

Статический анализ и ASOC: нулевая терпимость к ошибкам в проекте



CyberCodeReview
PROTECT YOUR CODE

Никита
Свиридов

PVS-Studio



Даниил
Коновалов

CyberCodeReview



Даниил Коновалов

Pre-sale AppSec-инженер

- 9+ лет опыта в information security
- Ведущий спикер Cybercodereview
- 50+ выступлений на закрытых и публичных мероприятиях



CyberCodeReview
PROTECT YOUR CODE

Никита Свиридов

C# разработчик, Tools&DevOps

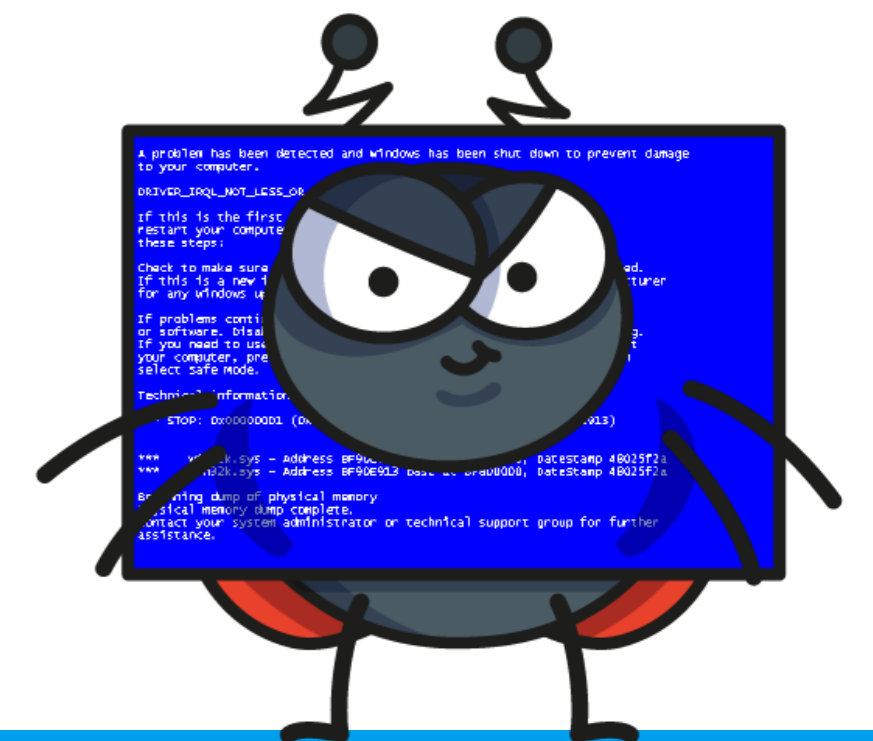
- Развиваю и поддерживаю инструменты PVS-Studio
- Тестирую совместимость нашего продукта с другими системами анализа кода
- Пишу статьи и делюсь опытом в области C#



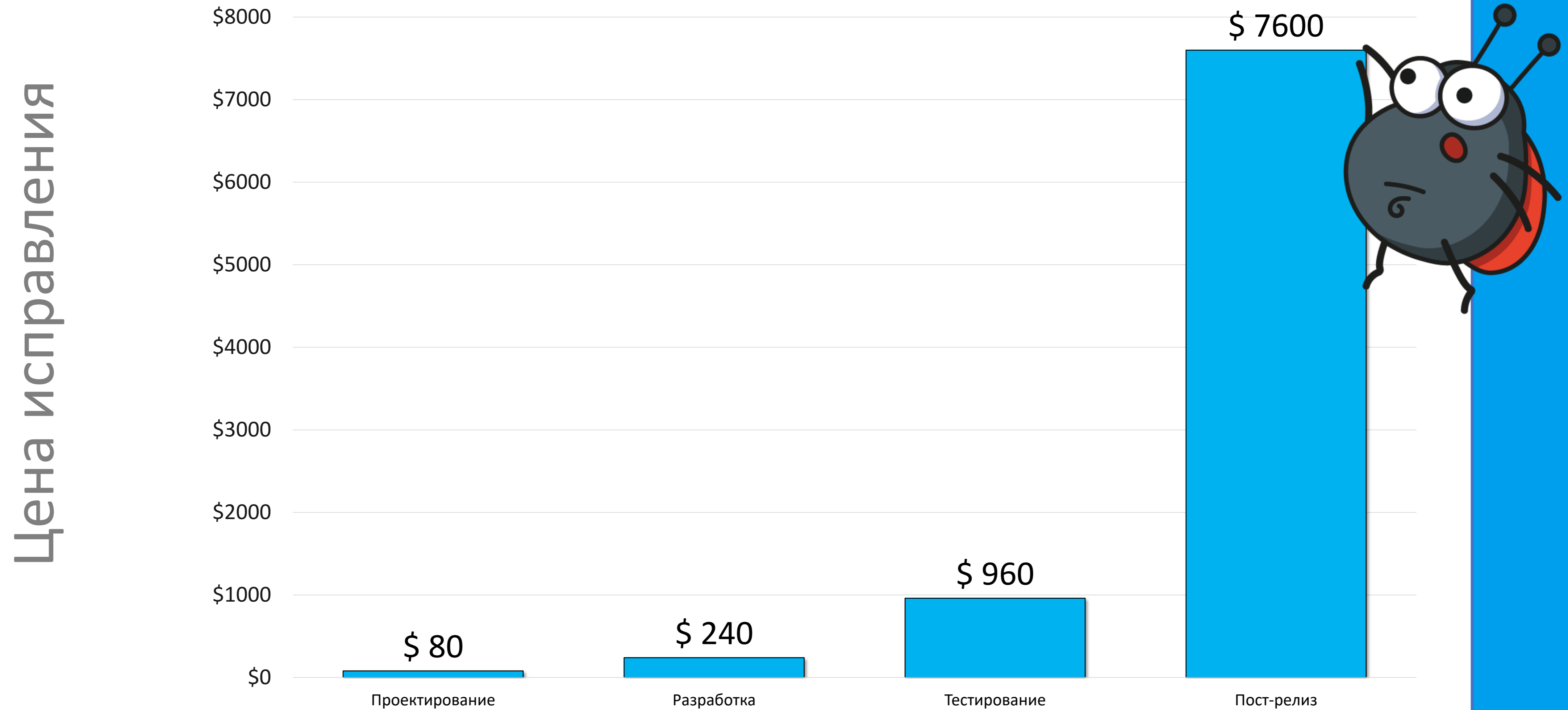
**Сколько стоит самая дорогая ошибка,
найденная на продкшене?**

Цена ошибки в современной разработке

- Классический цикл: Написали → Протестировали → Нашли баг → Исправили → (Повторить)
- **Стоимость исправления ошибки** растет стремительно на поздних стадиях
- Ручное ревью и тестирование не покрывают все сценарии



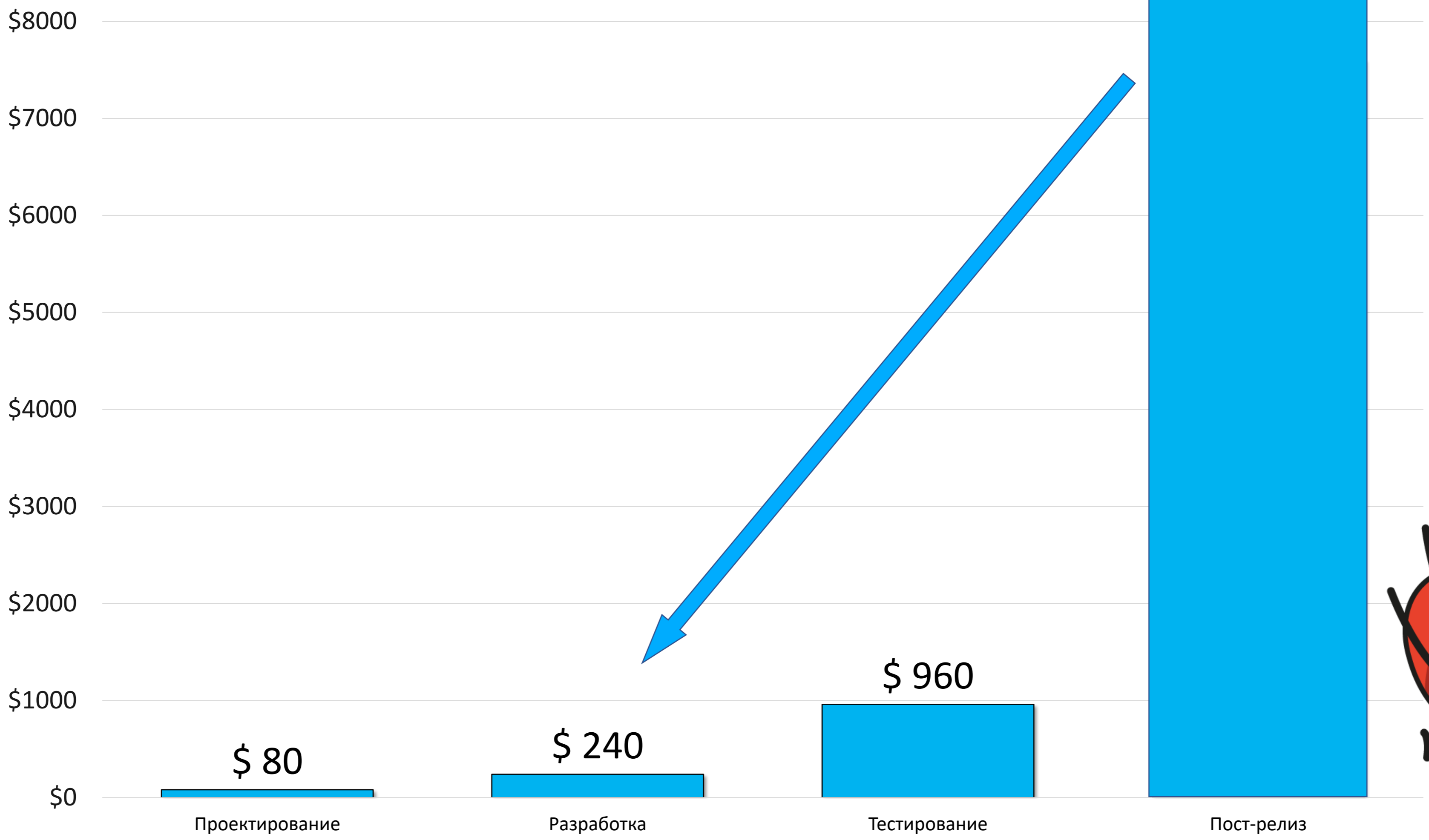
СКОЛЬКО СТОИТ ИСПРАВИТЬ ОШИБКУ?



Источник - [NIST](#): National Institute of Standards and Technology

СКОЛЬКО СТОИТ ИСПРАВИТЬ ОШИБКУ?

Цена исправления



**Что, если бы ошибки находились и
блокировались раньше?**

Принцип «Нулевой терпимости» к ошибкам в коде

Это не про «идеальный код», а про систему, которая не позволяет известным классам ошибок попадать в код проекта.

Три основных принципа нулевой терпимости к ошибкам в коде:

- Детекция
- Предотвращение
- Культура

Принцип «Нулевой терпимости» к ошибкам в коде

Детекция:

- Автоматическое обнаружение проблем на ранних этапах
- Использование статического анализатора как "радара"
- Постоянный мониторинг

Предотвращение:

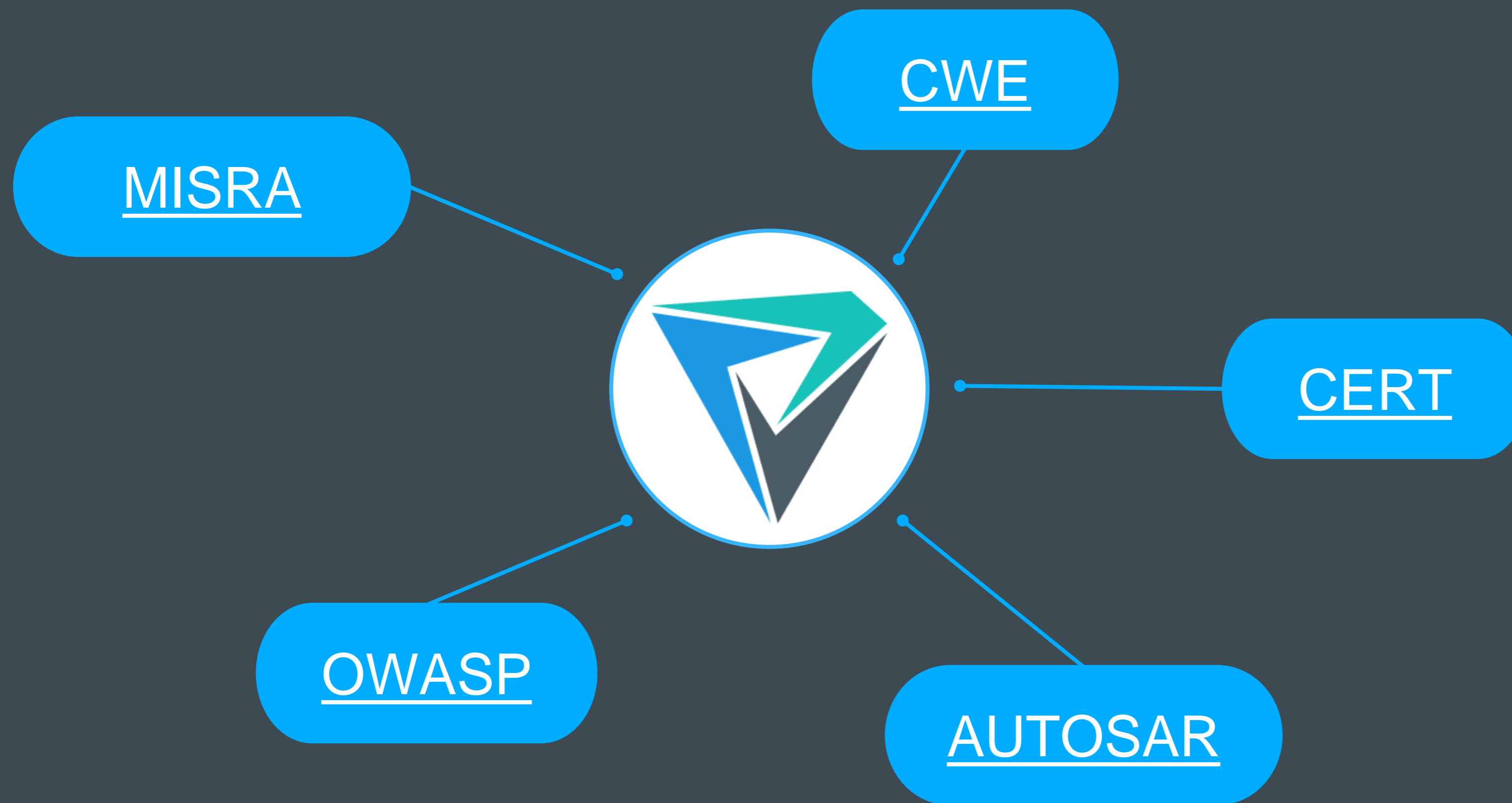
- Блокировка попадания проблемного кода в репозиторий
- Проверки в CI/CD пайплайне

Культура:

- Ответственность каждого разработчика за качество



Поддерживаемые стандарты



MISRA C
MISRA C++

MISRA: Готовый стандарт для нулевой терпимости

MISRA C и MISRA C++ — это стандарты разработки программного обеспечения, созданные организацией MISRA (Motor Industry Software Reliability Association). Цель стандартов: улучшение безопасности, переносимости и надёжности программ для встраиваемых систем.

- Это больше про ограничение сложности
- В основном за счёт ограничений на использование конструкций языка, библиотек, подходов программирования
- Последние версии:
 - MISRA-C – 2025
 - MISRA-C++ – 2023

Что может найти статический анализ?

Виды проблем

опечатки

ошибки сериализации /
десериализации

проблемы
безопасности

выход за
границы

неправильная работа
с методами

ошибки
синхронизации

недостижимый
код

переполнение
буфера

ошибки доступа к
памяти



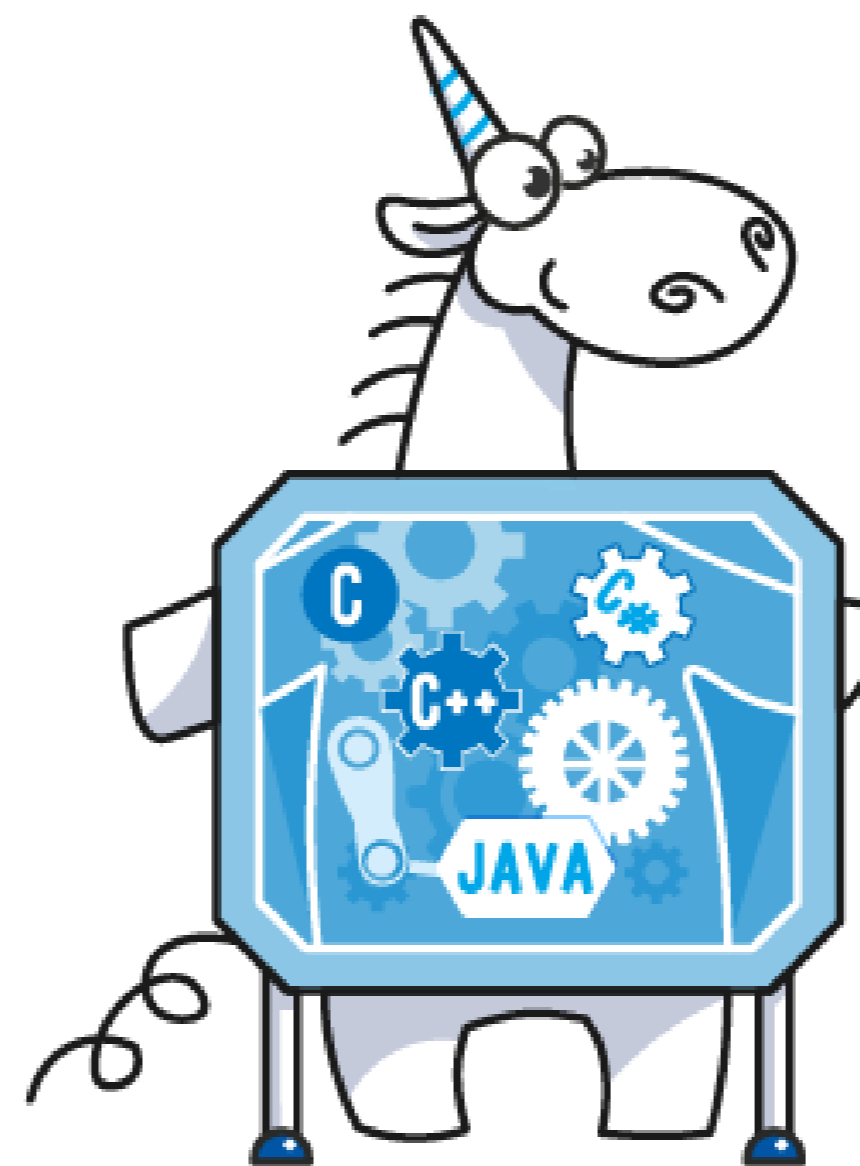
неправильная работа
с типами

и многое другое...

Как их искать?

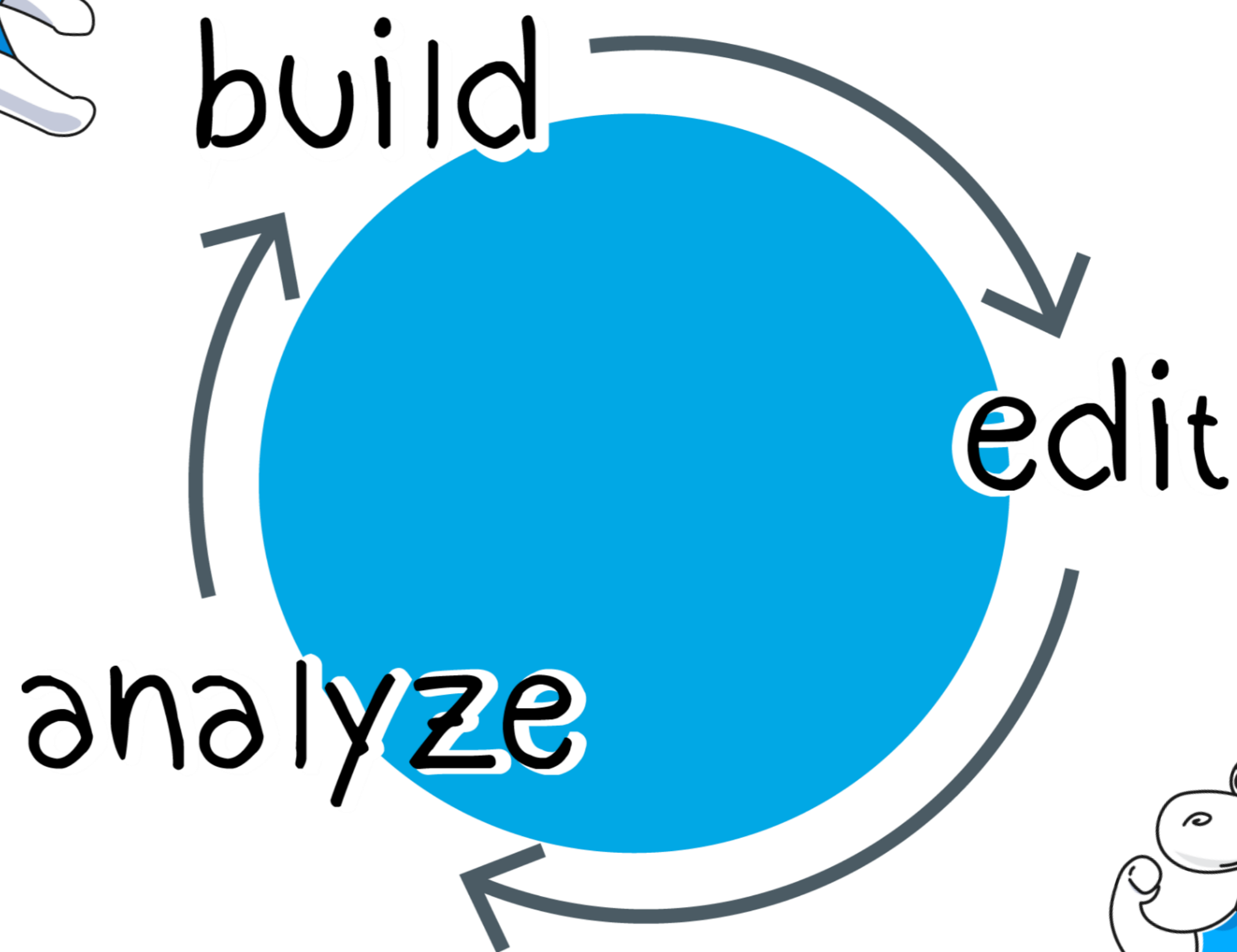
Локальный анализ

Базовый режим использования,
который запускается на машинах
разработчиков



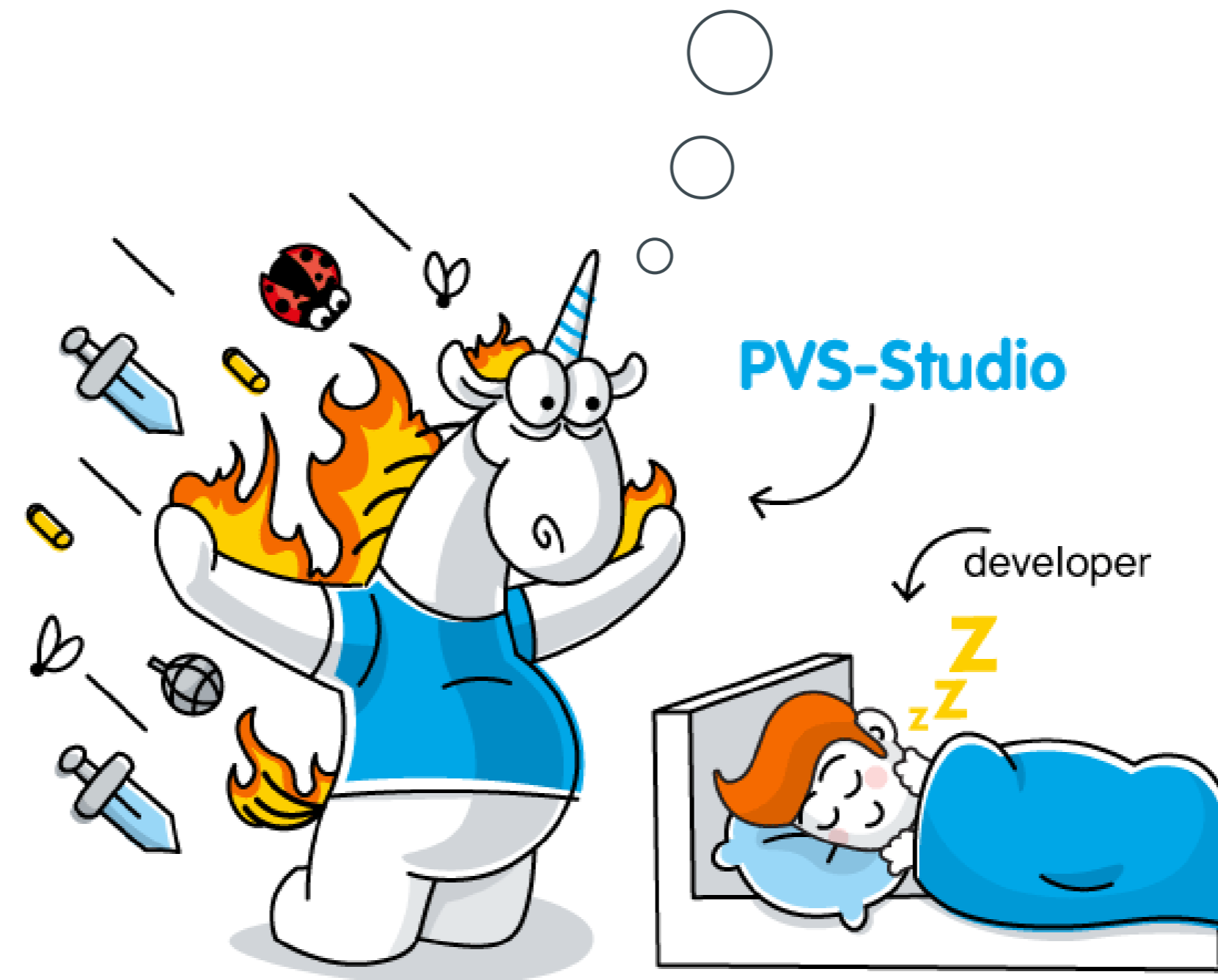
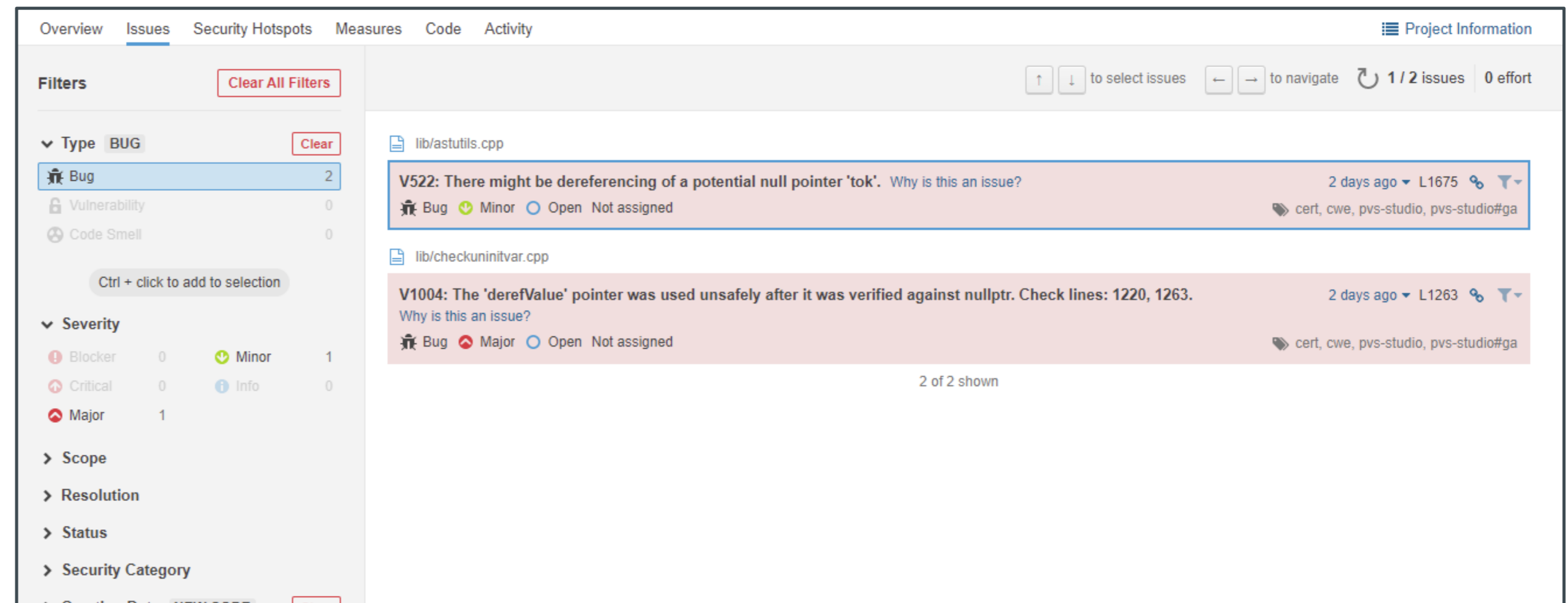
Инкрементальный анализ

Не нужно тратить время зря -
проверяйте только **изменённый**
код при помощи этого режима



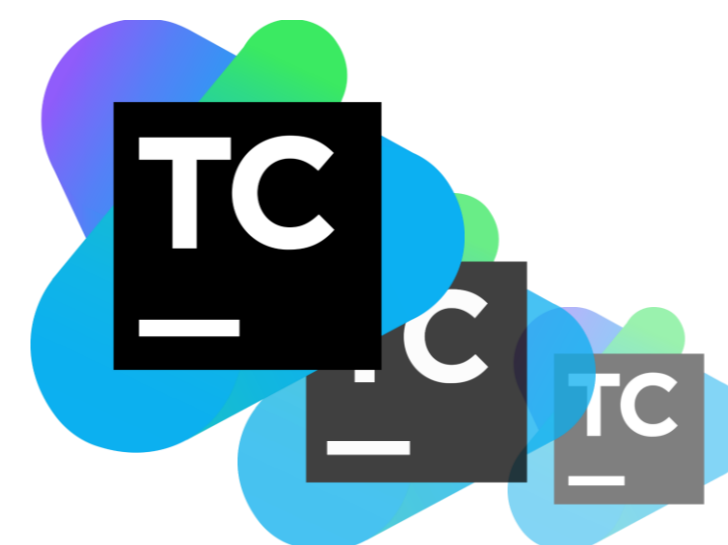
Серверный анализ

Интегрируйте PVS-Studio в ночные сборки и получайте каждое утро **подробный отчёт** о состоянии кодовой базы



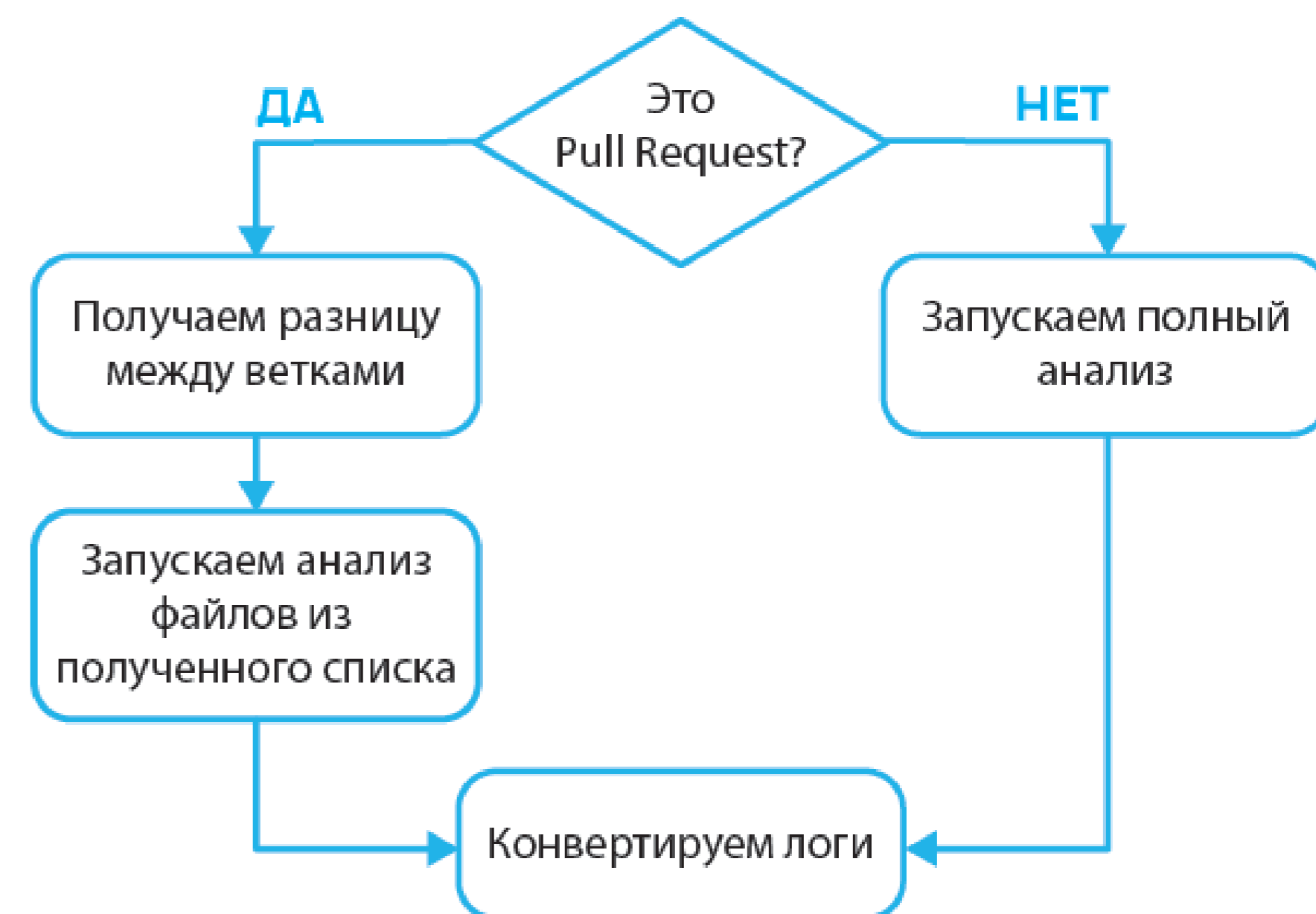
Облачный анализ

Помимо инфраструктуры компании вы всегда можете настроить анализ на **сервисах непрерывной интеграции**






Pull-request анализ

Проверяйте не только отдельные коммиты, но и целые **pull-request**, чтобы упростить процесс обзора кода



IDE и инструменты разработки



Облачные CI

-  CircleCI
-  Travis CI
-  GitLab
-  Azure DevOps

Сборочные системы

-  MSBuild
-  CMake
-  Make
-  Ninja
-  Maven
-  Gradle
-  JSON Compilation Database




CI

-  Jenkins
-  TeamCity






Виртуализация

-  Docker
-  WSL







Качество кода

-  SonarQube
-  DefectDojo
-  CodeChecker

Embedded

-  Keil μVision, DS-MDK
-  IAR Embedded Workbench
-  Platform.io
-  QNX Momentics
-  TI ARM Code Generation

IDE

-  Visual Studio
-  IntelliJ IDEA
-  Rider
-  CLion
-  Visual Studio Code
-  Qt Creator

Демонстрация работы с PVS-Studio

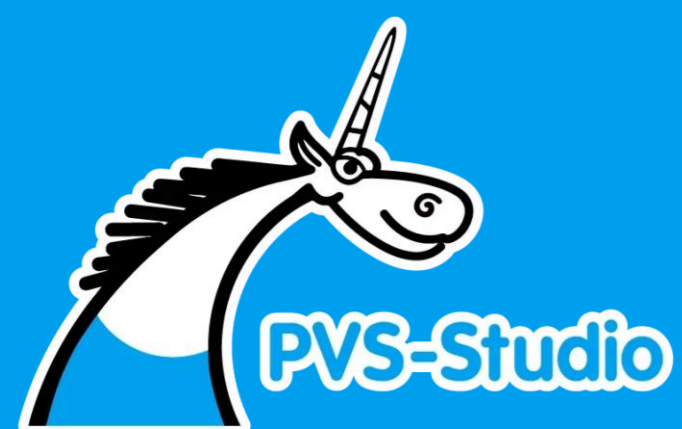
А что дальше? Анализ проведен, ошибки найдены...

- Использовать **несколько** статических анализаторов
- Работать с результатами анализа из **единого хранилища результатов**
- Интегрироваться с **системами трекинга ошибок**

Платформа, которая объединяет всё это **CyberCodeReview**

Q/A

Сделай свой проект чистым и безопасным вместе с PVS-Studio



Сайт
CyberCodeReview



LinkedIn CyberCodeReview



Попробуй пилотную версию продукта:

sales@cybercodereview.ru



CyberCodeReview
PROTECT YOUR CODE