Как стать первоклассным программистом: чем он отличается от кодера

Олег Бунин

HighLoad++

Олег Бунин

- Руководитель отдела веб-разработки компании Рамблер (ещё тогда, когда Рамблер был номером один);
- Руководитель компании по разработке и консалтингу в области высоконагруженных проектов:
  - Команда специалистов под технологии заказчика;
  - От украинской системы анализа комментариев до исламской социальной сети ☺️
- Председатель Программного комитета конференции разработчиков высоконагруженных систем HighLoad++ вот уже 11 лет;


ВКонтакте


sports.ru


ИНТЕРНЕТ ДЛЯ ЖЕНЩИН
WOMAN.ru


фото.сайт


Eva.Ru
Каждый день


SETUP


LOVE TIME


РИАНОВОСТИ


ЭЛЬДОРАДО


ЗДОРОВАЯ
РОССИЯ


НА ВКУС И ЦВЕТ
есть
ИМХОНЕТ


1С·БИТРИКС


ИТАР
ТАСС

Вы набираете в браузере http://highload.ru/. Что реально происходит?

1. Браузер отправляет запрос на сервер;
2. Сервер получает запрос;
3. Сервер запрашивает данные из базы данных;
4. Сервер строит страничку;
5. Сервер отправляет страничку браузеру;
6. Браузер строит страничку на экране для пользователя.

Вы набираете в браузере http://highload.ru/. Что реально происходит?

1. Браузер запрашивает DNS и вычисляет IP для highload.ru;
2. Браузер отправляет запрос на сервер;
3. Сервер получает запрос;
4. Сервер проверяет в кеше, есть ли такая страничка, если есть, то сразу к пункту 7;
5. Сервер запрашивает данные из базы данных;
6. Сервер строит страничку;
7. Сервер отправляет страничку браузеру;
8. Браузер строит страничку на экране для пользователя.

Вы набираете в браузере http://highload.ru/. Что реально происходит?

1. Браузер проверяет в локальном кэше, есть ли такая страничка;
2. Если есть, то проверяет заголовки и параметры кэширования (Expires), можно ли её использовать?
3. Браузер запрашивает DNS и вычисляет IP для highload.ru;
4. Браузер составляет http-запрос;
5. Браузер (прикладной уровень модели OSI) передаёт запрос ниже, транспортному уровню;
6. Транспортный уровень сетевой подсистемы упаковывает запрос в TCP-пакеты и отправляет ещё ниже, сетевому уровню;
7. Сетевой и нижележащие уровни отправляют запрос в сеть согласно таблицам маршрутизации;
8. Где-то в сети потенциальные кэширующие сервера проверяют, не могут ли они выполнить этот запрос без передачи дальше по маршруту:
9. Где-то в сети обрабатываются проверки блокировок от Роскомнадзора;
10. TCP/IP-пакеты передаются от сервера к серверу согласно таблицам маршрутизации и, наконец, доходят до сетевого уровня целевого сервера;
11. Сетевой и транспортный уровни сервера собирают запрос из пакетов;
12. Проверяется, может ли порт (в данном случае 80-й) принять запрос, есть ли место в очереди, слушает ли кто-нибудь этот порт;
13. Если порт слушается, то устанавливается сессия и запрос передаётся тому веб-серверу, который слушает порт;
14. В нашем случае это nginx, который аллоцирует буфер для принятия запроса;
15. Если буфера в памяти не хватает, то создаётся временный файл на диске;
16. Дисковая подсистема операционной системы решает, где создать временный файл – в буфере или реально на диске;
17. Nginx принимает весь запрос;
18. Только после этого nginx проверяет, требуется ли выполнение запроса или он может обработать его без передачи бекенду (например, он в кэше или это картинка);
19. Если запрос требует вычисления, то открывается upstream к бэкенду;
20. Сервер, который слушает порт бэкенда открывает соединение и принимает запрос;
21. В нашем случае это Apache, последовательно выполняются 11-ть стадий выполнения запроса: трансляция в URL, проверка заголовков, вычисление скрипта, который должен обработать запрос, запуск скрипта, передача скрипту информации запроса, ожидание выполнения скрипта, отдача результатов nginx'у, логгирование, этап очистки;
22. Скрипт получает запрос, запускается построение страницы;
23. Выполняется программа, данные с помощью SQL-запросов запрашиваются из базы данных;
24. В каждом случае устанавливается соединение с базой данных (если оно не постоянное);
25. База данных получает SQL-запрос;
26. База данных строит план выполнения запроса, затем выполняет его, используя те или иные буферы;
27. При каждом чтении таблицы сначала база данных проверяет, нет ли этой информации в кэше, затем дисковая подсистема делает тоже самое;
28. Скрипт собирает данные от базы данных и вставляет их в шаблон страницы;
29. Скрипт отдаёт собранную страницу веб-серверу Apache;
30. Apache передаёт собранную страницу nginx;
31. Nginx принимает страницу и только полностью записав её в локальный буфер начинает передачу пользователю. Всё это время соединение с пользователем открыто;
32. Собранная страница в обратном порядке спускается вниз по уровням модели OSI, бьется для передачи по сети на кусочки;
33. Сетевой и транспортный уровень машины пользователя собирают страницу и передают её выше;
34. Браузер парсит страницу и запускает аналогичный процесс для всех её составляющих: JS-файлов, CSS-файлов, картинок. Для каждого элемента весь процесс повторяется;
35. Только после получения всего этого начинается процесс построения странички для пользователя.
36. PS: Браузер пытается закэшировать всё, что можно.

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм, понятие SPF, закон Амдала, понятия shared nothing и stateless;

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм, понятие SPF, закон Амдала, понятия shared nothing;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;

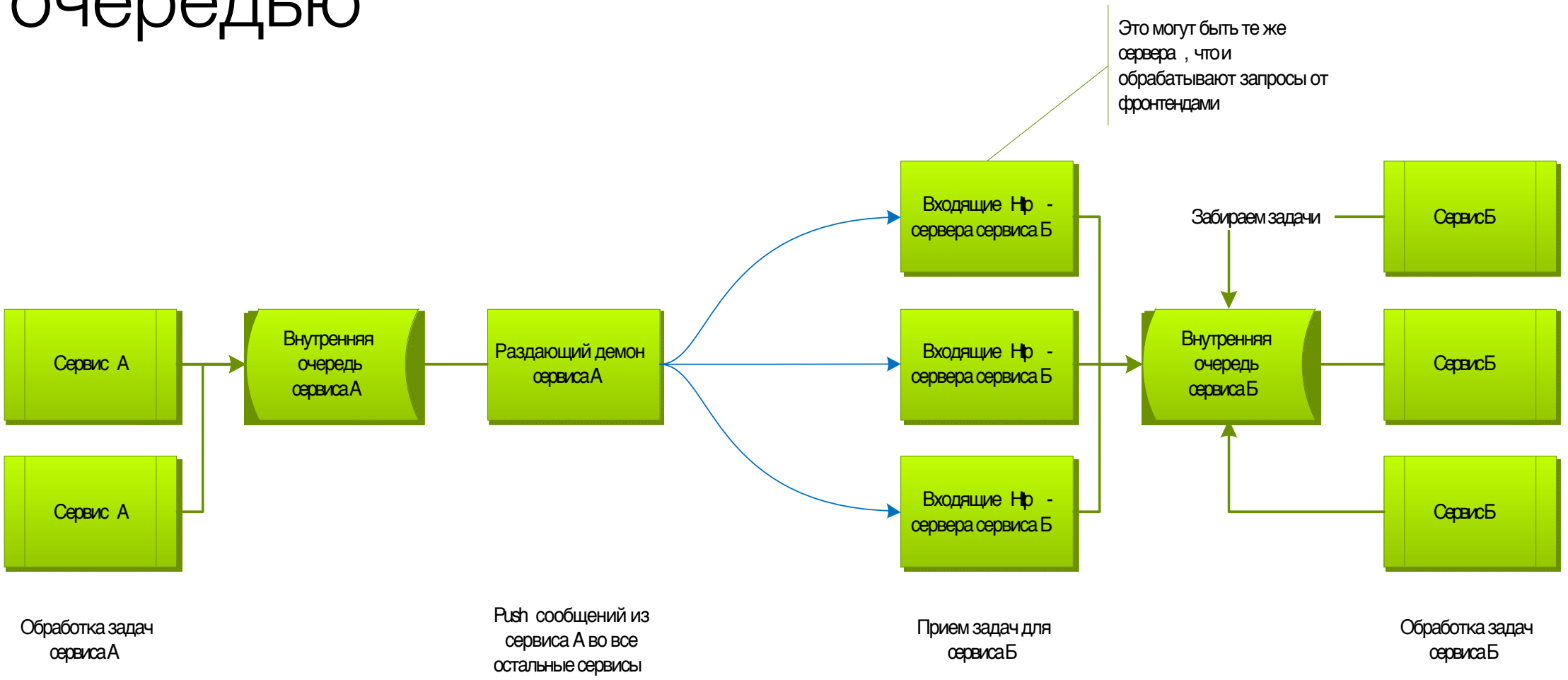
Архитектурные паттерны

1. Трёхзвенная структура
2. Кэширование
3. Толстый клиент
4. Деградация функциональности
5. Вертикальное масштабирование
6. Функциональное разделение
7. Горизонтальное масштабирование
8. SOA / микросервисы
9. Монолитное приложение
10. Отложенные вычисления
11. Асинхронная обработка
12. Конвейер
13. Репликация
14. Вертикальный шардинг
15. Горизонтальный шардинг
16. Виртуальные шарды
17. Центральный диспетчер
18. Партиционирование
19. Денормализация
20. Введение избыточности
21. Параллельное выполнение
22. Специализированные сервера
23. Comet-сервер

Кэширование – что может быть проще?

- Кэширование в браузере (выставляем правильные заголовки);
- Кэширование HTML-блоков;
- Кэширование данных;
- Кэширование HTML-страниц (например, с помощью nginx);
- Процедура одновременного перестроение кэшей;
- Когерентность кэша;
- Схема инвалидации кэшей;
- Учитывание зависимости элементов кэша (например, с помощью тегирования);
- Старт с холодным кэшем.

Интеркоммуникация / решение с очередью



Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;
- **Внутреннее устройство СУБД**: что означают различные буферы, как база данных выполняет ваш запрос;

Устройство СУБД

- MySQL и PostgreSQL – разные типы многозадачности, следовательно, разные типы буферов и разные возможности;
- Настройки по умолчанию в PostgreSQL;
- Достаточны ли буферы сортировки в PostgreSQL?
- Основные паттерны баз данных: репликация, шардирование, партиционирование;
- Теория СУБД: денормализация (избыточность) / нормализация;
- **Язык SQL vs ORM**. Бездумное использование ORM - одна из основных проблем низкого качества программного кода;

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;
- **Внутреннее устройство СУБД**: что означают различные буферы, как база данных выполняет ваш запрос;
- **Логика работы фронтенда**;

Логика работы фронтенда

- Понимая логику работы фронтенда можно подготовиться к запросам заранее, например, запрос дополнительных постов в френдленте;
- Протоколы работы с фронтендом: AJAX Long polling, WebSocket и другие.

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;
- **Внутреннее устройство СУБД**: что означают различные буферы, как база данных выполняет ваш запрос;
- **Логика работы фронтенда**;
- **Понимание работы операционной системы**: какие буферы и кэши и очереди, где они работают, на каких стадиях, как их настроить?

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;
- **Внутреннее устройство СУБД**: что означают различные буферы, как база данных выполняет ваш запрос;
- **Логика работы фронтенда**;
- **Понимание работы операционной системы**: что такое мультизадачность, какие архитектуры мультизадачности бывают;
- **Понимание работы операционной системы**: какие буферы и кэши и очереди, где они работают, на каких стадиях, как их настроить? Что такое индексы и почему это не панацея?

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;
- **Внутреннее устройство СУБД**: что означают различные буферы, как база данных выполняет ваш запрос;
- **Логика работы фронтенда**;
- **Понимание работы операционной системы**: что такое мультизадачность, какие архитектуры мультизадачности бывают;
- **Понимание работы операционной системы**: какие буферы и кэши и очереди, где они работают, на каких стадиях, как их настроить? Что такое индексы и почему это не панацея?
- **Веб-сервер**: выбранная схема обработки запросов в веб-сервере?

Стадии обработки запроса в Apache

- Инициализация дочернего процесса — фаза “**Child Init**”;
- Стадия “**Post-Read-Request**” — этап после-чтения запроса;
- Стадия “**URI Translation**” — этап трансляции URL в URI, в имя локального ресурса;
- Стадия “**Header Parsing**” — этап разбора заголовка;
- Стадия “**Access Control**” — этап проверки доступа;
- Стадия “**Authentication**” — этап идентификации и стадия “Authorization” — этап авторизации;
- Стадия “**Mime type checking**” — этап обработки и проверки MIME-типа;
- Стадия “**FixUp**”;
- Стадия “**Response**” (Content phase) — этап обработки запроса, создания ответа;
- Стадия “**Logging**” — этап записи информации в логи;
- Стадия “**Cleanup**” — этап очистки, выполняется уже ПОСЛЕ отдачи ответа пользователю;
- Стадия “**ChildExit**”.

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;
- **Внутреннее устройство СУБД**: что означают различные буферы, как база данных выполняет ваш запрос;
- **Логика работы фронтенда**;
- **Понимание работы операционной системы**: что такое мультизадачность, какие архитектуры мультизадачности бывают;
- **Понимание работы операционной системы**: какие буферы и кэши и очереди, где они работают, на каких стадиях, как их настроить? Что такое индексы и почему это не панацея?
- **Веб-сервер**: выбранная схема обработки запросов в веб-сервере?
- **Смежные области**: понимание основных алгоритмов DDoS-атак, сетевая модель OSI, понимание сетевой инфраструктуры;

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;
- **Внутреннее устройство СУБД**: что означают различные буферы, как база данных выполняет ваш запрос;
- **Логика работы фронтенда**;
- **Понимание работы операционной системы**: что такое мультизадачность, какие архитектуры мультизадачности бывают;
- **Понимание работы операционной системы**: какие буферы и кэши и очереди, где они работают, на каких стадиях, как их настроить? Что такое индексы и почему это не панацея?
- **Веб-сервер**: выбранная схема обработки запросов в веб-сервере?
- **Смежные области**: понимание основных алгоритмов DDoS-атак, сетевая модель OSI, понимание принципов построения сетевой инфраструктуры;
- **Смежные области**: понимание принципов балансировки нагрузки.

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;
- **Внутреннее устройство СУБД**: что означают различные буферы, как база данных выполняет ваш запрос;
- **Логика работы фронтенда**;
- **Понимание работы операционной системы**: что такое мультизадачность, какие архитектуры мультизадачности бывают;
- **Понимание работы операционной системы**: какие буферы и кэши и очереди, где они работают, на каких стадиях, как их настроить? Что такое индексы и почему это не панацея?
- **Веб-сервер**: выбранная схема обработки запросов в веб-сервере?
- **Смежные области**: понимание основных алгоритмов DDoS-атак, сетевая модель OSI, понимание принципов построения сетевой инфраструктуры;
- **Смежные области**: понимание принципов балансировки нагрузки;
- **Информационная безопасность:** начиная от экранирования переменных и до SQL- и Memcached-инъекций;

Что нужно знать?

- **Основные алгоритмы и теорию**: сортировки, требования ACID, CAP-теорему, MVCC-механизм;
- **Программные примитивы, паттерны программирования**: слои абстракции, объектно-ориентированное программирование, разбиение на программные модули;
- **Архитектурные примитивы**: 20 архитектурных паттернов;
- **Средства профилирования**: dtrace, NewRelic;
- **Внутреннее устройство СУБД**: что означают различные буферы, как база данных выполняет ваш запрос;
- **Логика работы фронтенда**;
- **Понимание работы операционной системы**: что такое мультизадачность, какие архитектуры мультизадачности бывают;
- **Понимание работы операционной системы**: какие буферы и кэши и очереди, где они работают, на каких стадиях, как их настроить? Что такое индексы и почему это не панацея?
- **Веб-сервер**: выбранная схема обработки запросов в веб-сервере?
- **Смежные области**: понимание основных алгоритмов DDoS-атак, сетевая модель OSI, понимание принципов построения сетевой инфраструктуры;
- **Смежные области**: понимание принципов балансировки нагрузки;
- **Информационная безопасность:** начиная от экранирования переменных и до SQL- и Memcached-инъекций;
- **Инфраструктура разработки:** системы контроля версий, понимание принципов работы DevOps;

Где это всё узнать?

- В первую очередь: конференции HighLoad++ (https://highload.ru/) и BackendConf (https://backendconf.ru/);

Где это всё узнать?

- В первую очередь: конференции HighLoad++ (https://highload.ru/) и BackendConf (https://backendconf.ru/);
- Обучающие митапы (https://www.meetup.com/HighLoad-User-Group/);

Где это всё узнать?

- В первую очередь: конференции HighLoad++ (https://highload.ru/) и BackendConf (https://backendconf.ru/);
- Обучающие митапы (https://www.meetup.com/HighLoad-User-Group/);
- Материалы прошлых конференций: https://www.youtube.com/user/profyclub

Где это всё узнать?

- В первую очередь: конференции HighLoad++ (https://highload.ru/) и BackendConf (https://backendconf.ru/);
- Обучающие митапы (https://www.meetup.com/HighLoad-User-Group/);
- Материалы прошлых конференций: https://www.youtube.com/user/profyclub
- Обучающий email-курс: http://highload.guide/

Где это всё узнать?

- В первую очередь: конференции HighLoad++ (https://highload.ru/) и BackendConf (https://backendconf.ru/);
- Обучающие митапы (https://www.meetup.com/HighLoad-User-Group/);
- Материалы прошлых конференций: https://www.youtube.com/user/profyclub
- Обучающий email-курс: http://highload.guide/
- Расшифровки докладов: https://habr.com/ru/company/oleg-bunin/

Где это всё узнать?

- В первую очередь: конференции HighLoad++ (https://highload.ru/) и BackendConf (https://backendconf.ru/);
- Обучающие митапы (https://www.meetup.com/HighLoad-User-Group/);
- Материалы прошлых конференций: https://www.youtube.com/user/profyclub
- Обучающий email-курс: http://highload.guide/
- Расшифровки докладов: https://habr.com/ru/company/oleg-bunin/
- **Конечно!** Обучающие курсы, например, https://course.skillbox.ru/webdev/

Как учиться самому?

Советы от **Ильи Космодемьянского**:

1. Прочитать документацию целиком хотя бы один раз;
2. Поскольку 1. никто никогда не делает, можно принимать полумеры, которые все равно помогают. Например прочитать доку по всем параметрам в конфиге (в случае посгреса очень хорошо работает);
3. Внимательно читать все релиз нотс даже на минорные релизы и где применимо проверять на практике фичи;
4. Учится делать все стандартные процедуры быстро. Например последний релиз посгреса это около 70 фиксов - если хочешь пощупать руками хотя бы 10 из них, операции по разворачиванию кластера (например) должны быть автоматическими;
5. Рассказывать кому-нибудь о новых для тебя технологиях. Хочешь разобраться - напиши статью в блог, на хабр, расскажи коллегам на семинаре, подай доклад куда-нибудь.

Вопросы?

Олег Бунин
oleg.bunin@ontico.ru
https://www.facebook.com/oleg.bunin
https://vk.com/oleg.bunin