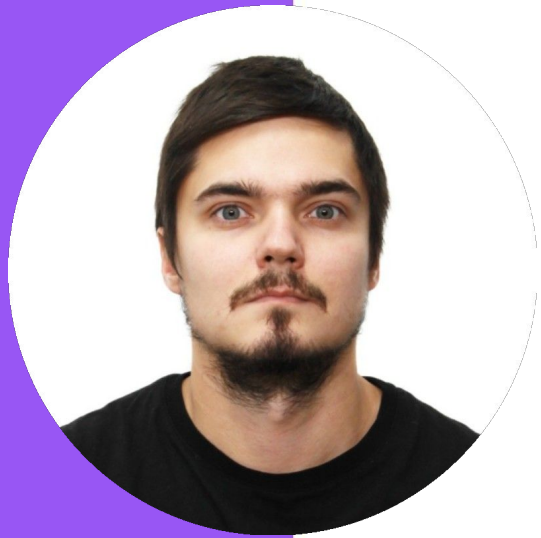
Жизненный цикл, Workflow и подходы в разработке

CI/CD на основе GitLab

REC **Проверить, идет ли запись**

Меня хорошо видно & слышно?



Николай Осипов

MLOps Engineer

- 5+ лет в области **Data Science**
- **MLOps**, DS инфраструктура, табличные данные
- Выступаю наставником и преподавателем на курсах по **ML**
- Преподаватель курса **CI/CD на основе GitLab**

@NickOsipov

github.com/NickOsipov

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в учебной группе

Задаем вопрос
в чат

Вопросы вижу в чате,
могу ответить не сразу

Карта курса

СТАРТ

CI/CD - системы, подходы и workflow

Gitlab CI

Безопасность

Проектная работа

ФИНИШ

Маршрут вебинара

- Знакомство
- Жизненный цикл разработки ПО
- Workflow в разработке ПО
- Основные подходы к разработке ПО
- Git Workflows
- Рефлексия

Цели вебинара

К концу занятия вы сможете

1. Изучить жизненный цикл разработки программного обеспечения и его ключевые этапы: инициирование, анализ требований, проектирование, разработка, тестирование, внедрение и сопровождение.

2. Различать основные подходы к разработке программного обеспечения: водопадная модель, Agile, Scrum, Twelve-Factor App, Git Flow, GitHub Flow.

3. Разобраться в workflow процесса разработки и роли каждого шага: создание задачи, планирование, разработка, code review, тестирование, релиз и поддержка.

Кто из вас знает и применяет подходы к разработке? Какие подходы?

Жизненный цикл разработки ПО

Жизненный цикл разработки программного обеспечения

это процесс, включающий последовательность этапов от идеи до реализации и поддержки программного продукта.

Основные этапы:

1. Инициирование.
2. Анализ требований.
3. Проектирование.
4. Разработка.
5. Тестирование.
6. Внедрение и сопровождение.

https://xbsoftware.ru/blog/zhiznennyj-tsykl-testirovaniya-po-preimuschestva-shagi/

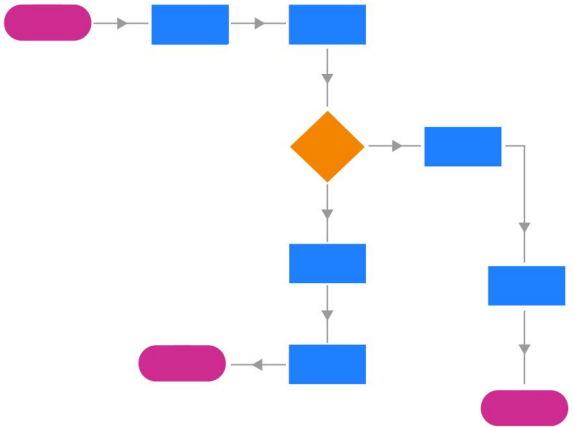
Workflow в разработке ПО

Workflow в разработке программного обеспечения

Workflow — это последовательность шагов и действий, которые выполняются в процессе разработки программного обеспечения.

Примерный Workflow в разработке ПО:

1. Создание задачи.
2. Планирование.
3. Разработка.
4. Код ревью.
5. Тестирование.
6. Релиз.
7. Поддержка.



Основные подходы к разработке ПО

Основные подходы к разработке ПО (Waterfall)

Водопадная модель разработки (Waterfall) - одна из самых классических методологий, которая использовалась с первых этапов эволюции программного обеспечения.

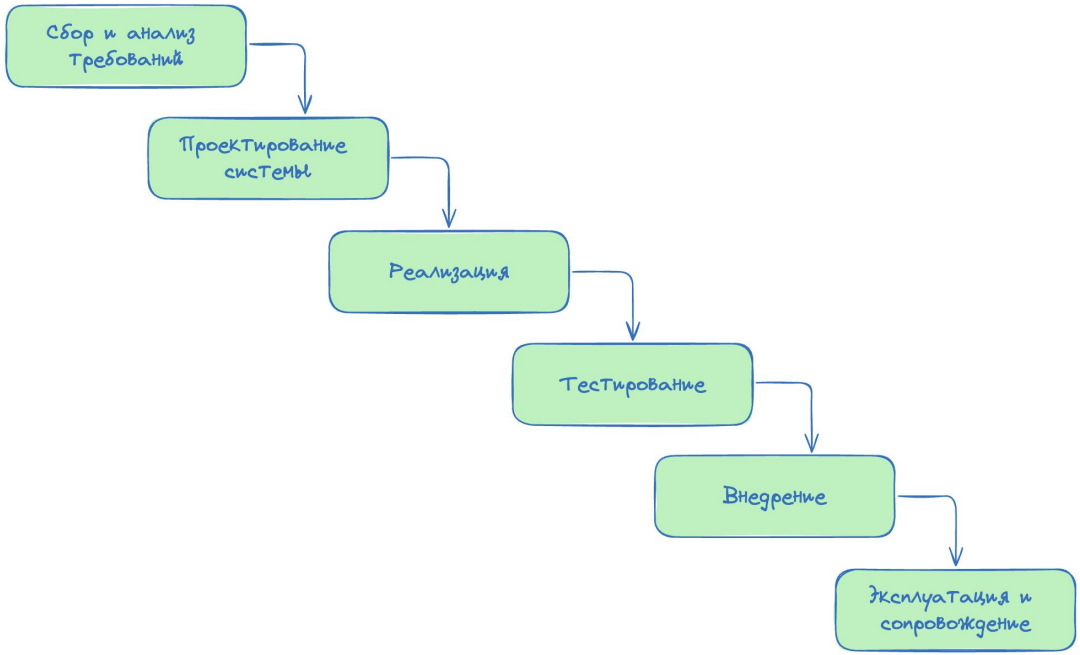
Основные принципы водопадной модели:

- Последовательные фазы
- Жесткие требования
- Документирование на каждом этапе

Waterfall

Этапы водопадной модели:

1. Сбор и анализ требований
2. Проектирование системы
3. Реализация
4. Тестирование
5. Внедрение
6. Эксплуатация и сопровождение



Waterfall

Преимущества водопадной модели:

- Простота и четкость
- Четкое документирование
- Легкость управления

Недостатки водопадной модели :

- Отсутствие гибкости
- Высокие риски
- Длительное время до получения результата

Когда использовать водопадную модель?

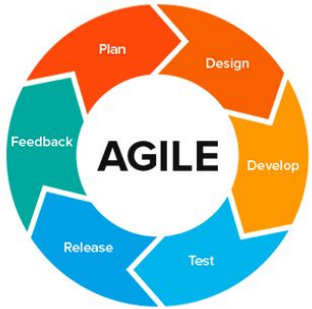
- Когда требования четко определены и не будут изменяться.
- Когда проект небольшой и сроки фиксированы.
- Когда есть высокая степень уверенности в технологическом решении.

Основные подходы к разработке ПО (Agile)

Agile — одном из самых популярных подходов к разработке программного обеспечения, который отличается своей гибкостью и быстрой адаптацией к изменениям.

Основные принципы Agile:

- Итеративный подход
- Гибкость и адаптивность
- Приоритет взаимодействия с заказчиком
- Кросс-функциональные команды



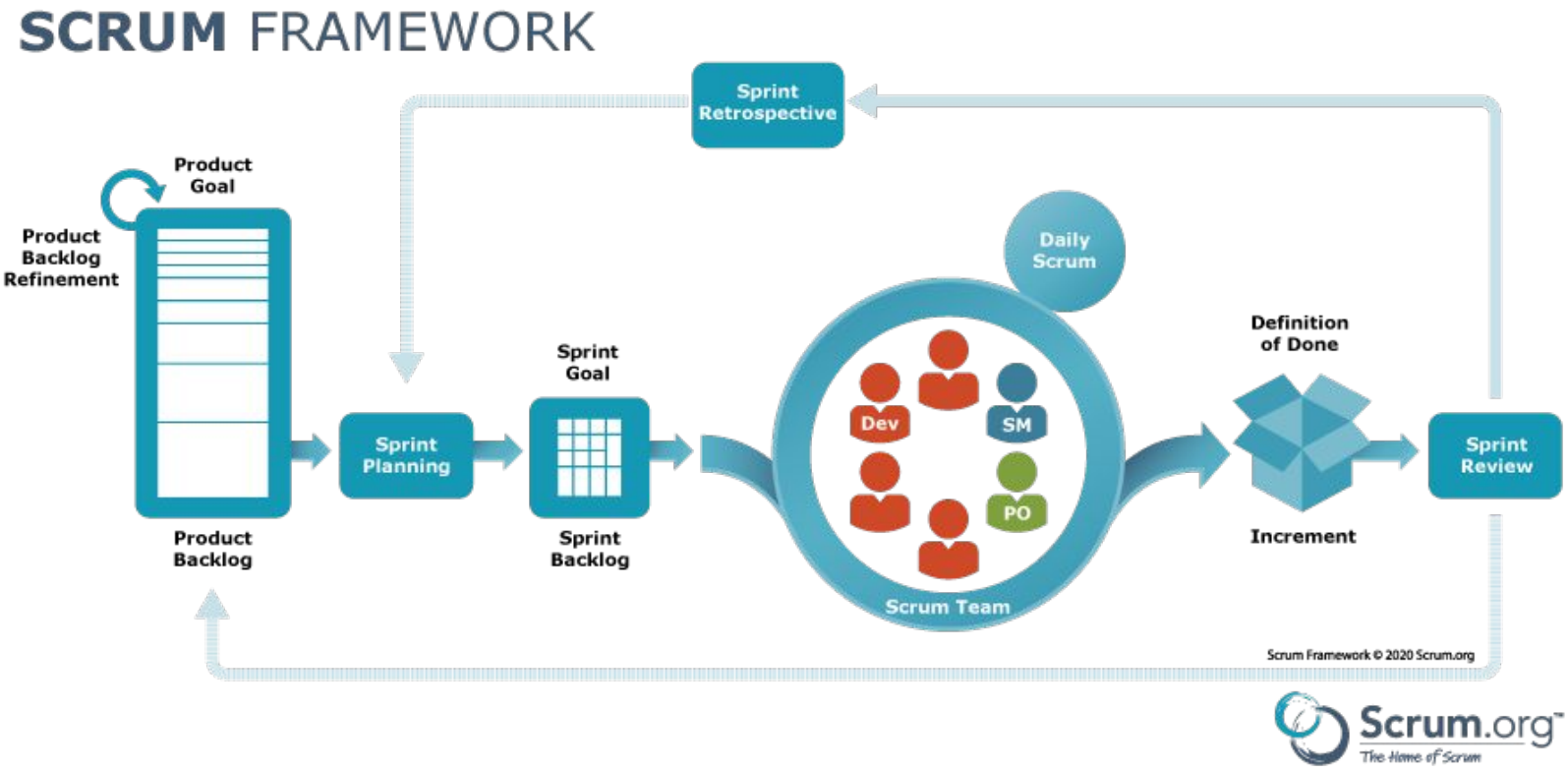
https://mymonday.by/dispelling-common-agile-misconceptions

Agile

Основные этапы Agile:

1. Формирование бэклога
2. Планирование спринта
3. Реализация и разработка
4. Ежедневные встречи
5. Демонстрация результата
6. Ретроспектива

Scrum


SCRUM FRAMEWORK
Sprint Retrospective
Product Goal
Product Backlog Refinement
Product Backlog
Sprint Planning
Sprint Goal
Sprint Backlog
Daily Scrum
Dev
SM
PO
Scrum Team
Definition of Done
Increment
Sprint Review
Scrum Framework © 2020 Scrum.org
Scrum.org
The Home of Scrum

Agile

Преимущества Agile:

- Гибкость
- Регулярная обратная связь
- Меньше рисков
- Улучшение взаимодействия в команде

Недостатки Agile:

- Сложность прогнозирования
- Требование высокой вовлеченности заказчика
- Неопределенность на ранних этапах

Когда использовать Agile?

- Когда проект сложный и требования могут изменяться.
- Когда важна быстрая доставка продукта.
- Когда необходим тесный контакт с заказчиком.

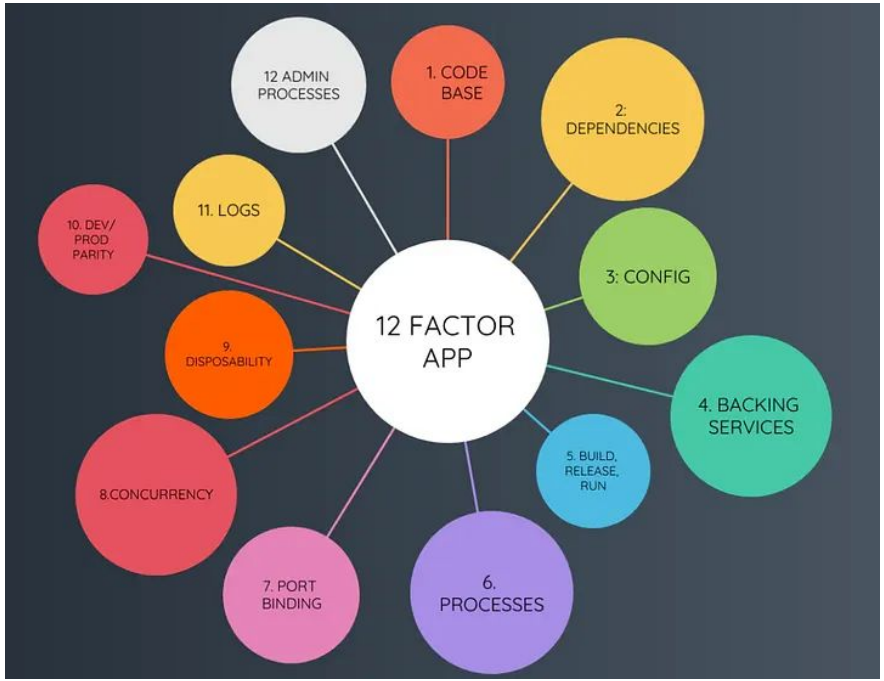
Основные подходы к разработке ПО (Twelve-Factor App)

Twelve-Factor App — это методология, созданная для разработки современных веб-приложений, которые легко масштабируются, разворачиваются и поддерживаются в облачной среде.

Twelve-Factor App

Основные принципы Twelve-Factor App:

1. Кодовая база
2. Зависимости
3. Конфигурация
4. Сервисы
5. Сборка, релиз, запуск
6. Процессы
7. Привязка портов
8. Параллелизм
9. Одноразовость
10. Совместимость между средами
11. Логирование
12. Административные процессы



Twelve-Factor App

1. **Единая кодовая база:** У каждого приложения должна быть одна кодовая база, отслеживаемая в системе контроля версий, с возможностью развертывания в нескольких экземплярах.

2. **Явное объявление зависимостей:** Все зависимости должны быть явно указаны и изолированы, чтобы избежать неожиданных проблем при развертывании в разных средах.

3. **Конфигурация в среде:** Конфигурационные данные, такие как ключи API или строки подключения, должны храниться в переменных среды, а не в коде.

Twelve-Factor App

4. **Сторонние сервисы как ресурсы:** Все внешние сервисы (базы данных, очереди сообщений и т.д.) рассматриваются как прикрепляемые ресурсы, доступ к которым осуществляется через конфигурацию.

5. **Разделение этапов сборки, релиза и запуска:** Процесс разработки должен быть разделен на три этапа: сборка (build), релиз (release) и запуск (run), чтобы обеспечить четкое разграничение и управляемость.

6. **Процессы:** Приложение должно работать как один или несколько stateless-процессов, где состояние хранится во внешних сервисах.

Twelve-Factor App

7. **Привязка к порту:** Приложение должно быть самодостаточным и предоставлять свои услуги через привязку к определенному порту.

8. **Параллелизм:** Приложение должно быть способно масштабироваться горизонтально, запуская несколько экземпляров процессов для обработки увеличивающейся нагрузки.

9. **Одноразовость:** Процессы должны быть легко запускаемыми и останавливаемыми, что способствует устойчивости и быстрому восстановлению после сбоев.

Twelve-Factor App

10. **Паритет между разработкой и продакшеном:** Среды разработки, тестирования и продакшена должны быть максимально схожими, чтобы минимизировать ошибки при переходе между ними.

11. **Логирование как поток событий:** Приложение должно записывать логи как непрерывный поток событий, который может быть перенаправлен в различные системы для хранения и анализа.

12. **Административные процессы:** Одноразовые административные задачи (например, миграция базы данных) должны выполняться как отдельные процессы в той же среде, что и основное приложение.

Twelve-Factor App

Преимущества использования Twelve-Factor App:

- Масштабируемость
- Упрощение деплоя
- Облачная совместимость
- Гибкость в управлении окружениями
- Независимость от платформы

Когда использовать Twelve-Factor App?

- При разработке облачных приложений
- При микросервисной архитектуре
- При построении CI/CD пайплайнов

Git Workflows

Git Flow

Git Flow — это одна из самых популярных моделей работы с ветвлениями в Git.

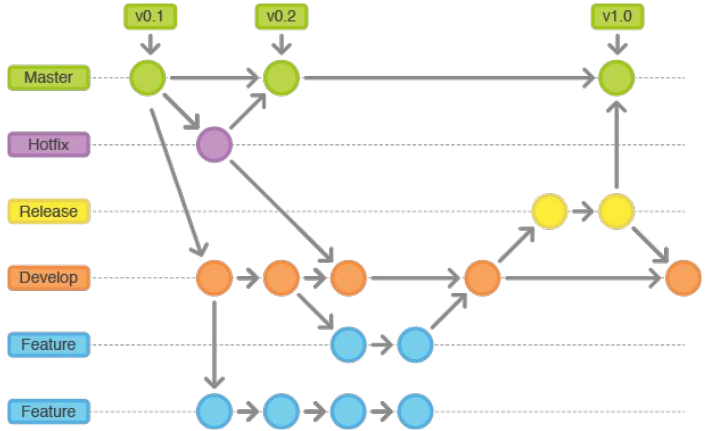
Git Flow основывается на четком разделении работы в различных ветках для упрощения процесса разработки, тестирования и релиза продукта.

Основные ветки:

- main
- develop

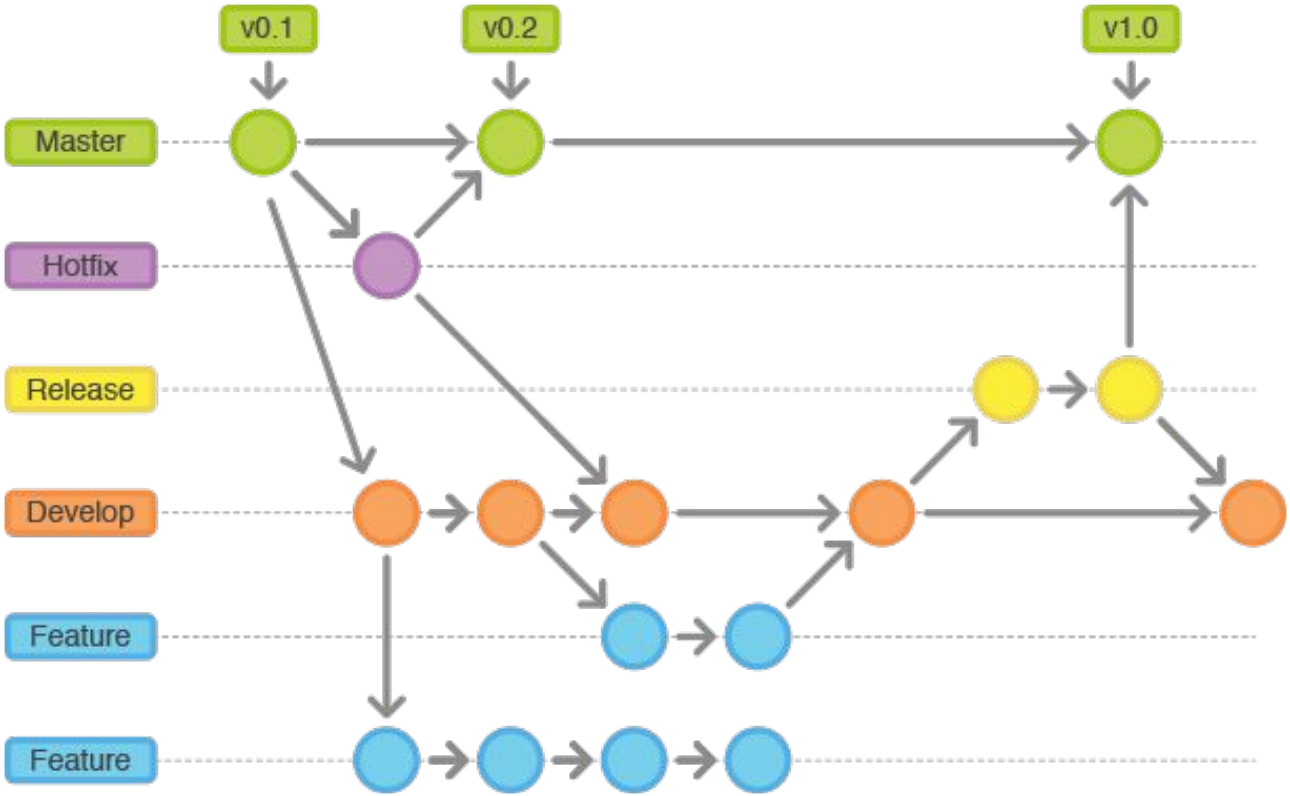
Поддерживающие ветки:

- feature
- release
- hotfix



https://leanpub.com/git-flow/read

Git Flow



Git Flow

Основные этапы работы в Git Flow:

1. Создание новой фичи
2. Подготовка релиза
3. Экстренное исправление
4. Завершение работы над ветками

Основные подходы к разработке ПО (Git Flow)

Преимущества Git Flow:

- Структурированная разработка
- Параллельная работа
- Минимизация риска ошибок в продакшн
- Удобная система релизов

Недостатки Git Flow:

- Сложность для небольших проектов
- Медленность в быстрых релизах
- Обучение команды

Когда использовать Git Flow?

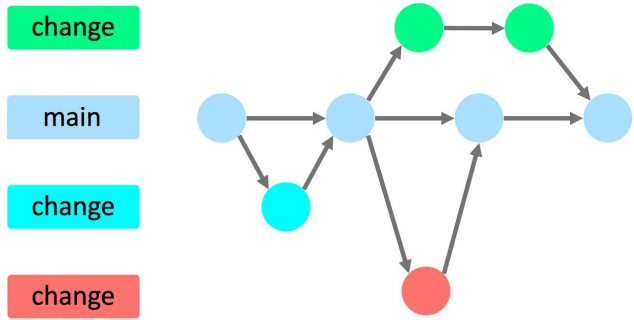
- Большие проекты
- Четкий релизный цикл
- Наличие отдельной команды для релизов

GitHub Flow

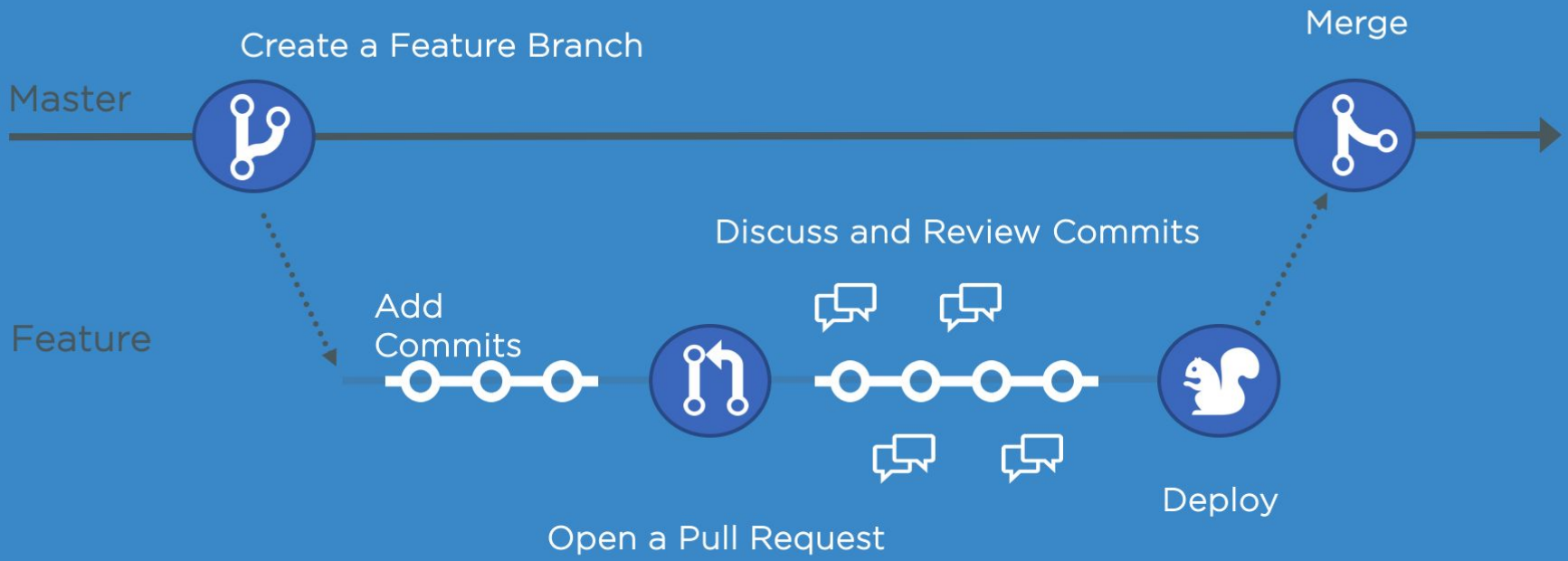
GitHub Flow — это простая и эффективная модель работы с ветвлением в Git, созданная специально для проектов, которые разрабатываются с использованием платформы GitHub.

Основные принципы GitHub Flow:

- Основная ветка (main)
- Создание новой ветки для изменений
- Постоянные коммиты в ветку
- Открытие Pull Request (PR)
- Тестирование и Review изменений
- Слияние Pull Request
- Деплой после слияния



GitHub Flow


Merge
Create a Feature Branch
Master
Discuss and Review Commits
Add
Commits
Feature
Deploy
Open a Pull Request

GitHub Flow

Преимущества GitHub Flow:

- Простота и гибкость
- Быстрые релизы
- Простота для Continuous Delivery
- Эффективность при небольшом количестве разработчиков
- Прозрачность через Pull Requests

Недостатки GitHub Flow:

- Не подходит для сложных релизов
- Трудности с длинными фичами
- Нет поддержки для старых версий

Когда использовать GitHub Flow?

- Небольшие проекты или стартапы
- Проекты с непрерывной интеграцией и доставкой (CI/CD)
- Частые обновления продукта

Основные подходы к разработке ПО (Итеративная модель)

Итеративная модель предполагает многократное повторение этапов разработки с целью улучшения продукта на каждом цикле.

Ключевые особенности:

- Несколько циклов разработки и тестирования.
- Продукт совершенствуется по мере прохождения итераций.

Преимущества:

- Возможность раннего обнаружения ошибок.
- Постоянное улучшение продукта.

Недостатки:

- Требует постоянной коммуникации и управления.
- Подходит для: проектов с постепенным улучшением требований и дизайна.

Основные подходы к разработке ПО (V-модель)

V-модель акцентирует внимание на тестировании, которое происходит параллельно с каждым этапом разработки.

Ключевые особенности:

- Тестирование интегрировано в каждый этап разработки.
- Предполагается строгая структура тестирования.

Преимущества:

- Возможность раннего обнаружения ошибок.
- Постоянное улучшение продукта.

Недостатки:

- Невозможность гибко адаптироваться к изменениям.
- Подходит для: проектов с высокими требованиями к качеству.

Вопросы

+ если есть вопросы

– если вопросов нет

Домашнее задание

1. Напишите, какие ошибки вы видите в процессе разработки. Какой совет (ы) вы можете дать разработчику.
2. Опишите два примера разработки с разными вариантами flow.

Подведём итоги занятия

Теперь вы сможете:

Отправьте в чат номера достигнутых целей

1. Изучить жизненный цикл разработки программного обеспечения (ЖЦРП) и его ключевые этапы: инициирование, анализ требований, проектирование, разработка, тестирование, внедрение и сопровождение.

2. Различать основные подходы к разработке программного обеспечения: водопадная модель, Agile, Scrum, Twelve-Factor App, Git Flow, GitHub Flow.

3. Разобраться в workflow процесса разработки и роли каждого шага: создание задачи, планирование, разработка, code review, тестирование, релиз и поддержка.

Заполните, пожалуйста, опрос о занятии

Мы читаем все ваши сообщения
и берем их в работу ✏️❤️

Приглашаем на следующий вебинар

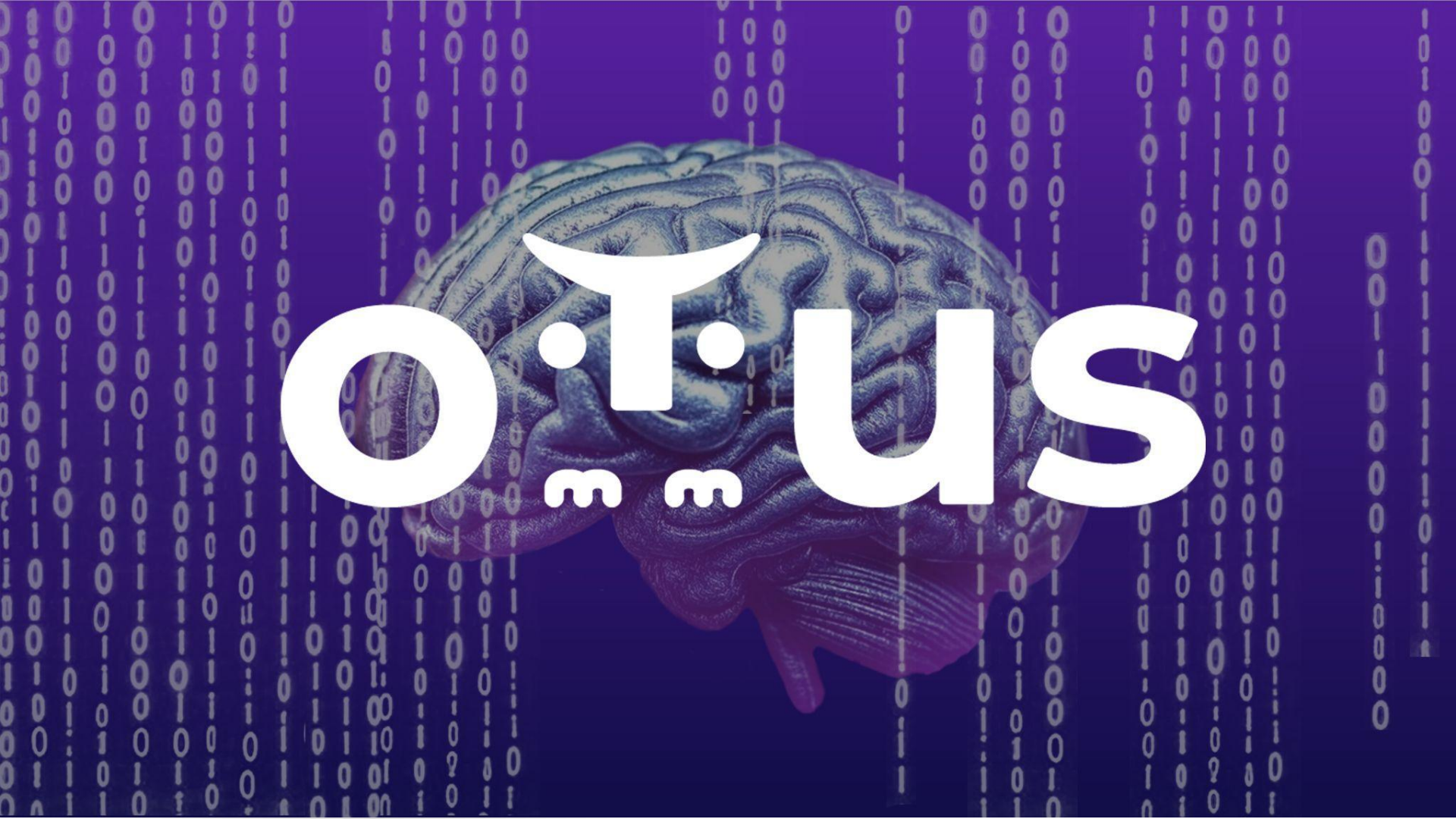
14 октября 2025

Создание и настройка проектов. Тюнинг GitLab Runner

Ссылка на вебинар будет в ЛК за 15 минут

Материалы к занятию в ЛК — можно изучать

Обязательный материал обозначен красной лентой


OTIUS

СКЛАДЧИК

ЭТОТ КОНТЕНТ КУПЛЕН НА САЙТЕ SKLADCHIK.ORG

SKL.TO

ТЫ ЕЩЕ НЕ В КЛУБЕ?
 присоединяйся и начинай экономить!

ЧТО ТАКОЕ КЛУБ «СКЛАДЧИК»?

Платформа
Это платформа, где каждый день тысячи людей собираются вместе, чтобы общими усилиями находить, приобретать и изучать курсы интересные именно им.

Сообщество
Это сообщество из 500 000 людей, открывших для себя выгодный способ быть в тренде самых актуальных и получать ценные знания по минимальной цене.

Библиотека
Это крупнейшая библиотека инфопродуктов в которой можно найти практически любой курс или тренинг продававшийся за последние 10 лет, а также уникальные авторские инфопродукты, которые не получиться найти больше нигде.