

# Разработка масштабируемых веб-сервисов

—

Denis Tribushkov  
Technical Consultant / Software Engineer

# Agenda

- Что такое web?
- Как устроен простейший веб-сервис?
- Что такое асинхронный веб-сервер?
- Как устроен высоконагруженный веб-сервис?
- Облака и DevOps.

# Agenda

- Что такое web?
- Как устроен простейший веб-сервис?
- Что такое асинхронный веб-сервер?
- Как устроен высоконагруженный веб-сервис?
- Облака и DevOps.

# Internet vs WWW

## Internet

Internet - глобальная сеть передачи данных

## Протоколы

**HTTP**, SSH, P2P - прикладные протоколы

DNS - система имен

TCP - надежная последовательная передача данных

IP - глобальная адресация, передача в гетерогенной среде

## World Wide Web

WWW - множество **взаимосвязанных документов**, располагающихся на машинах подключенных к Internet

WWW - набор протоколов, серверного и клиентского ПО, позволяющих получать доступ к документам



JAN  
2018

# DIGITAL AROUND THE WORLD IN 2018

KEY STATISTICAL INDICATORS FOR THE WORLD'S INTERNET, MOBILE, AND SOCIAL MEDIA USERS

TOTAL  
POPULATION



we  
are  
social

**7.593**  
BILLION

URBANISATION:  
**55%**

INTERNET  
USERS



**4.021**  
BILLION

PENETRATION:  
**53%**

ACTIVE SOCIAL  
MEDIA USERS



we  
are  
social

**3.196**  
BILLION

PENETRATION:  
**42%**

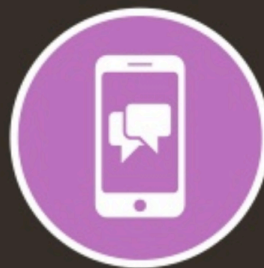
UNIQUE  
MOBILE USERS



**5.135**  
BILLION

PENETRATION:  
**68%**

ACTIVE MOBILE  
SOCIAL USERS



**2.958**  
BILLION

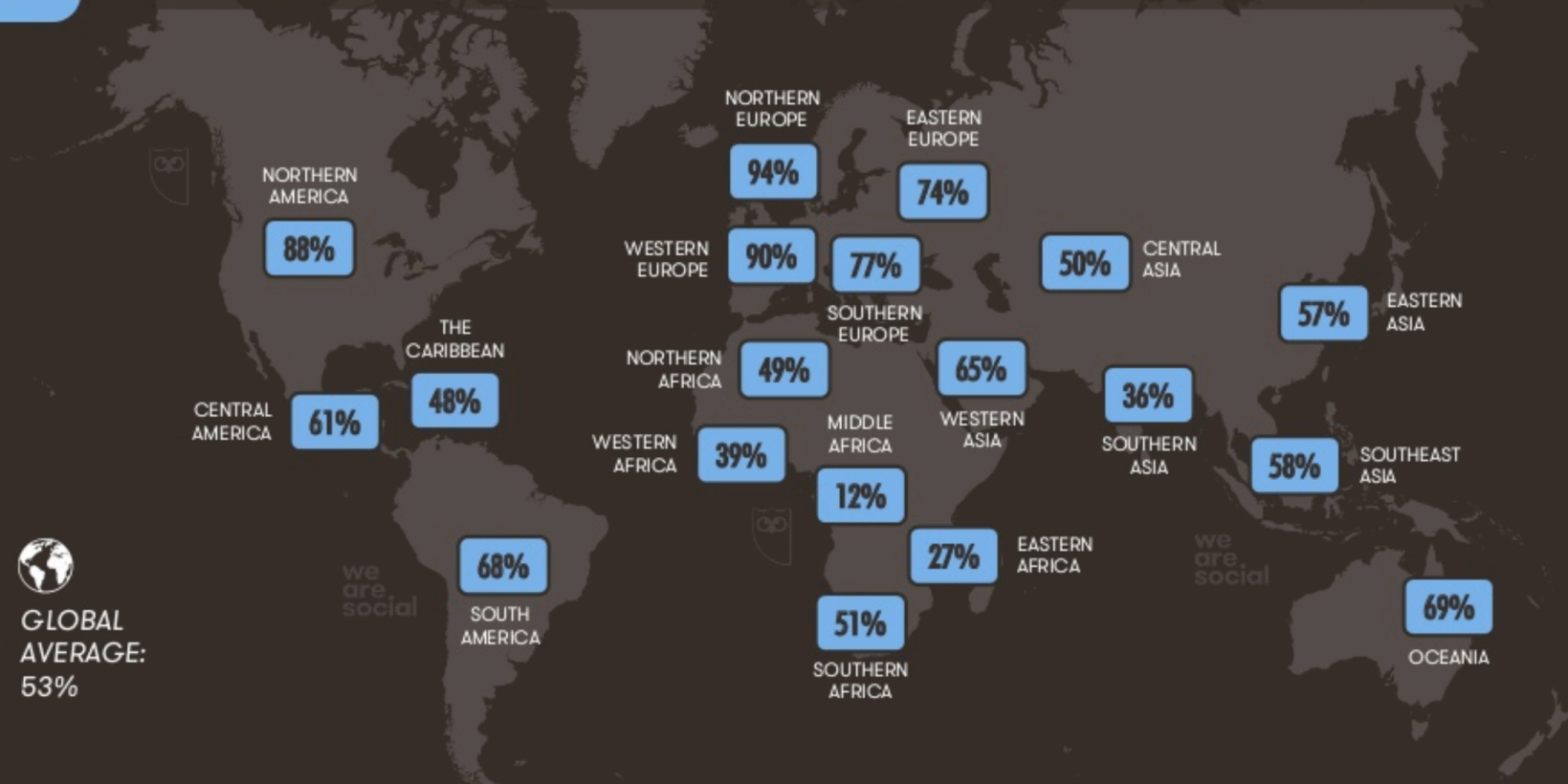
PENETRATION:  
**39%**



JAN  
2018

# INTERNET PENETRATION BY REGION

REGIONAL PENETRATION FIGURES, COMPARING INTERNET USERS TO TOTAL POPULATION



JAN  
2018

# SHARE OF WEB TRAFFIC BY DEVICE

BASED ON EACH DEVICE'S SHARE OF ALL WEB PAGES SERVED TO WEB BROWSERS

LAPTOPS &  
DESKTOPS



**43%**

YEAR-ON-YEAR CHANGE:

**-3%**

MOBILE  
PHONES



**52%**

YEAR-ON-YEAR CHANGE:

**+4%**

TABLET  
DEVICES



**4%**

YEAR-ON-YEAR CHANGE:

**-13%**

OTHER  
DEVICES

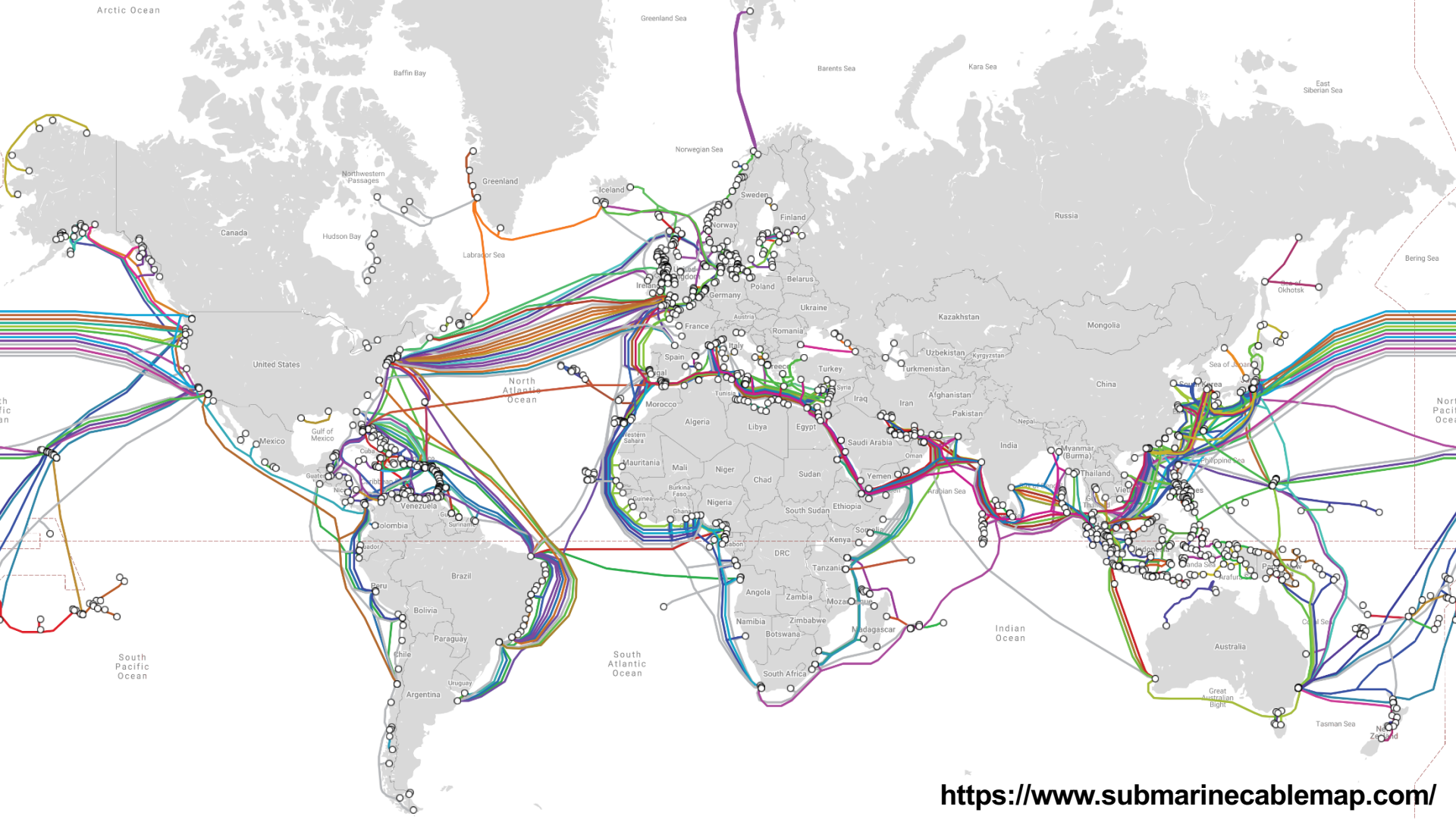


**0.14%**

YEAR-ON-YEAR CHANGE:

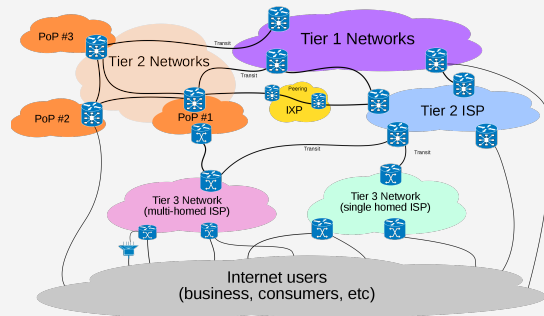
**+17%**





# Операторы связи

- **Tier-1**  
оператор, который имеет доступ к сети Интернет исключительно через пиринговые соединения (равноправны бесплатный обмен);
- **Tier-2**  
оператор, который имеет доступ к части сети Интернет через пиринговые соединения, но покупает транзит IP-трафика для доступа к остальной части Интернета;
- **Tier-3**  
оператор, который для доступа к сети Интернет использует исключительно каналы, которые покупает у других операторов.

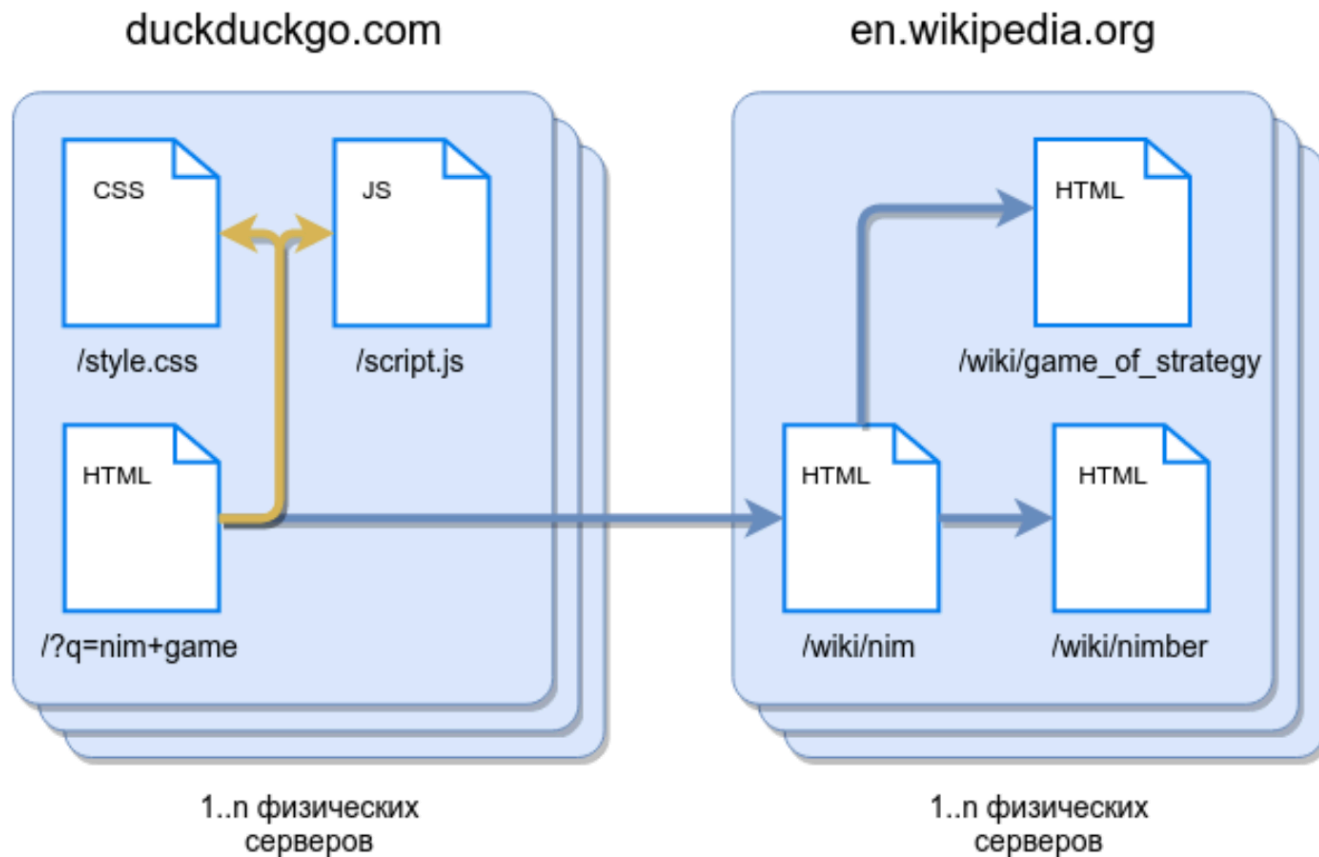




# URL - unified resource locator

<http://server.org:8080/path/doc.html?a=1&b=2#part1>

- http - протокол
- server.org - DNS имя сервера
- 8080 - TCP порт
- /path/doc.html - путь к файлу
- a=1&b=2 - опции запроса
- part1 - якорь, положение на странице



# Agenda

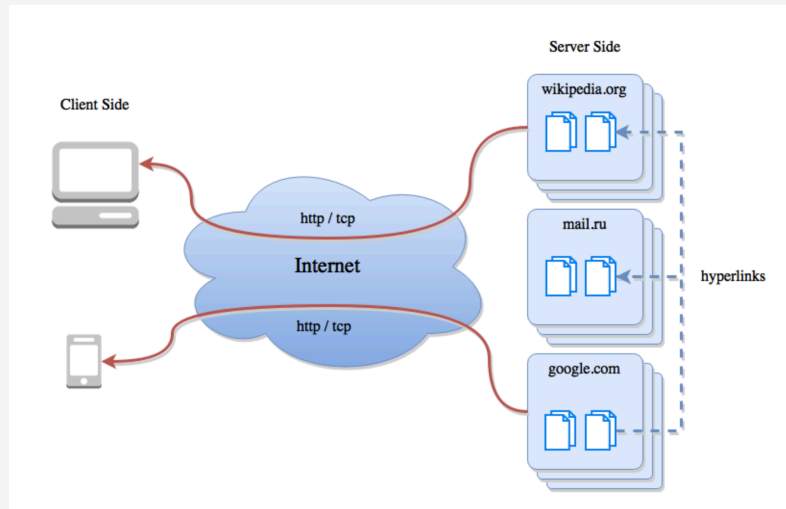
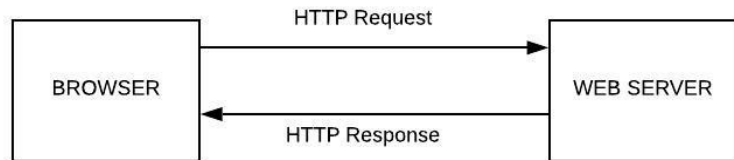
- Что такое web?
- Как устроен простейший веб-сервис?
- Что такое асинхронный веб-сервер?
- Как устроен высоконагруженный веб-сервис?
- Облака и DevOps.



# Клиент-серверная архитектура, виды web-клиентов

- Библиотеки в ЯП: libcurl, urllib, и т.д.
- Консольные утилиты: wget, curl, telnet!
- Роботы: поисковики, вредоносные скрипты
- Браузеры:
  - Полноценные: firefox, chrome и т.д.
  - Встроенные: web-view, webkit и т.д.

```
# pip install requests
import requests
url = 'https://api.site.com/method/'
params = {'argument1': 'value1',
          'argument2': 'value2'}
headers = {'User-Agent': 'python requests'}
response = requests.get(url, params=params, headers=headers)
r.text
# '{"type": "User", "name": "Pupkin"...}'
response.json()
# {'type': 'User', 'name': 'Pupkin', ...}
```



# Как происходит HTTP запрос? (1)

## HTTP/1.1 запрос

```
GET /robots.txt HTTP/1.1
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Cache-Control: max-age=0
Connection: keep-alive
Host: www.ru
User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/39.0
```

## HTTP/1.1 ответ

```
HTTP/1.1 404 Not Found
Server: nginx/1.5.7
Date: Sat, 25 Jul 2015 09:58:17 GMT
Content-Type: text/html; charset=iso-8859-1
Connection: close

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>...
```

## HTTP коды ответа

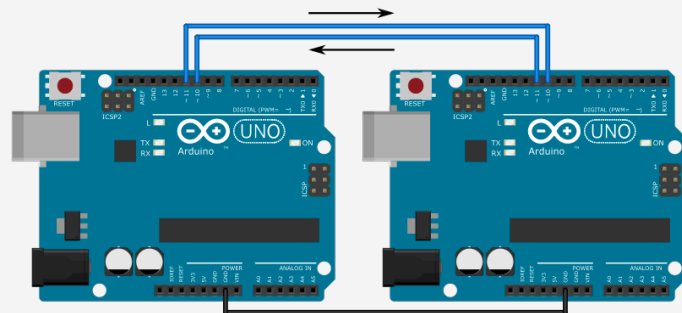
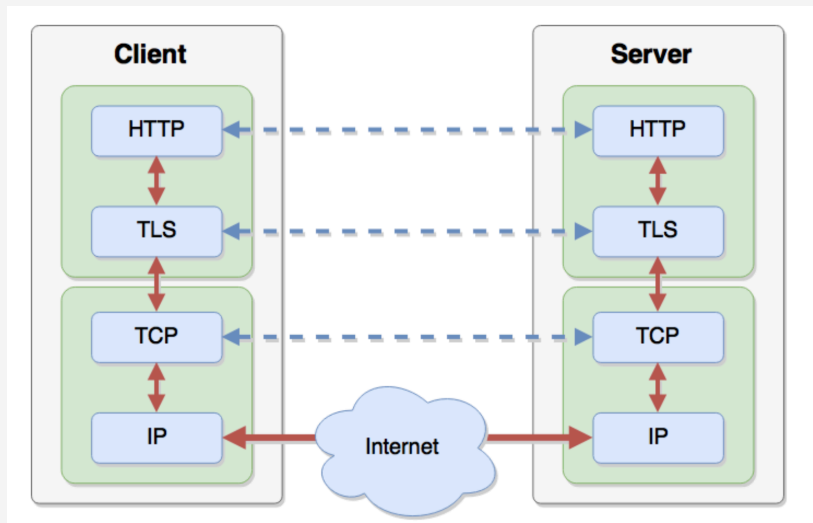
- 1xx - информационные
- 2xx - успешное выполнение
- 3xx - перенаправления
- 4xx - ошибка на стороне клиента
- 5xx - ошибка на стороне сервера

## Заголовки HTTP запросов

- Authorization - авторизация, чаще всего логин/пароль
- Cookie - передача состояния (сессии) на сервер
- Referer - URL предыдущего документа, контекст запроса
- User-Agent - описание web-клиента, версия браузера
- If-Modified-Since - условный GET запрос
- Accept-\* - согласование (negotiation) содержимого

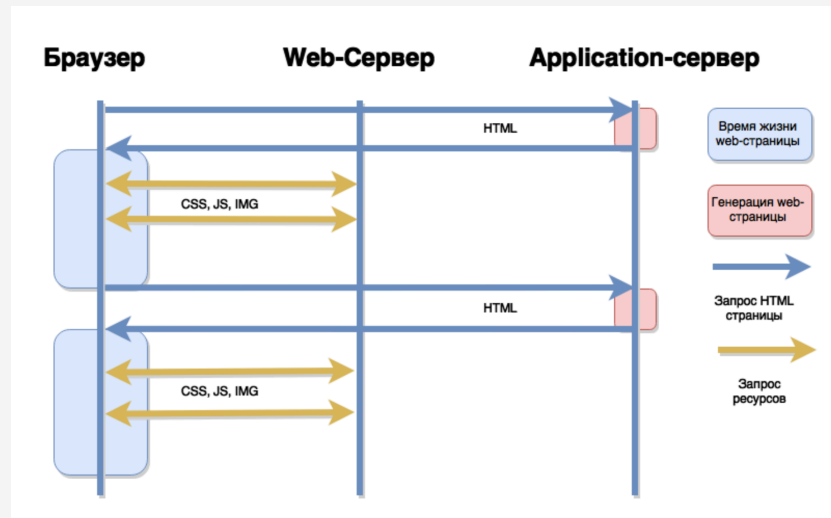
# Как происходит HTTP запрос? (2)

- Браузер анализирует введенный URL и извлекает имя хоста
- Используя систему DNS, браузер преобразует домен в ip адрес
- Устанавливает TCP соединение с web-сервером
- Если протокол https, устанавливает TLS соединение поверх TCP
- Формирует HTTP запрос, отправляет его, HTTP ответ
- Браузер закрывает соединение (для HTTP/1.0)
- Далее процесс парсинга и отображения документа ...



# Сценарий работы web приложения

- Пользователь вводит URL
- Браузер загружает Web страницу - HTML документ
- Браузер анализирует (parse) HTML и загружает доп. ресурсы
- Браузер отображает (rendering) HTML страницу
- Пользователь переходит по гиперссылке или отправляет форму
- Цикл повторяется



```

<!DOCTYPE html>
<html>
  <head>
    <title>Страница</title>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8">
    <meta name="description" content="Сайт">
    <link rel="stylesheet" href="./style.css">
  </head>
  <body id="the_body">
    <p class="article">...</p>
    <script src="./script.js"></script>
  </body>
</html>

```

```

var saveApiUrl = '/items/save/';
var newTitle = 'Duck tales';
$.ajax({
  type: 'POST',
  url: saveApiUrl,
  data: { id: 10, title: newTitle }
});

```

application/json

```

{
  "status": "ok",
  "friends": [
    { "id": 1, "name": "v.pupkin" },
    { "id": 2, "name": "a.pushkin" },
    { "id": 3, "name": "n.tesla" }
  ]
}

```

text/css

```

.hljs-subst,
.hljs-title,
.json .hljs-value {
  font-weight: normal;
  color: #0000;
}

```

# Документы могут быть

## **Статические**

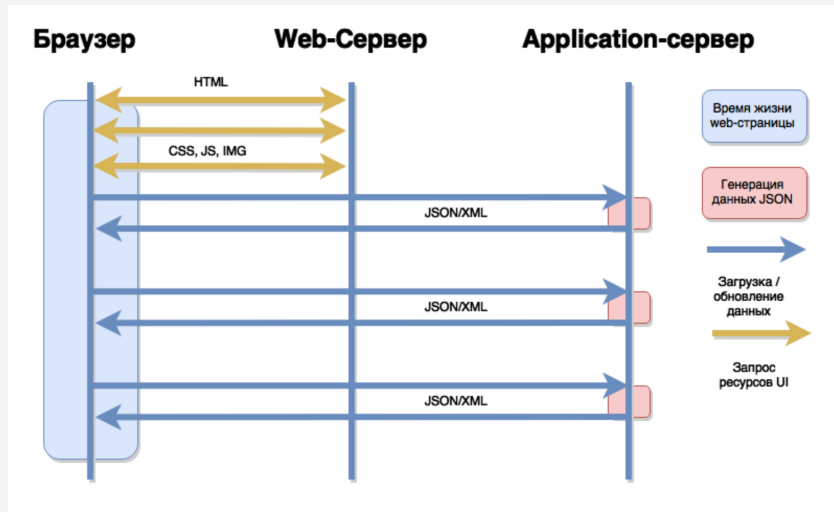
- Это файлы на дисках сервера
- Как правило, обладают постоянным адресом

## **Динамические**

- Создаются на каждый запрос
- Содержимое зависит от времени и пользователя
- Адрес может быть постоянным или меняться

# Сценарий работы современного приложения

- Браузер загружает Web страницу, ресурсы и отображает ее
- JavaScript загружает данные с помощью AJAX запросов
- JavaScript обеспечивает полноценный UI на странице
- Пользователь взаимодействует с UI, что приводит к вызову JavaScript обработчиков
- JavaScript обновляет данные на сервере или загружает новые данные, используя AJAX



# Agenda

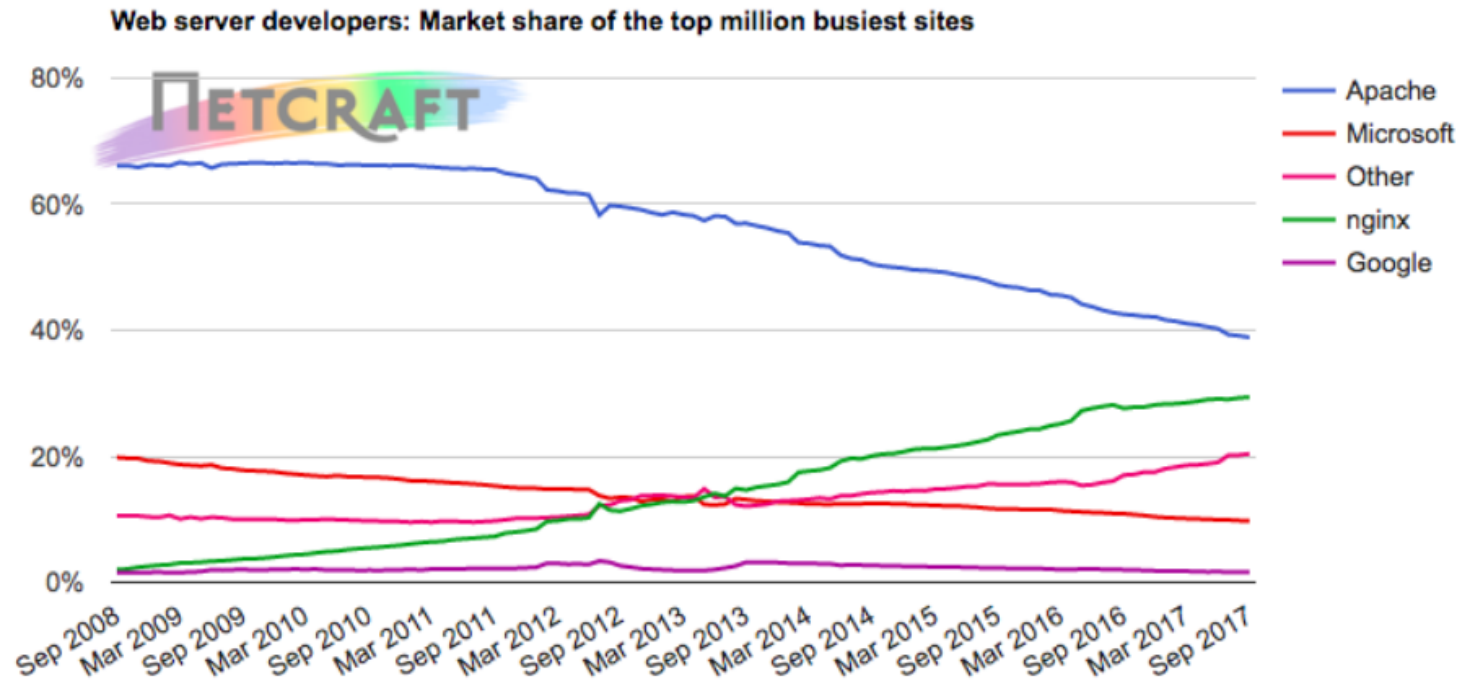
- Что такое web?
- Как устроен простейший веб-сервис?
- Что такое асинхронный веб-сервер?
- Как устроен высоконагруженный веб-сервис?
- Облака и DevOps.



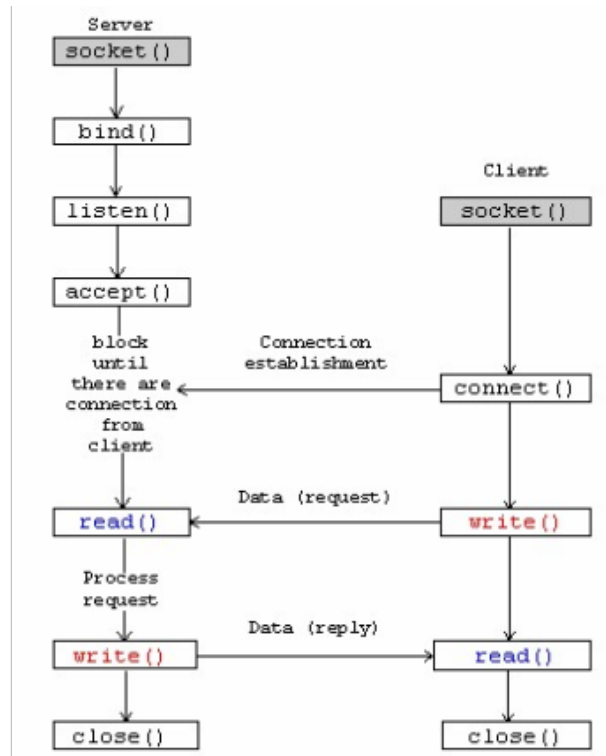
# Веб-сервер



# Статистика по распространенности серверов



# Блокирующая обработка соединений



# Методы обработки большого кол-ва соединений

- fork
- prefork
- threads
- threads prefork
- pooling
- coroutines

# Неблокирующая обработка соединений

## **Системные вызовы:**

- select
- kqueue (FreeBSD 4.1+)
- epoll (Linux 2.6+)

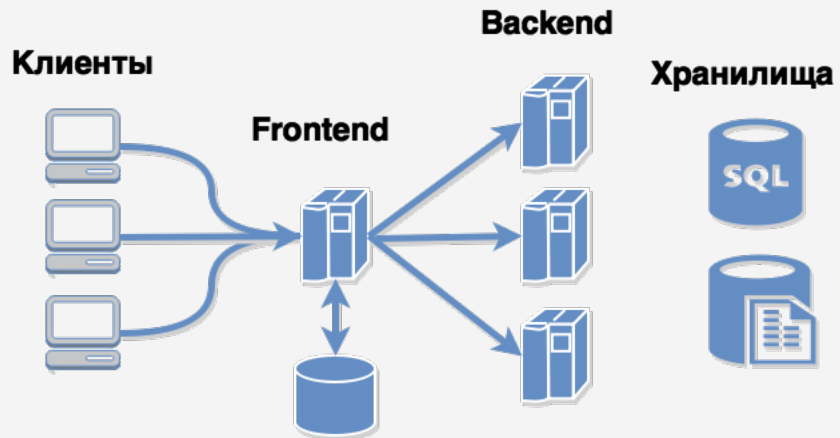
## **Прикладные библиотеки:**

- libevent
- libev

## **Веб-серверы:**

- httpd
- nginx
- Tornado
- Node.js
- Go: net/http

# Статистика по распространенности серверов



# Agenda

- Что такое web?
- Как устроен простейший веб-сервис?
- Что такое асинхронный веб-сервер?
- Как устроен высоконагруженный веб-сервис?
- Облака и DevOps.

# Определения

**Система** — множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность, единство. (М.: БРЭ. — 2003, с. 1437)

В нашем случае – множество серверов и программ на них работающих, представляющих в сумме сервис для конечного пользователя.

**Нагрузка** — совершаемая полезная работа

**Высоконагруженная система** – система при проектировании и эксплуатации которой фактор нагрузки является определяющим



# Предметная область: веб-сервисы

- Ярко выраженный эффект масштаба
- Быстрый рост успешных проектов
- Могут использоваться миллионами людей
- Необходимо учитывать нагрузку при проектировании
- Умение держать нагрузку – вопрос выживания

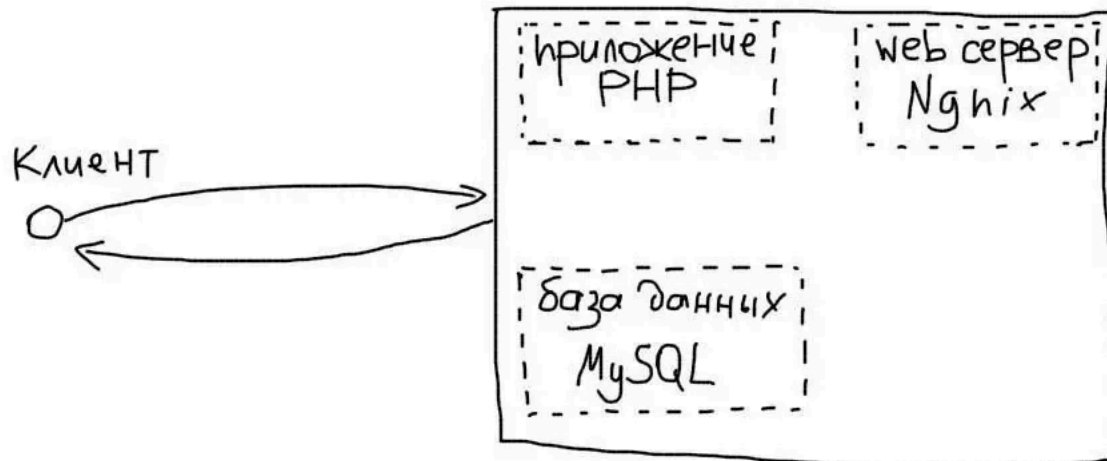
# Разработка ПО в больших системах

- Разработка ПО это только первый (небольшой) шаг
- Дальнейшая эксплуатация и модификация системы это большая часть работы
- Условия эксплуатации диктуют требования к ПО (а не наоборот)
- ПО не проверенное боевой эксплуатацией не считается рабочим

# Доступность

Доступность %	Время простоя в год	Время простоя в месяц
99% ("две девятки")	3.65 дней	7.20 часов
99.5%	1.83 дней	3.60 часов
99.9% ("три девятки")	8.76 часов	43.2 минут
99.95%	4.38 часов	21.56 минут
99.99% ("четыре девятки")	52.56 минут	4.32 минут
99.999% ("пять девяток")	5.26 минут	25.9 секунд
99.9999% ("шесть девяток")	31.5 секунд	2.59 секунды

# Простая архитектура веб-приложения



# СУБД и Реляционная модель данных

- Данные хранятся в виде таблиц. У каждой таблицы фиксированное число столбцов. Все данные в столбце одного типа.

**blog\_post**

id	title	category_id
1	Python	1
3	Python for dummies	1
15	Perl	2
7	C++ in 21 day	3

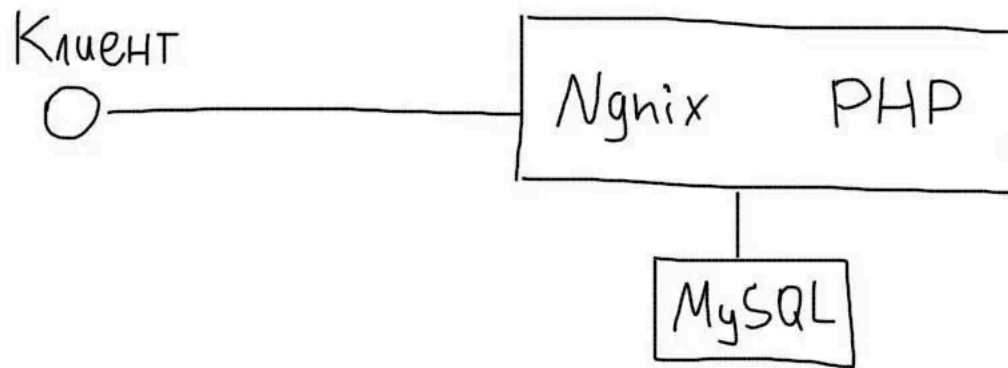
**blog\_category**

id	title	auditory
1	Python	4000
2	Perl	2000
3	C++	10000
4	Java	8000

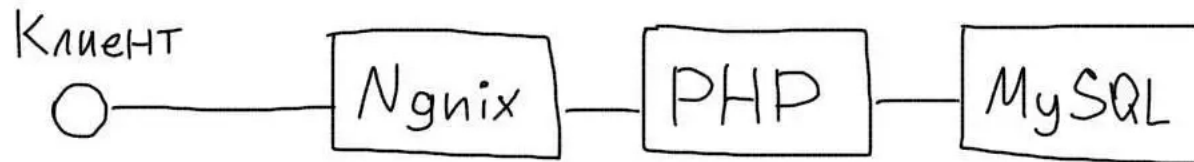
```
SELECT * FROM instructor
WHERE salary BETWEEN 50000 AND 100000;
```



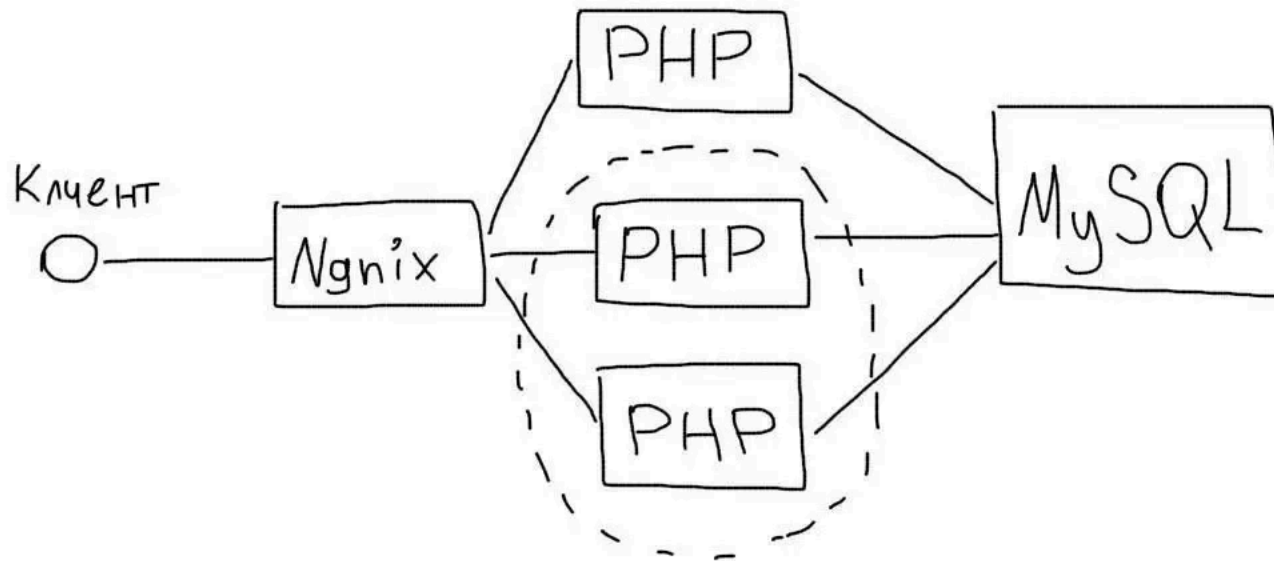
# Отделение базы данных



# Отделение Web сервера

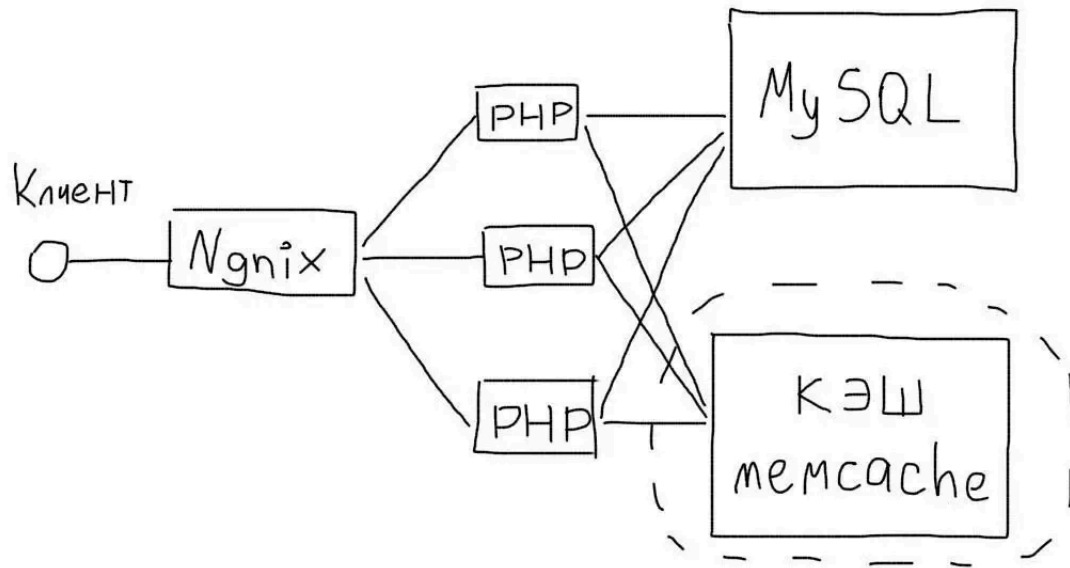


# Несколько бекендов

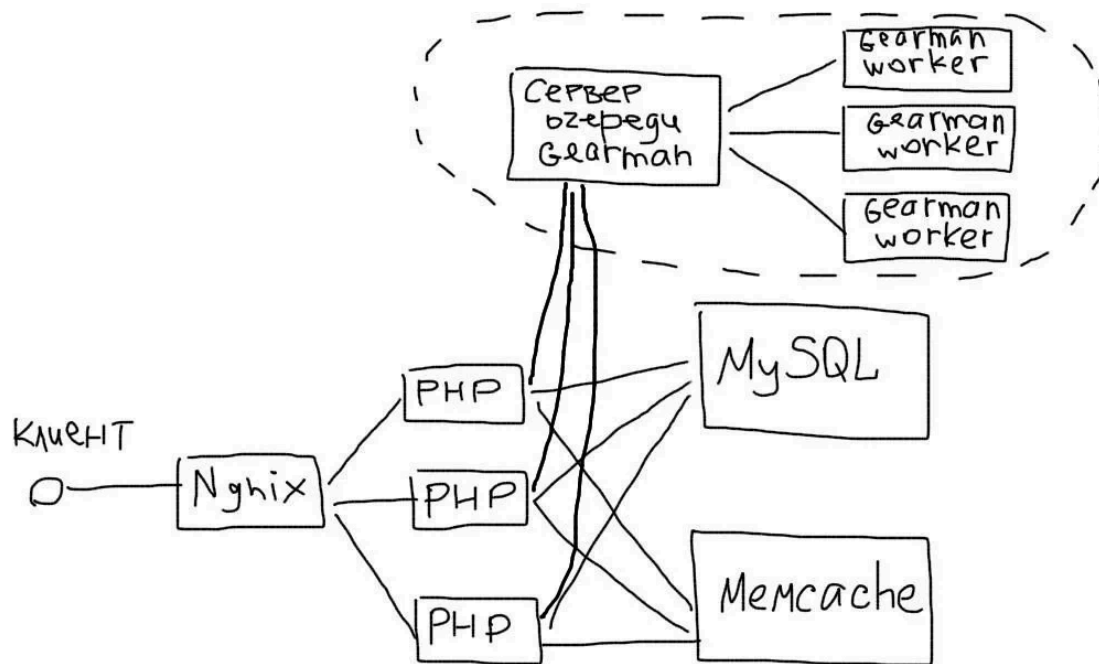




# Кэширование



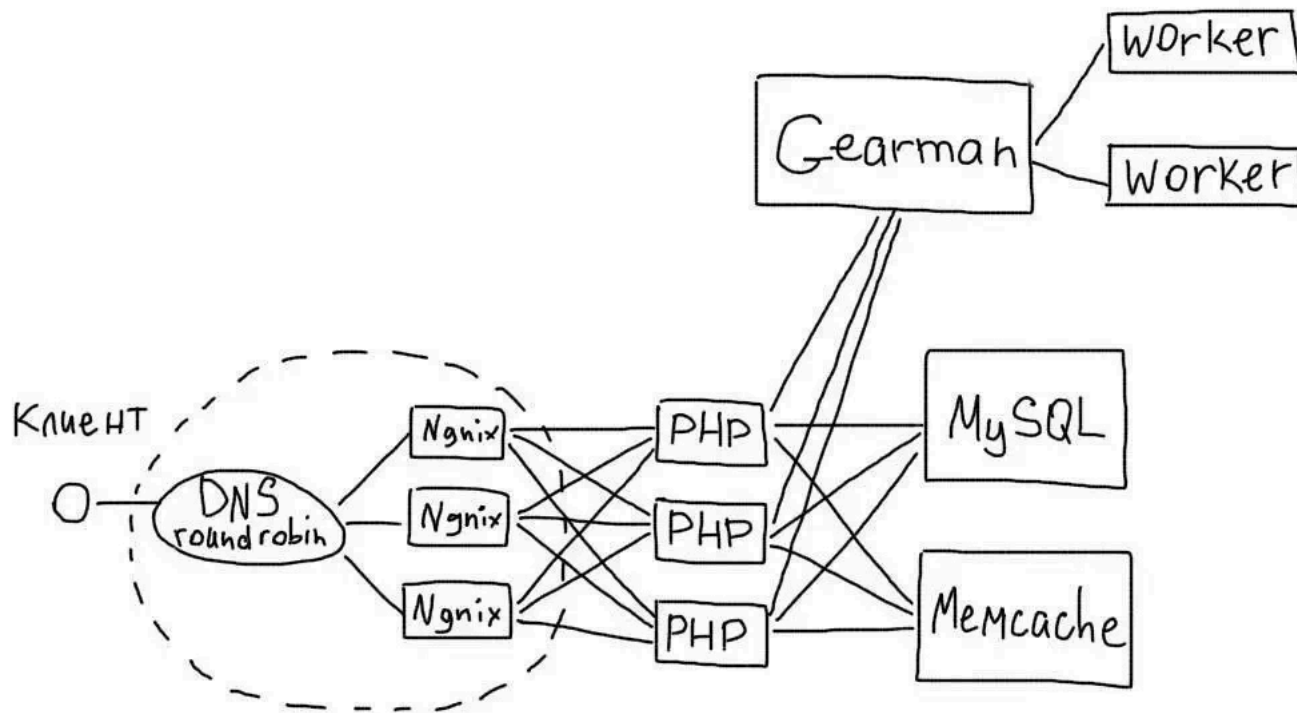
# Очереди задач



 RabbitMQ™

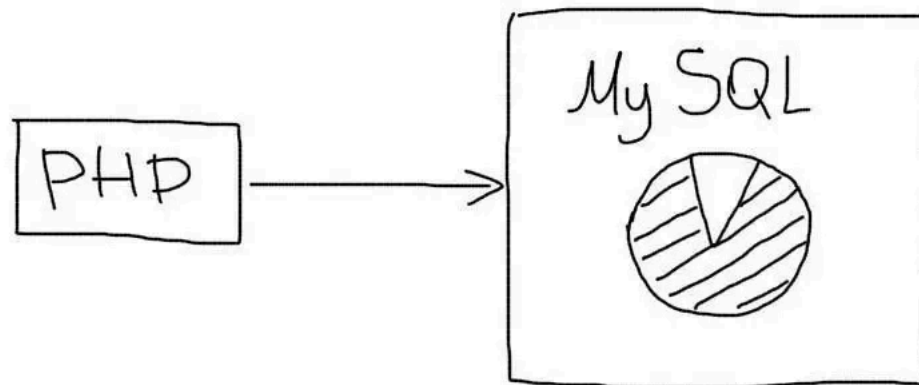


# Балансировка DNS

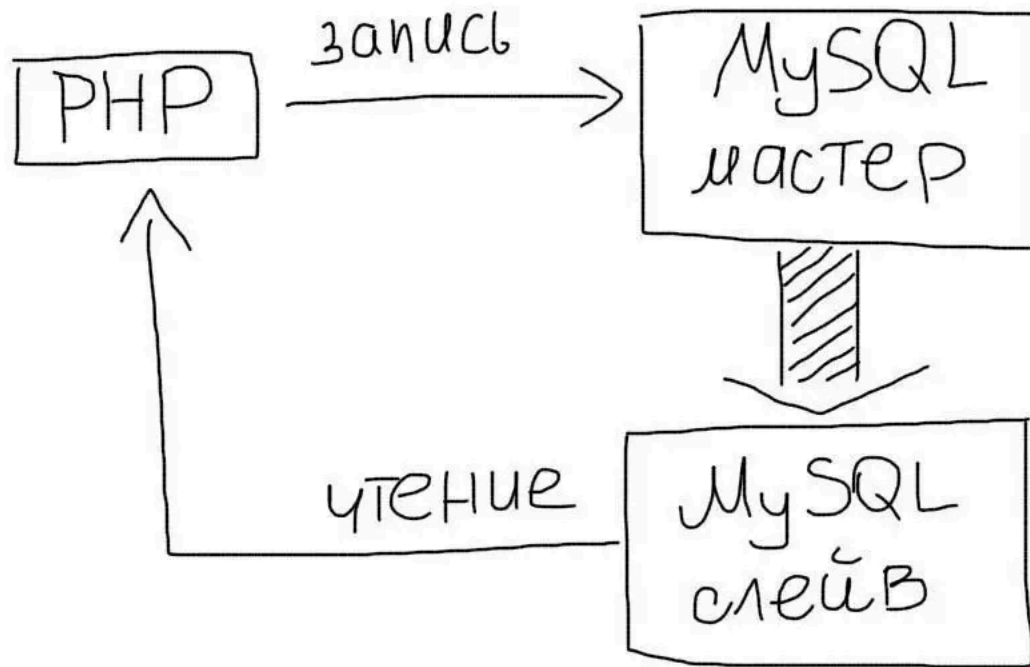


# Шардинг и репликация

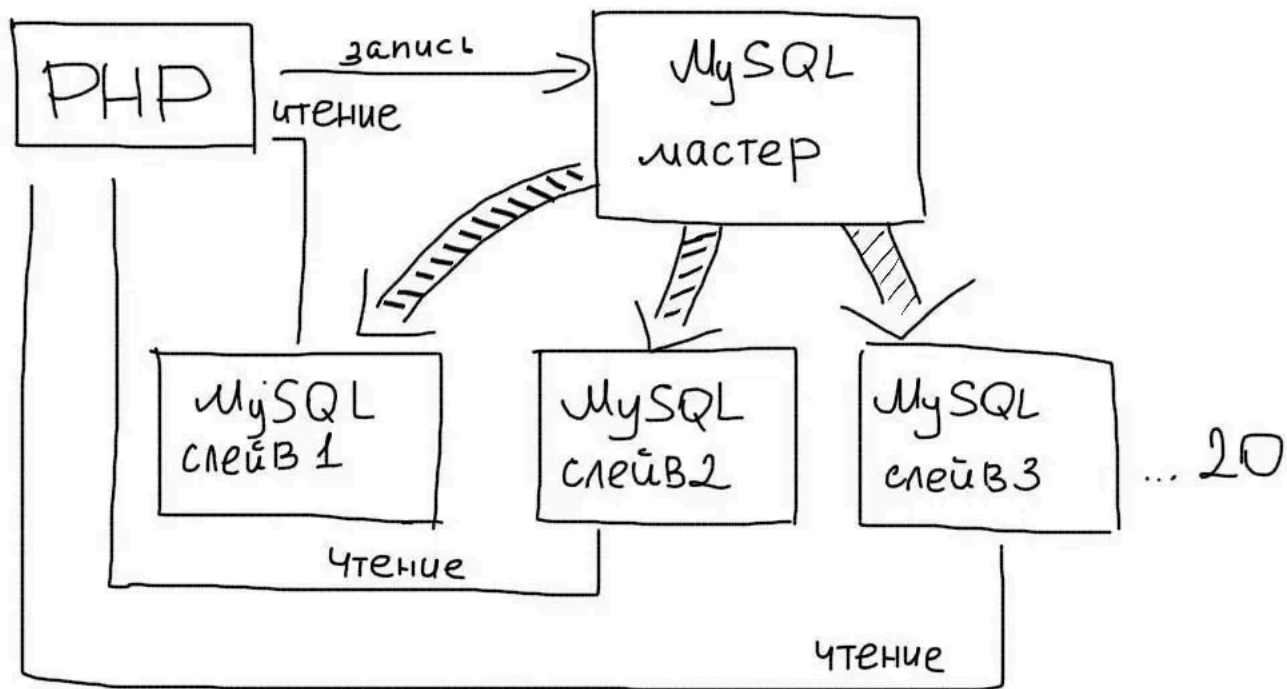
Классическая схема работы приложения с базой данных



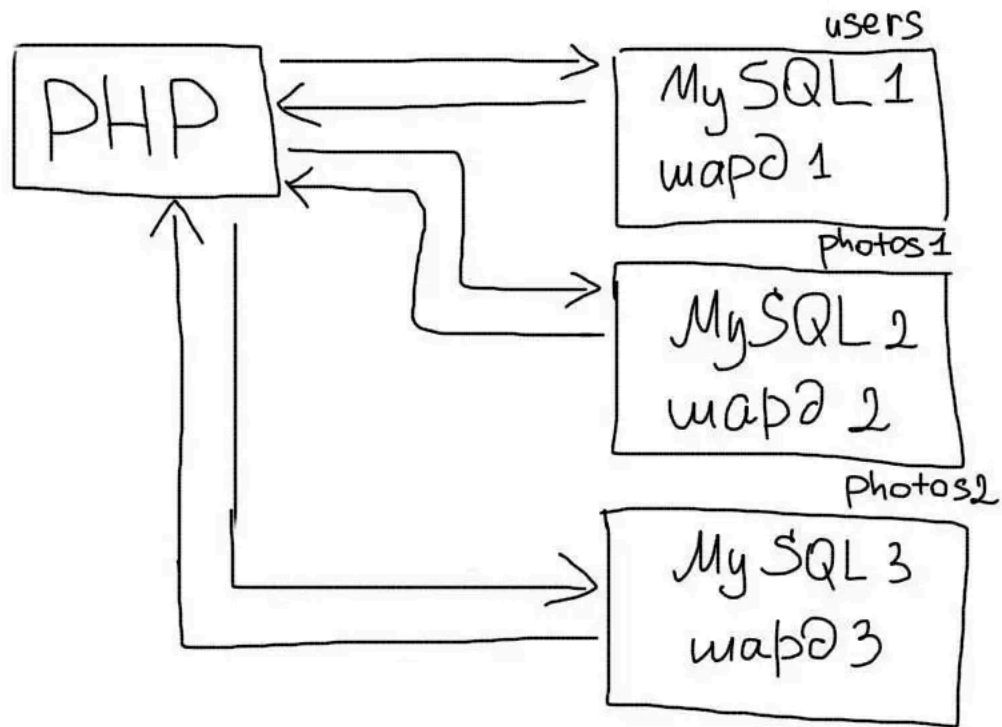
# Репликация (1)



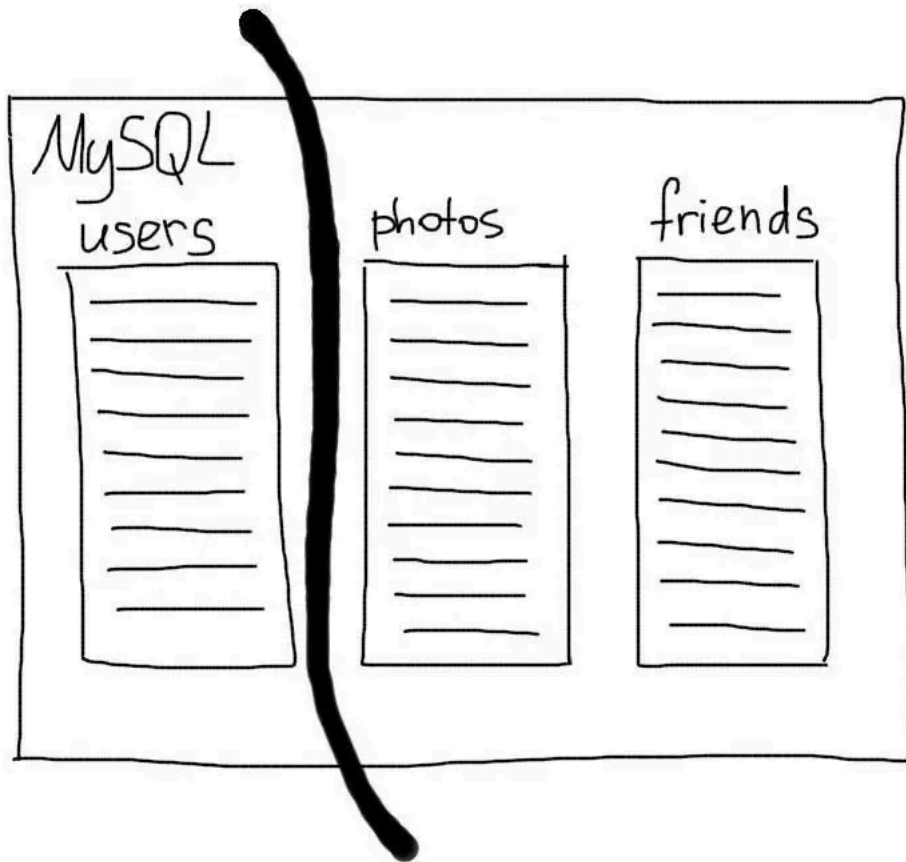
## Репликация (2)



# Шардинг

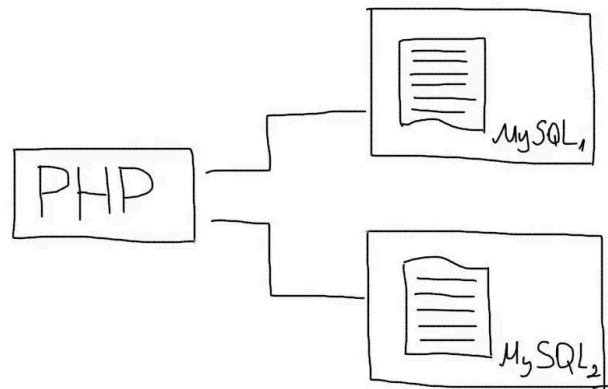
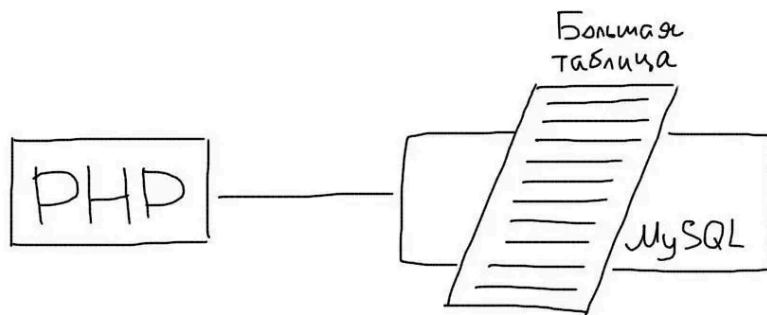


# Вертикальный шардинг





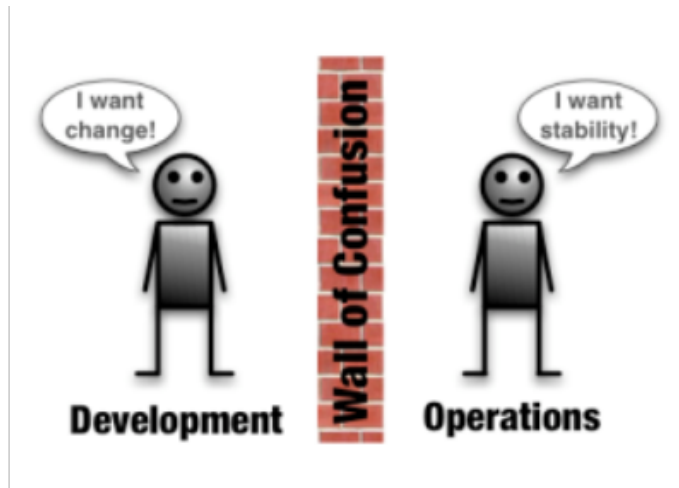
# Горизонтальный шардинг



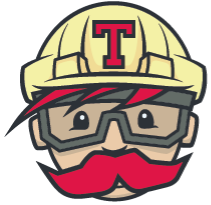
# Agenda

- Что такое web?
- Как устроен простейший веб-сервис?
- Что такое асинхронный веб-сервер?
- Как устроен высоконагруженный веб-сервис?
- Облака и DevOps.

# Методология DevOps



# Continuous Integration / Continuous delivery (CI/CD)



**Travis CI**

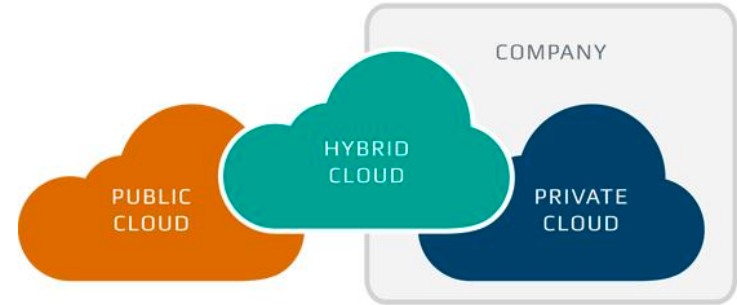
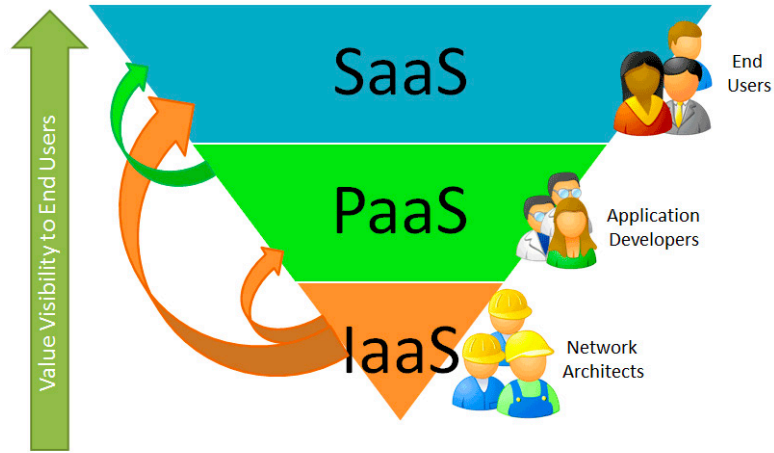


**Jenkins**



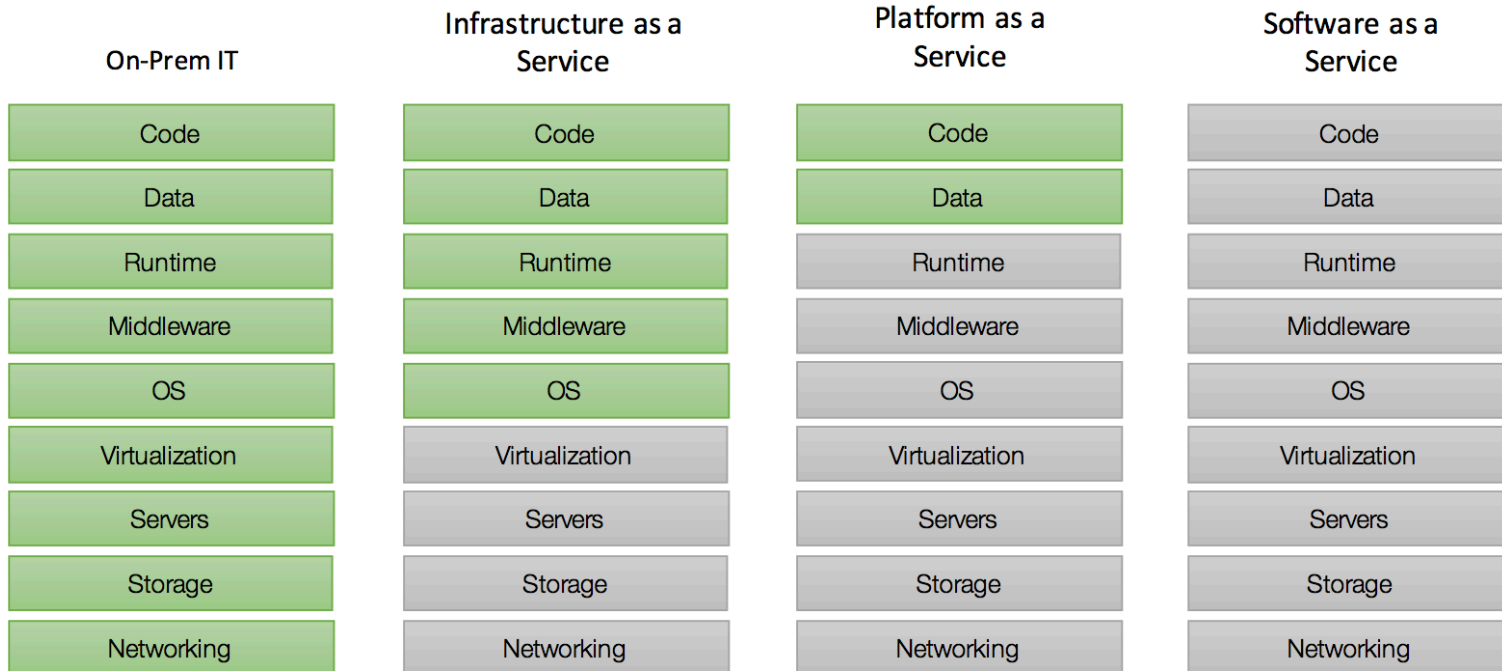
**CI  CD**

# Clouds



# Clouds

- Customer Managed
- Service Provider Managed



# Thank you

Denis Tribushkov  
Technical Consultant

—

[tribushkov@ru.ibm.com](mailto:tribushkov@ru.ibm.com)