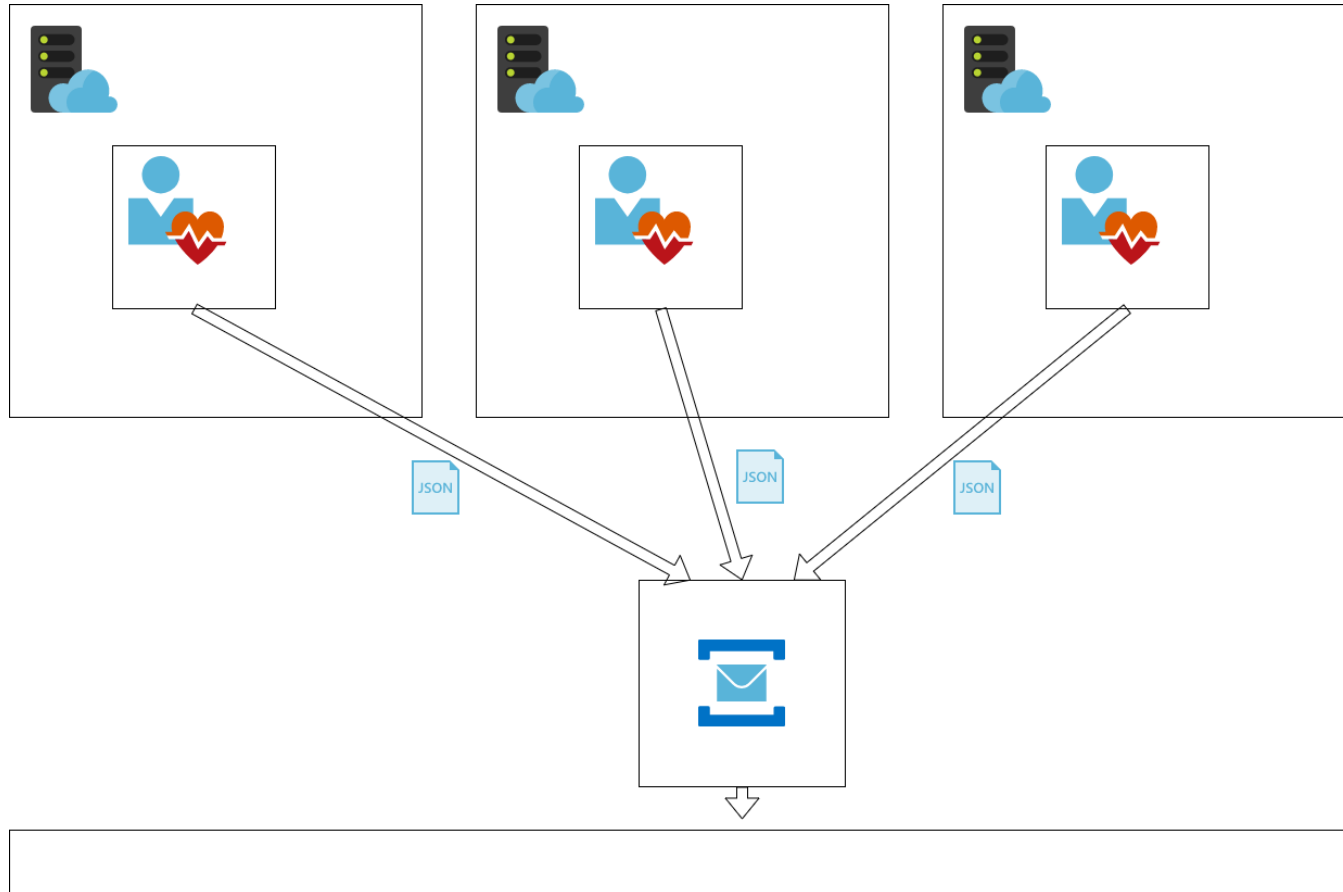


Host 3

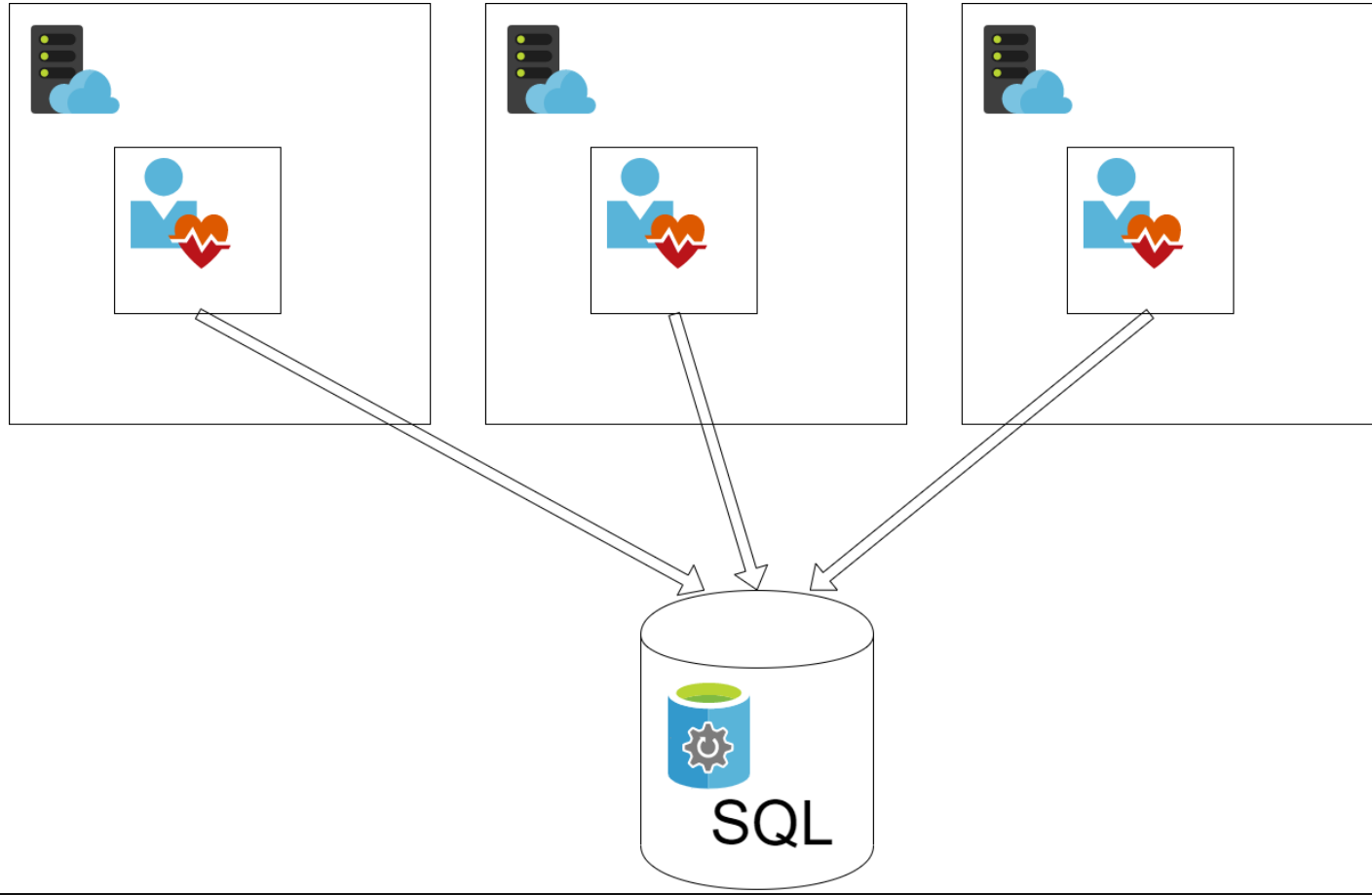


# External Storages

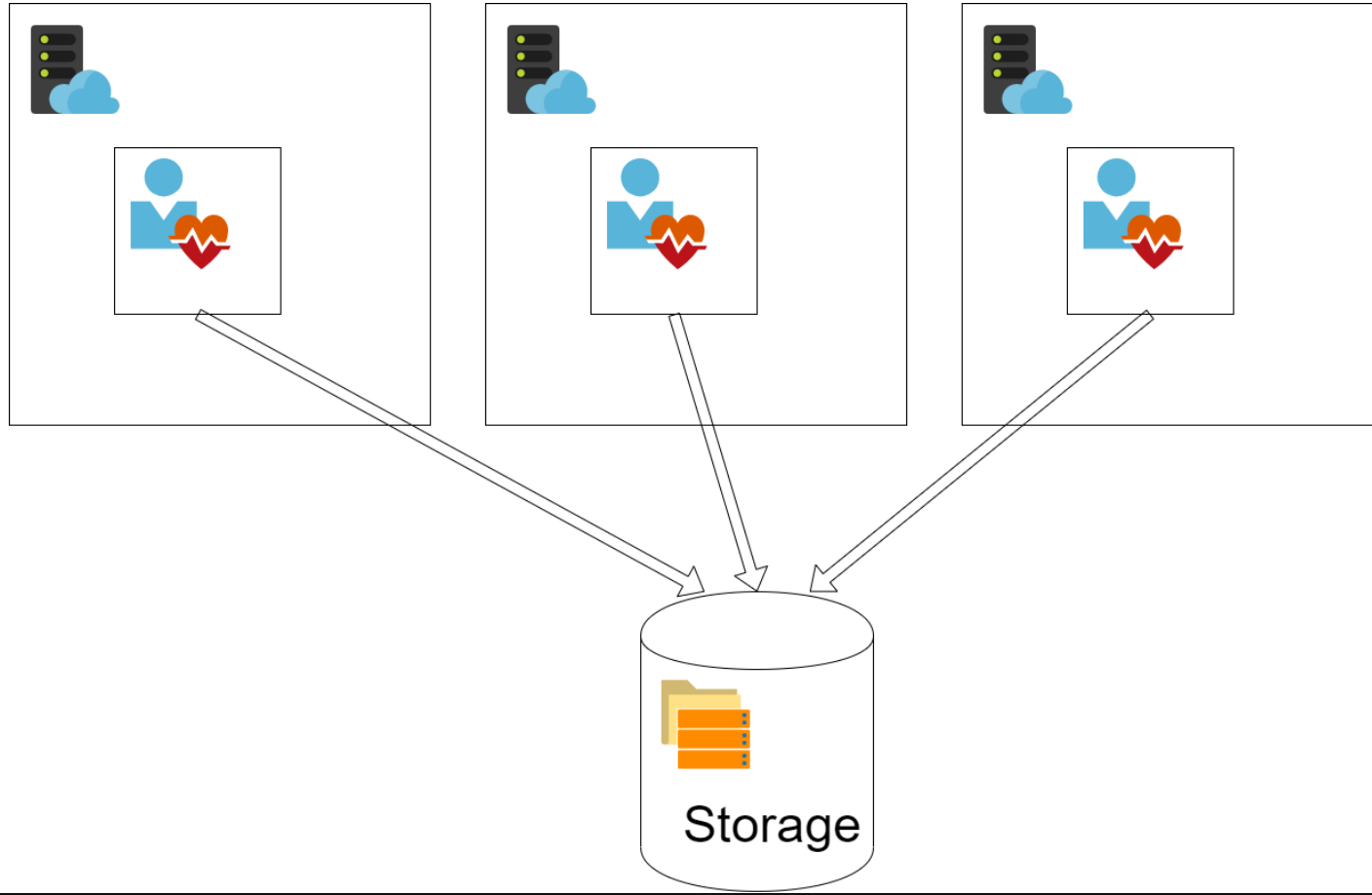
# Message brokers



# Databases



# Network drive





# Messenger

Монолит

Можно обмениваться сообщениями в  
комнатах (PtP)

Messenger Messenger Messenger

DB

# Messenger

Authorization

Consumer 1

Producer 1

Settings

Consumer 2

Producer 2

Messages

Producer 3

# Авторизация

# Зачем?

Использую Basic Auth и не парюсь!

# Token

- 12231244124
- 8178e09f-4c9b-46c4-b98c-dcd9d3c1e5ca
- RRFHFGW183H9o5yb5jPH0fLFDEjDF2R2

# Пара токенов

**Access token** - 15 минут (много раз)

**Refresh token** - 1 неделя (один раз)

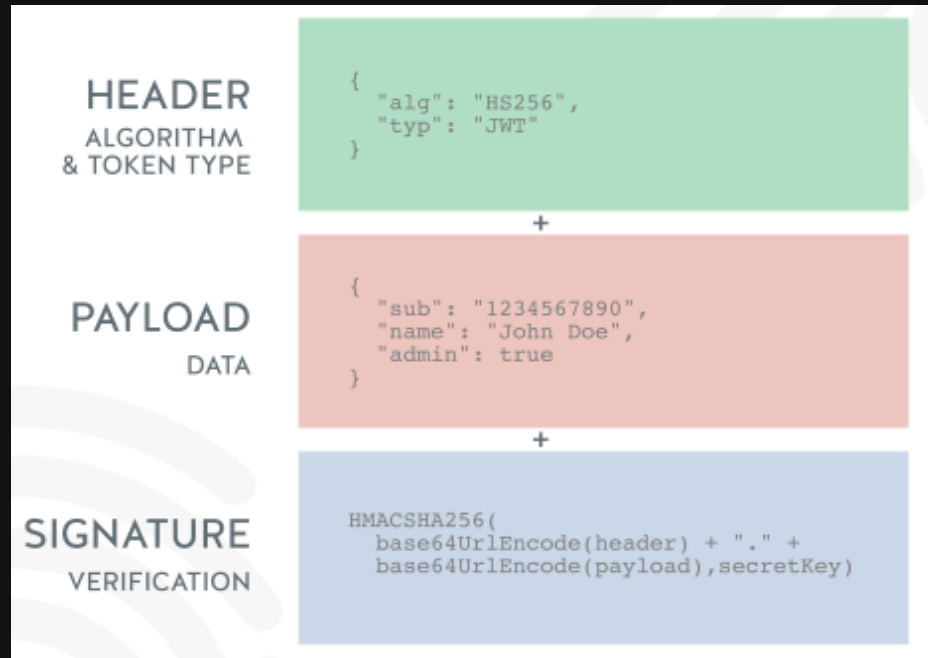
# Требования

- Хочу подписывать токен ключом
- Хочу хранить информацию внутри токена

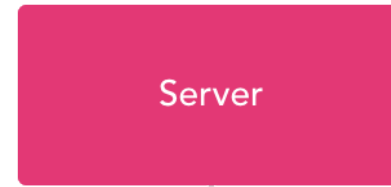


# JWT

json web token



<https://jwt.io/>



1. POST /users/login with username and password

2. Creates a JWT  
with a secret

3. Returns the JWT to the Browser

4. Sends the JWT on the Authorization Header

5. Check JWT signature.  
Get user information  
from the JWT

6. Sends response to the client

# Структура

```
1 {  
2   "alg": "HS256",  
3   "typ": "JWT"  
4 }
```

```
1 {  
2   "sub": "1234567890",  
3   "name": "John Doe",  
4   "admin": true  
5 }
```

`iss, sub, aud, exp, nbf, jti, iat`

```
1 >>> import jwt
2
3 >>> encoded_jwt = jwt.encode({'some': 'payload'}, 'secret', algorithm='HS256')
4 >>> encoded_jwt
5 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.'
6 'eyJzb211IjoicGF5bG9hZCJ9.4twFt5NiznN84AWoold7K01T_yoc0Z6XOpOVswacPZg'
7
8 >>> jwt.decode(encoded_jwt, 'secret', algorithms=['HS256'])
9 {'some': 'payload'}
```

# OAuth2



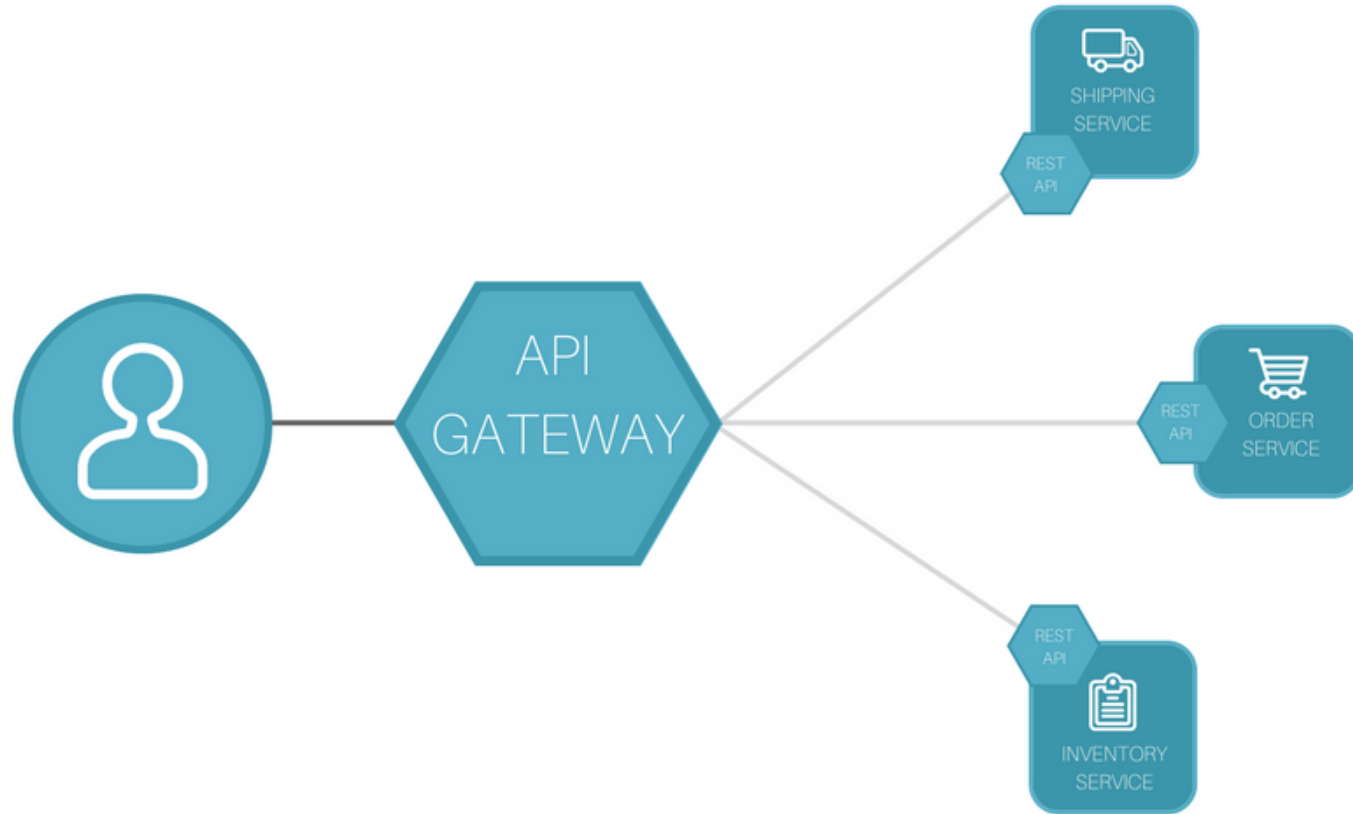
# Архитектура

# Серебряная пуля

# ОСНОВНЫЕ КОМПОНЕНТЫ



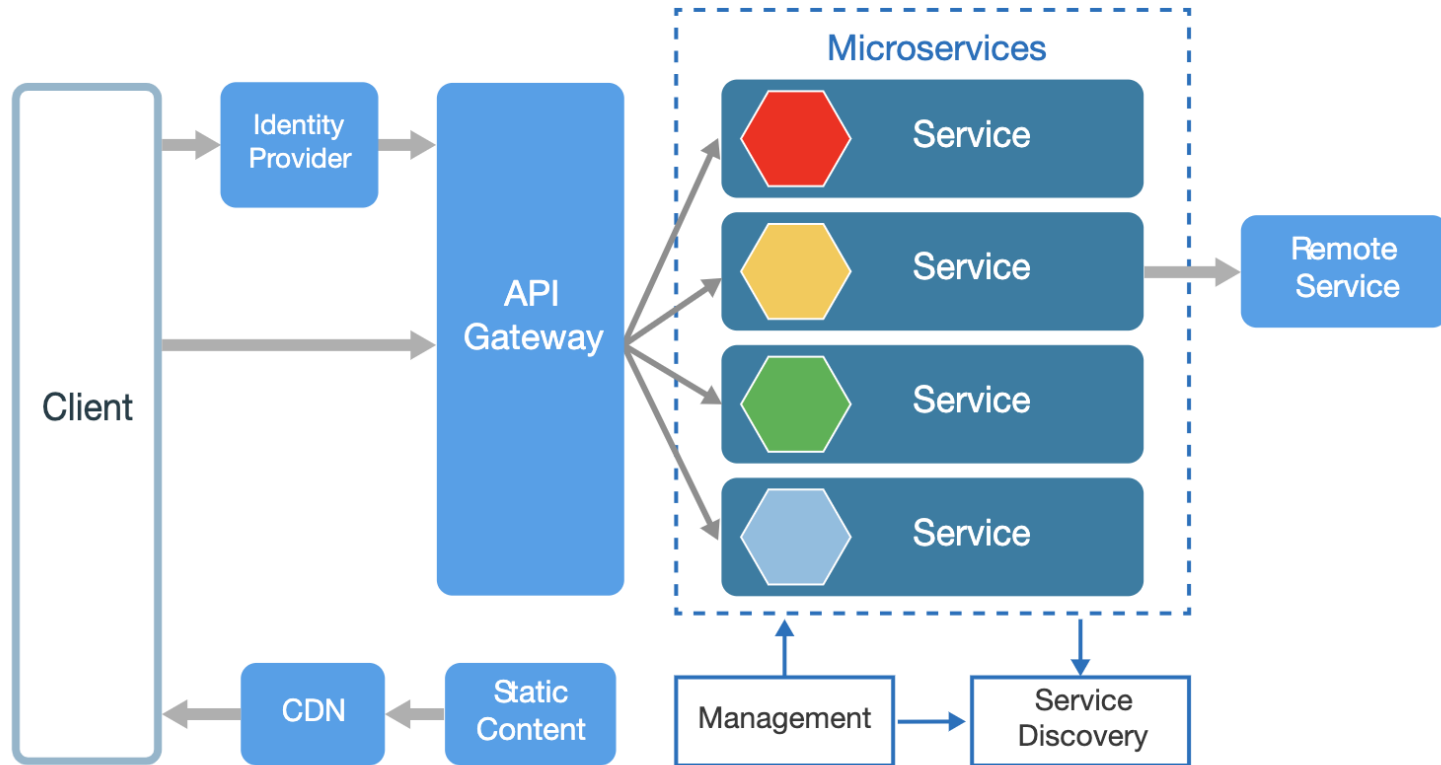
# API Gateway



# Auth Service

# Storage

# Your Service



# Вопросы?

# Мониторинг

# Зачем?



# Logging

**graylog** Search Streams Alerts Dashboards Sources System ▾ In 0 / Out 0 msg/s Help ▾ Jonathan ▾

Search in the last 1 day ▾

▶ Not updating ▾

Saved searches ▾

message:"Processing time" AND (function:download\_file OR function:email\_file OR function:ftp\_file)|

### EZ Exporter - Production

Found **533 messages** in 187 ms, searched in [1 index](#).  
Results retrieved at 2017-02-23 23:36:46.

Add count to dashboard ▾

Save search criteria

More actions ▾

Fields

Decorators

Default

All

None

Filter fields

☐ environment

☐ facility

☐ file

☒ function

☒ level

☒ line

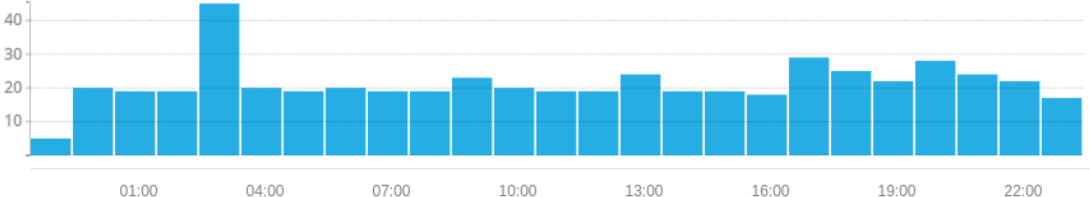
☒ message

List fields of [current page](#) or [all fields](#).

### Histogram

Add to dashboard ▾

Year, Quarter, Month, Week, Day, **Hour**, Minute



### Messages

Previous


1

2

3

4

Next

Timestamp 	source	function	level	line
2017-02-23 23:30:14.492 <a href="#">Processing time (seconds): 10</a>	ezexporter	ftp_file	6	991
2017-02-23 23:30:04.424 <a href="#">Processing time (seconds): 0</a>	ezexporter	email_file	6	889
2017-02-23 23:30:04.270 <a href="#">Processing time (seconds): 0</a>	ezexporter	email_file	6	889

15.3

# Метрики

GET /metrics

```
1 python_threads_total 10
2 ...
3 app_version_info 1.0
4 app_connections_total{role="admin"} 10
5 app_connections_total{role="user"} 110
6 ...
```

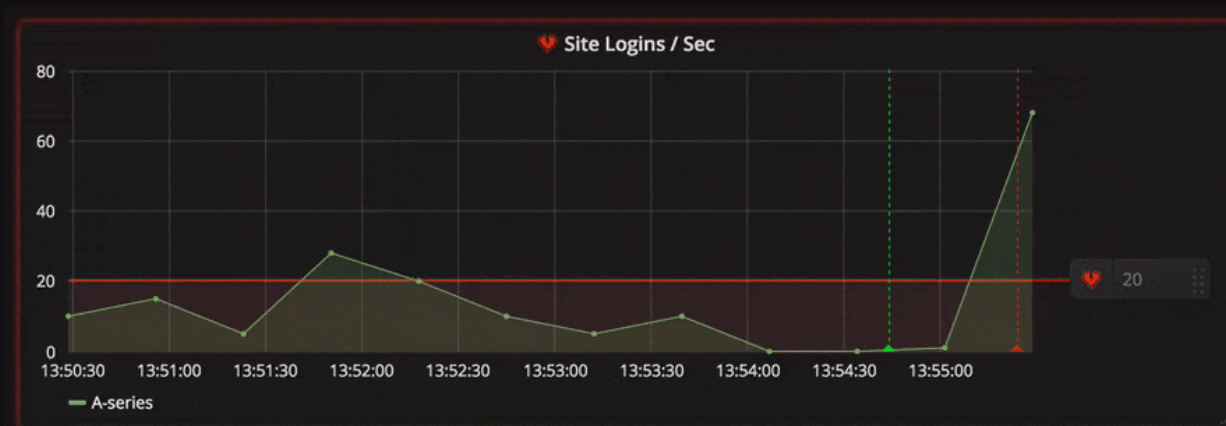
# Metric types

- Counter
- Gauge
- Summary
- Histogram
- Info
- Enum (Current state "stopped")

# Визуализации



# Alerts



Graph

General

Metrics

Axes

Legend

Display

Alert

Time range

×

Alert Config

Notifications (1)

State history

Delete

Alert Config

Name

Site Logins Too Low

Evaluate every

10s

Conditions

WHEN

avg ()

OF

query (A, 5m, now)

IS BELOW

20

✕

+

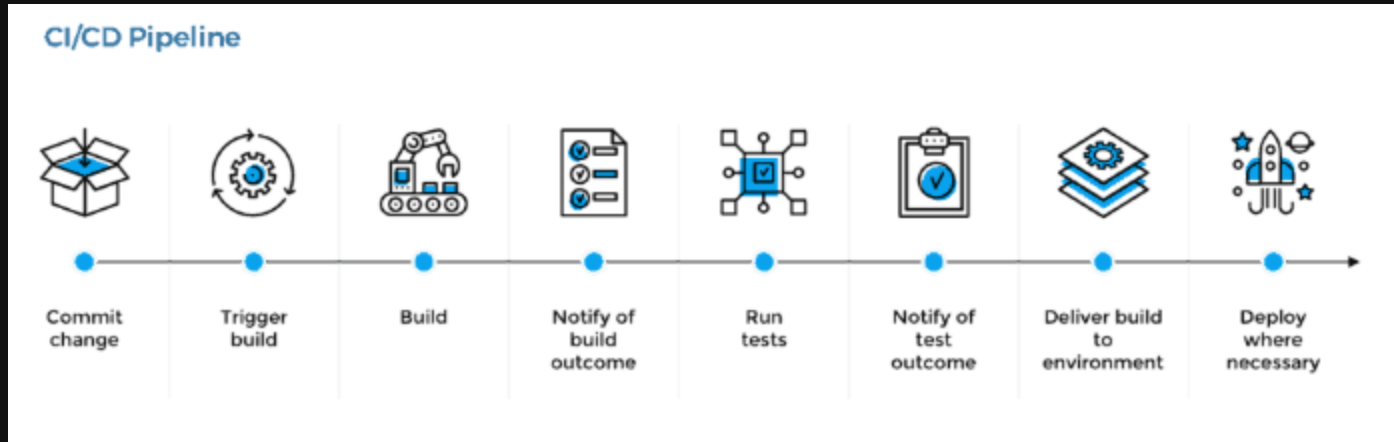
If no data points or all values are null

SET STATE TO

No Data

▼

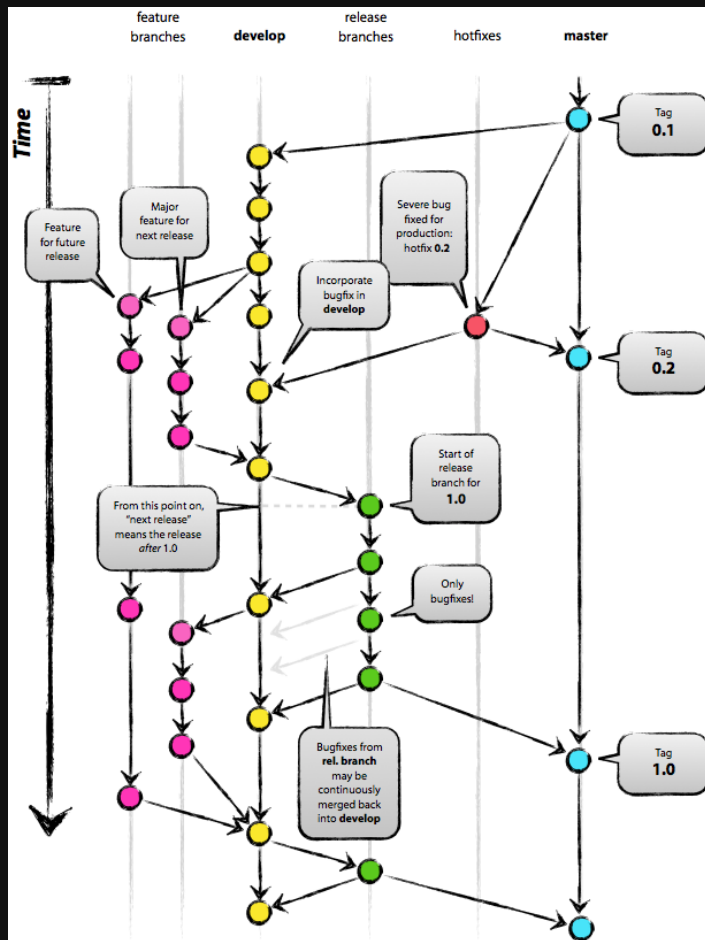
# CI/CD



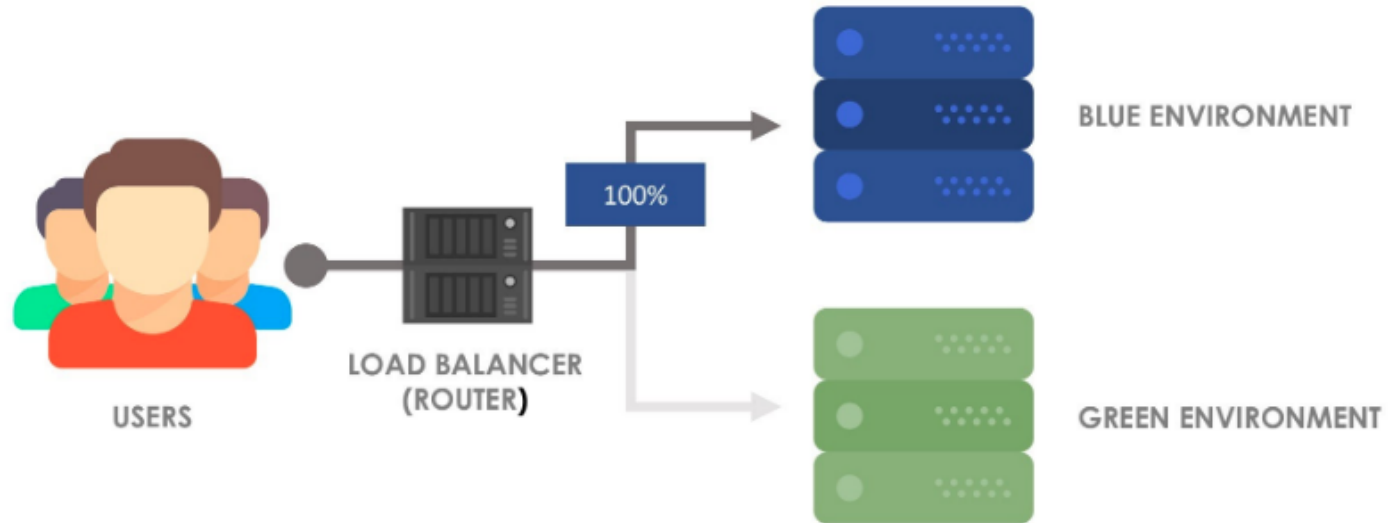
# Gitflow

- master
  - develop
  - feature/\*
  - bugfix/\*
  - release/\*
  - hotfix/\*
- master
  - develop
  - feature/STORY-1
  - bugfix/TASK-1
  - release/v1.0
  - hotfix/v1.0.1
  - TASK-6

# Gitflow







# Configs

**Переменные окружения**

# Pydantic

```
1 from pydantic import BaseSettings
2
3
4 class Config(BaseSettings):
5     MAX_SIZE: int = 1024
6     KEY: str
7
8     class Config:
9         env_prefix = 'MY_APP_'
10
11
12 config = Config()
```

# The Twelve-Factor App

Двенадцать факторов

1. Кодовая база
2. Зависимости
3. Конфигурация
4. Сторонние службы (Backing Services)
5. Сборка, релиз, выполнение
6. Процессы
7. Привязка портов (Port binding)
8. Параллелизм
9. Утилизируемость (Disposability)
10. Паритет разработки/работы приложения
11. Журналирование (Logs)
12. Задачи администрирования