OTUS

C# разработчик

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Введение в CI/CD

Преподаватель

Новиков Александр

Cloud Architect

Microsoft Certified Trainer

Цели вебинара

После занятия вы сможете:

1. Понимать преимущества CI/CD
2. Создавать собственные CI/CD pipelines

План полета

- DevOps
- Continious Integration
- Build-серверы для CI
- Continuous Delivery, инструменты для CD
- Continuous Delivery VS Continuous Deployment

Ссылка на проект

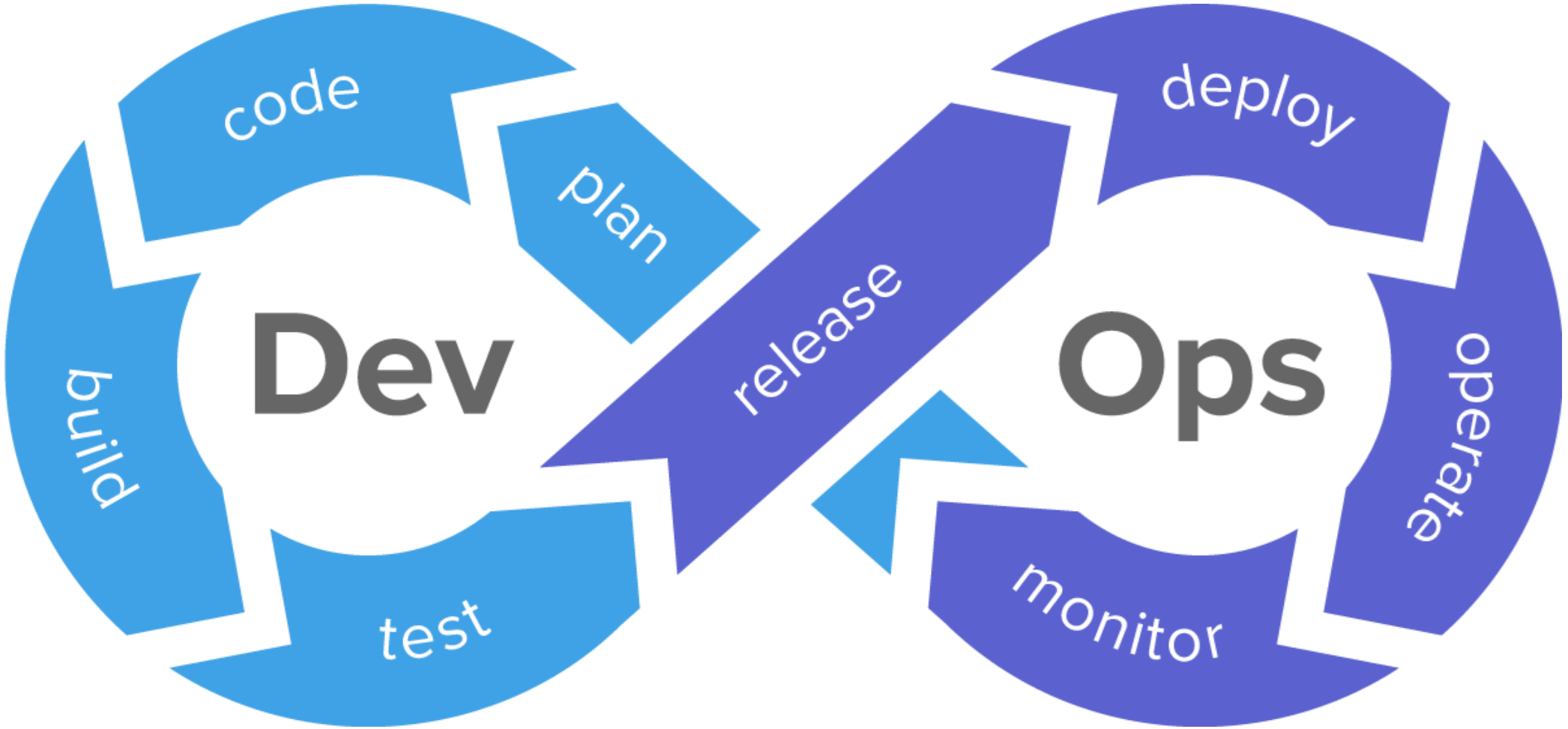
1. https://gitlab.com/AleksandrN/otus-cicd-dev
2. https://dev.azure.com/alexNew/OTUS
3. https://github.com/AleksandrNew/otus-cicd

DevOps

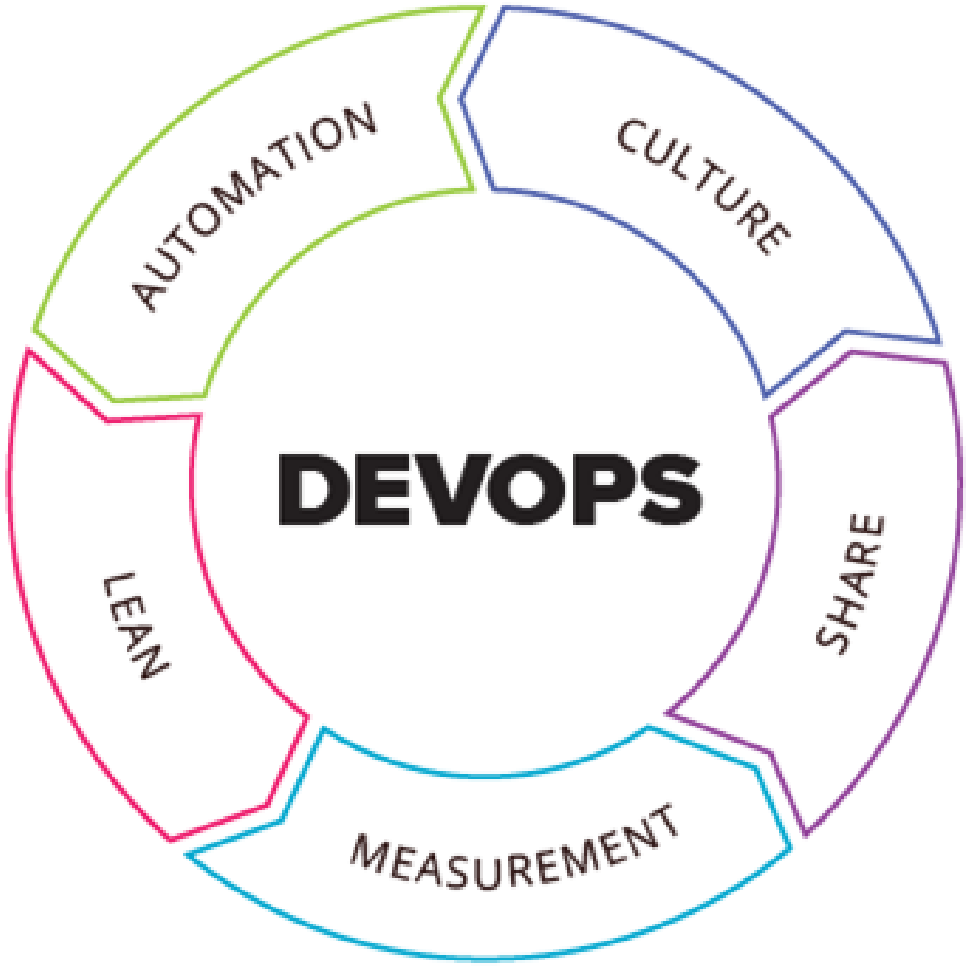
DevOps

DevOps (акроним от англ. **development и operations**) — это методология разработки ПО, сфокусированная на предельно активном взаимодействии и интеграции в одной упряжке программистов, тестировщиков и админов, синхронизировано обслуживающих общий для них сервис/продукт.

DevOps


code
plan
build
test
Dev
release
deploy
operate
monitor
Ops

CALMS



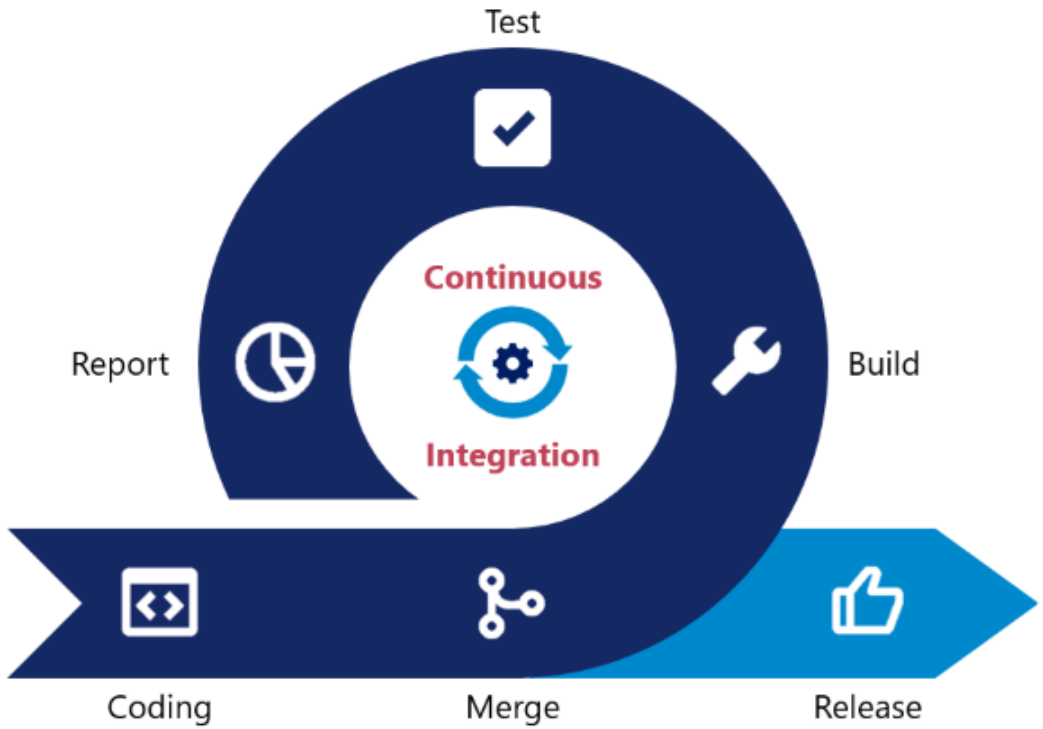
Преимущества DevOps

- Быстрый выход в продакшен
(reduce time to market)
- Уменьшение количества сбоев, откатов и времени на восстановление
- Постоянная обратная связь
- Улучшение качества продукта
- Автоматизация монотонных задач
- Экономия денег

Continuous Integration

Continuous Integration (CI)

Continuous Integration — практика разработки программного обеспечения, которая заключается
в постоянном слиянии рабочих копий в общую основную ветвь разработки (до нескольких раз
в день) и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем.



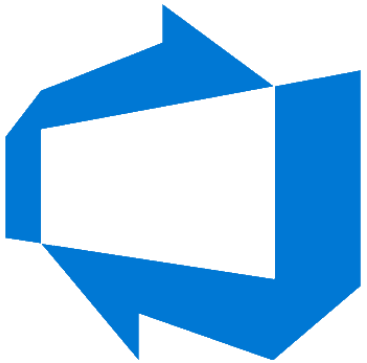
Преимущества CI

- Мёрджить маленькие изменения проще
- Code review будет проще
- Автотесты будут проверять атомарные изменения
- Запуск на каждый коммит и/или мёрдж-реквест
- Выкатка фичи по готовности
- С основной ветки разработки можно собрать рабочий билд

Требования хорошего CI

- Фича-бранчи
- Автотесты
- Правила мёрдж-реквестов (code review, linters, smoke tests)
- Тулы для запуска ci

CI tools



Azure DevOps

GitLab

Jenkins

Различие тулов

- Стоимость и лицензия
- Установка (on-premise, SaaS)
- Платформа (windows, linux, macos, public clouds)
- Интеграция и плагины
- Сложность ci-процессов
- Способ запуска агентов

Типичные шаги CI-процесса

1. создать фича-ветку
2. git commit && git push
3. собрать билд
4. запустить unit-тесты
5. задеплоить его на тестовое окружение
6. запустить smoke-тесты
7. если что-то упало - поправить и вернуться к пункту 2
8. если всё прошло успешно - создать мёрдж реквест
9. code review и опционально дополнительные проверки (статический анализ кода, расширенный набор тестов)
10. слить ветку в основную, закрыть мр и удалить фича-ветку

LIVE

Continuous Delivery

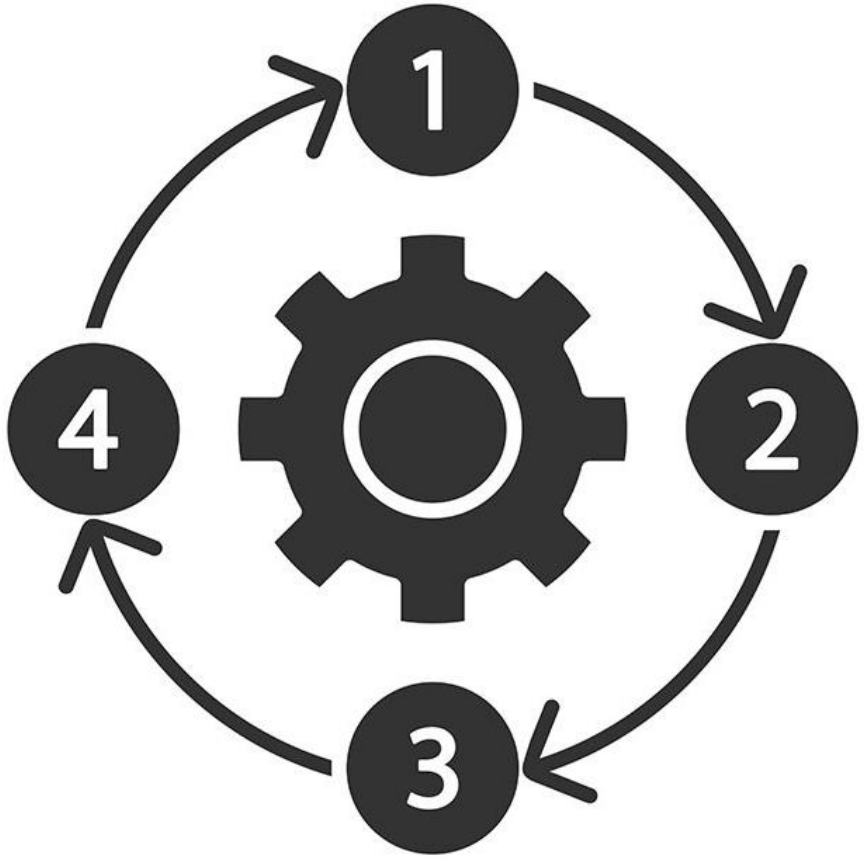
Continuous Delivery

Continuous Delivery – подразумевает развертывание кода после каждой новой интеграции. Эта операция проводится вручную или автоматически.

Continuous Deployment

Continuous Deployment

Continuous Deployment — уже полностью автоматический процесс, проходящий без контроля со стороны команды.



Приемущества Continuous Deployment

- Быстрый деплой
(reduce time to market)
- Быстрый roll-back в случае проблем
- Уменьшение человеческого фактора

Приемущества Continuous Deployment

- сходить руками на каждый сервак
- удалять старую версию приложения
- поставить новую
- и надеяться, что ничего не забыл

LIVE

Стратегии деплоя

Пересоздание (Recreate)

Recreate – полная остановка текущей работающей версии приложения и запуск новой версии.

Плюсы	• Позволяет мигрировать данные • Позволяет деплоить при запрету на одновременную работу двух версий одного приложения. • Состояние приложения полностью обновлено
Минусы	• Downtime

Инкрементальный (Rolling-update)

Rolling – при выборе этой стратегии платформа дожидается старта новой версии приложения и только после этого завершает работу старой.

Плюсы	• Нет Downtime
Минусы	• Может занять долгое время • Саппорт двух версий API • Плохой контроль трафика

Blue/Green

Blue/Green – предусматривает одновременное развертывание старой (зеленой) и новой (синей) версий приложения. После размещения обеих версий обычные пользователи получают доступ к зеленой, в то время как синяя доступна для QA-команды для автоматизации тестов через отдельный сервис или прямой проброс портов:

Плюсы	• Нет Downtime • Возможность быстрого отката к старой версии приложения • Возможность тестирование в продакшен
Минусы	• Дублирование ресурсов • Саппорт двух версий API

Канареечный деплой (Canary)

Canary – данная стратегия даёт возможность проверить работоспособность новой версии приложения на ограниченном числе пользователей. Обычно это делается при помощи правил распределения трафика – например 10% пользователей, проходя по одному и тому же адресу в браузере, попадают на новую версию приложения.

Плюсы	• Нет Downtime • Возможность быстрого отката к старой версии приложения • Возможность тестирование в продакшен на небольшом количестве реальных пользователей
Минусы	• Дублирование ресурсов • Медленное развертывание

A/B

A/B – реализовано путем добавления дополнительных функций к канареечному развертыванию за счет перенаправления траффика по определенным критериям (browser cookie, query parameters, geolocalisation, etc).

Плюсы	• Нет Downtime • Несколько версий работают параллельно • Полный контроль над распределением трафика
Минусы	• Дублирование ресурсов • Требуется интеллектуальный механизм балансировки нагрузки • Сложность траблшутинга

Shadow

Shadow – дублирование траффика на дополнительно развернутые новые версии приложения.

Плюсы	• Нет Downtime • Тестирование производительности приложения с продакшн трафиком • Нет импакта для пользователей • Стабилизация перед выходом в продакшен
Минусы	• Дублицирования ресурсов • Сложная установка • Требуются mock’и

Deployment strategies

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

Container Solutions

Вопросы для самопроверки

1. Что такое DevOps?
2. Что такое Continuous Integration?
3. Что такое Continuous Delivery?
4. Что такое Continuous Deployment?
5. Какие бывают стратегии деплоя?

Напишите в чат/ответьте голосом

5 мин

Заполните, пожалуйста, опрос

Спасибо за внимание! Приходите на следующие вебинары