Кивер Данила Андреевич

danila.kiver@mail.ru или +7-909-645-38-84

Образование: МГУ имени М. В. Ломоносова, физический факультет, кафедра компьютерных методов физики, бакалавр (2015).

Коммуникация: upper-intermediate английский; есть опыт работы в англоязычных командах; есть опыт работы в распределенных командах с разбросом во временных зонах до 10 часов.

Публичные профили (кликабельно): StackOverflow, Unix StackExchange, GitHub.

1 Опыт работы

Общий стаж — 9 лет.

- **Творческий отпуск**
  январь 2024 – н. в., в стаж не входит

  – Путешествия, языки, книги, pet-проекты и расширение технического кругозора.

- **Старший разработчик, команда Архитектура**
  МойСклад (`https://moysklad.ru` и `https://kladana.com`), июль 2020 – декабрь 2023

  – Спроектировал систему управления окружениями и деплойментами для всей компании с учетом нюансов существующих в компании процессов.

  – Спроектировал новый вид девелоперских окружений для тестирования изменений с реальными данными по отдельному шарду (`microstage`, известный внутри компании как `minipig`) и реализовал с нуля инструментарий для управления этими окружениями: подготовки данных для окружения, оркестрации деплоймента и т. д.; разгрузил основное `stage`-окружение.

  – Внедрил разделение сборочной фермы на зоны под разные виды задач для оптимизации распределения нагрузки.

  – Внедрил использование статического анализа для прикладных сборок (spotbugs / maven dependency analyzer / hadolint / shellcheck), перевел все интеграционные тесты на использование testcontainers.

  – Участвовал в переработке системы построения отчетов (вынесении отчетов из основной транзакционной БД в отдельную аналитическую БД) — проводил эксперименты с разными технологиями с замерами и анализом результатов; оптимизировал производительность прототипа новой системы.

  – Выполнял архитектурный надзор за изменениями, проектируемыми и разрабатываемыми продуктовыми командами: ревьюил архитектурные проекты и реализации новых прикладных сервисов.

  – Проводил технические интервью senior-разработчиков и техлидов.

  – Ни разу не уронил прод, имея полные доступы 

- **Senior Software Engineer**
  EPAM Systems, май 2018 – декабрь 2019

  – Проект PERF: мигрировал решение (Java-монолит на SpringBoot + PostgreSQL) в облако (VM с Ubuntu Server → GKE).
  – Проект Axway Syncplicity (`https://www.syncplicity.com`): разрабатывал и поддерживал бэкенд системы (message queue, content search, cloud storage) и сервисы, устанавливаемые on-premises (on-prem storage). Технический стек — Java-сервисы на SpringBoot + MariaDB на VM с CentOS 7 в AWS. Legacy-сервисы на Scala + Play Framework.
  – Участвовал во внутренней программе менторинга в роли ментора.
  – Выступал на технических митапах, в т. ч. на внешнем (тема — детали реализации Linux-контейнеров и используемые ими интерфейсы ядра).

- **Software Engineer / Senior Software Engineer**
  NetCracker Technology, ноябрь 2014 – май 2018

  – Разработал прототип системы Zero Touch Provisioning для проекта SD-WAN. Технический стек — Java-сервисы на SpringBoot + PostgreSQL в OpenShift Origin.
  – Разработал и поддерживал HA-дистрибутив PostgreSQL на базе фреймворка Patroni.
  – Мигрировал существующее решение (Java-монолит на WildFly + Spring + PostgreSQL) в облачное окружение (OpenShift Origin).
  – Развивал и поддерживал HA-инфраструктуру на базе Pacemaker.
  – Развивал и поддерживал низкоуровневые Java-библиотеки (persistence, history, platform abstraction) корпоративной платформы.

2 Основные технологии

- Java (основной язык), Spring / SpringBoot, PostgreSQL.
- Linux: 9 лет в проде с RHEL-like дистрибутивами; 15+ лет по жизни со всем остальным.
  на рабочей станции **исключительно** Linux с максимально кастомизированной под себя конфигурацией; Windows и Mac не использую в качестве основной системы ни при каких условиях.
- Ansible, Docker, K8s и OKD (включая администрирование).
- OpenStack, AWS.

3 Мы совместимы?

Небольшой список из утверждений для проверки на культурную совместимость:

✓ Разработчик *решает проблемы пользователей*, а не слепо и механически пишет код по ТЗ.

Используемые инструменты, технологии и формализованное ТЗ — не догма и могут меняться в ходе работы, если это решит проблемы пользователей *лучше*.

✓ использование слоев абстракций не снимает с разработчика обязанности знать или быть готовым разобраться, что происходит внутри используемых технологий.

Абстракции текут, и когда это происходит, зачастую проще залезть в исходники, чем отлаживать черный ящик;

✓ CI/CD — это подход к организации процесса разработки, а не конкретные инструменты и факт их использования сам по себе;

✓ DevOps — это культура и методология разработки, применяемая командой, а не выделенная должность разработчика пайплайнов и манифестов.

There's No Such Thing as a "Devops Team";

✓ по указанным выше причинам корректное применение практик CI/CD и DevOps невозможно без надлежащего образа мышления в команде разработки, без глубокого понимания этой командой используемых инструментов, инфраструктуры под разрабатываемыми приложениями и процессов разработки;

✓ код, как и история проекта, пишется единожды, а читается сотни раз:

- производительный, но нечитаемый и неподдерживаемый код — мусорный код;
- история проекта без грамотного разбиения на минимальные и атомарные коммиты со внятными commit message'ами — мусорная история;

✓ удаленка — это когда никого не волнует, в какой точке планеты, как долго и зачем ты находишься, как часто и куда перемещаешься. Важен только продукт, который ты создаешь.