

Введение в DevOps



Андрей
Борю



Андрей Борю

Principal DevOps Engineer, Snapcart



План занятия

1. Принципы обучения на курсе
2. Что такое DevOps
3. Шаги создания приложения
4. Настройка рабочей среды
5. Этапы разработки ПО
6. Подходы к разработке
7. Итоги
8. Домашнее задание

Принципы обучения на курсе

Два принципа обучения

Принцип 1. Выполнение домашнего задания	Принцип 2. Изучение дополнительных материалов
<ul style="list-style-type: none">● Выполнять домашние задания, чтобы <i>получить практические навыки</i>● Сдавать домашнее задание до начала следующей лекции● Спрашивать непонятные моменты темы домашнего задания в чате Slack	<ul style="list-style-type: none">● Основные принципы темы вместе разбираем на лекции● Рассказываем, что еще из этой темы может понадобиться для самостоятельного изучения● Даем ссылки на дополнительные материалы, чтобы вы <i>глубже поняли тему</i>

Что такое DevOps

DevOps

Главная задача DevOps инженера – максимально увеличить предсказуемость, эффективность и безопасность разработки ПО.

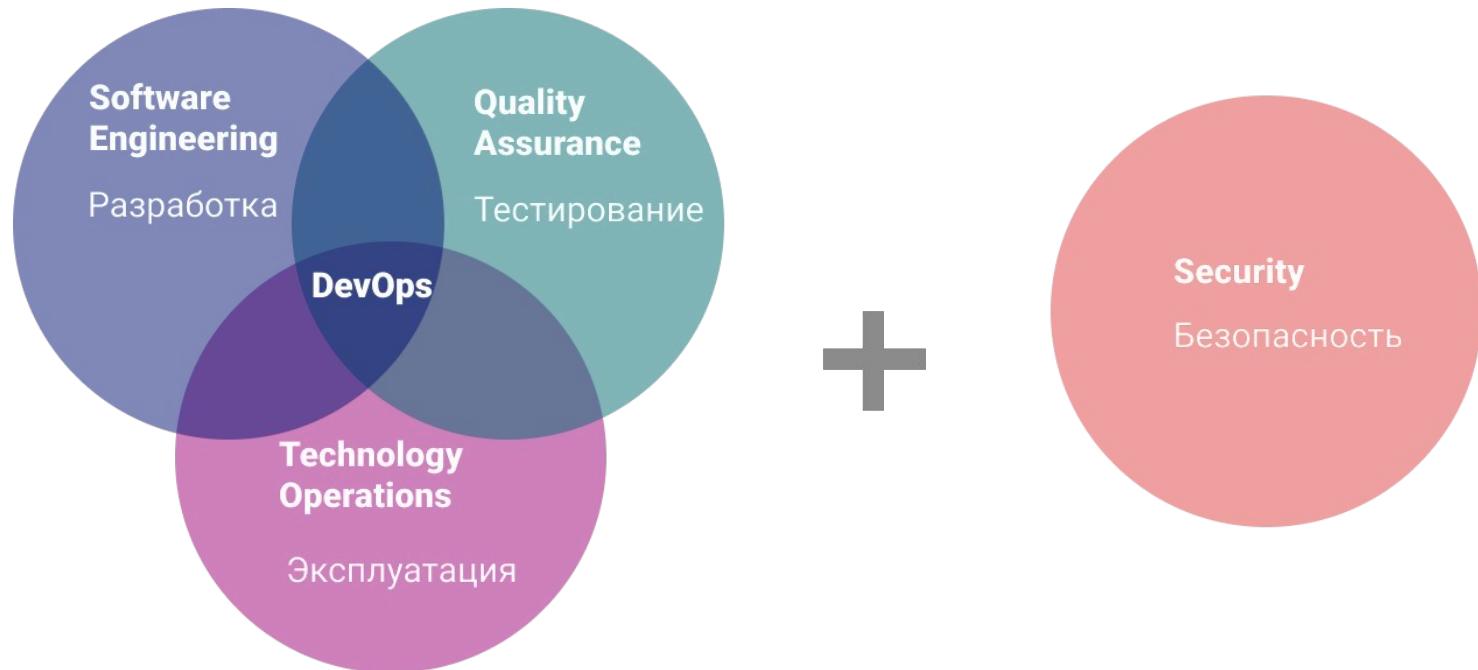
DevOps

DevOps - работа на стыке нескольких должностей:



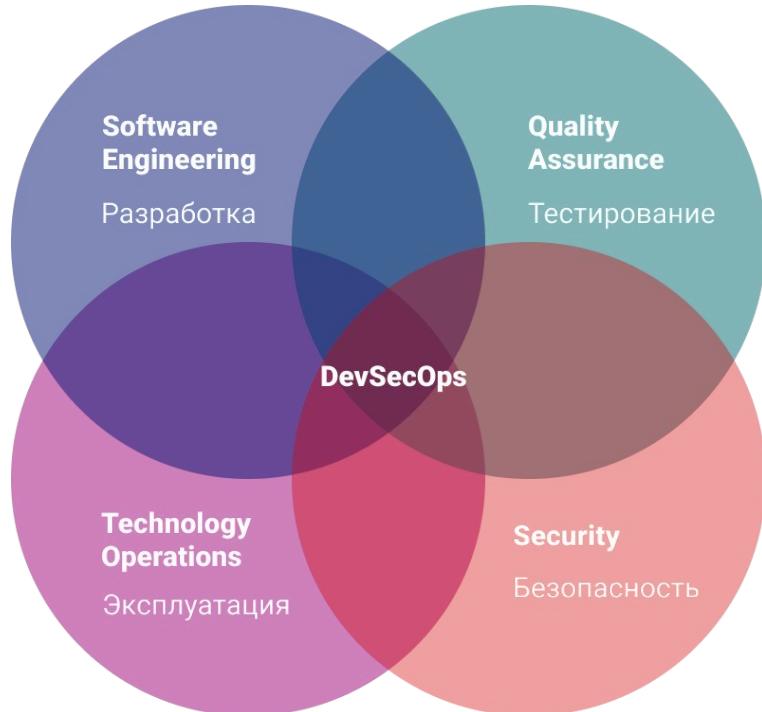
Добавляем Security: DevSecOps

Также иногда в это понятие добавляют и ответственность за безопасность:



DevSecOps

В итоге специалист DevOps имеет достаточно большую ответственность:



Шаги создания приложения

Разработка

Особенности шага:

- Все разработчики используют идентичную среду разработки
- Разработчики должны писать тесты (юнит, функциональные, приемочные)
- Необходимо иметь возможность проводить код-ревью и быстро получать результаты сложных тестов

Тестирование

Особенности шага:

- Тестирование должно быть автоматизировано
- В основную ветку не должен попасть нерабочий код
- Новый код должен быть покрыт тестами

Безопасность

Особенности шага:

- Наличие контроля доступа для разных сотрудников
- Исключение известных уязвимостей в используемых библиотеках
- Наличие контроля доступа отдельных сервисов

Создание песочницы, выкладка на стейдж

Особенности шага:

- Каждая фича должна быть протестирована на изолированном окружении
- Все окружения должны быть идентичны
- Процесс создания и уничтожения песочниц должен быть автоматизирован

Выкладка в продакшн

Особенности шага:

- Выкладка в продакшн должна быть только после успешного прохождения тестов
- Выкладка и отклад изменений должны происходить автоматически
- Должна быть возможность откатки изменений

Мониторинг

Особенности шага:

- Все ошибки во время тестирования на стейдже и при работе приложения в продакшене должны стать известны
- Должен быть простой доступ к логам всех составляющих системы
- Должно быть автоматическое оповещение об ошибках

Настройка рабочей среды

Выбор общих инструментов

Зачем использовать общие инструменты?

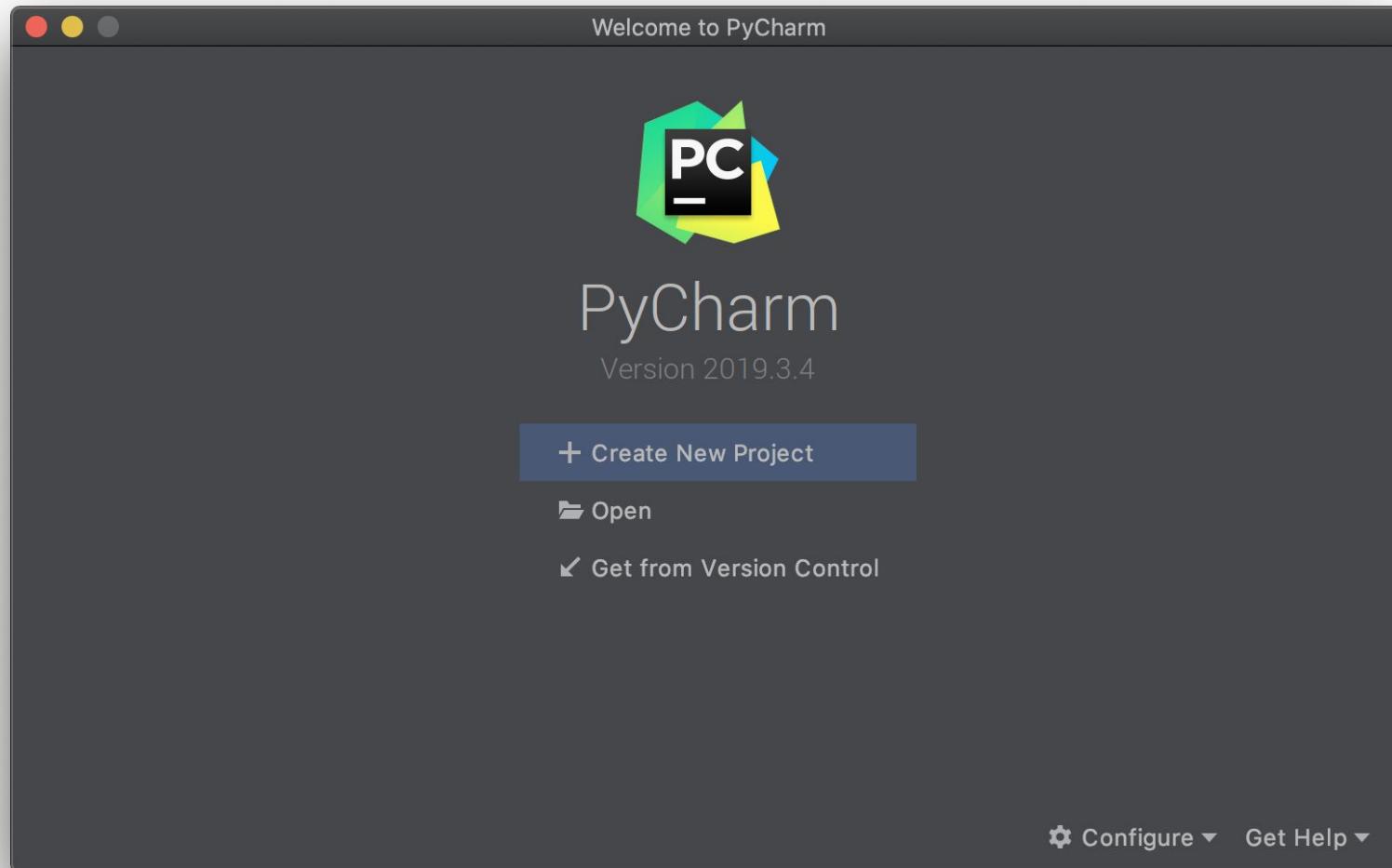
- Будет проще понимать друг друга
- Подсветка синтаксиса
- Автодополнение кода и команд

IDE PyCharm Community Edition

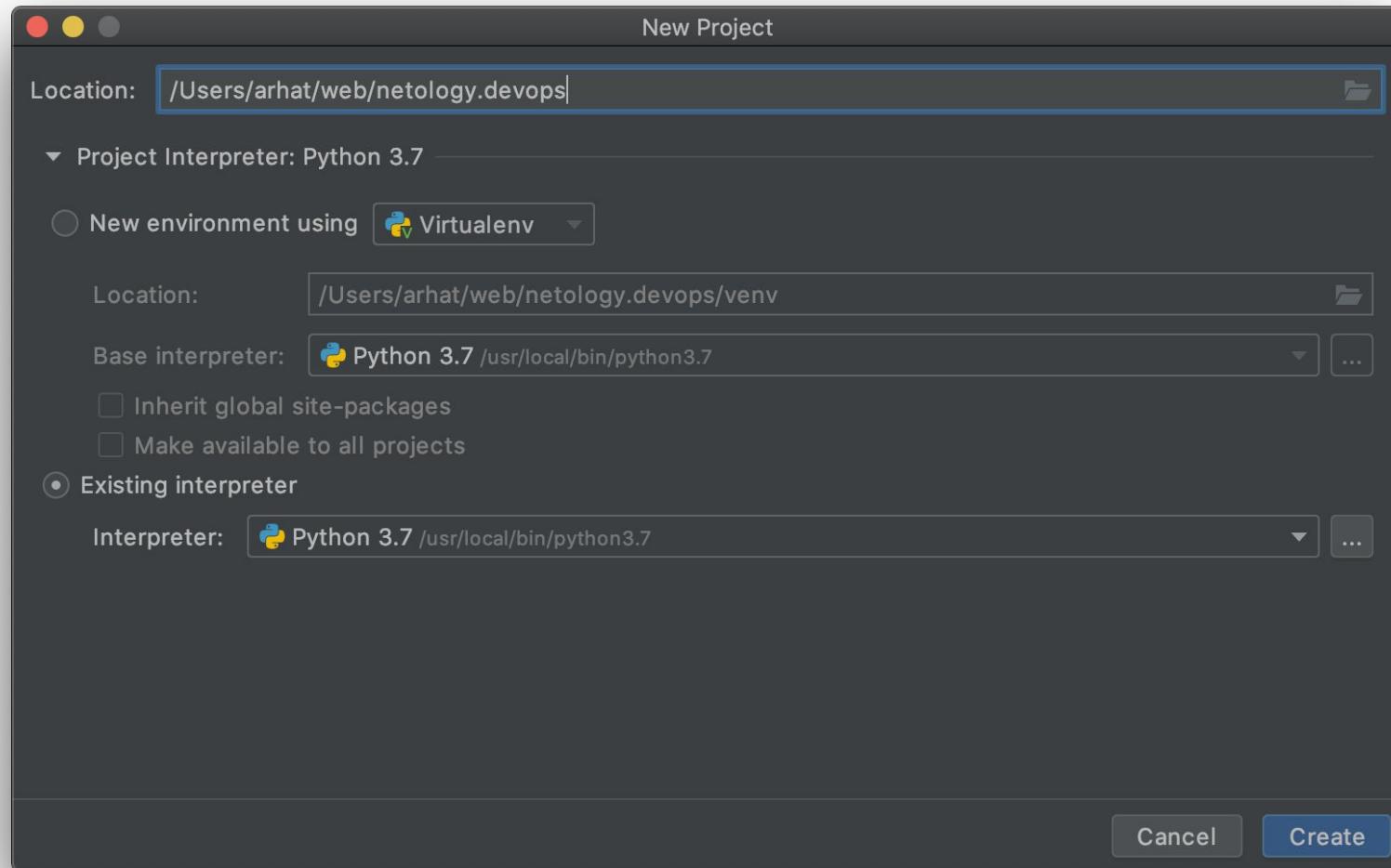
Почему IDE PyCharm?

- Бесплатная редакция
- Доступны все необходимые плагины
- На базе самого популярного решения от IDEA

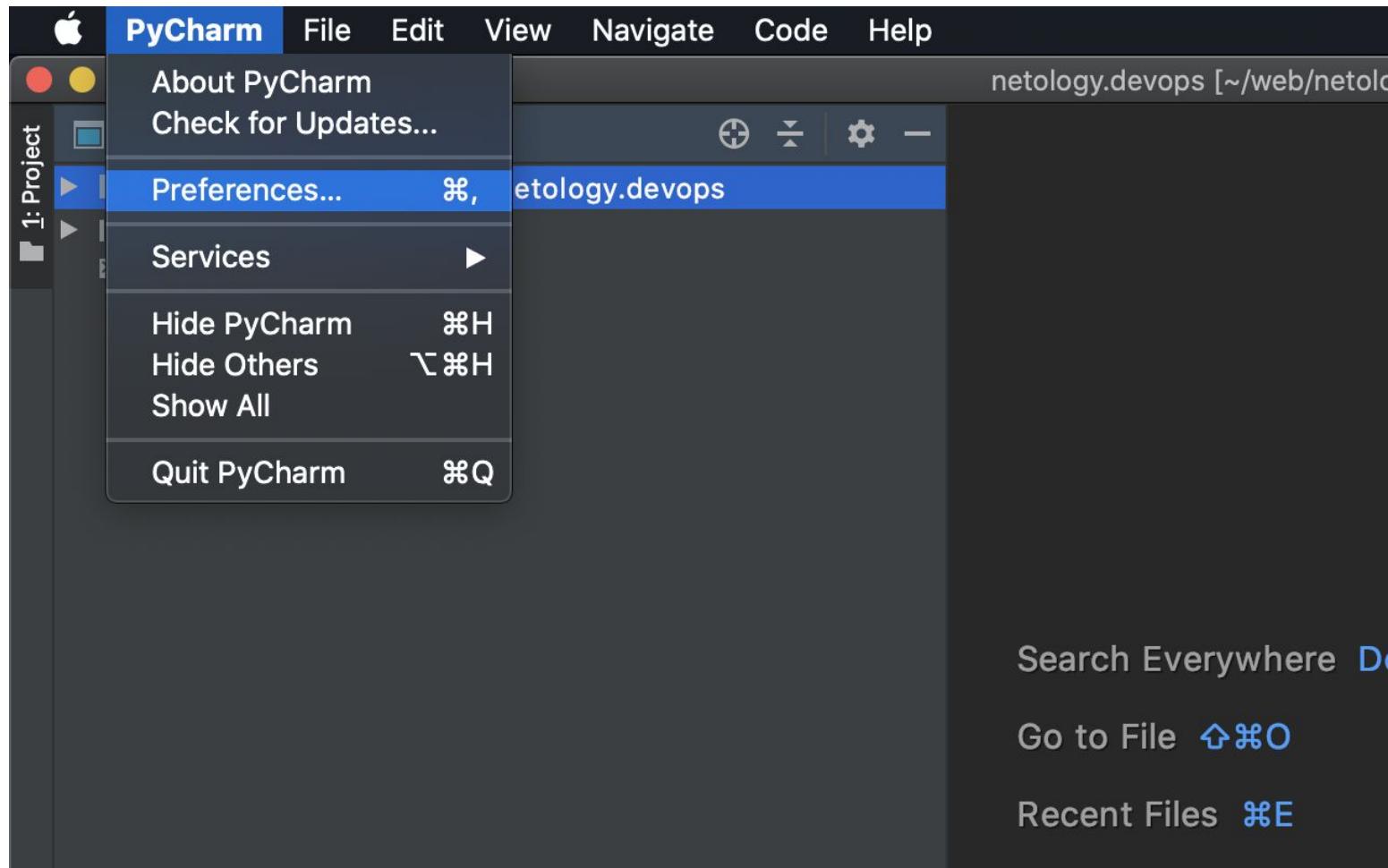
Установка PyCharm Community Edition



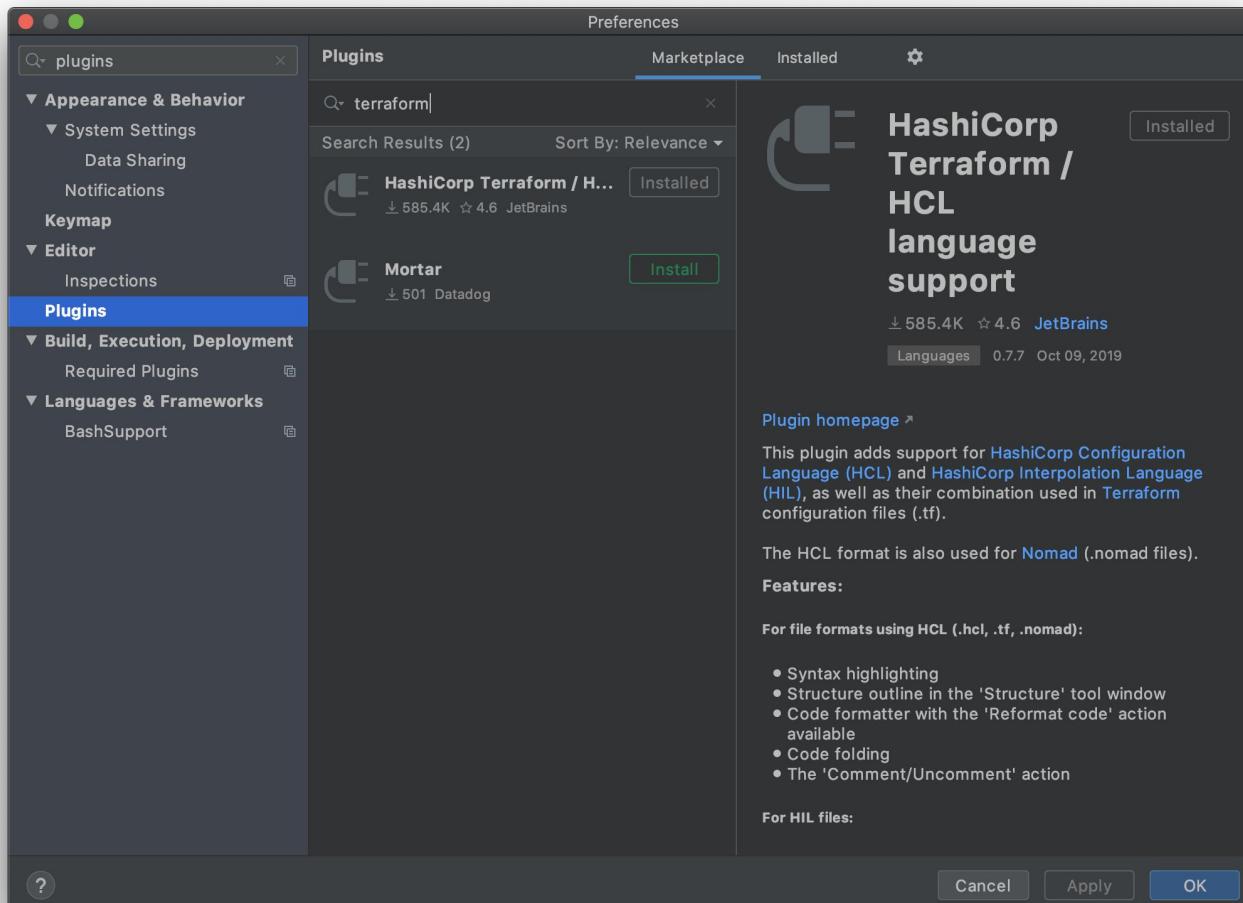
Создаем проект



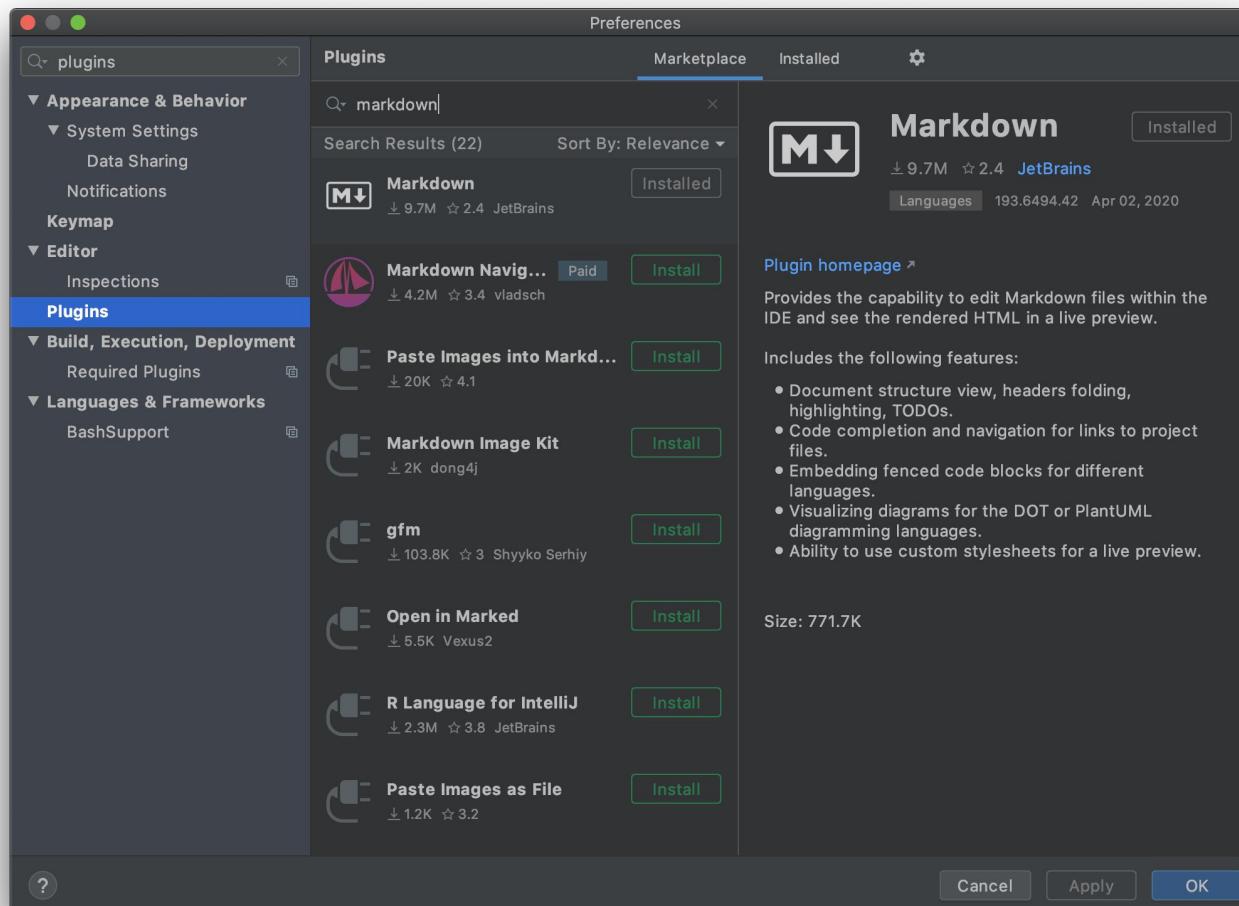
Переходим в настройки



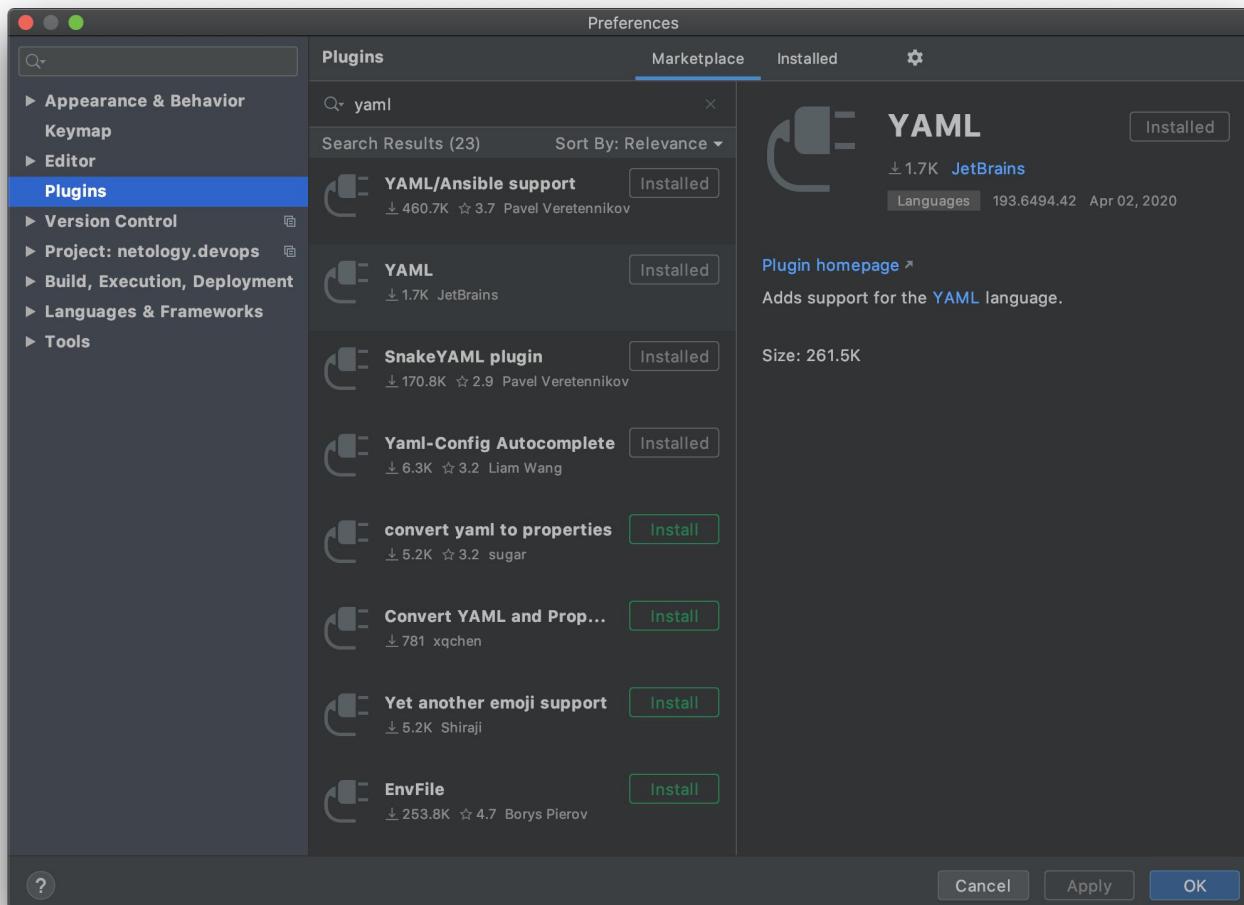
Устанавливаем плагин Terraform



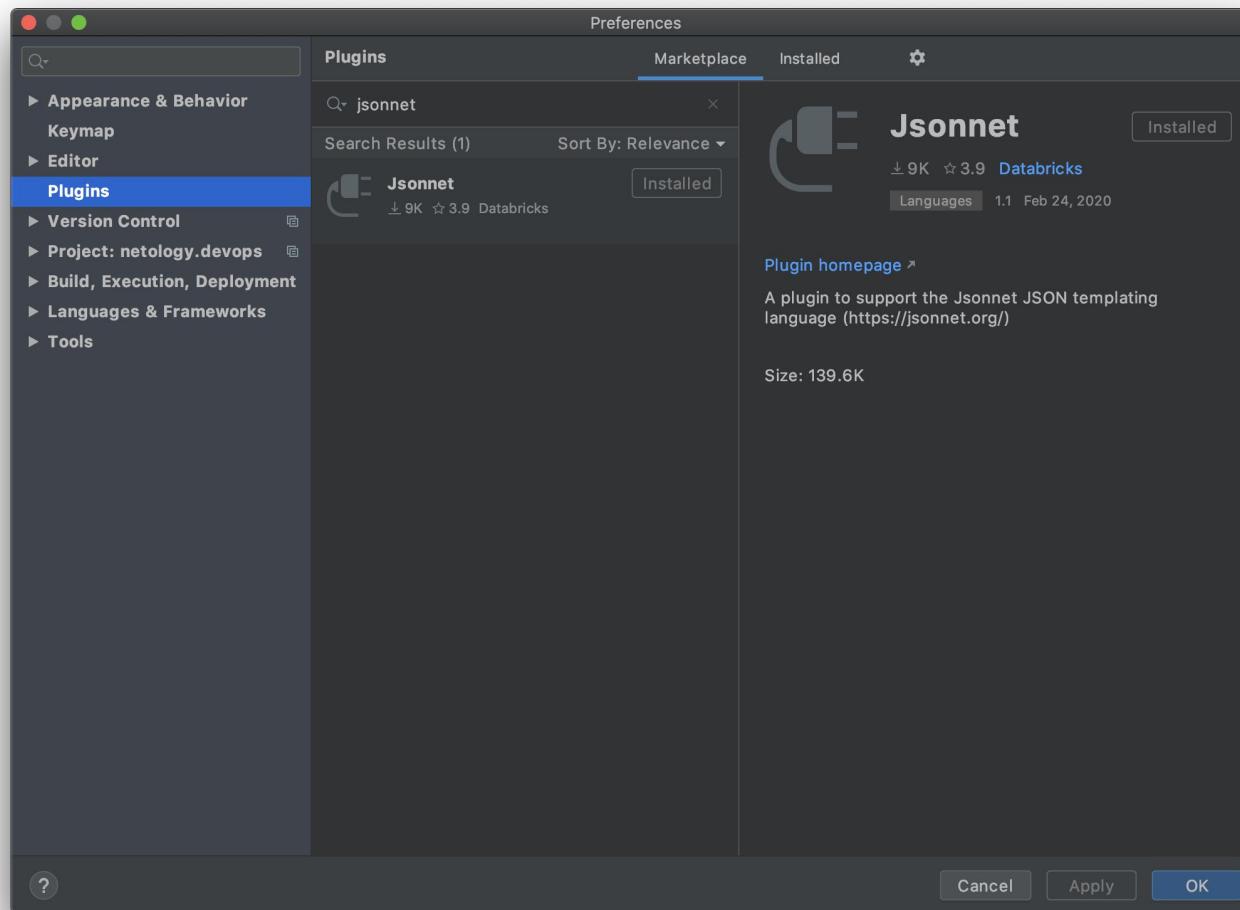
Устанавливаем плагин Markdown



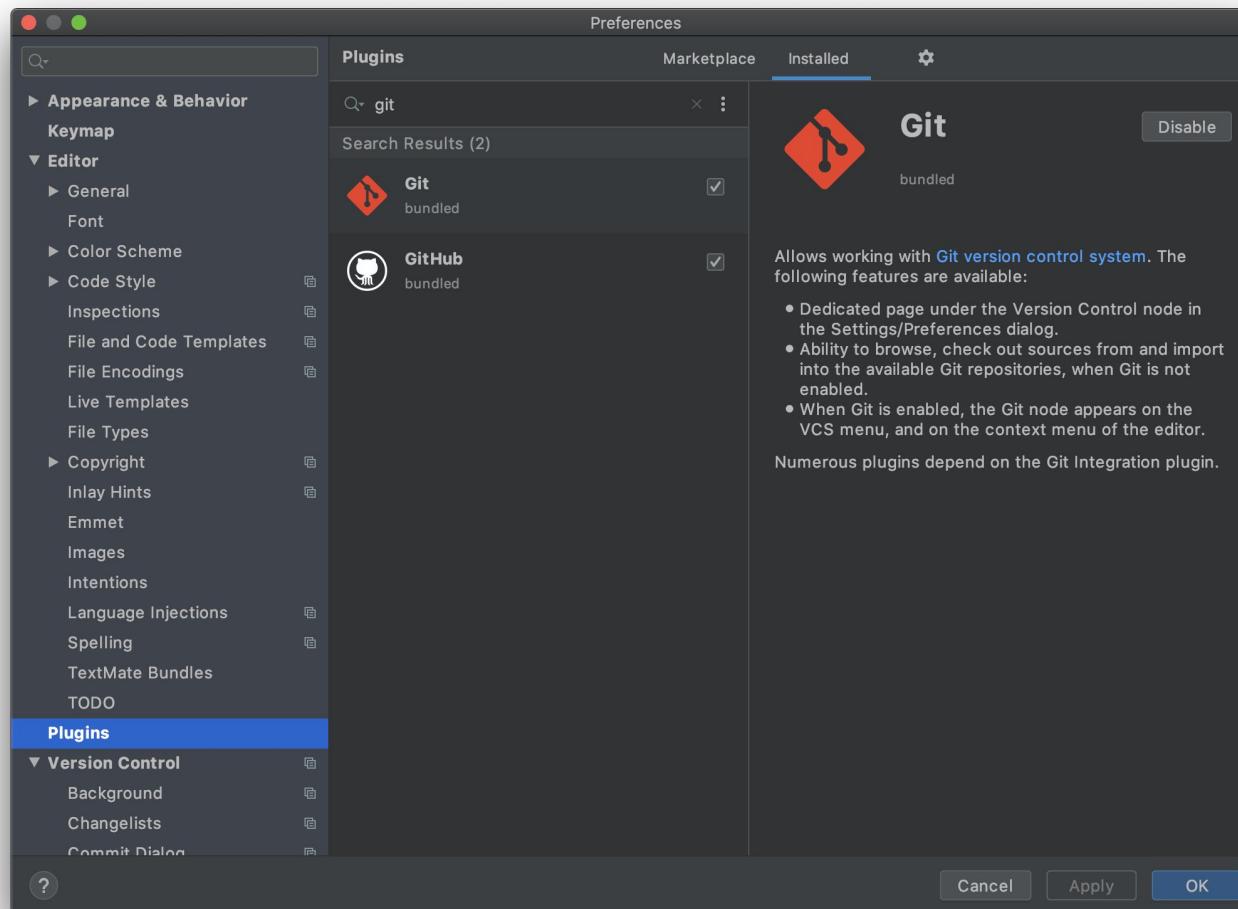
Устанавливаем плагин YAML/Ansible support



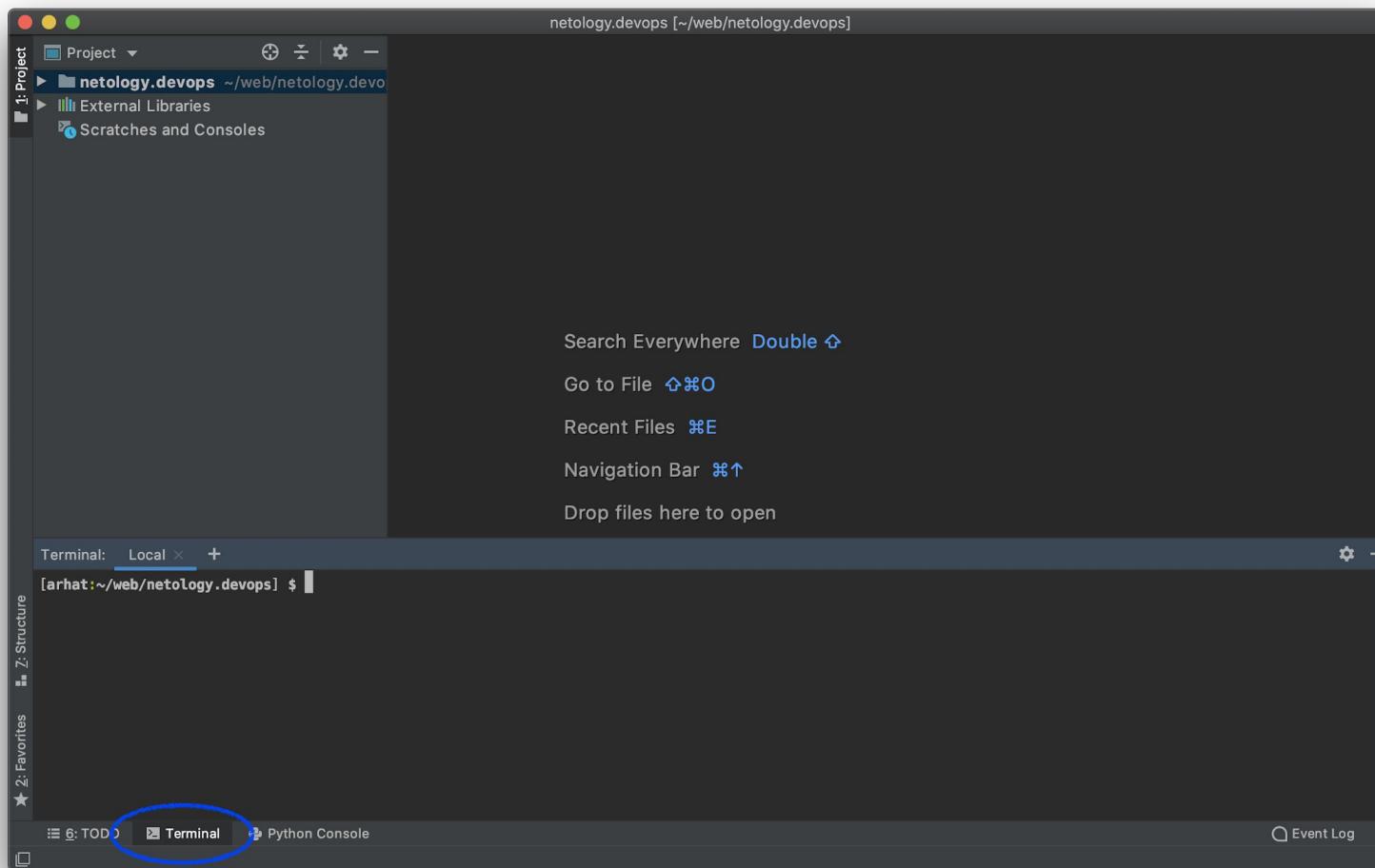
Устанавливаем плагин Jsonnet



Проверяем активацию Git



Открываем консоль



Консоль в разных ОС

Выбор терминала / консоли:

- **MacOS.** Iterm. <https://www.iterm2.com/>
- **Windows.** WSL:
<https://docs.microsoft.com/ru-ru/windows/wsl/install-win10>
- **Linux.** Стандартный или ваш любимый терминал

Этапы разработки ПО

Коммуникация

Задача этапа - отложенное взаимодействие:

- Между заказчиком и менеджером
- Между менеджерами
- Между менеджером и разработчиками
- Между разработчиками
- И др.

Разработка

Задачи этапа:

- Выбор инструментов
- Написание кода
- Внутреннее тестирование
- И др.

Внешние интеграции

Общие задачи этапа:

- Лицензирование
- Соглашение о модели взаимодействия
- Поддержка изменений
- И др.

Инфраструктура

Задача этапа – создать отказоустойчивое решение:

- Выбор оптимального решения
- Развертывание нескольких площадок
- Дальнейшая поддержка
- И др.

Тестирование

Задача этапа - прохождение тестирований:

- Юнит тестирование
- Функциональное тестирование
- Интеграционное тестирование
- И др.

Подходы к разработке.

Agile

Agile - гибкая методология, которая позволяет вашему проекту меняться легко и быстро. Обычно включает себя практики Scrum и Kanban. Работает по принципу: *требования > план > работа > ревью > повтор*.

Как работать?

- оцените этапы работы по часам (часто оценка является приблизительной)
- установите приоритетность задач (самые важные – в начале очереди)
- проанализируйте уже выполненную работу (если вы не вписываетесь в план, увеличьте рабочую нагрузку над этим спринтом)

Agile манифест

Главное в Agile манифесте:

- **Люди и взаимодействие** важнее процессов и инструментов
- **Работающий продукт** важнее исчерпывающей документации
- **Сотрудничество с заказчиком** важнее согласования условий контракта
- **Готовность к изменениям** важнее следования первоначальному плану

Закон Парето

20% усилий дают **80%** результата.

Это значит, что даже небольшое, но значимое усилие приводит к необходимому результату.

Правило универсальное, и может быть применено как ко времени разработки, так и к планированию скачков трафика и другим оценкам.

Основные фреймворки

Фреймворки, которые включает в себя
методология Agile:

- Scrum
- Kanban

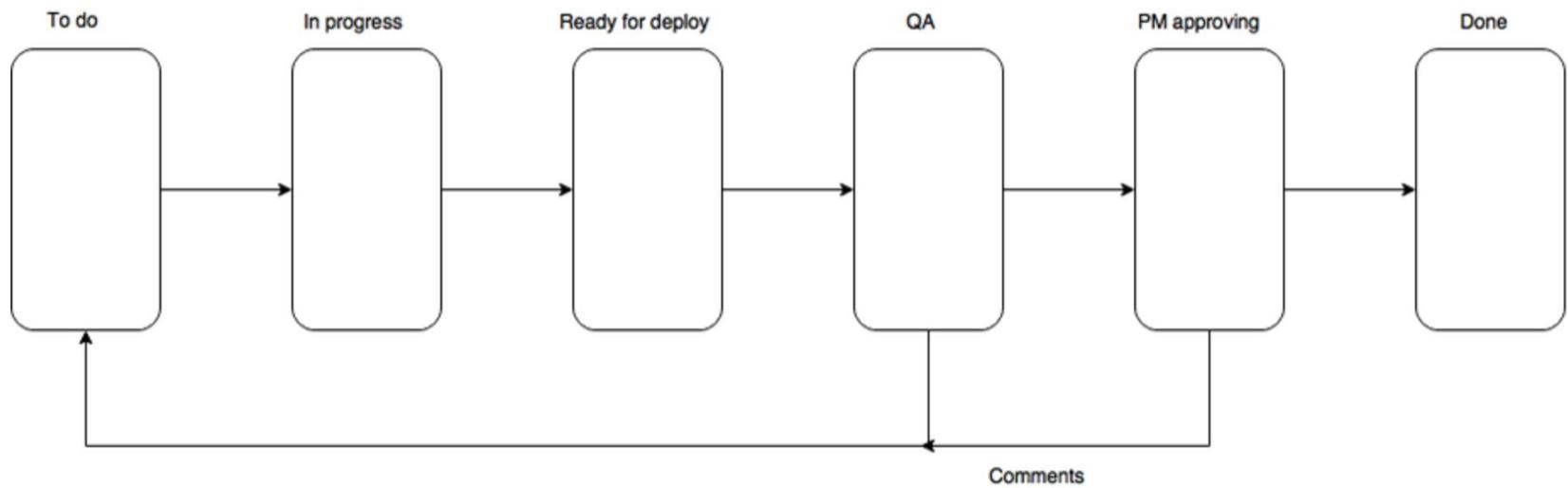
Пример организации процесса

- Деление работы на части - спринты (1-2 недели)
- Спринты планируются исходя из требований для этого конкретного момента времени
- Относительная оценка времени выполнения работ
- Ревью каждого спрингта, чтобы понять, как он прошёл и что можно было бы улучшить
- Фидбек по поставляемому продукту
- Ежедневные собрания (очень короткие)

Пример организации процесса

- Еженедельные собрания
- Непрерывная разработка
- Визуализация процесса на доске
- Решение сначала самых важных задач
- Поэтапные улучшения

Ограничение количества задач в каждом статусе



Итоги

Чему мы научились

- Обсудили принципу учебы на курсе.
- Разобрались что включает в себя DevOps.
- Договорились об общих инструментах.
- Разобрались в основных подходах к разработке.

Домашнее задание

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.



Задавайте вопросы и
пишите отзыв о лекции!

Андрей Борю



[andrey.borue](#)



[andrey.borue](#)



[andreyborue](#)