DevOps в банковском секторе

или какая команда нам больше подходит

Андрей Пономарев

Зам. Начальника Управления ИТ эксплуатации

По мотивам публикаций Matthew Skelton

Введение

- Основной целью перехода на идеологию DevOps является желание улучшить и ускорить обслуживание клиентов. Сокращение затрат, увеличение автоматизации, улучшение управления конфигурацией, выполнение бизнес-запросов - это лишь сопутствующие задачи. При этом разным типам организаций может потребоваться разная структура команд для эффективного взаимодействия Dev и Ops.

Кратко и по существу

- Какая структура или топология команды DevOps подходит для организации, зависит от многих вопросов, например:
  - Каков набор выпускаемых и поддерживаемых продуктов – чем меньше продуктов, тем больше взаимосвязь Dev и Ops.
  - Степень, сила и эффективность технического руководства - имеет ли Dev и Ops общую цель.
  - Готова ли организация на трансформацию роли ИТ эксплуатации от «принеси-подай» во встроенной в бизнес процесс.
  - Есть ли в организации лидеры, которые готовы указывать на проблемные области и сопровождать изменения.
- Чаще всего построение «команды мечты» потребует последовательной смены нескольких моделей.

Итак, какая структура команды больше подходит для DevOps?

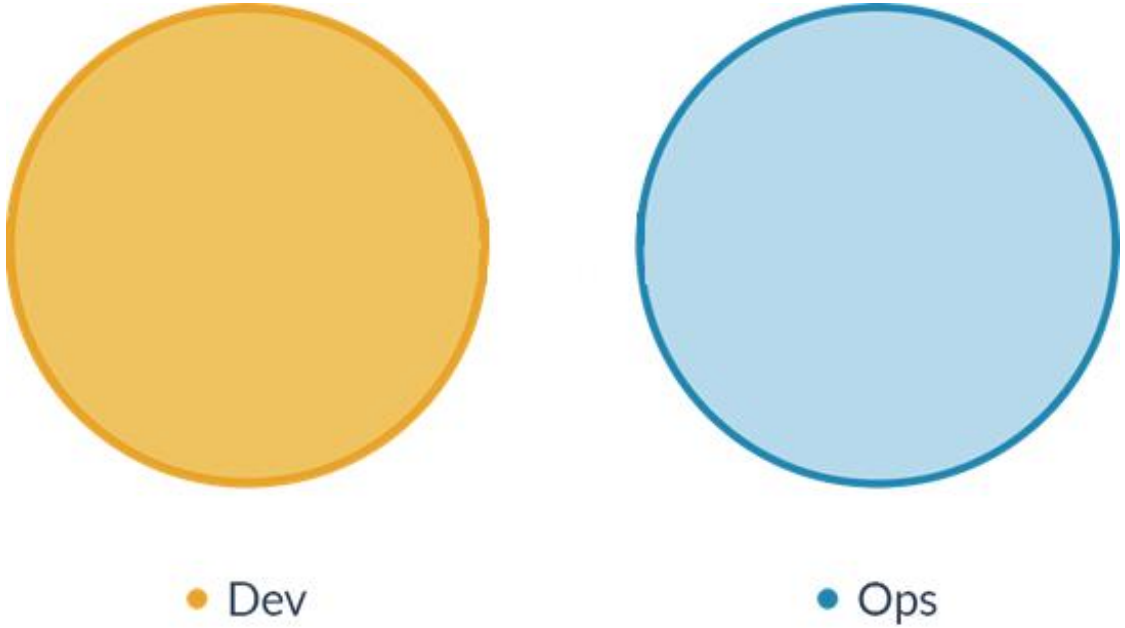
- Очевидно, что нет никакой магической структуры или командной топологии, которая подойдет для каждой организации. Тем не менее, есть классификация различных моделей (или «топологий). Изучая сильные и слабые стороны этих топологий, мы можем определить структуру команды, которая могла бы лучше всего работать в практике DevOps в наших собственных организациях.

DevOps Анти-Типы

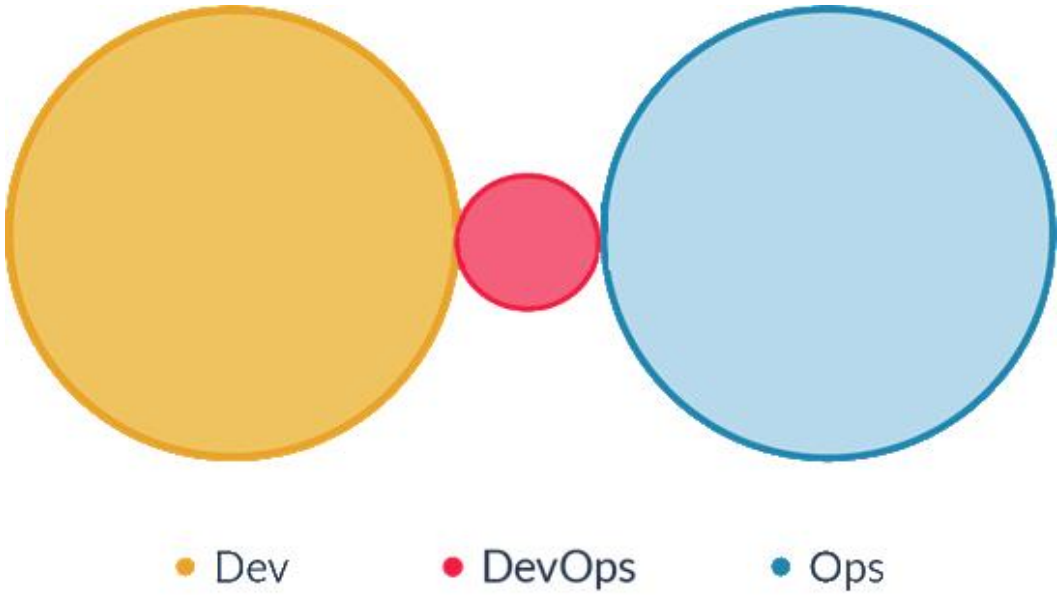
- A: Dev и Ops отдельно
- B: Выделенная DevOps
- C: Dev не нужны Ops
- D: DevOps как внутр. инструмент
- E: Новые СисОп’ы
- F: Ops внутри Dev

Анти-Тип А: Dev и Ops отдельно

- Это классическая «стена недопонимания» между Dev и Ops. Это означает, что Dev считает что переданное на тестирование уже завершено и выпущено, Ops не может сопровождать потому что нет документации. У Ops нет желания участвовать в архитектурных комитетах. Dev не желает принимать участие в разборе ошибок.
- Мы все знаем, что это плохая структура взаимодействия Dev и Ops, но она хотя бы не приводит к войне внутри ИТ, как следующая схема.



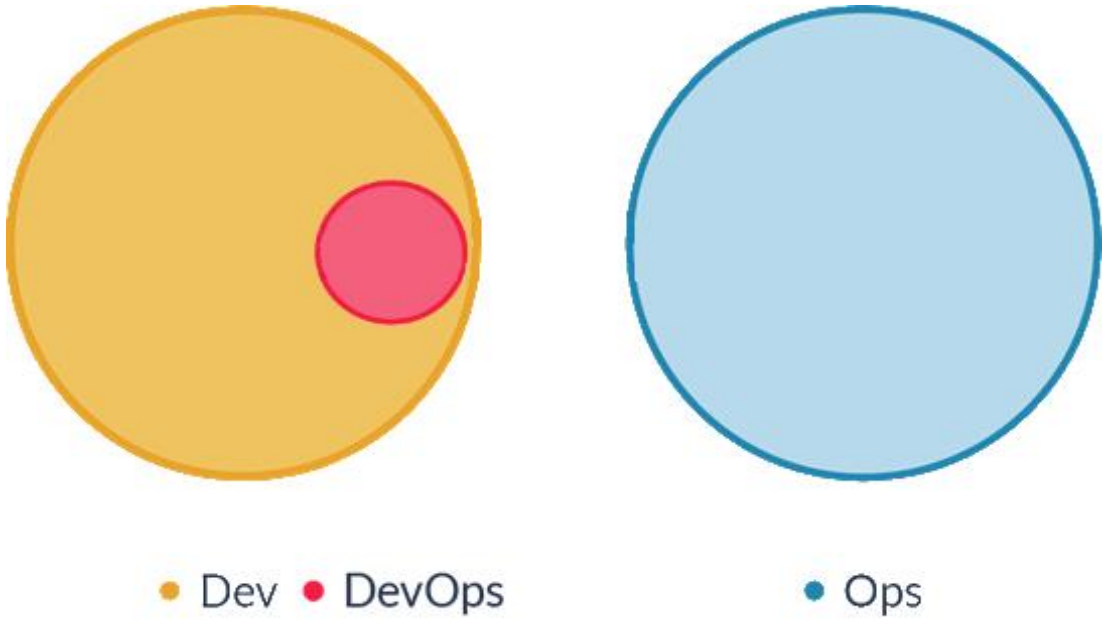
Анти-Тип B: Выделенная DevOps



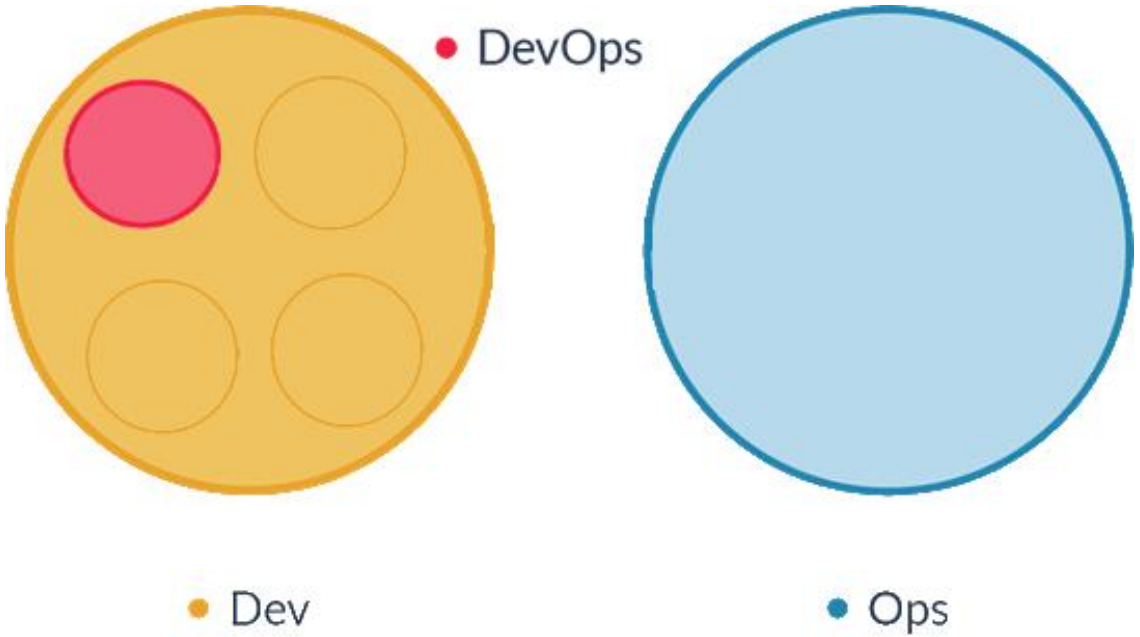
- Изолированная группа DevOps обычно возникает, когда менеджер или директор решает что «нам нужно немного этого DevOps» и организуют новое подразделение. Сотрудники этой группы еще дальше отодвигаются и от Dev и от Ops, пряча свои знания супергероев от «невежественных Dev» и «динозавров Ops»
- Такая конфигурация может быть только один шанс на существование – это времнное явление (не более 12 -18 мес.) и все четко знают, что в результате выделенная DevOps команда будет распределена между Dev и Ops и мы получим **Тип 5 DevOps**.

Анти-Тип С: Dev не нужны Ops

- Эта топология несет в себе сочетание наивности и высокомерия от разработчиков и менеджеров по развитию, особенно при запуске новых проектов или систем. Предполагая, что Ops теперь ушла в прошлое («у нас ведь теперь есть Облако, верно?»), Dev явно недооценивают сложность и важность знаний и навыков Ops и считают, что они могут обойтись без них и просто закрыть все вопросы самостоятельно.
- Такая топология Anti-Type C, скорее всего нуждается в другом Ops (либо «**Тип 2 Ops как IaaS**», либо в «**Тип 3 DevOps как Service**»). Это становится явным, когда ПО выходит за рамки планирования и тестирования и операционная деятельность начинает занимать существенное время у разработчиков.



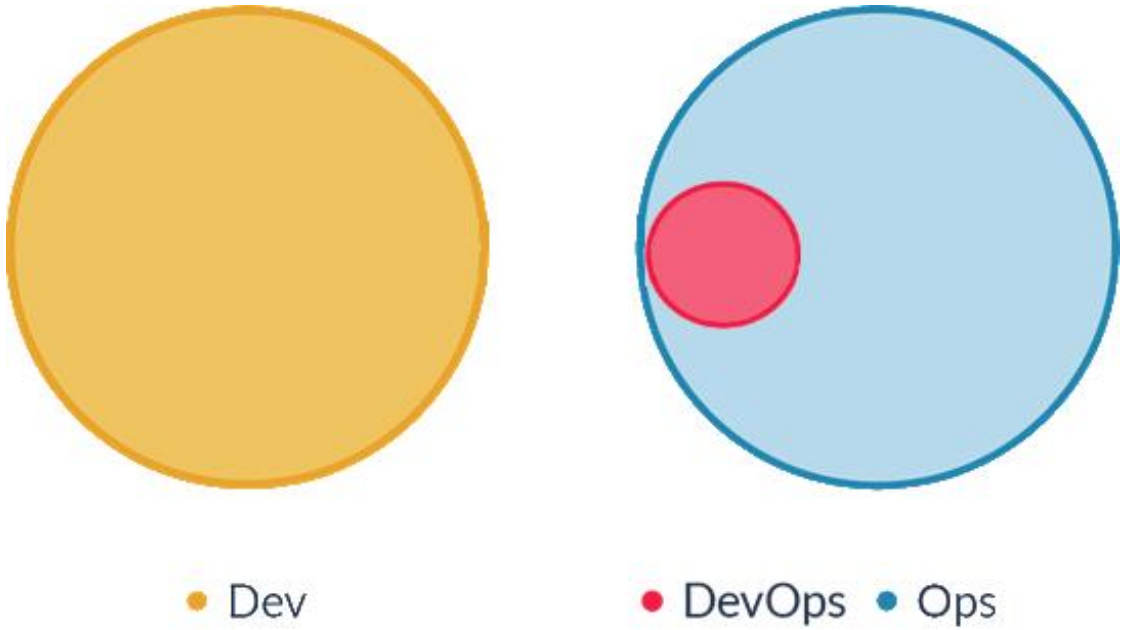
Анти-Тип D: DevOps как внутр. инструмент



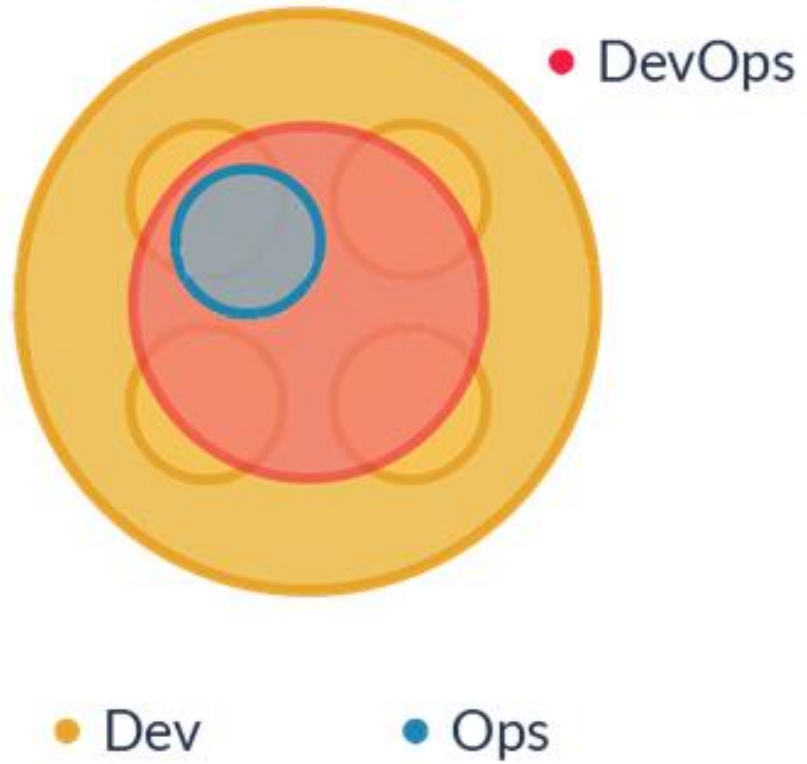
- Для того, чтобы не терять скорость разработки, специфичные инструменты хранения кода, сборки и развертывания, настраиваются и используются только внутри команды Dev. Ops продолжает получать готовые программы на выходе и пытаться работать в полной изоляции от Dev.
- Хотя инструменты применяются правильные и в нужных местах, но их использование ограничено только одной командой. По прежнему присутствует фундаментальная ошибка изоляции Ops от построения архитектуры решения.

Анти-Тип E: Новые СисОп’ы

- Этот антитип типичен для организаций с низкой инженерной зрелостью. Они хотят улучшить свою практику и сократить расходы, но они не видят ИТ в качестве двигателя бизнеса. Поскольку успехи DevOps на слуху, они также хотят «сделать DevOps». К сожалению, вместо того, чтобы подумать о пробелах в структуре и отношениях, они идут путем найма «инженеров DevOps» для команды Ops.
- DevOps становится просто другим названием роли Системного Администратора, без реальных культурных и организационных изменений. Этот антитип становится все более распространенным, так как легче найти кандидата по ключевым словам, чем создать почву для коммуникации между Dev и Ops.



Анти-Тип F: Ops внутри Dev



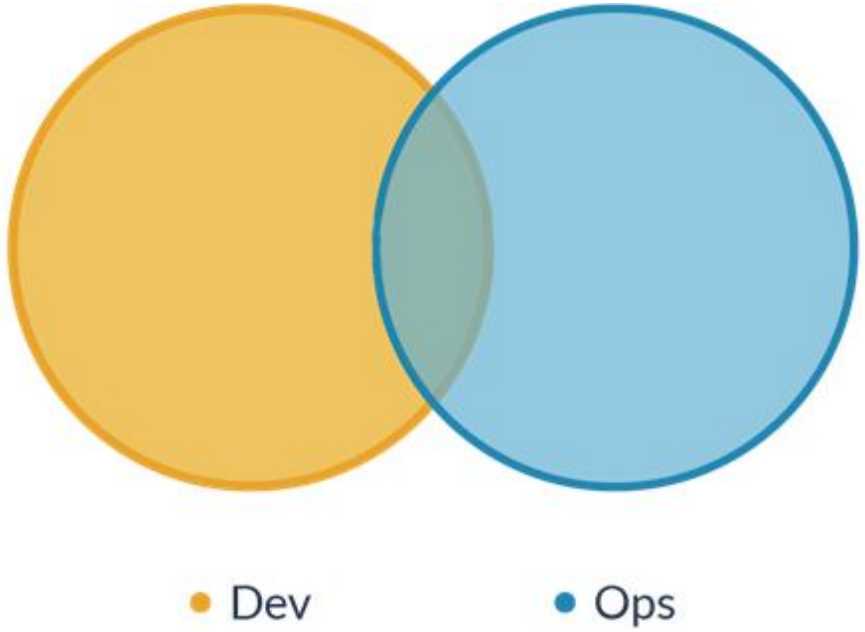
Организация не хочет поддерживать отдельную команду Ops, поэтому команды разработчиков берут на себя ответственность за инфраструктуру, управление средами, мониторинг и т.п. Однако, когда это делается в рамках проекта, это означает, что Ops элементы зависят от ограничений по ресурсам, их перераспределения, от постановки приоритетов, которые приводят к недо… или полу … решениям.

В этом анти-типе организация демонстрирует недостаточную оценку важности и навыков, необходимых для эффективных Ops.

DevOps Модели

- 1: Dev+Ops
- 2: Ops as IaaS
- 3: DevOps-as-a-Service
- 4: Группа SRE
- 5: Все в контейнерах

Тип 1: Dev и Ops

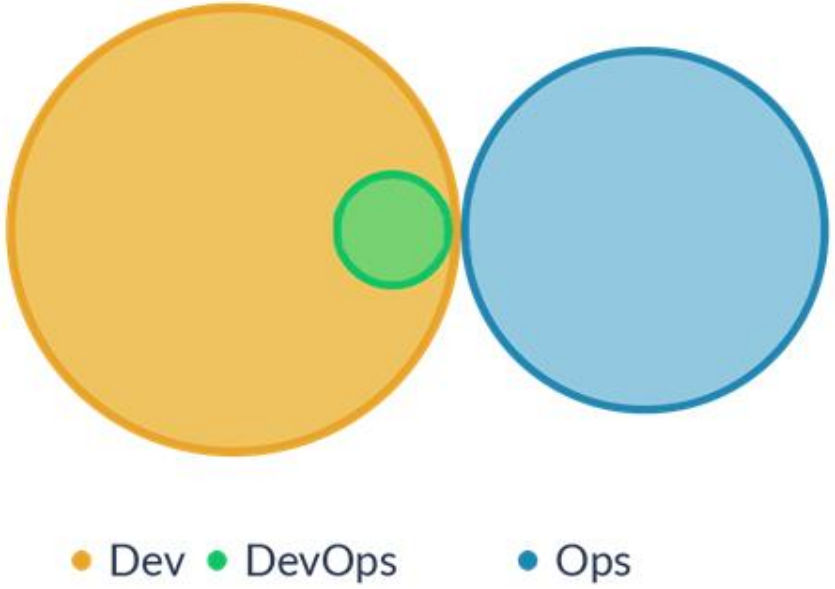


Применение Тип 1: в организациях с сильным техническим лидером.
Эффективность: **ВЫСОКАЯ**

- Это «команда мечты» DevOps: хорошее взаимодействие между командами Dev и Ops, каждая из которых прикладывает силы именно там, где это необходимо. Возможна ситуация, когда есть много отдельных команд Dev, каждая из которых работает над отдельным проектом, но все они замыкаются на Ops команде.
- Модель этого типа нуждается в достаточно существенных организационных изменениях для ее создания, а также в высокой степени компетентности в команде технического руководства. Dev и Ops должны иметь четко выраженную и явно эффективную общую цель («внедрение надежных, частых изменений» или что-то типа того). Сотрудникам Ops и Dev должны быть комфортно работать друг с другом.

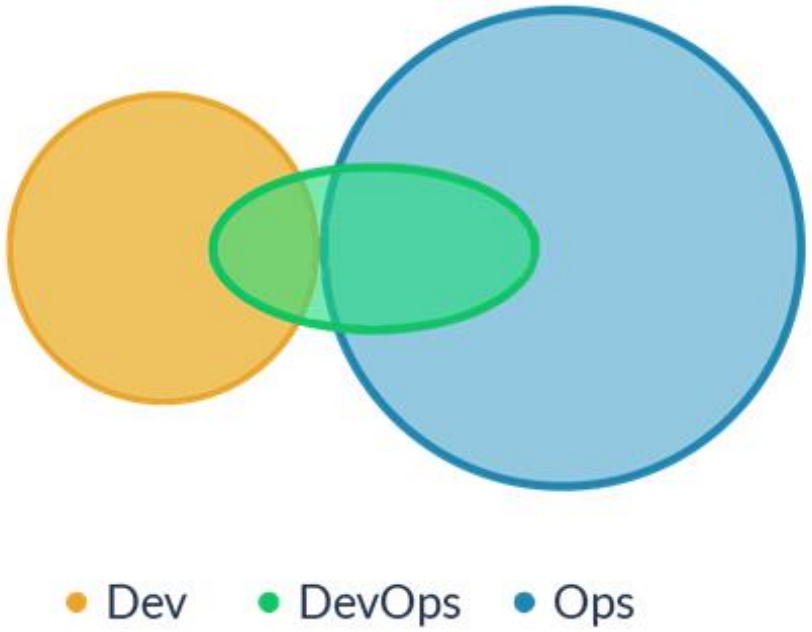
Тип 2: Ops - Infrastructure-as-a-Service

- Подходит для организаций с «традиционным» отделом ИТ, который не может или не будет быстро меняться по требованиям бизнеса. Так же подходит и для организаций, которые запускают все свои приложения в общедоступном облаке (Amazon EC2, Rackspace, Azure и т.п.). Модель позволяет обрабатывать запросы и предоставлять эластичную инфраструктуру, на которой развертываются и запускаются приложения. Часть группы Dev выступает в качестве источника знаний об Ops функциях, показателях, мониторинге, серверах и т.п.
- Топология IaaS имеет в себе некоторую потенциальную эффективностью, т.к. может дать более легкое и быстрое развертывание проекта, возможно даже быстрее, чем если развертывать **Тип 1 (Dev+Ops)**, на которую можно попытаться перейти позже.



Применение Тип 2: организации с традиционным Ops или полностью в публичном облаке.
Эффективность: СРЕДНЯЯ

Тип 3: DevOps как внешний сервис

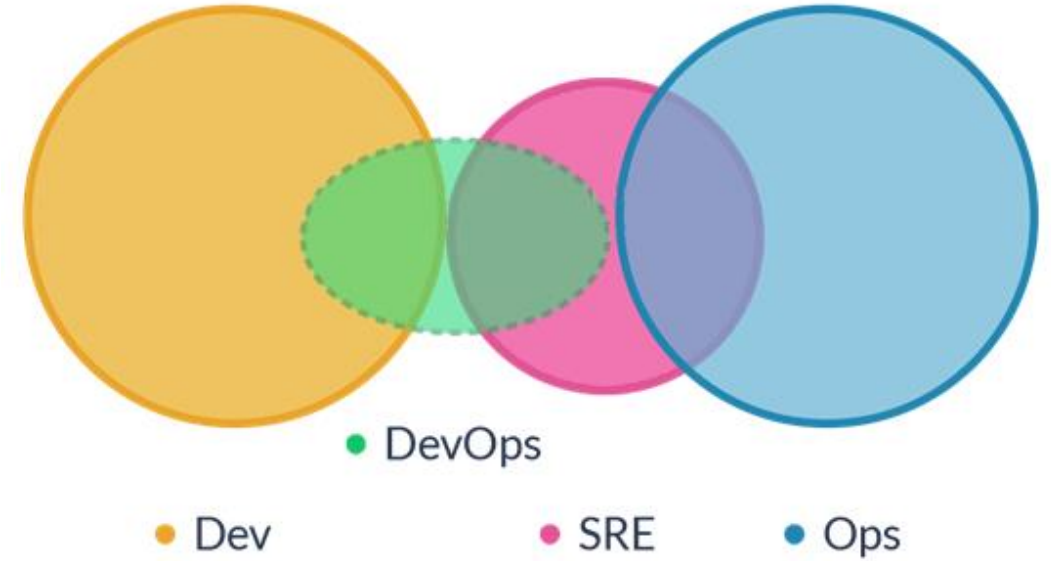


- Некоторые небольшие организации из-за отсутствия опыта или персонала могут передавать функции DevOps во внешние компании (например Rackspace), которые могут помочь в создании тестового окружения, настройке и автоматизации развертывания, мониторинга и т.п.
- То, что можно назвать DevOps-as-a-Service может быть вполне полезным опытом для небольшой организации. Нормальным переходом из такого типа будет переход в **Тип 2 (Ops как IaaS)** или, по мере роста внутренней экспертизы, часть команды может вырасти в Ops и тогда получится полноценный **Тип 1 (Dev+Ops).**

Применение Тип 3: небольшие группы или организации с небольшим опытом.
Эффективность: **СРЕДНЯЯ**

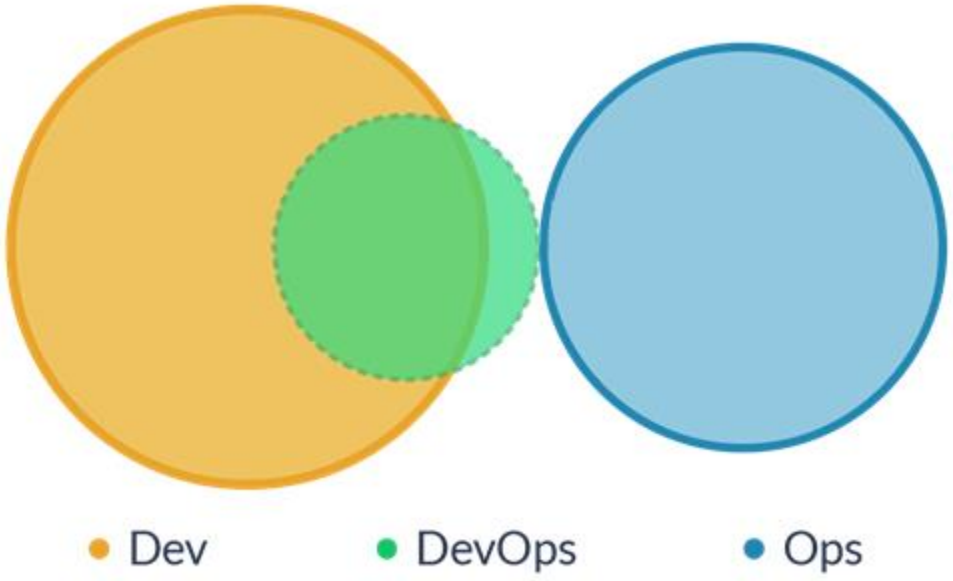
Тип 4: Группа SRE (модель Google)

- Фактически, некоторые организации (в том числе Google) запускают расширенную модель с явным «отрывом» Dev от команды Ops. На уровне DevOps появляется промежуточный слой, команда Site Reliability Engineering (SRE), которая отвечает за работоспособность продукта. В этой модели команда Dev должна предоставить тестовые доказательства (логи, тестовые отчеты, показатели мониторинга и т.п.) команде SRE, как доказательство, что их программное обеспечение работает как запланировано. При этом группа SRE может полностью отменить релиз и отправить код на доработку.



Применение Тип 4: применимо только для организаций с высокой степенью организационной зрелости. Если применять принцип “JFDI”, то все скатится к **Anti-Type A**.
Эффективность: **НИЗКАЯ или ВЫСОКАЯ**

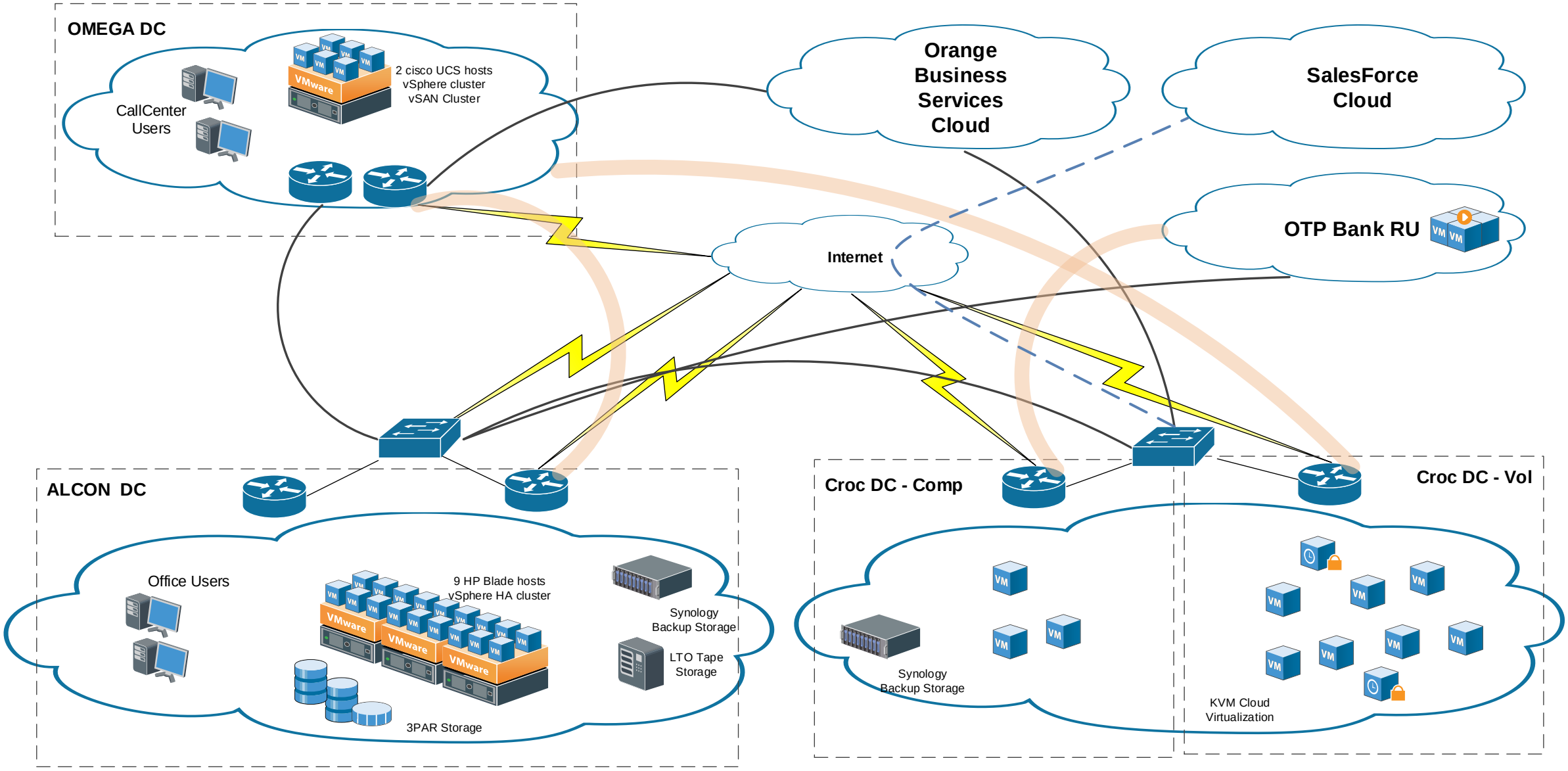
Тип 5: Все в контейнерах



Применение Тип 5: Контейнеры могут работать очень хорошо. Стоит остерегаться только ситуации **Anti-Type A**, когда Ops запускает что ни попадя, что к ним бросает Dev.
Эффективность: **от СРЕДНЕЙ до ВЫСОКОЙ**

- Контейнеры устраняют необходимость в некоторых видах сотрудничества между Dev и Ops, инкапсулируя требования к развертыванию и времени выполнения приложения в контейнер. Таким образом, контейнер действует как граница ответственности Dev и Ops. Модель контейнеров работает хорошо до того момента, как Dev начинает игнорировать инфраструктурные требования, тогда эта модель может вернуться к состязательному «мы и они».

Ландшафт Ops



Ландшафт DevOps