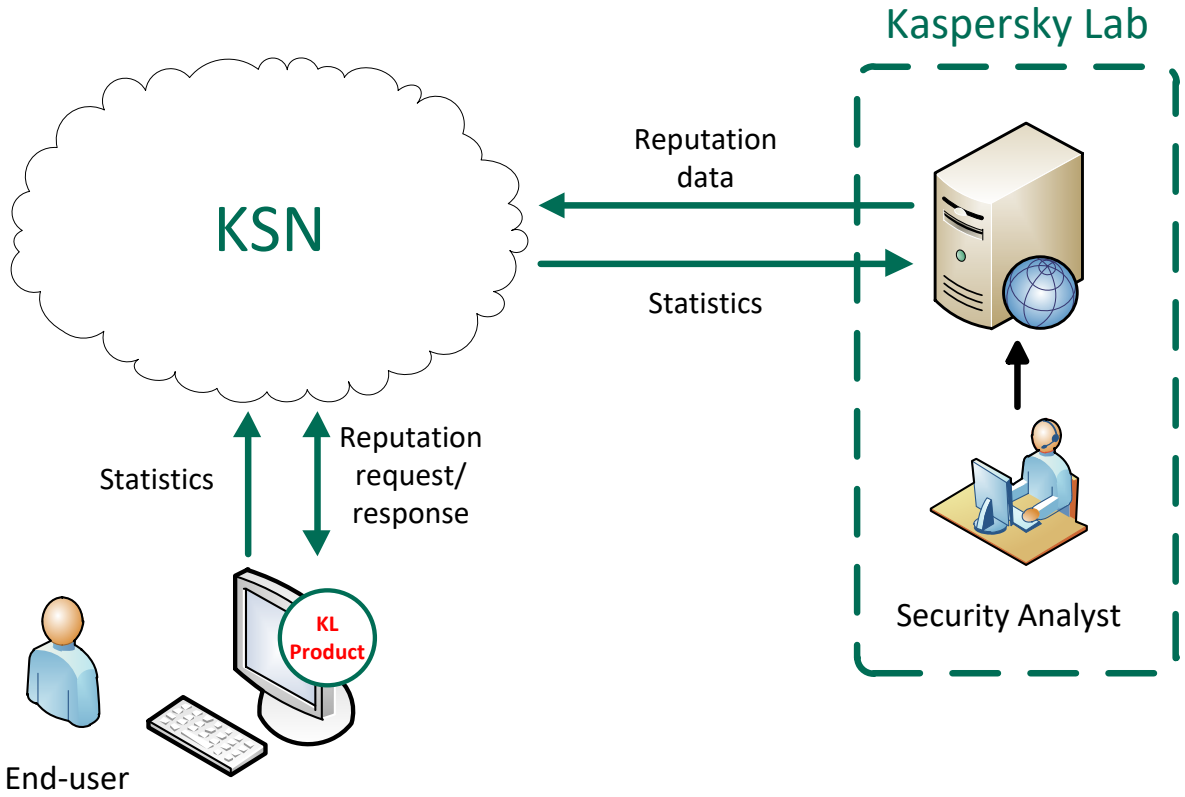
SafeBoard 2019-2020 / Python. Automation.

Валерия Попова
Разработчик тестовой инфраструктуры и автоматизированных тестов KSN

kaspersky

KASPERSKY SECURITY NETWORK



Концептуальная модель KSN

“Антивирусное” облако

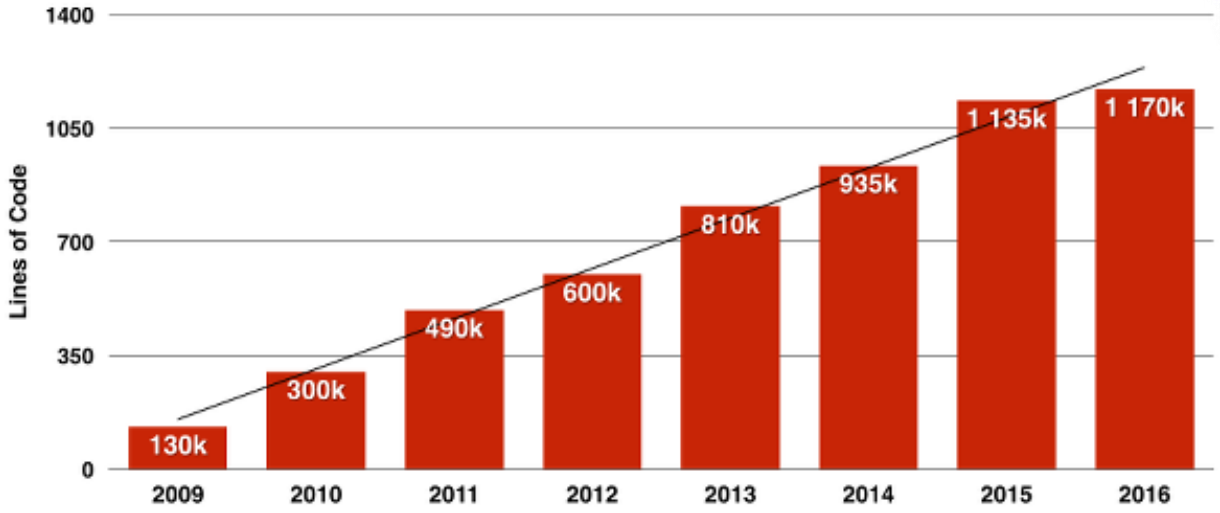
Для чего нужно тестирование?

- Основная задача тестирования – получение актуальной информации о состоянии проекта
- Без тестирования пользователи получат неизвестно как работающий (или не работающий функционал)
- Программа, работающая у одного человека, может сломаться при широком использовании
- Проверка совместимости с различными устройствами, браузерами и операционными системами
- ПО должно быть качественным и удовлетворять требованиям пользователя

Как выглядит проект без тестирования

А если изменений много?

- **Новый релиз** – надо проверить регресс и new features
- **~20** различных сервисов
- **7** разрабов одновременно коммитят в ~~master~~ parent ветку



Для чего нужно АВТОтестирование?

- Повторяемость
- Быстрое выполнение
- Минимальные затраты на поддержку
- Выполнение тестов без непосредственного участия человека
- Экономия времени
- Автоматические отчеты

Кто должен заниматься автоматизацией тестирования?

Аналитика

- Общение с коллегами
- Анализ требований
- Составление тестовой документации

Разработка

- Реализовать архитектуру автотестов
- Написать сами автотесты
- Автоматизация WEB

Тестирование

- Разобраться в сервисе
- Провести тест-дизайн
- Нагрузочное тестирование

DevOps

- Управление VM
- Организация CI/CD
- Мониторинг

Автотесты на сложный проект – это еще один сложный проект

- Архитектура
- Большое количество модулей
- Сами автотесты
- CI

```
valeriia@valeriia:~/projects/ksn_test_py3$ grep -r "def test_" ./ | wc -l
4981
```

```
valeriia@valeriia:~/projects/ksn_test_py3$ find . -name "*.py" | xargs wc -l | grep total
  76391 total
```

Python
Linux
Unittest
Pytest
Selenium
Django
Docker
CI, CD
Jenkins
C++, Go, Erlang

Unittest

- Модуль встроен в Python
- Нужно помещать тесты в классы, как методы
- Нужно использовать специальные методы утверждения. Класс TestCase вместо обычного встроенного выражения assert.

Pytest

- Поддерживает Unittest
- Возможность запускать тесты параллельно (*pytest-xdist*). Скорость выполнения в 2 и более раз выше
- Можно параметризовать тесты. Переиспользуем существующий код для множества тестов
- Возможность помечать тесты для их организации в отдельные группы
- Интеграция различных расширений. Например, *pytest --flake8*
- Автоматическая генерация тест-репортов, *pytest --html=report.html*
- Используем встроенное выражение assert

Pytest vs Unittest

Преимущество pytest — простота разработки тест-кейсов. Тест-кейсы pytest — серия функций в Python-файле с test_ в начале названия.

Есть в нем и другие полезные функции:

Поддержка встроенных выражений assert вместо использования специальных self.assert*() методов;

- Поддержка фильтрации тест-кейсов;
- Возможность повторного запуска с последнего проваленного теста;
- Экосистема из сотен плагинов, расширяющих функциональность.