Летняя школа DevOps и CI/CD

Введение

Почему решили сделать эту школу

- Тема «горячая», но мы ее выбрали не поэтому
- Необходимость давать актуальные навыки студентам.
  - Навыки работы с git, навыки DevOps – входят в базовый набор требований современного специалиста
- Востребовано индустрией
- Хорошо ложится на проектную деятельность в МИЭМ
  - Она должна готовить к реальной жизни (цель проектного обучения)
  - Чтобы успешно работать в проектах, надо знать те технологии, которые мы изучим

Кому может быть полезно

- **Разработчику**
  - чтобы научиться настраивать процесс сборки, тестирования и разворачивания приложения
- **Системному администратору**
  - автоматизировать рутинные операции, оптимизировать нагрузки
- **QA-инженеру**
  - научиться строить автоматизированные тестовые среды, настраивать предпродакшн тестирование
- **Специалисту по безопасности**
  - при внедрении CI/CD в организации, часть его экспертизы так же может быть автоматизирована.
  - Должен научится взаимодействовать с разработчиками и тестировщиками (DevSecOps)
- **DevOps инженер**
  - сейчас отдельная профессия

DevOps – Development и Operations

- Есть несколько стадий, которые проходит любой код от разработки до пользователя

DevOps – Development и Operations

- В зависимости от принятой модели разработки, могут быть разные

DevOps – Development и Operations

- Основные стадии
  - разработка (кодирование),
  - сборка,
  - тестирование,
  - упаковка,
  - выпуск релиза,
  - настройка инфраструктуры,
  - внедрение на продуктовый сервер,
  - мониторинг
- Разные люди этим занимаются – разработчики, тестировщики, администраторы, специалисты QA, специалисты по безопасности
- DevOps – это про то, как сделать их совместную работу эффективнее

DevOps – Development и Operations

- Две большие группы занимаются следующими вопросами
- Development
  - Разработка приложения
  - Тестирование, выпуск релизов
- Operations
  - Развертывание приложения
  - Поддержка и мониторинг

DevOps – Development и Operations

- В частности, внедрение DevOps призвано
  - Ускорить прохождение всех этапов жизненного цикла ПО, обеспечить эффективное взаимодействие разных членов команды
  - Ускорить получение обратной связи и исправление ошибок
  - Обеспечить возможность быстрого отката в случае ошибок
  - Ускорить процесс обновлений ПО
  - Упростить процессы оптимизации нагрузок и восстановления после сбоев (косвенно)
- Все это происходит за счет автоматизации всех процессов

DevOps – Development и Operations

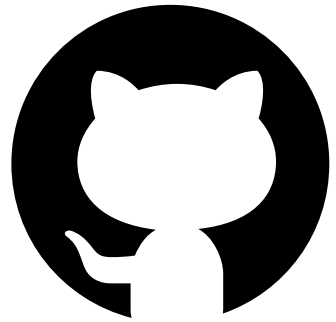
- Идея автоматизировать рутинные операции не нова
- Что же изменилось, откуда всплеск интереса к DevOps?
- За последние 15 лет появилось много технологий, которые существенно изменили многое, в том числе, подход к разработке и внедрению приложений
  - Посмотрим некоторые из них

Git (2005)

- Распределенная система управления версиями
- Системы контроля версий были и раньше, просто git оказался лучшим
  - в плане работы из командной строки,
  - работы с ветками,
  - разрешения конфликтов,
  - синхронизации,
  - работы оффлайн и т.п.

Git

- Постепенно появлялись публичные и частные репозитории
- Например github – основное хранилище opensource решений
- Были и другие
  - GitLab, Bitbucket, etc.
- Больше чем просто репозитории



Технологии виртуализации

- Аппаратная поддержка виртуализации
- Появление гипервизоров
  - Специальные приложения, которые позволяли запускать на одной машине несколько виртуальных платформ
  - Откреплление сервера от железа, работа в виртуальном окружении
- IBM LPAR, VMware, Hyper-V, Xen, KVM, Bhyve
- Платформы управления облачными ресурсами

Публичные облака

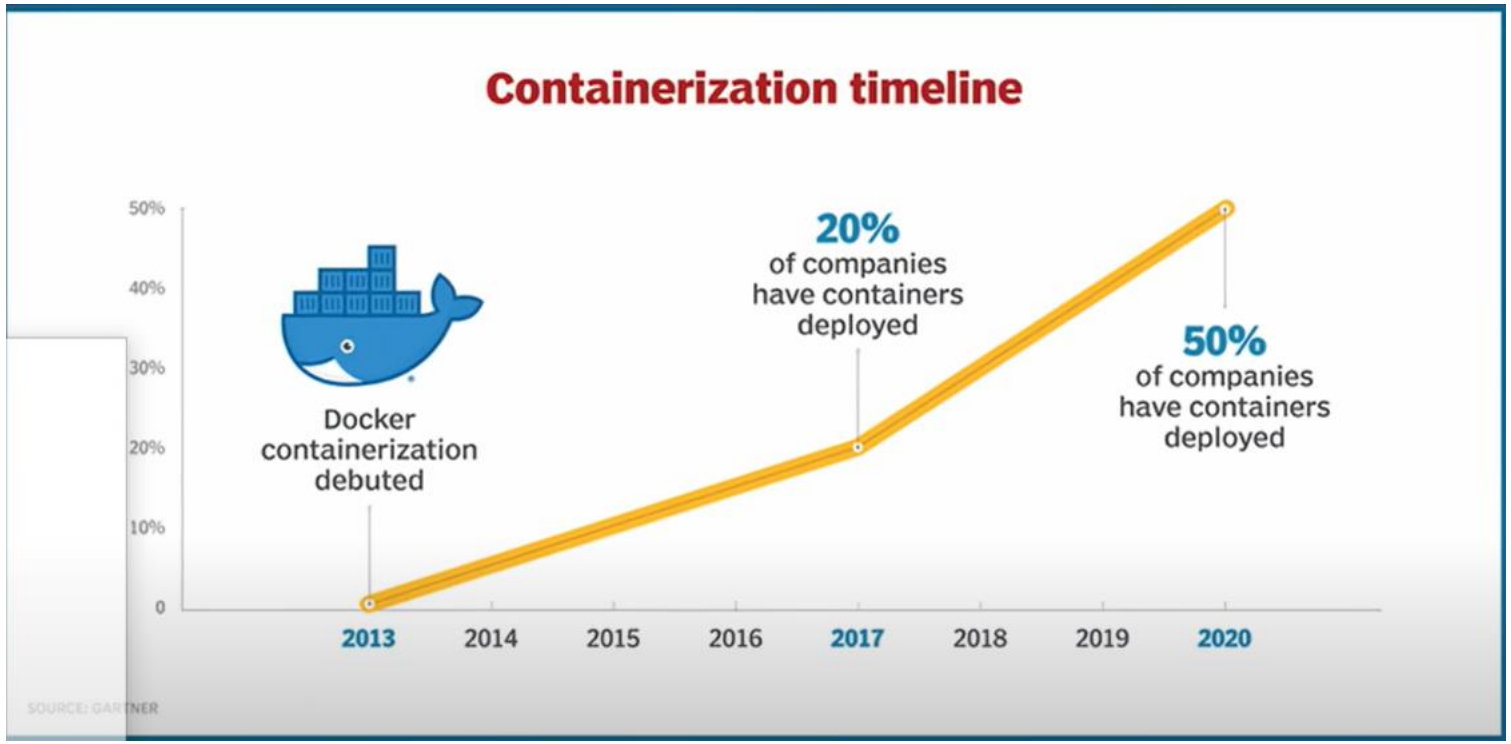
- Далее естественно появились публичные облачные платформы
  - Amazon Web Services (2006),
  - Google Cloud (с 2008),
  - Azure (2010)
  - и т.п.

Контейнерная виртуализация

- Контейнер это не вполне виртуальная машина
  - Об этом будет подробно рассказано завтра
- Были разные решения
  - FreeBSD Jail (2000), Virtuozzo Containers (2000), Solaris Containers (2005), Linux-VServer[en], OpenVZ (2005), LXC (2008), iCore Virtual Accounts (2008)
- Но потом появился Docker (2013)
  - и достаточно быстро стал наиболее используемым приложением для контейнеризации
  - (и завоевал мир)

Контейнерная виртуализация

- Сейчас использование docker распространилось очень широко
  - Работа с docker - must-have навык



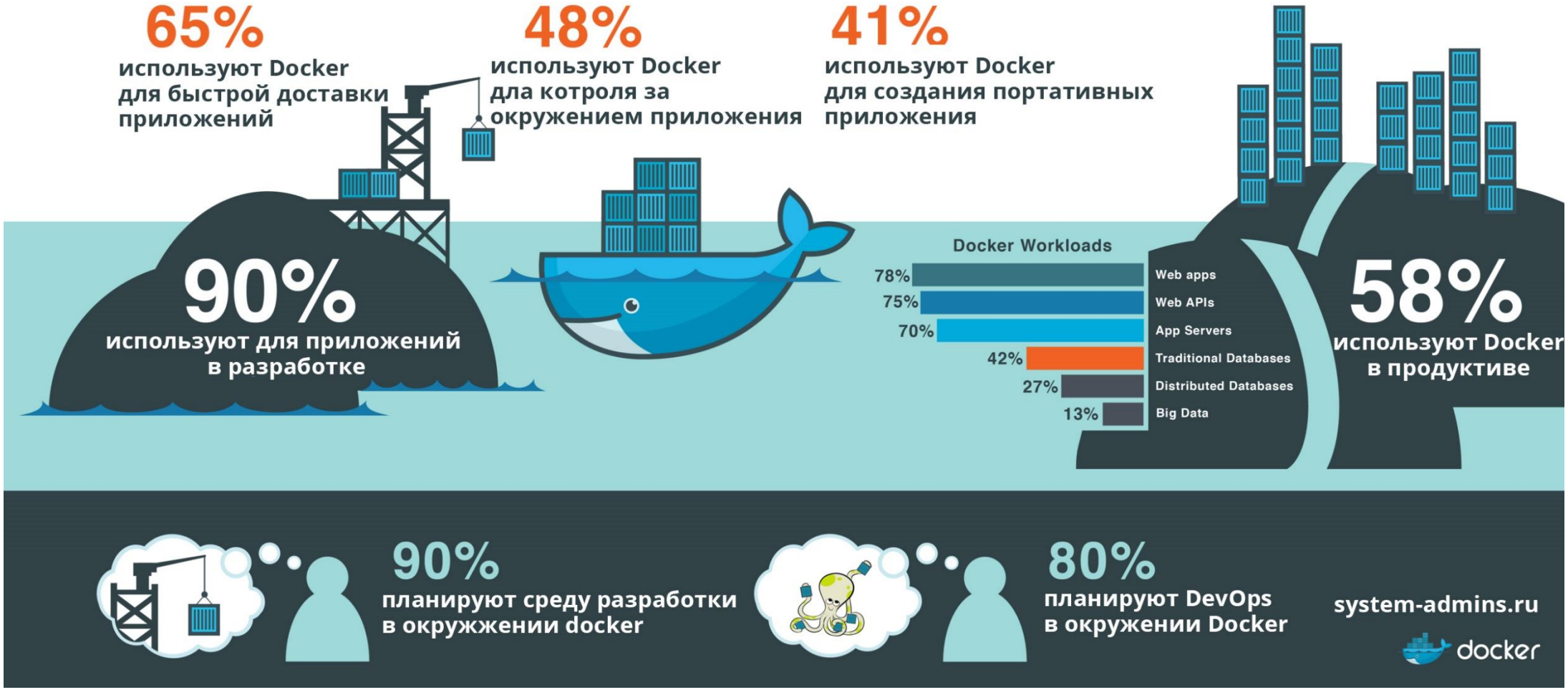
Контейнерная виртуализация

- Почему это так популярно?
- Рассмотрим аналогию с появлением стандартных морских контейнеров
- Использование стандартных морских контейнеров позволяет в отдельных случаях снизить затраты на перевозку материалов и изделий практически вдвое.

Контейнерная виртуализация

- Использование стандартных контейнеров для упаковки программных решений имеет схожие плюсы
  - Поддержка на разных платформах
  - Возможность написать dockerfile и запустить его в любом месте, получив гарантированный повторяемый результат
  - Можно упаковать свое приложение в стандартный docker контейнер и запустить его сколько угодно раз
  - Можно развернуть, например, в docker compose

Контейнерная виртуализация

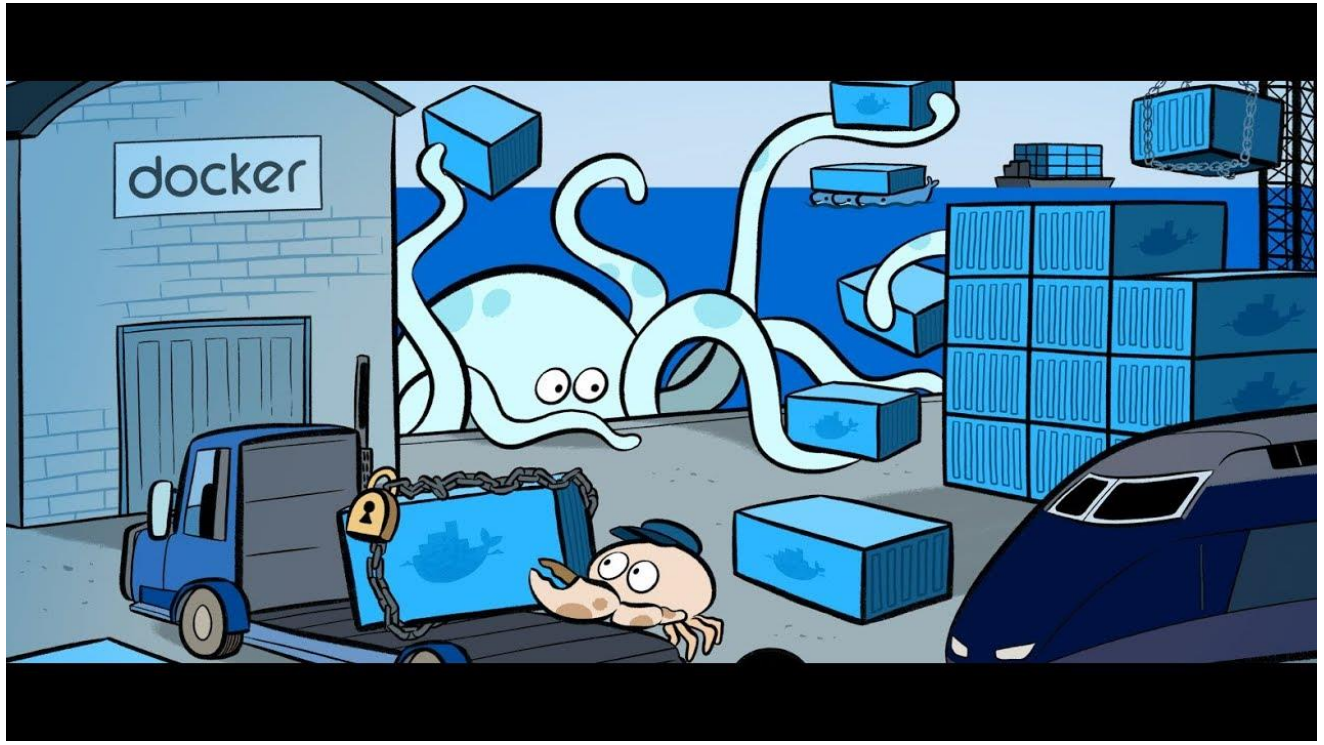


Оркестрация

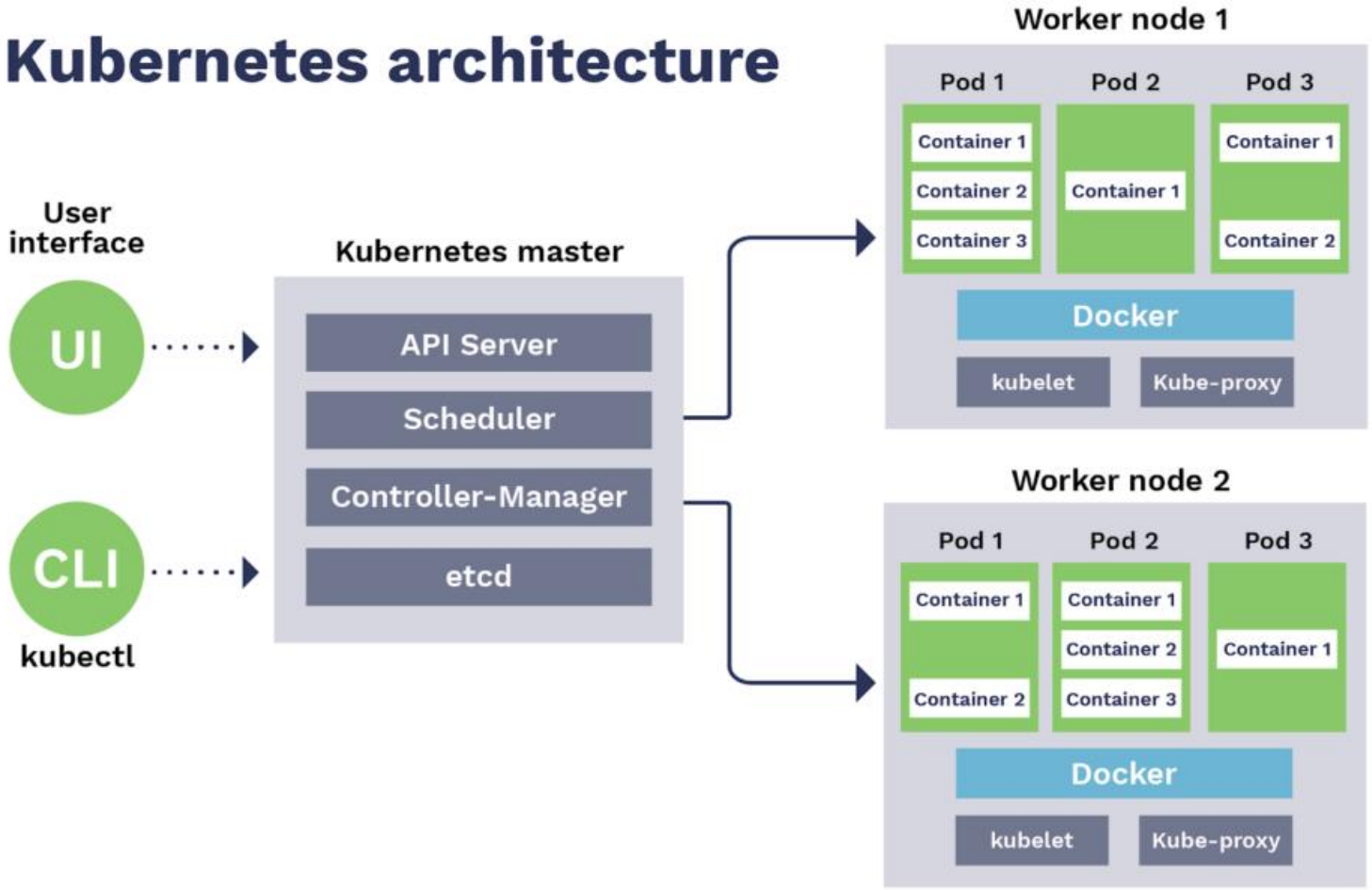
- Управление большим количеством контейнеров
  - На самом деле большим – сотнями, тысячами
  - Контроль их состояния, выделение ресурса, перезапуск, балансировка, развертывание дополнительных серверов (масштабирование) и т.п.
- Для этого так же есть специализированные решения
- Прежде всего - Kubernetes

Оркестрация

- Стандартная схема работы
  - Разрабатываем приложение
  - Упаковываем его в docker контейнер
  - Разворачиваем в kubernetes (частный, или в публичное облако)



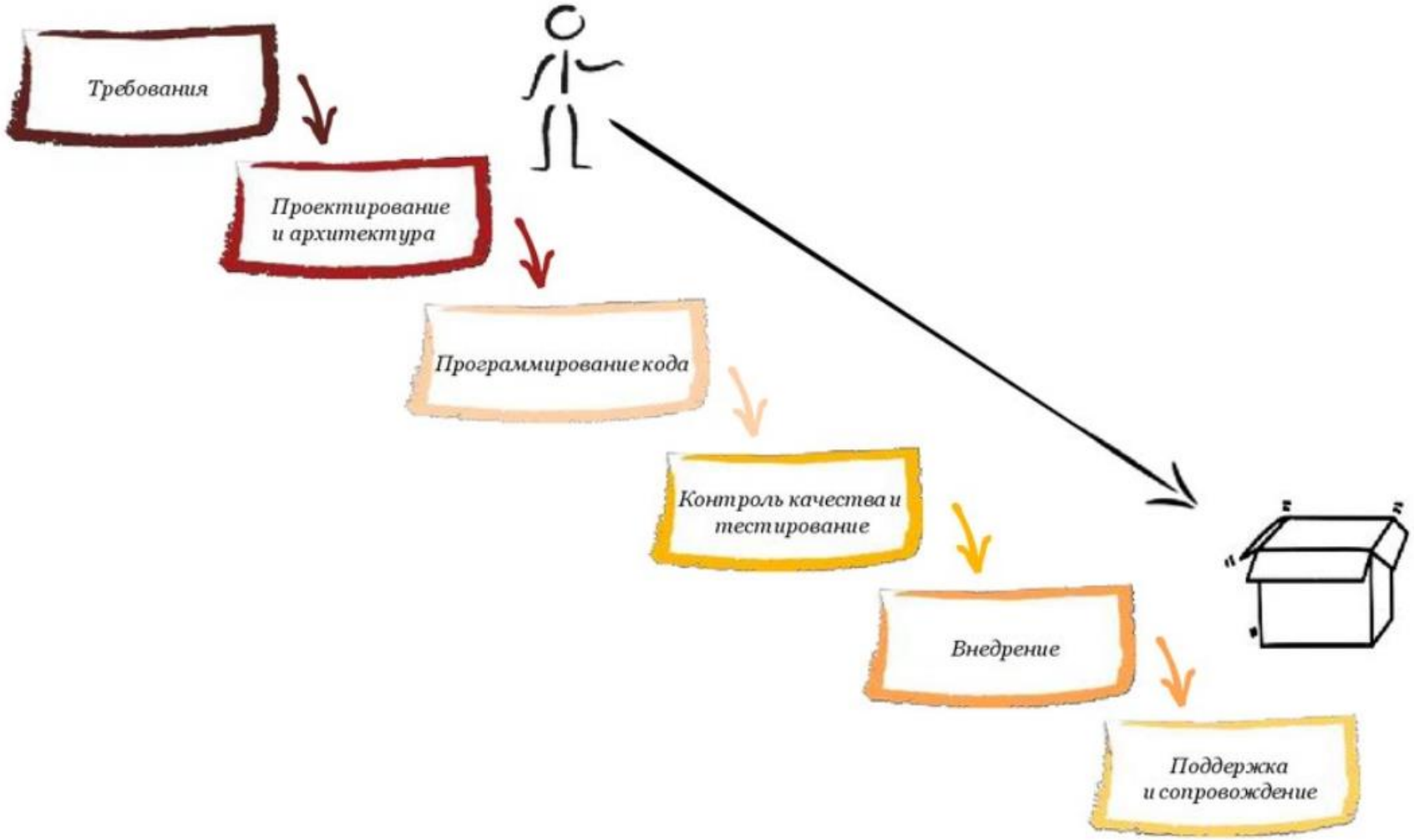
Оркестрация



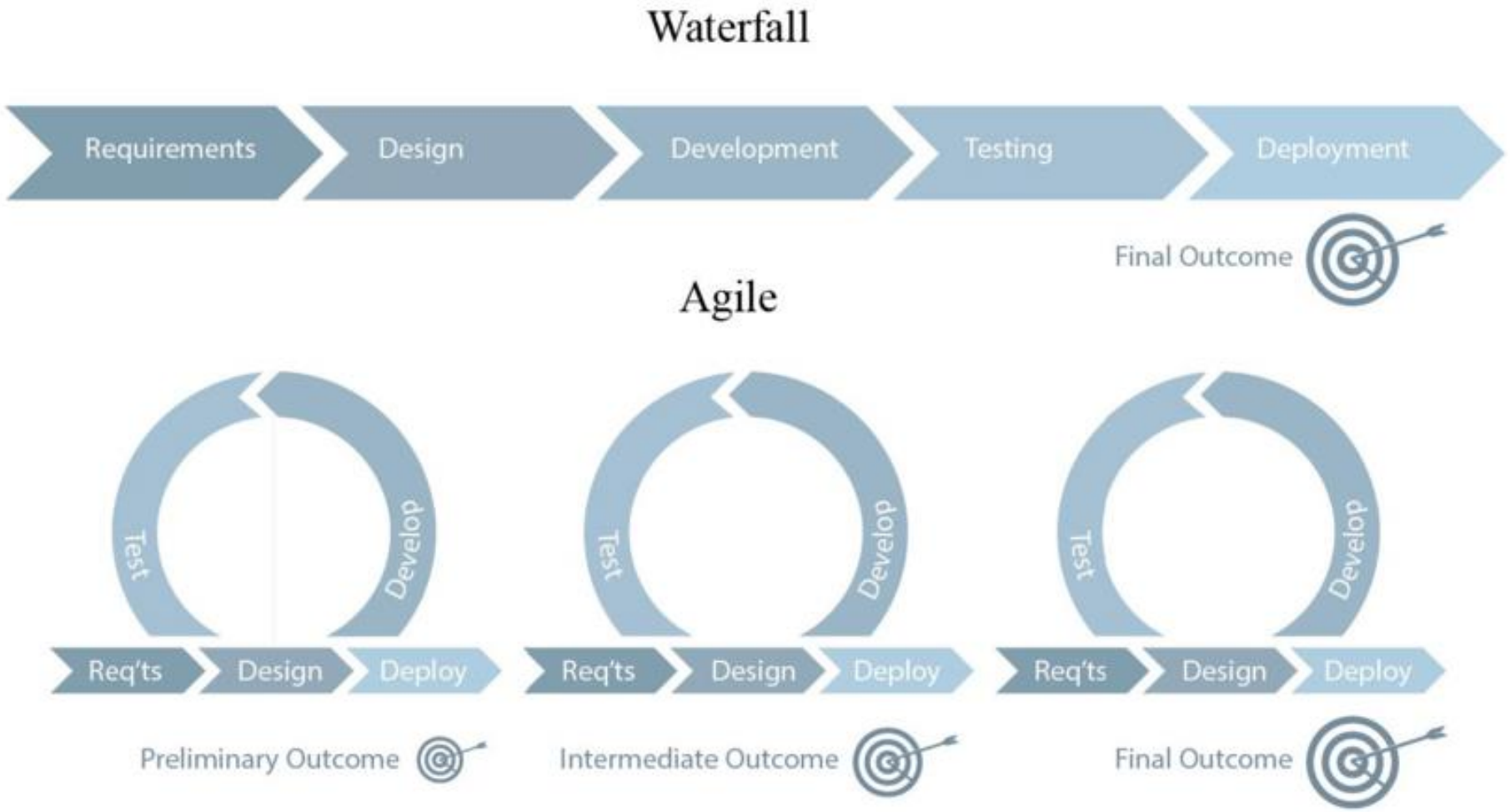
Жизненный цикл ПО

- Помимо технологий, серьезные изменения произошли и в методологии разработки, в частности, это касается жизненного цикла разработки ПО
  - Скорее, развитие одного делало возможным изменение другого
- Известны классические стадии разработки
  - Анализ требований – проектирование – разработка – тестирование – внедрение
  - Модель «waterfall»

Каскадный метод

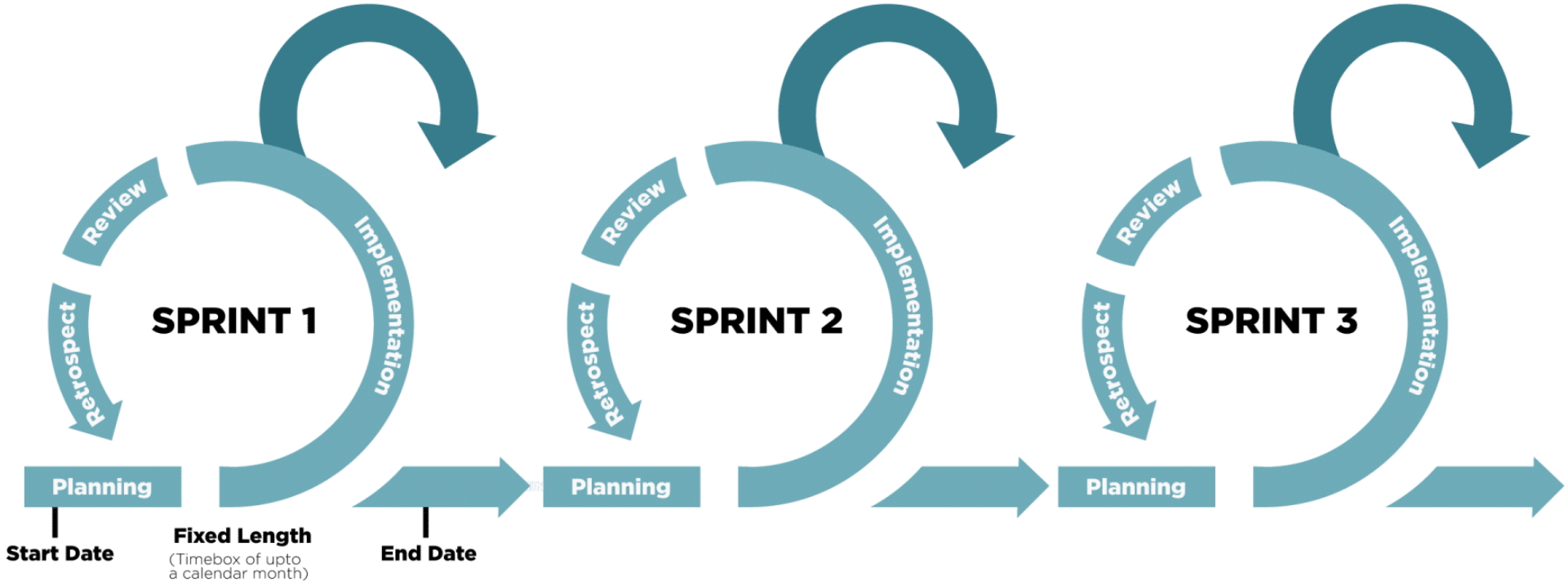


Гибкая методология разработки



Гибкая методология разработки

- При разработке используются спринты – короткие периоды, на выходе которых отдается готовый результат



Гибкая методология разработки

- Такой подход, например, внедрен в проектную модель обучения
  - Ну или, по крайней мере, его пытаются внедрить
  - Вся методология работы в проектах построена по модели Agile
- Есть еще такие связанные термины такие как Scrum, Kanban
- Главное отличие – такая методология позволяет часто выпускать обновление продукта
  - Сокращается «time-to-market»
  - И это влечет за собой изменение всей организации работы

Гибкая методология разработки

- Частая выкатка обновлений требует изменения всей структуры работы
  - Другой жизненный цикл ПО
  - Изменение культуры разработки и тестирования
    - Внедрение автоматизированного тестирования, налаживание тесного взаимодействия между development и operations командами
  - Внедрение средств автоматизации при всех стадиях создания приложения
- И мы постепенно подходим к continuous integration, continuous delivery

Автоматизация DevOps

- При частой выкатке обновлений, невозможно продолжать делать все операции вручную
  - Главное это не нужно
- Несколько тезисов
  - Лень и изобретательность – двигатель прогресса
    - Зачем повторять рутинные операции, если их можно делать автоматически
  - Все, что может быть формализовано, может быть автоматизировано
    - Процедуры сборки, упаковки, доставки на сервер, тестирования, достаточно хорошо формализуются

Автоматизация DevOps

- Важный момент – корень доверия
  - Или корень правильных исходников, настроек и т.п.
  - Что именно должно быть собрано/оттестировано/доставлено
- Иными словами, «где именно лежит правильная версия»
  - Это нужно знать, чтобы знать что именно и когда разворачивать в продакшн, как откатываться, если все упало и т.п.
- И тут на помощь приходит система контроля версий
  - При условии правильного выстраивания workflow

Автоматизация DevOps

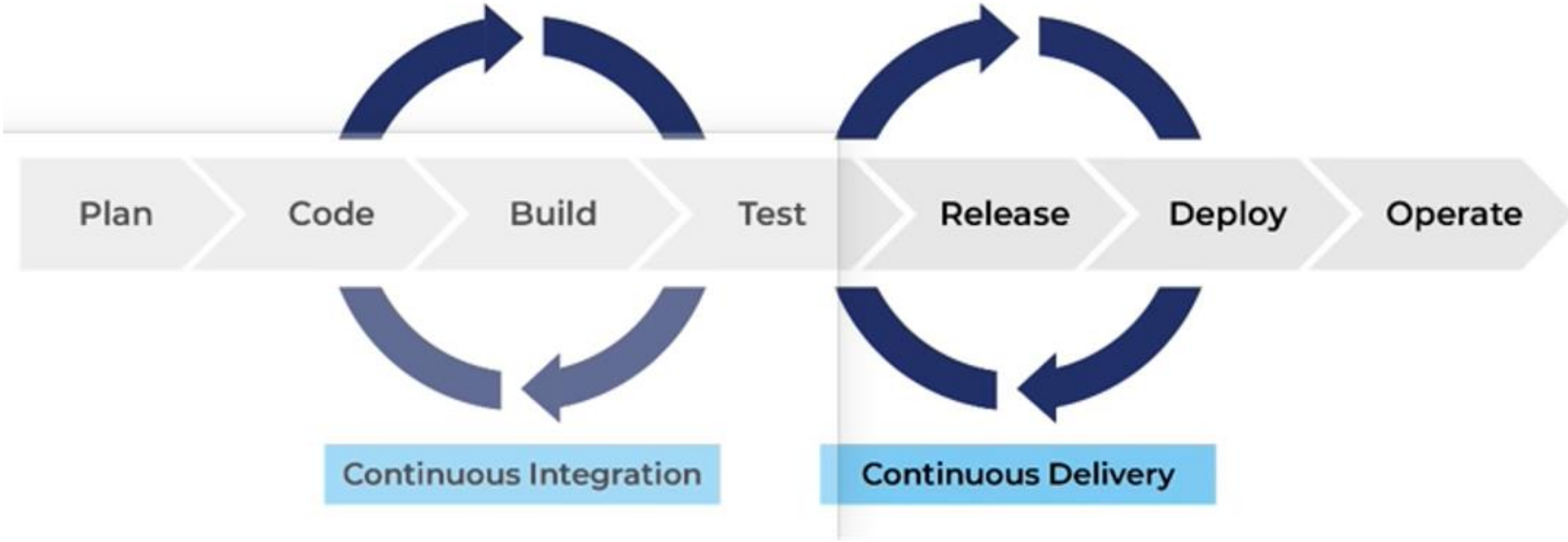
- Все кусочки пазла сошлись
  - Новая методология разработки
    - Необходимость регулярно и быстро выкатывать приложение
  - Код, который лежит в репозитории – корень доверия
    - Код, который лежит в предыдущих коммитах – предыдущие версии приложения
  - Форма передачи исходников – стандартный контейнер
  - Кластер или публичное облако – берут на себя решение задачи поддержания работоспособности
- Осталось только настроить скрипты

CI/CD

- Скрипты – это собственно автоматизация всех рутинных процессов
  - Тут нет ничего нового, они были всегда
  - Но теперь они стали более унифицированы, появились технические средства поддержки такой автоматизации, стандартные контейнеры и т.п.
- Все это называется непрерывная интеграция и непрерывное внедрение (доставка)
  - Про них мы более подробно поговорим позже
- Это скорее не какой-то конкретный продукт, и идеология конвейера при прохождении стандартных стадий жизни кода и внедрения его в продакшн

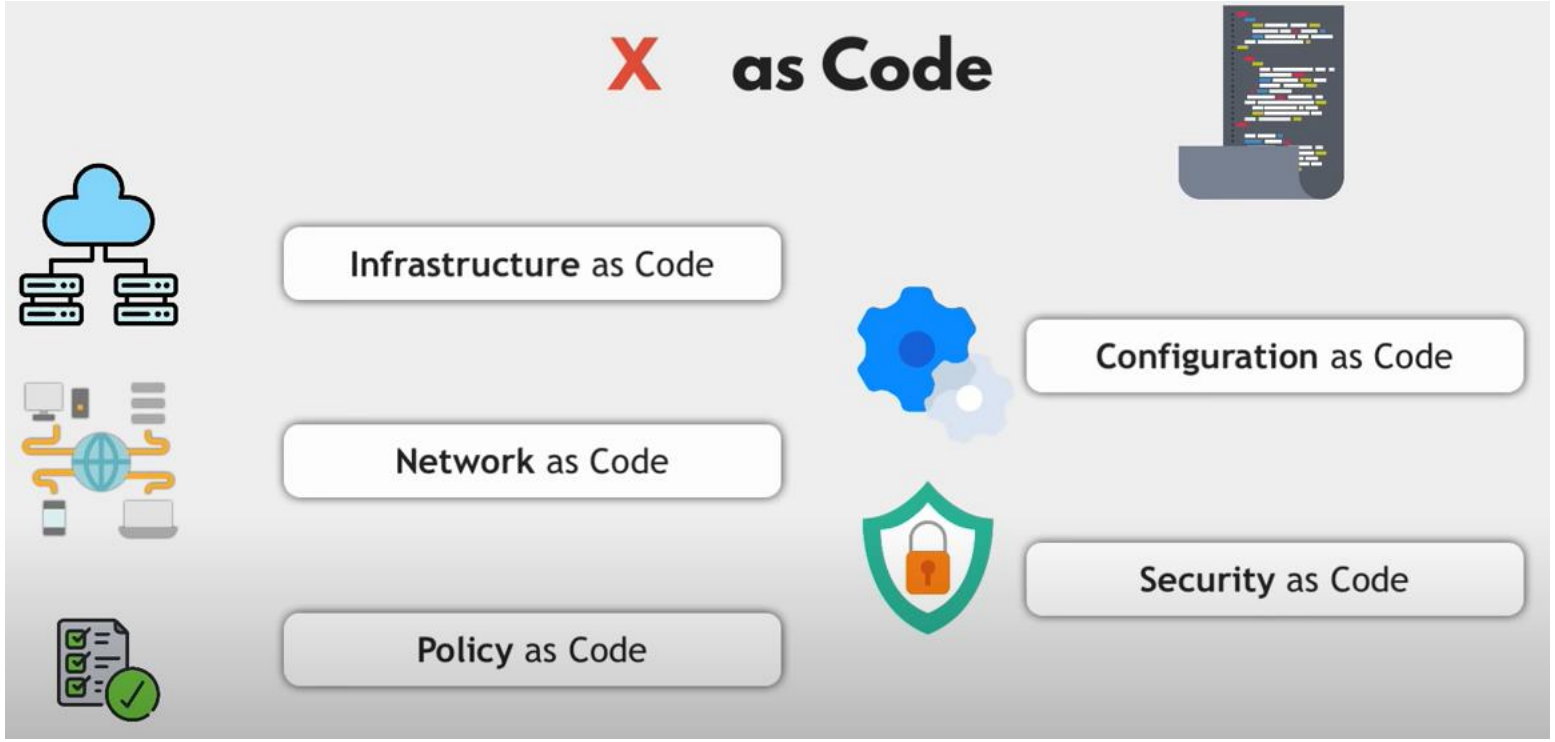
CI/CD

- CI/CD – это собственно continuous integration (непрерывная интеграция) и continuous delivery (непрерывная доставка)



X as Code

- Все что можно формализовать – можно автоматизировать
- В git можно хранить не только исходники



IaC

- Сейчас очень активно развивается тема Infrastructure as Code
  - Решения вопросов управления инфраструктурой через конфигурационные файлы
  - Terraform для подготовки инфраструктуры
  - Ansible для управления конфигурацией
- Все это так же интегрируется с CI/CD

GitOps

- Новый термин GitOpts
  - **GitOps** = IaC + MRs + CI/CD
- IaC - Infrastructure as code
- MR – Merge requests
- Continuous integration and deployment