нетология

DevOps и SRE

Алексей
Метляков

Алексей Метляков

DevOps Engineer

OpenWay

Алексей Метляков

План занятия

1. DevOps
2. Структуры команд DevOps
3. Антипаттерны
4. SRE
5. Итоги
6. Домашнее задание

DevOps

DevOps

DevOps - идеологически был выведен из принципов **agile.** Не имеет точного определения и до сих пор интерпретируется индивидуально.

- Часто DevOps считают системным администрированием;
- Не реже DevOps называют процессом непрерывной поставки;
- Кто-то считает, что DevOps - это поддержка разработки ПО.

И каждое из этих мнений является равнозначно истинным. В целом, **DevOps** можно назвать мультиинструментом для обеспечения работоспособности конвейера жизненного цикла ПО, включающего в себя не только сопровождение сборки, тестирования, поставки продукта, но и управление командой в части использования всех этих процессов.

DevOps

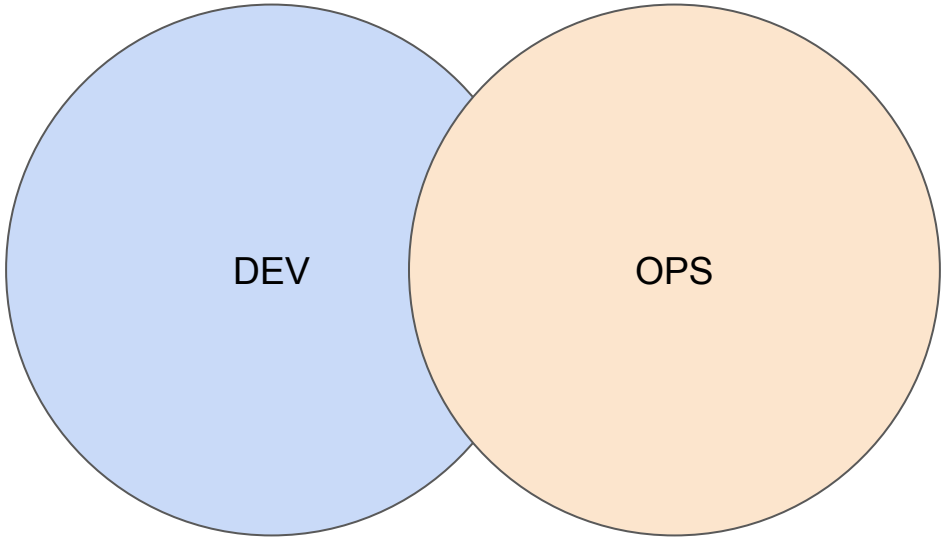

Непрерывная разработка
Непрерывная интеграция
Непрерывная поставка
Непрерывное тестирование
Управление окружением
Управление состоянием

DevOps

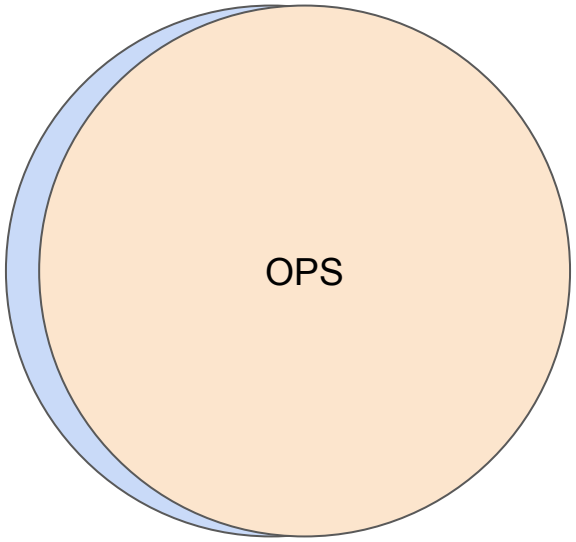

Непрерывная разработка
Непрерывная интеграция
Непрерывная поставка
Непрерывное тестирование
Управление окружением
Управление состоянием

Структуры команд DevOps

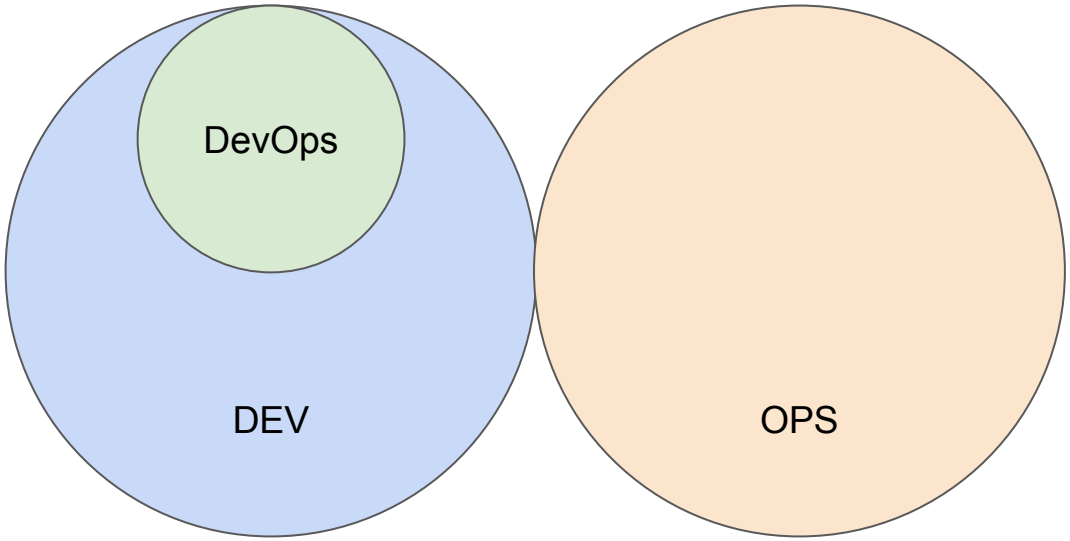
Сотрудничество dev и ops



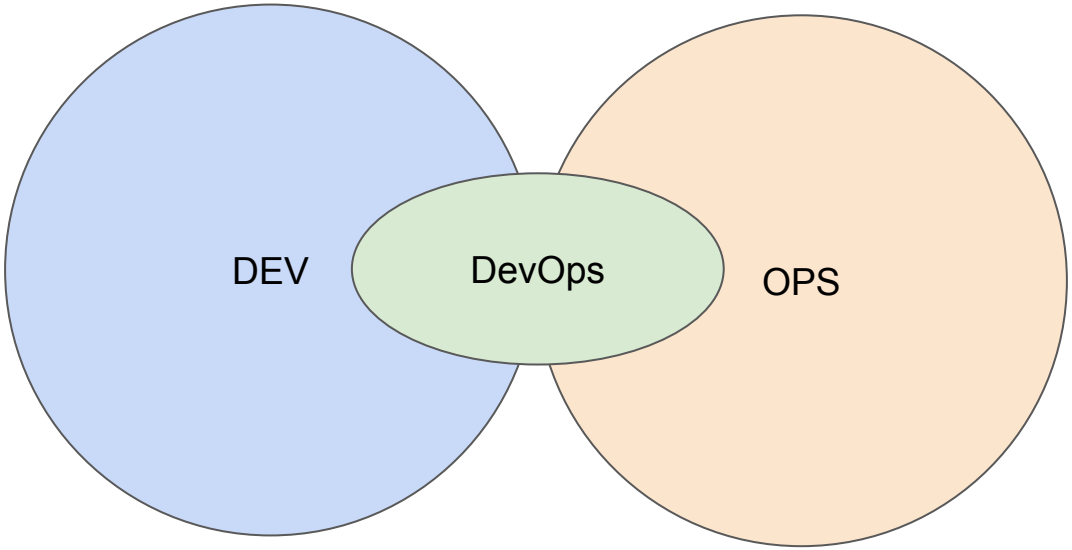
NoOPs



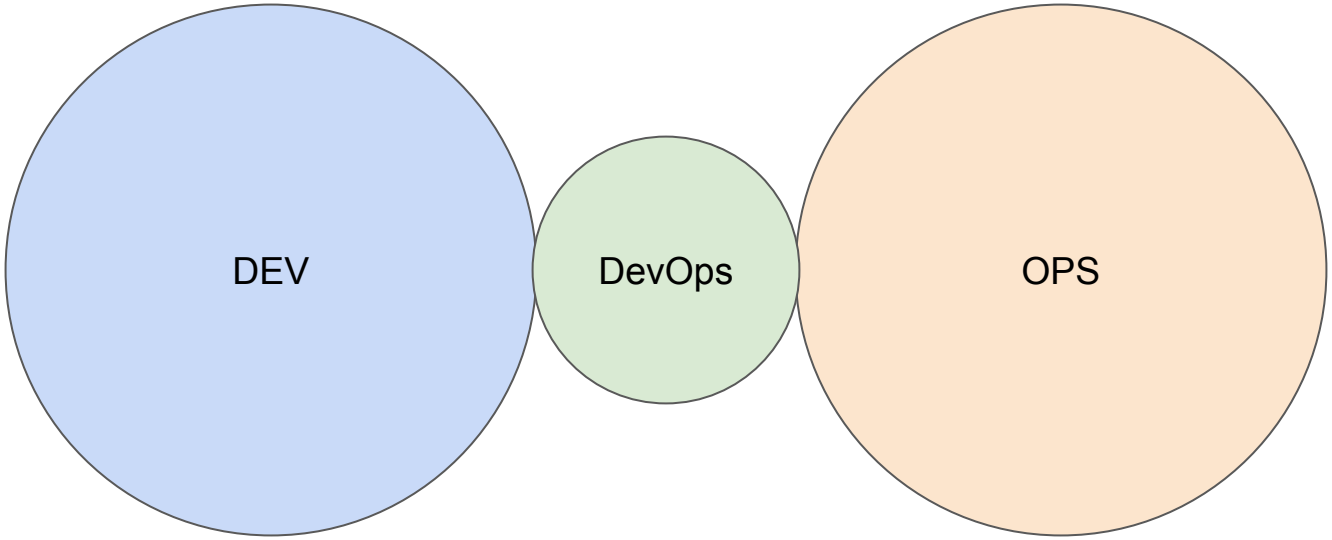
IaaS (Инфраструктура как сервис)



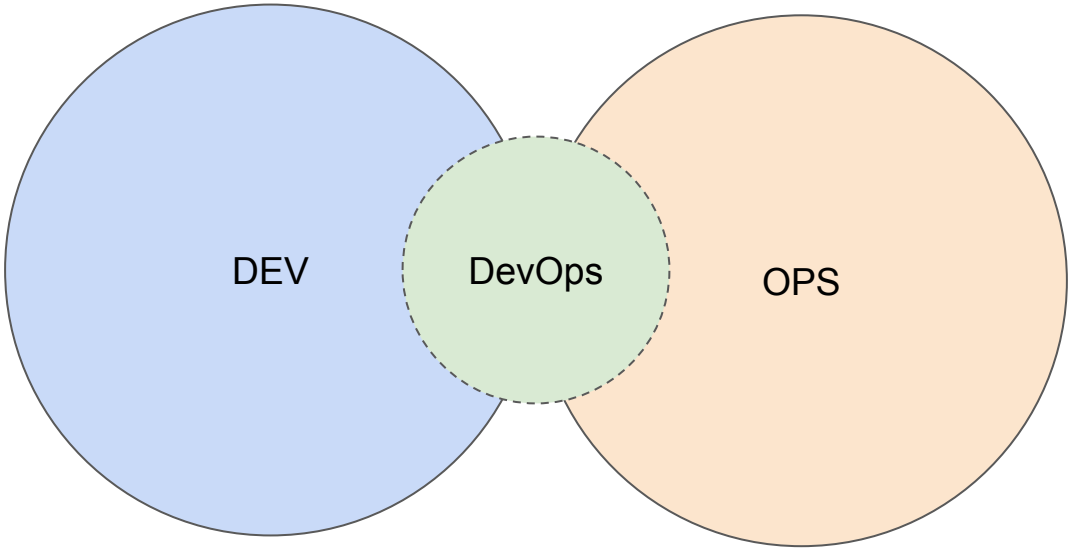
DevOps как внешний сервис



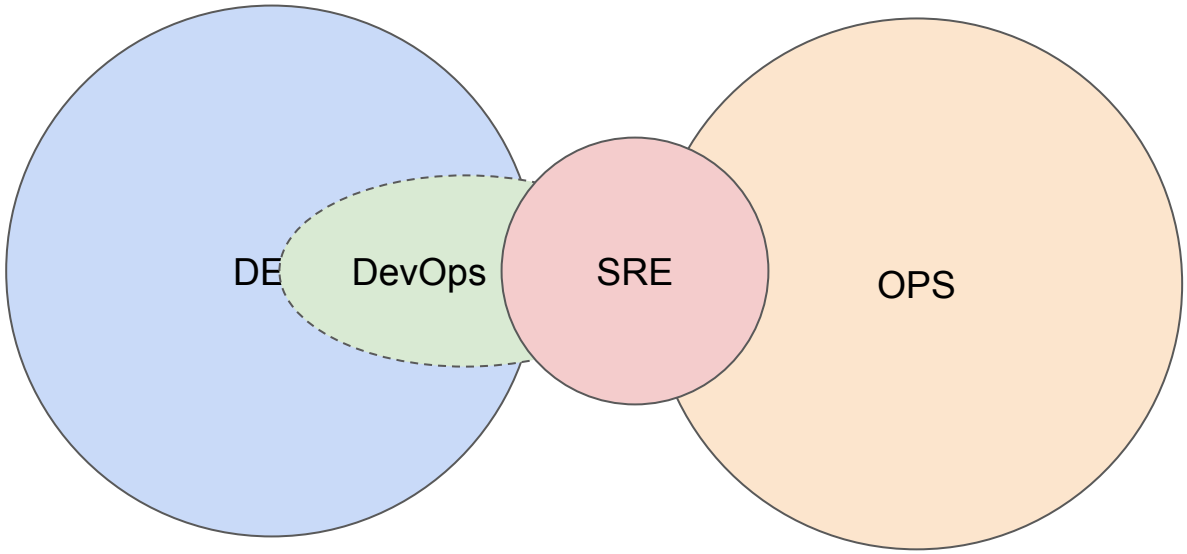
DevOps отдельная команда с ограниченным сроком действия



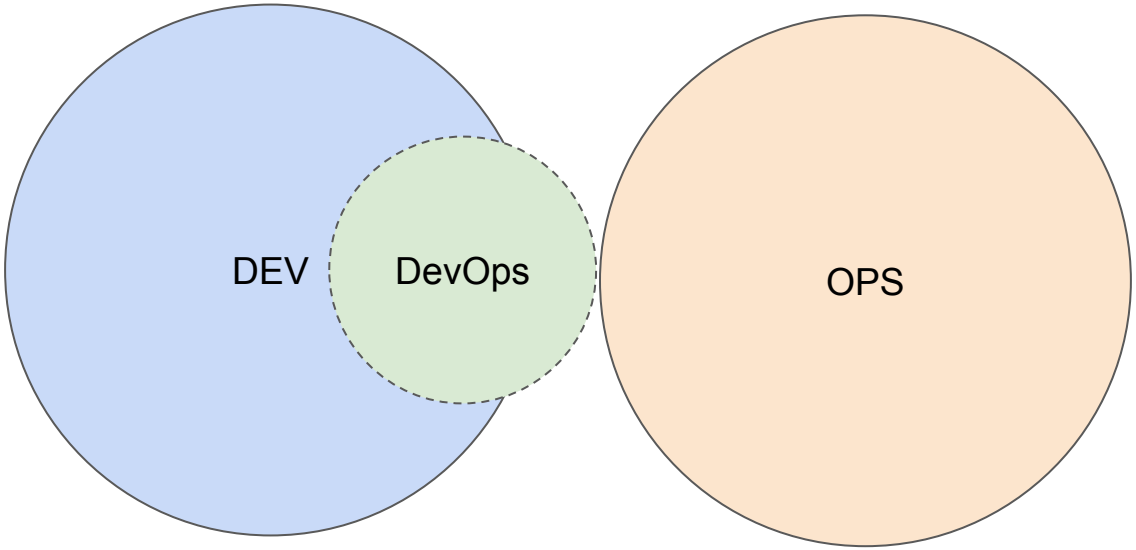
DevOps евагнелисты



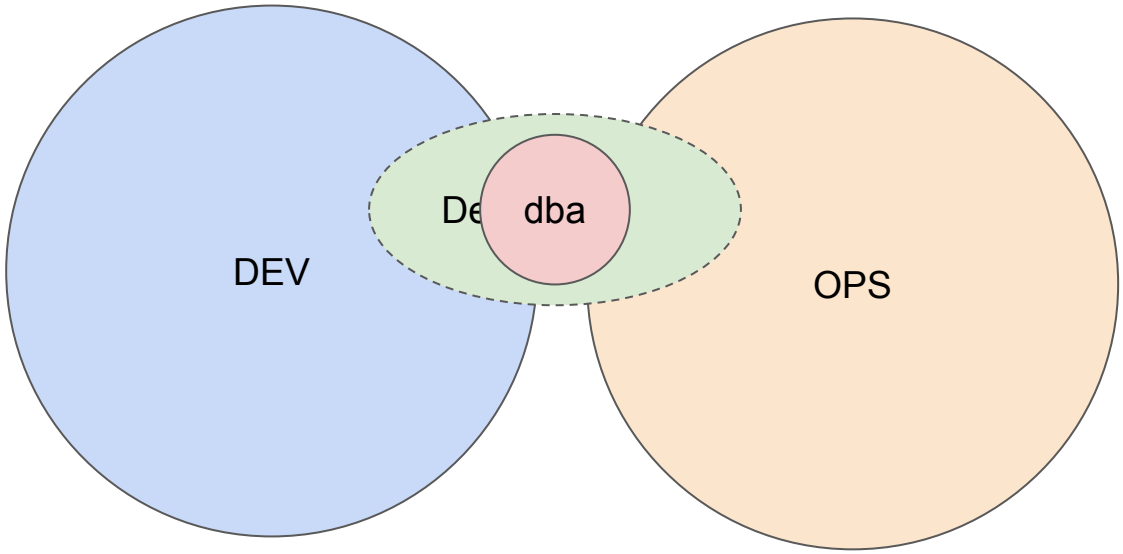
SRE



Взаимодействие через контейнеры

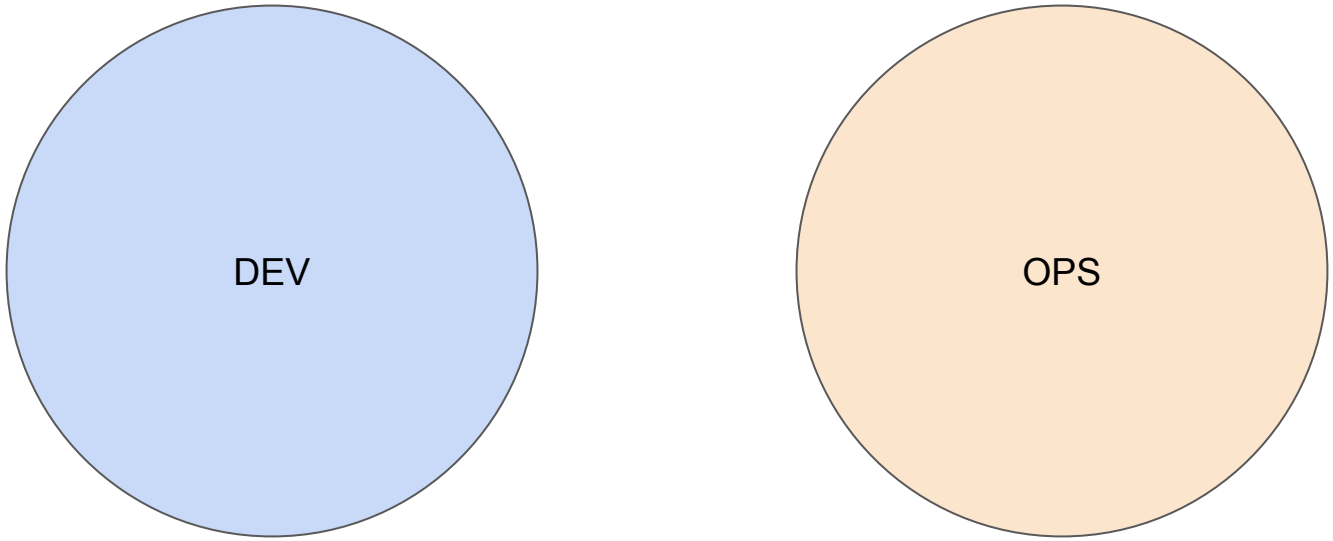


Взаимодействие разработки и DBA

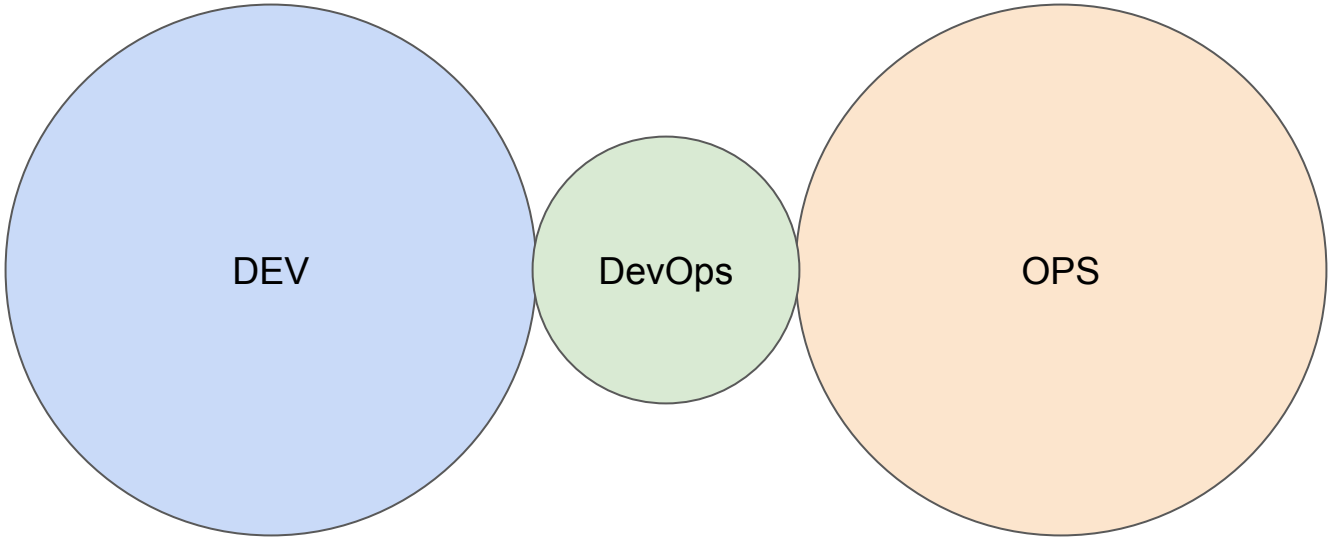


Антипаттерны

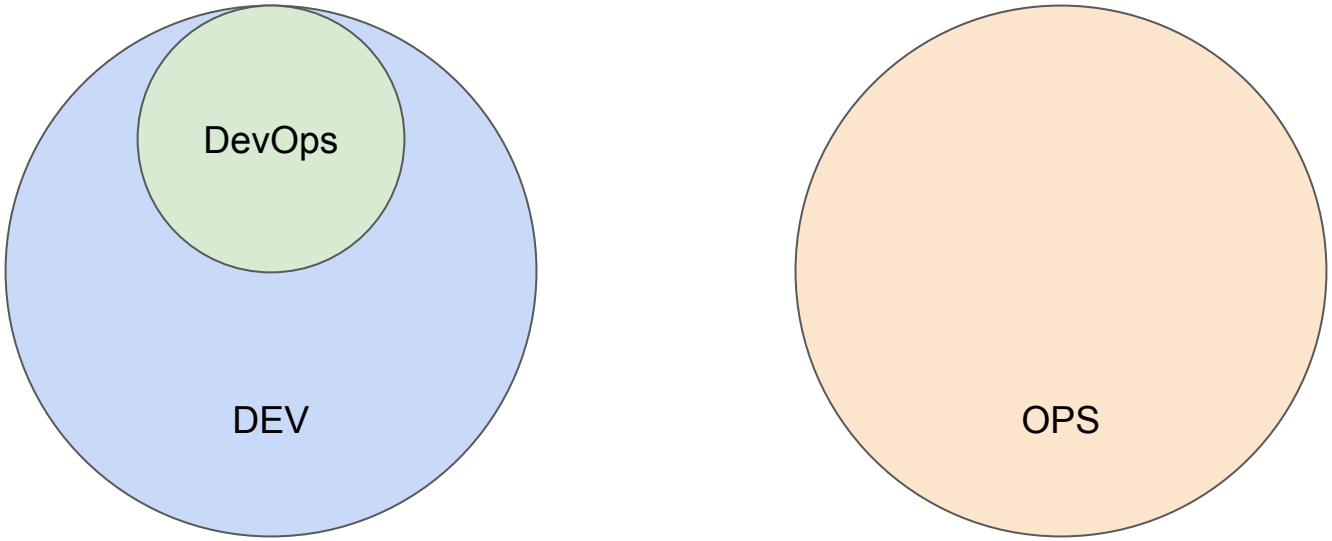
Изоляция Dev и Ops



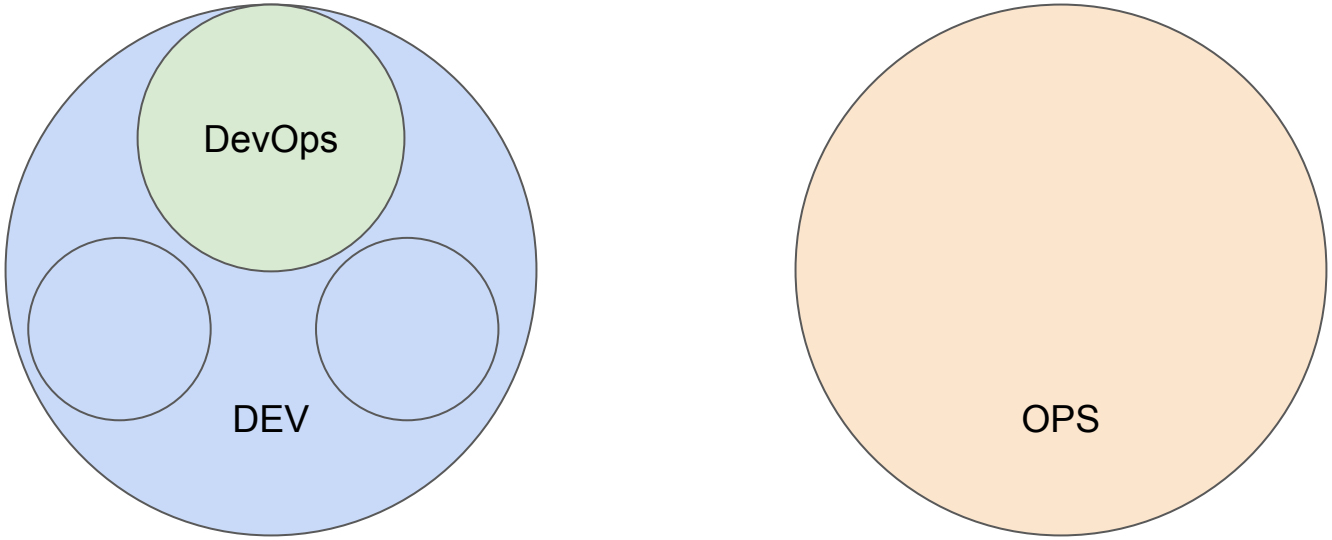
DevOps отдельная команда



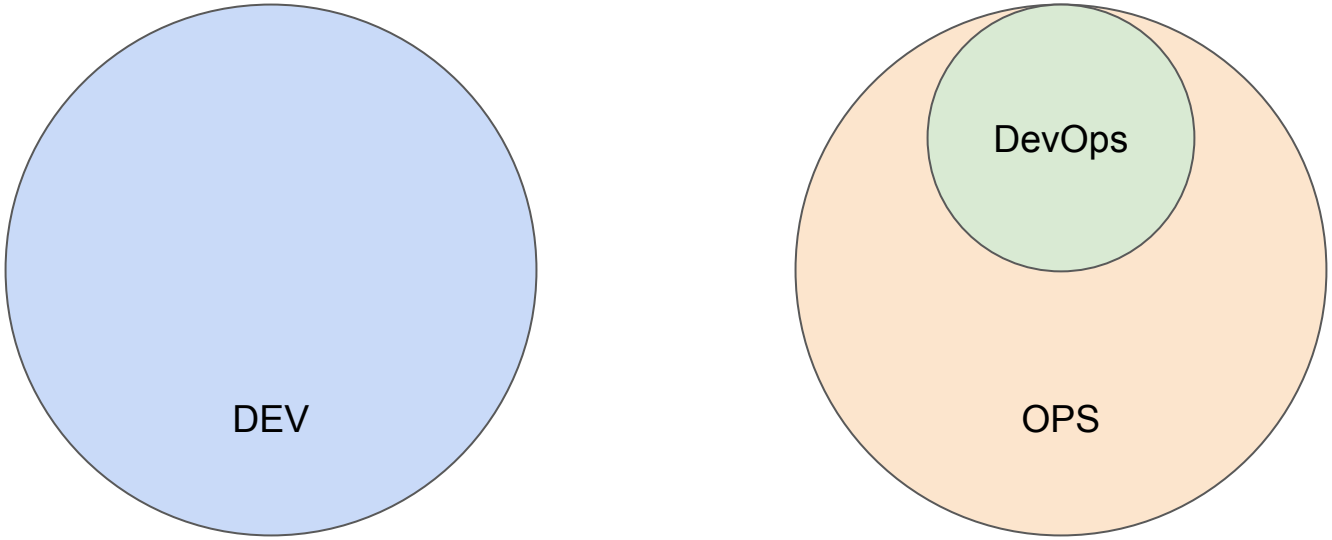
Dev не нужен Ops



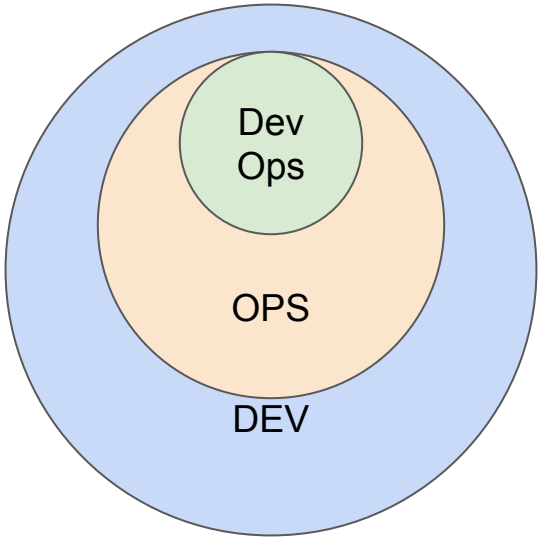
Dev как разработка инструментария



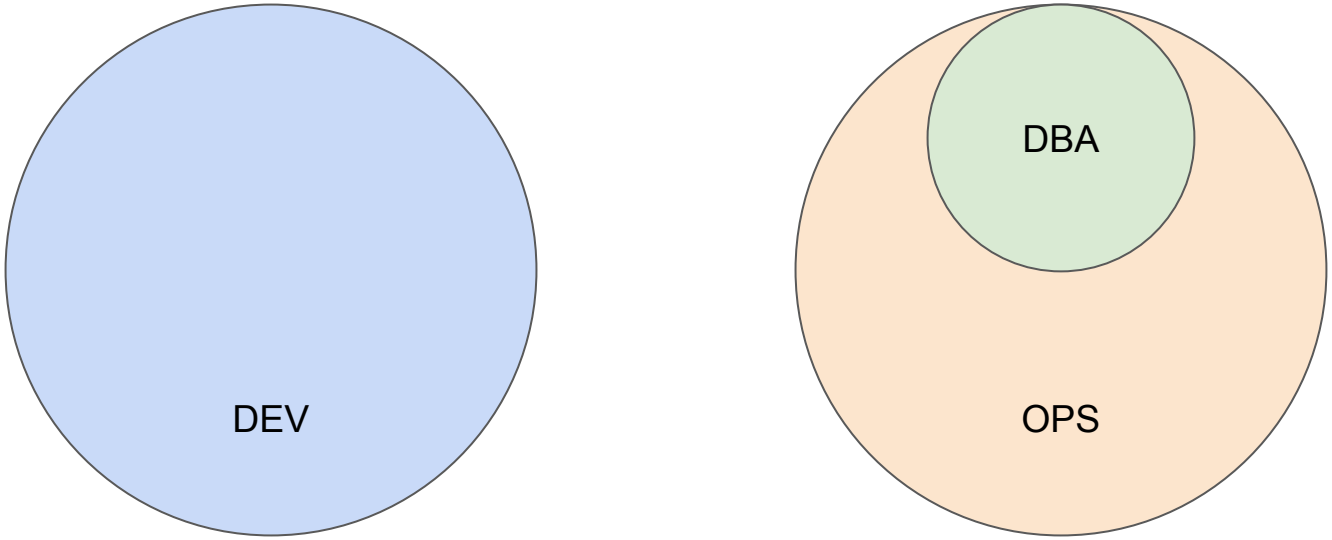
Переименованные сисадимны



Ops встроенный в Dev



Изоляция DBA от Dev



DevOps

Резюме:

Если **Agile** (**Scrum**, **Kanban**) в большей степени применяет изменчивость и непрерывность для стадии разработки, то **DevOps** расширяет эти практики на тестирование и сопровождение и добавляет технологический бэкграунд.

DevOps имеет множество разновидностей, к каждой конкретной команде стоит подбирать собственный подход

SRE

SRE

SRE (Site Reliability Engineering) - производная от концепции DevOps в исполнении Google.

Концепция:

Взять на роль администраторов (OPS) разработчиков (DEV) или бывших администраторов с большим опытом в разработке.

Цель:

Создать команду OPS, которая будет склонна к автоматизации рутины и нацелена на развитие продукта без ущерба надёжности.

Задача:

сократить MTTR, улучшить исполнение SLA и получить свободный error budget.

Глоссарий

- **SLA (Service Level Agreement)** - общее соглашение о доступности сервиса, штрафах за ошибки, метрики и прочее;
- **SLI (Service Level Indicator)** - метрика доступности сервиса;
- **SLO (Serivce Level Objective)** - сочетание метрики, её значения и периода;
- **MTTR (Mean Time To Recovery)** - среднее время на восстановление;
- **MTBF (Mean Time Between Failures)** - среднее время между сбоями;
- **Error budget** - статья в бюджете, запланированная на расходы по восстановлению сервиса.

SRE

- В **SLO** часто запланирована доступность:
  - 90% - 36 дней и 12 часов
  - 99% - 3 дня, 15 часов и 36 минут
  - 99,9% - 8 часов 45 минут и 36 секунд
  - 99,99% - 52 минуты и 33 секунды
  - 99,999% - 5 минут и 15 секунд
  - 99,9999% - 31 секунда
- Задача **SRE** правильно обработать ошибки, приводящие к недоступности (ошибки это нормально);
- Чтобы увеличить **MTBF**, нужно следить за тем, что делает команда в новом релизе и не допускать внедрение кода с большим количеством ошибок;
- Чтобы увеличить **MTTR**, нужно работать над **SLO**, а значит уменьшить время реакции до исправления недоступности (внедрение автоматизаций)

SRE

- Каждая недоступность должна быть проанализирована, выявлены недостатки системы и предприняты меры по их устранению - **PostMortem;**
- Нужно искать не виновных, а недостатки системы;
- Автоматизация рутин может стоить дороже, чем исполнение рутины руками, так как в долгосрочной перспективе это компенсируется уменьшением затрат в **Error Budget;**
- **Chaos engineering** - использование инструментов, которые подключаются к вашему рабочему процессу и в случайном порядке отключает функциональные блоки в промышленной системе;
- Использование **chaos engineering** - важная составляющая работы **SRE** (Chaos Monkey, Chaos Gorilla).

SRE

Резюме:

SRE предписывает смотреть на сопровождение с точки зрения разработчика и **наоборот**. Смысл использования в уменьшении трат времени на **рутинную работу** в пользу использования этого времени на проведение экспериментов.

Итоги

Итоги

Сегодня узнали, что:

- **Жизненный цикл ПО** - перечисление этапов проектирования, разработки, тестирования, сопровождения ПО до процесса вывода из эксплуатации;
- Существует несколько принципов разработки ПО: **Watefall**, **Agile**, **Lean;**
- На основе принципов **Agile** и **Lean** созданы методы ведения разработки и сопровождения: **Kanban**, **Scrum, DevOps, SRE**
- Эффективное использование гибких методов достигается путём **комбинирования** всех в индивидуальном порядке для каждой команды.

Домашнее задание

Домашнее задание будет у вас в личном кабинете в виде теста из 5 вопросов.

Вопросы по домашней работе задавайте **в чате** мессенджера Slack.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Алексей Метляков

Алексей Метляков