

Непрерывный статический анализ кода

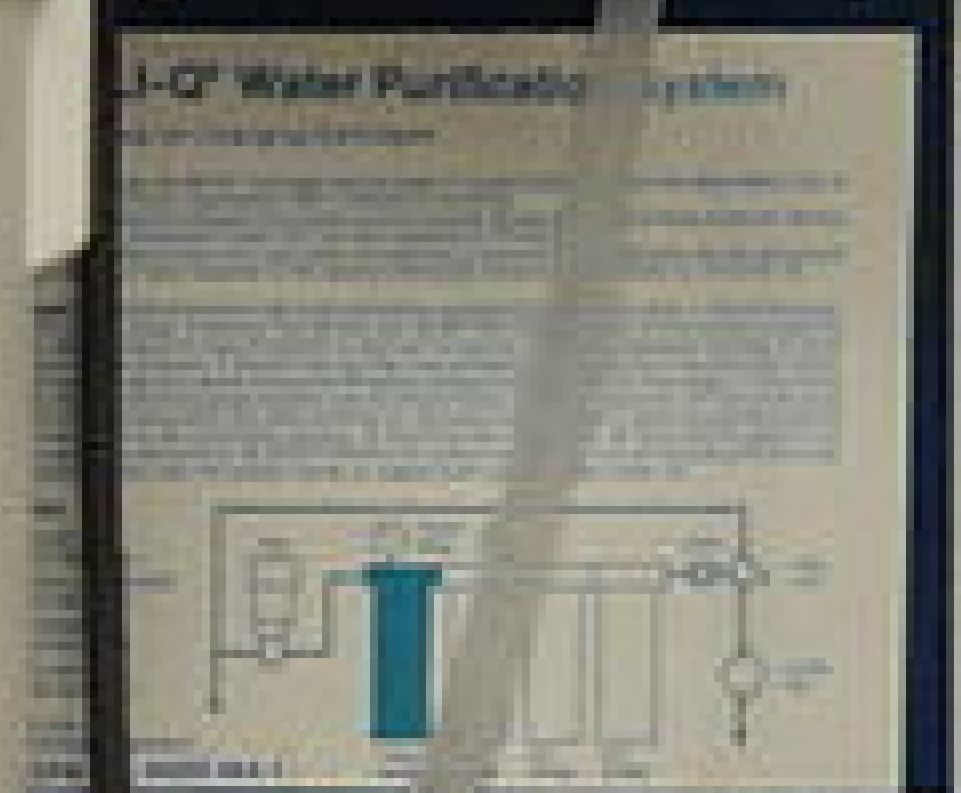
Иван Пономарёв, КУРС/МФТИ

ponomarev@corchestra.ru

 [@inponomarev](https://twitter.com/inponomarev)







1
Carbon

2
Mixed Bed

3
Mixed Bed

4
Organic

Carbon

U.S. FILTER

U.S. FILTER

U.S. FILTER



New!!!
9999\$ only

Что такое статанализ?

- Wikipedia: «Анализ программного обеспечения, производимый без реального выполнения исследуемых программ».
- Здравый смысл: Любая проверка исходного кода, требующая только исходный код (без тестов).

Чего в принципе не может СА?

Чего в принципе не может СА?

Зависнет или остановится?

```
def halts(f):  
    # false, если программа зависает  
    # true, если завершается за конечное время
```

Чего в принципе не может СА?

Зависнет или остановится?

```
def halts(f):  
    # false, если программа зависает  
    # true, если завершается за конечное время  
def g():  
    if halts(g):  
        while(True):  
            pass
```

Теорема Райса

Вычисляет ли функция квадрат числа?

```
def is_a_squaring_function(f):  
    # true, если функция вычисляет квадрат  
    # false, если не вычисляет
```


Теорема Райса

Вычисляет ли функция квадрат числа?

```
def is_a_squaring_function(f):  
    # true, если функция вычисляет квадрат  
    # false, если не вычисляет  
  
def halts(f):  
    def t(n):  
        f()  
        return n * n  
    return is_a_squaring_function(t)
```

Статанализ не найдёт то, что решается на уровне языка

- `object has no attribute` (если динамическая типизация)
- NPE (если нет Null Safety)



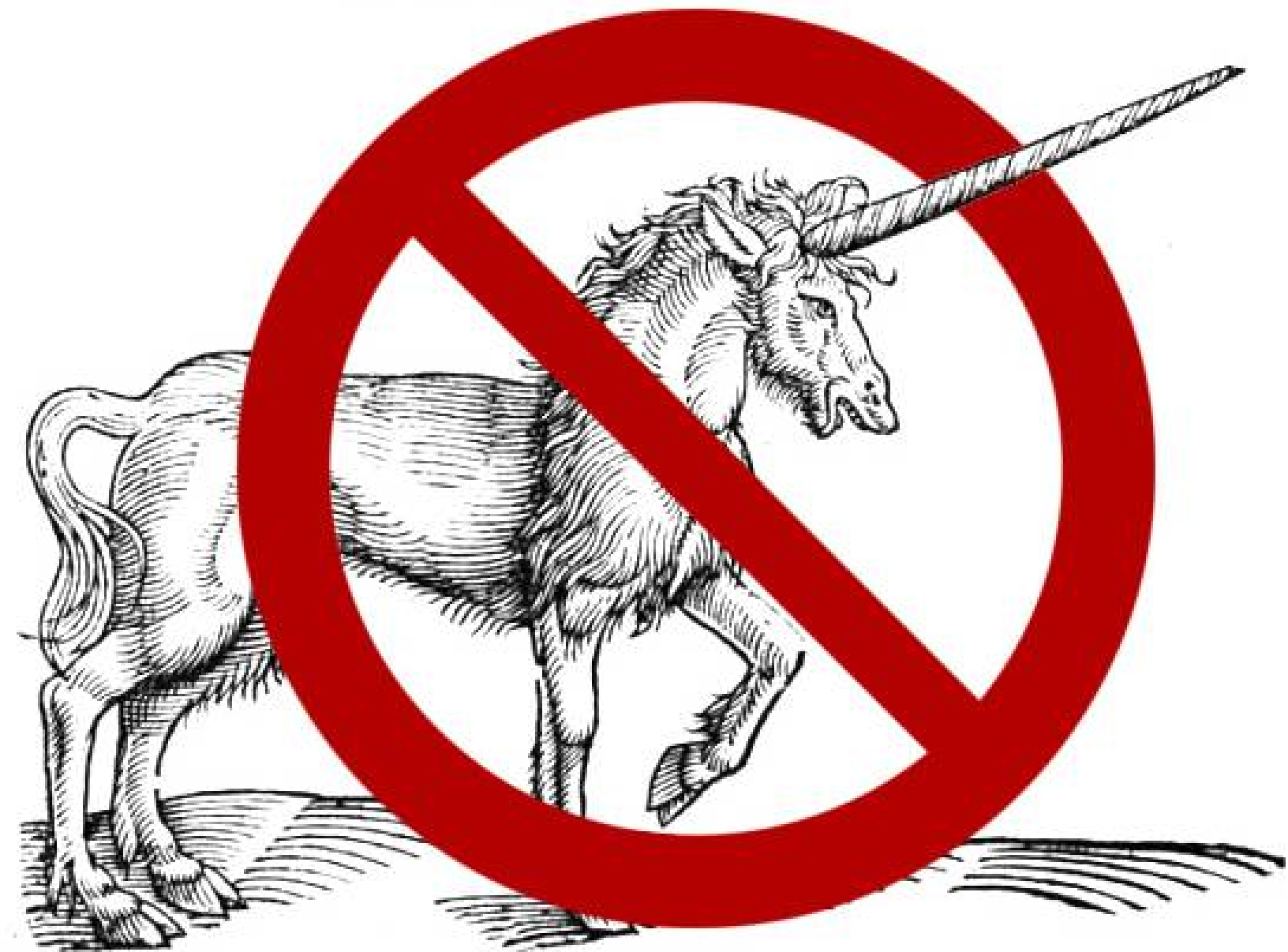
Близкое динамическое болото



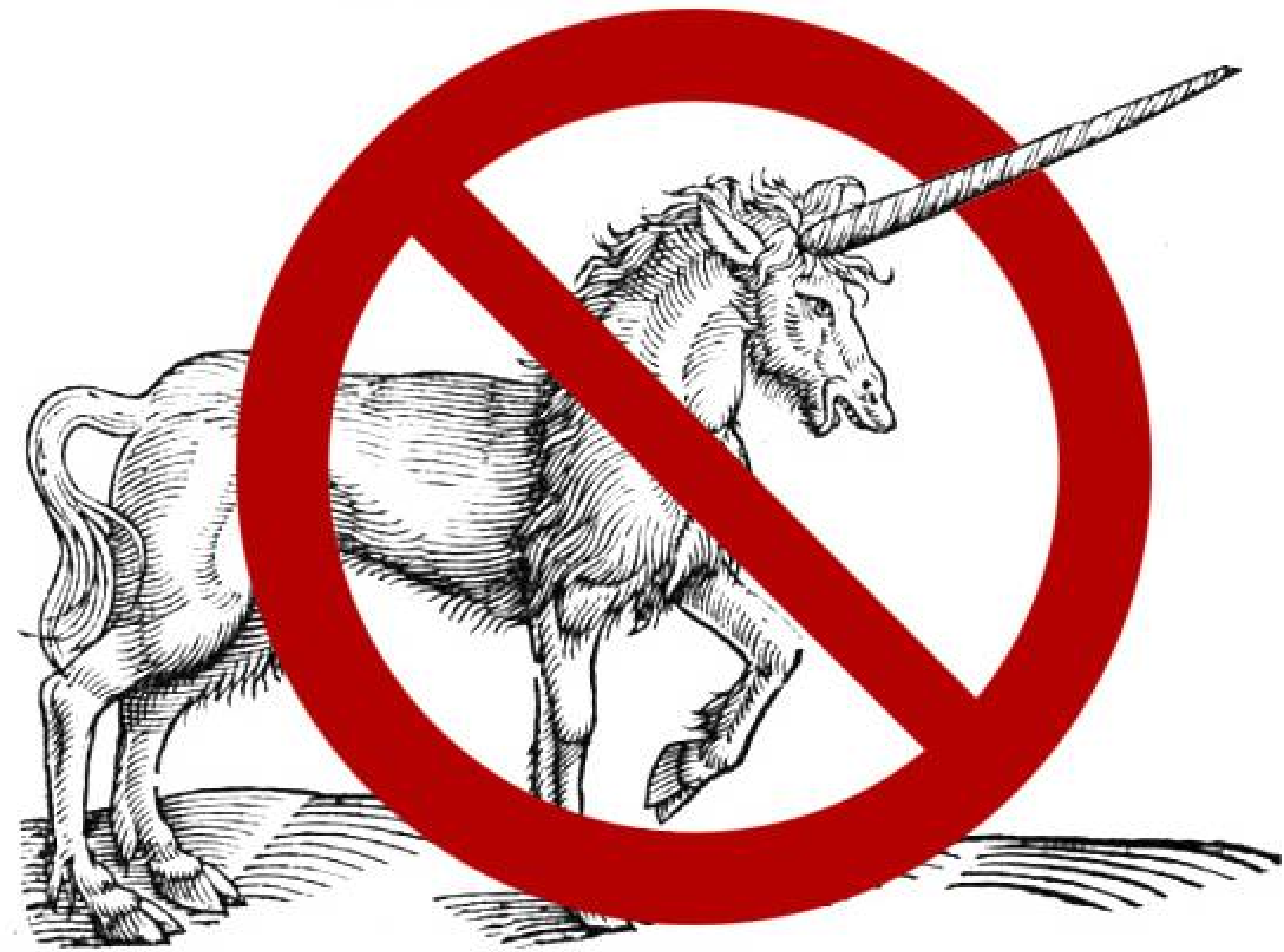
Труднодоступное статическое озеро

А что же статанализ?

А что же статанализ?



А что же статанализ?



Разнообразие средств статанализа



Разнообразие средств статанализа



- Проверка стиля кодирования (checkstyle, flake8)

Разнообразие средств статанализа



- Проверка стиля кодирования (checkstyle, flake8)
- Поиск характерных ошибок в коде (spotbugs, IDEA, PVS-Studio)

Разнообразие средств статанализа



- Проверка стиля кодирования (checkstyle, flake8)
- Поиск характерных ошибок в коде (spotbugs, IDEA, PVS-Studio)
- Проверка валидности ресурсных файлов (xmllint, YAMLLint, JSONLint)

Разнообразие средств статанализа



Разнообразие средств статанализа



- Компиляция/парсинг (`ansible --syntax-check`, `terraform validate`)

Разнообразие средств статанализа



- Компиляция/парсинг (`ansible --syntax-check`, `terraform validate`)
- Предупреждения компиляторов

Разнообразие средств статанализа



- Компиляция/парсинг (`ansible --syntax-check`, `terraform validate`)
- Предупреждения компиляторов
- Проверка правописания

Разнообразие средств статанализа



- Компиляция/парсинг (`ansible --syntax-check`, `terraform validate`)
- Предупреждения компиляторов
- Проверка правописания
- Конфигурационные тесты

Больше анализаторов!

- Google <Your Language> static analyzer
- Google <Your Language> linter

ГЕЙЗЕР
СТАНДАРТНАЯ ОЧИСТКА
КЛАССИК

ОЧИСТКА ОТ:
Механических частиц
Активного хлора
Связанного хлора
Хлорорганических соединений
Тяжелых металлов
Неприятных запахов и привкусов

ВЫСТРОСЕМ
Быстрая и легкая смена картриджа

РЕСУРС 8 000 литров*

2 590₽

rgio НОВАЯ ВОДА®
многоступенчатый фильтр для очистки воды

Expert
комплексная очистка воды

В ПРЕМИАЛЬНОМ ИСПОЛНЕНИИ

German
INNOVATION

www.filter.ru

4 510₽

АКВАФОР
фильтры для воды

ТРИО УНИВЕРСАЛ УМЯГЧАЮЩИЙ

СНИЖЕНИЕ ЖЕСТКОСТИ / ДОПОЛНИТЕЛЬНАЯ ОЧИСТКА ОТ ЖЕЛЕЗА

Очищает воду от активного хлора, раскисляет, убирает жесткость, тяжелые металлы, пестициды.

Классическая замена картридей

Рекомендовано для детского питания**

Ресурс 4000 литров

2 227₽

Вы заботитесь о своем здоровье?
Сделайте анализ воды!

370₽

ГЕЙЗЕР

СНИЖЕНИЕ ЖЕСТКОСТИ

ГЕЙЗЕР-БИО 331

ОЧИСТКА ОТ:
Бактерий и вирусов
Соединений жесткости
Хлора
Железа и ржавчины
Тяжелых металлов и радионуклидов
Органических и хлорорганических соединений
Соединения азота и фосфора

МОЖНО ПИТЬ БЕЗ КИПЯЧЕНИЯ

5 000 000

7 900

3 041₽

БАРЬЕР
внутренние фильтры для воды

СНИЖЕНИЕ ЖЕСТКОСТИ

ОЧИСТКА ОТ:
Хлора
Органических и хлорорганических соединений
Солей жесткости
Тяжелых и токсичных металлов
Неприятных запахов и привкусов

КЛАССИЧЕСКАЯ СМЕНА КАРТРИДЖЕЙ

РЕСУРС 10 000 литров*

3 041₽

АКВАФОР
фильтры для воды

ЭКО
УМЯГЧАЮЩИЙ

СНИЖЕНИЕ ЖЕСТКОСТИ

КРИСТАЛЛ ЭКО УНИВЕРСАЛ УМЯГЧАЮЩИЙ

Полностью удаляет активный хлор, тяжелые металлы, неабсорбируемые вредные примеси, улучшает вкус и запах воды благодаря уникальному ионнообменному полимеру AQUALEN™

Сверхлегкая замена смесных модулей одним нажатием

Рекомендовано для детского питания**

Ресурс 6000 литров

Питьевая вода без кипячения***

5 074₽

ГЕЙЗЕР
5 000 000

16800

68800

31000

50500

100000

АКВАФОР
МАКСИФОР

20600

13000

14200

18700

53400

ГЕЙЗЕР
БЕСПЛАТНАЯ УСТАНОВКА

СМАРТ

МОМЕНТАЛЬНАЯ ЗАМЕНА КАРТРИДЖА

QUICKLOCK

2 590₽

БАРЬЕР
внутренние фильтры для воды

СТАНДАРТНАЯ ОЧИСТКА

КЛАССИК

ОЧИСТКА ОТ:
Механических частиц
Активного хлора
Связанного хлора
Хлорорганических соединений
Тяжелых металлов
Неприятных запахов и привкусов

ВЫСТРОСЕМ
Быстрая и легкая смена картриджа

РЕСУРС 8 000 литров*

2 590₽

3 04600

АКВАФОР
фильтры для воды

КРИСТАЛЛ Н УНИВЕРСАЛ

СНИЖЕНИЕ ЖЕСТКОСТИ / ДОПОЛНИТЕЛЬНАЯ ОЧИСТКА ОТ ЖЕЛЕЗА

Полностью удаляет активный хлор, тяжелые металлы, неабсорбируемые вредные примеси, улучшает вкус и запах воды благодаря уникальному ионнообменному полимеру AQUALEN™

Сверхлегкая замена смесных модулей одним нажатием

Классическая, экономичная смена картридей

Выдерживает максимальное давление (рабочее) в течение 100 000 циклов

Ресурс 4000 литров

1 85100

СИНГЛ
преимущества БАРЬЕР ЭКСПЕРТ Сингл

БАРЬЕР
внутренние фильтры для воды

АКВАЛИНИЯ

Компактный фильтр для очистки воды

86800

81400

АКВАФОР
фильтры для воды

55800

23200

53400

ГЕЙЗЕР
СНИЖЕНИЕ ЖЕСТКОСТИ

5 000 000

26200

БАРЬЕР
внутренние фильтры для воды

ЭКСПЕРТ

СНИЖЕНИЕ ЖЕСТКОСТИ И ЖЕЛЕЗА

КОМПЛЕКС

ОЧИСТКА ОТ:
Механических частиц
Активного хлора
Хлорорганических соединений
Растворенного железа
Солей жесткости
Неприятных запахов

19800

БАРЬЕР
внутренние фильтры для воды

ЭКСПЕРТ

СНИЖЕНИЕ ЖЕСТКОСТИ И ЖЕЛЕЗА

КОМПЛЕКС

ОЧИСТКА ОТ:
Механических частиц
Активного хлора
Хлорорганических соединений
Растворенного железа
Солей жесткости
Неприятных запахов

19700

HotFrost
Помпа механическая АБ

50600

ГЕЙЗЕР
фильтры для воды

301

В ПОДАРОК 2-ой КАРТРИДЖ

4

50600

Кто программирует на Bash?

Кто программирует на Bash?

The screenshot shows the GitHub repository page for `koalaman/shellcheck`. At the top, the repository name is displayed with a copy icon. To the right, there are buttons for 'Watch' (367), 'Star' (14,503), and 'Fork' (734). Below this, a navigation bar contains links for 'Code', 'Issues' (400), 'Pull requests' (7), 'Projects' (0), 'Wiki', and 'Insights'. The main description reads: 'ShellCheck, a static analysis tool for shell scripts' followed by the URL <https://www.shellcheck.net>. At the bottom, a summary bar lists: '1,293 commits', '4 branches', '21 releases', '76 contributors', and 'GPL-3.0' license.

koalaman / shellcheck

Watch 367 Star 14,503 Fork 734

Code Issues 400 Pull requests 7 Projects 0 Wiki Insights

ShellCheck, a static analysis tool for shell scripts <https://www.shellcheck.net>

1,293 commits 4 branches 21 releases 76 contributors GPL-3.0

Кто программирует на Bash?

koalaman / shellcheck

Watch 367 Star 14,503 Fork 734

Code Issues 400 Pull requests 7 Projects 0 Wiki Insights

ShellCheck, a static analysis tool for shell scripts <https://www.shellcheck.net>

1,293 commits 4 branches 21 releases 76 contributors GPL-3.0

```
aaronkili@tecmint ~/bin $ shellcheck test.sh

In test.sh line 24:
E_NOTROOT=50
^-- SC2034: E_NOTROOT appears unused. Verify it or export it.

In test.sh line 25:
E_MINARGS=100
^-- SC2034: E_MINARGS appears unused. Verify it or export it.

In test.sh line 30:
echo $E_NONROOT
    ^-- SC2153: Possible misspelling: E_NONROOT may not be assigned, but E_NOTROOT is.
    ^-- SC2086: Double quote to prevent globbing and word splitting.

aaronkili@tecmint ~/bin $
```

Intellij IDEA B CI

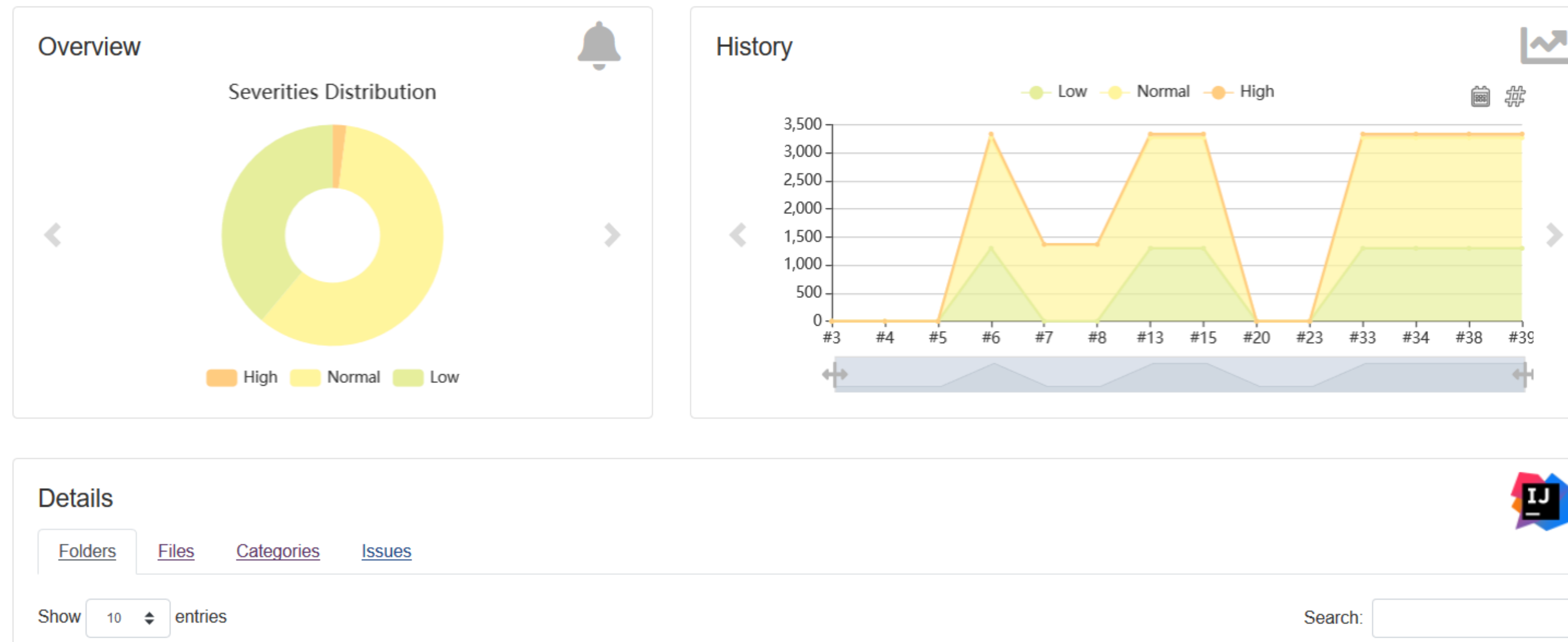
IntelliJ IDEA в CI

- `bin/format.sh` – форматирование кода
- `bin/inspect.sh` – инспекции (с выводом в `.xml`)

IntelliJ IDEA в CI

- `bin/format.sh` – форматирование кода
- `bin/inspect.sh` – инспекции (с выводом в `.xml`)

IntelliJ IDEA Inspections Warnings



Что из этого использовать?

Что из этого использовать?

Всё!

Как это всё использовать?

Как это всё использовать?

- Однократное применение анализа бессмысленно

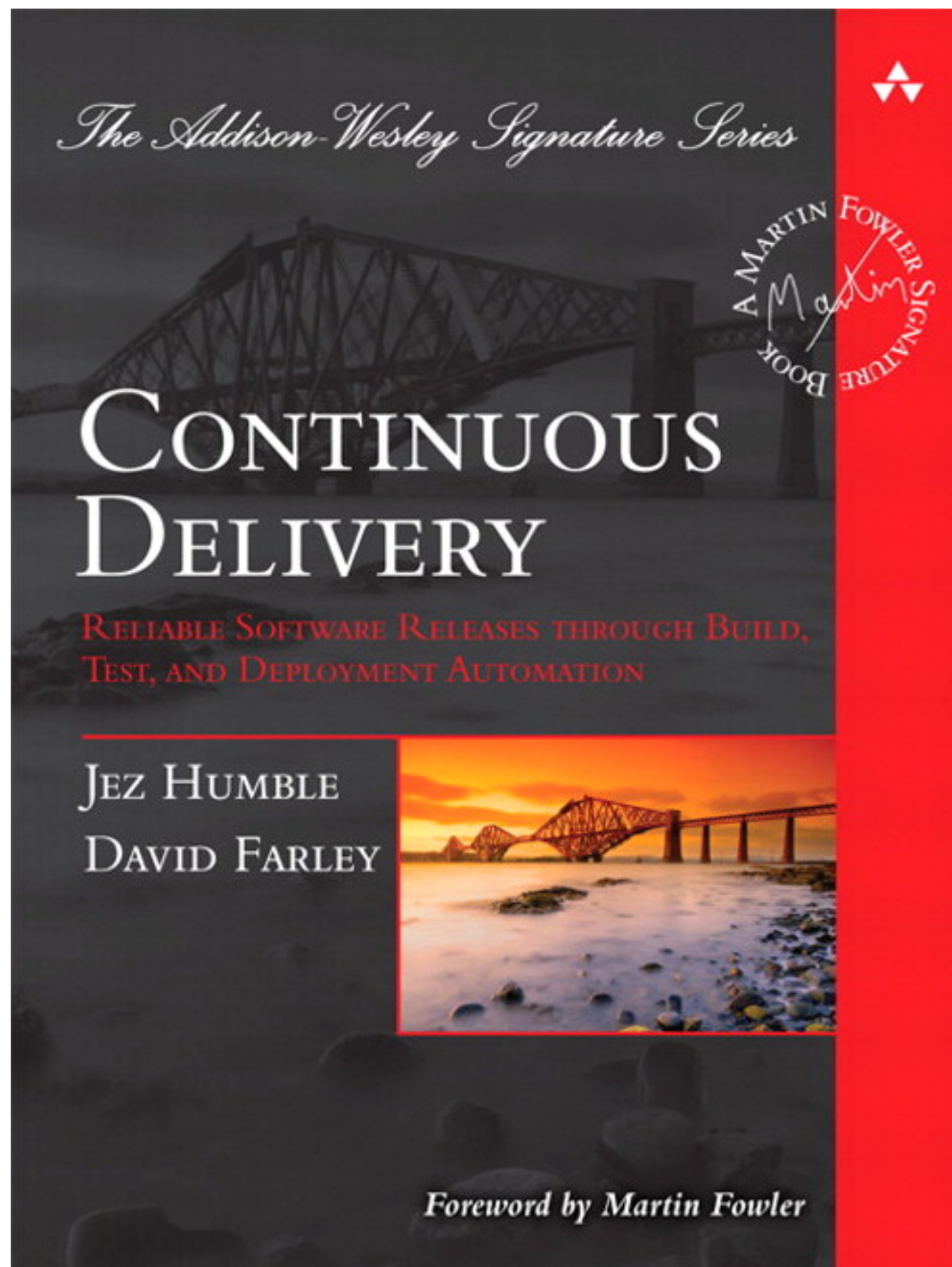
Как это всё использовать?

- Однократное применение анализа бессмысленно
- Анализ должен производиться **непрерывно и автоматически**

Как это всё использовать?

- Однократное применение анализа бессмысленно
- Анализ должен производиться **непрерывно и автоматически**
- Результаты анализа должны определять **quality gates**

Роль и место СА в конвейере поставки



Jez Humble, David Farley. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley, 2011

Типовой конвейер сборки



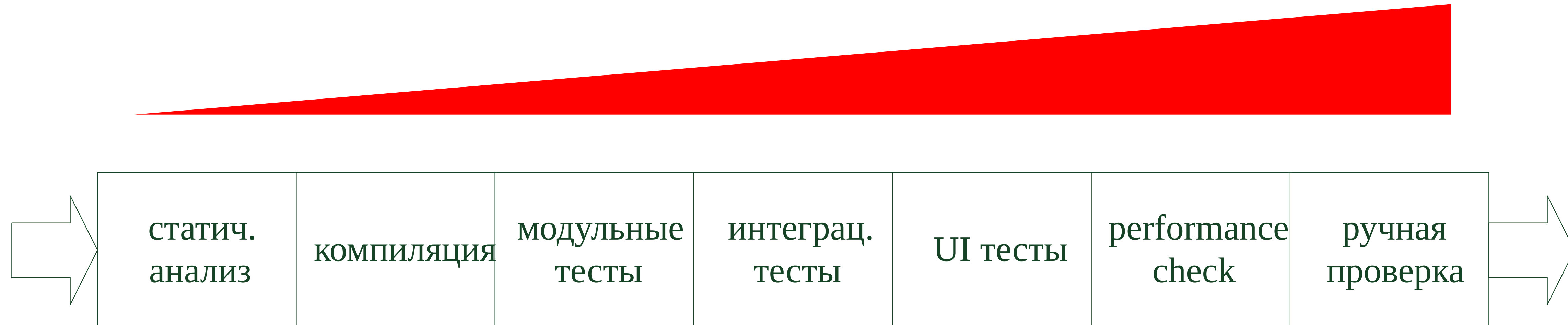
Типовой конвейер сборки

«Фильтрующая способность»



Типовой конвейер сборки

Сложность, стоимость,
время работы, вероятность сбоя



Многоступенчатый фильтр



Многоступенчатый фильтр

Размер пропускаемого загрязнения
Пропускная способность



Многоступенчатый фильтр

Сложность, стоимость



Вывод

Вывод

- Статанализ – «фильтр грубой очистки» в начале каскада фильтров

Вывод

- Статанализ — «фильтр грубой очистки» в начале каскада фильтров
- В отдельности от других — не работает

Вывод

- Статанализ — «фильтр грубой очистки» в начале каскада фильтров
- В отдельности от других — не работает
- Другие без него работают хуже

Случай из практики: долгий отклик

resource.json

```
{  
  "key": "value with "unescaped quotes" "  
}
```

Случай из практики: долгий отклик

resource.json

```
{  
  "key": "value with "unescaped quotes" "  
}
```

- Все UI тесты падают.

Случай из практики: долгий отклик

resource.json

```
{  
  "key": "value with "unescaped quotes" "  
}
```

- Все UI тесты падают.
- Но это происходит **спустя дни**.

Случай из практики: лечение

- Добавляем JSONLint в начало пути

```
find . -name \\.json -print0 | xargs -0 -n1 -t jsonlint -q
```

Случай из практики: лечение

- Добавляем JSONLint в начало пути

```
find . -name \\.json -print0 | xargs -0 -n1 -t jsonlint -q
```

- Отклик на проблему идёт **сразу**

Случай из практики: лечение

- Добавляем JSONLint в начало пути

```
find . -name \\.json -print0 | xargs -0 -n1 -t jsonlint -q
```

- Отклик на проблему идёт **сразу**

- PROFIT

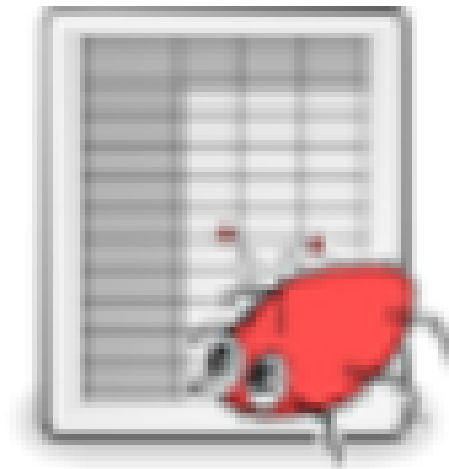
Внедрение в legacy-проект

Внедрение в legacy-проект

Знакомая картина?



Checkstyle: 2,496 warnings from one analysis.



FindBugs: 130 warnings from one analysis.

Внедрение в legacy-проект

Знакомая картина?



Checkstyle: 2,496 warnings from one analysis.



FindBugs: 130 warnings from one analysis.

Оставить нельзя пофиксить!



Пофиксить автоматически?

Google + Stackoverflow:

• 'sed remove trailing spaces'

```
find . -name '*.py' -print0 | xargs -0 -n1 -t \  
sed -i -r 's/\s+$//'
```

• 'bash add a newline to the end of a file'

```
find . -name '*.java' -print0 | xargs -0 -L1 bash \  
-c 'test "$(tail -c 1 "$0")" && printf "\r\n" >> $0'
```

• etc etc

Автофикс

- `Javascript: eslint --fix`

Spotless: идемпотентный автоформаттер

Spotless can format

<java | kotlin | scala | sql | groovy | javascript | flow | typeScript | css | scss | less | jsx
| vue | graphql | json | yaml | markdown | license headers | anything>

using

<gradle | maven | anything>



Quality Gates



Пороговое значение находок

Пороговое значение находок

- «Если меньше 100 находок, то код ОК»

Пороговое значение находок

- «Если меньше 100 находок, то код ОК»
- ДАНО: в коде 90 находок и код ОК.

Пороговое значение находок

- «Если меньше 100 находок, то код ОК»
- ДАНО: в коде 90 находок и код ОК.
- Добавляем Null Pointer Dereference.

Пороговое значение находок

- «Если меньше 100 находок, то код ОК»
- ДАНО: в коде 90 находок и код ОК.
- Добавляем Null Pointer Dereference.
- У нас 91 находка, код всё ещё ОК?

Пороговое значение находок

- «Если меньше 100 находок, то код ОК»
- ДАНО: в коде 90 находок и код ОК.
- Добавляем Null Pointer Dereference.
- У нас 91 находка, код всё ещё ОК?

Вывод: не используйте данный метод!

Suppression Profile

- Старые находки — в игнор
- Новые находки — не пропускаем

Suppression Profile

Наивный подход:

```
<file name="AppProperties.java">  
  <error line="31" column="5" message="Missing a Javadoc comment."/>  
  <error line="36" column="5" message="Missing a Javadoc comment."/>  
</file>
```

Suppression Profile

Наивный подход:

```
<file name="AppProperties.java">  
  <error line="31" column="5" message="Missing a Javadoc comment."/>  
  <error line="36" column="5" message="Missing a Javadoc comment."/>  
</file>
```

Добавляем текст в начало файла...

Suppression Profile

Наивный подход:

```
<file name="AppProperties.java">  
  <error line="31" column="5" message="Missing a Javadoc comment."/>  
  <error line="36" column="5" message="Missing a Javadoc comment."/>  
</file>
```

Добавляем текст в начало файла...

...номера строк "уползли" и все находки снова появились.

Suppression Profile

Подход PVS-Studio -- хеши строк:

```
{  
  "FileName": "CelestaParser.java",  
  "ErrorCode": "V6021",  
  "CodePrev": -1464702071,  
  "CodeCurrent": -1679070819,  
  "CodeNext": 35764079  
}
```


Suppression Profile

Вывод: метод хорош, но труднодоступен

Проверка правописания

prefer?ed

ac?om?odate

cred?t

over?id?en

defer?ed

deb?t

over?ide

anal?s?s

progra?

Проверка правописания

GNU Aspell



Проверка документации:

```
for f in $(find . -name '*.adoc'); do \  
  cat $f | aspell --master=ru \  
  --personal=./dict list; done \  
| sort | uniq
```

Проверка литералов и комментариев:

```
for f in $(find . -name '*.java'); do \  
  cat $f | aspell --mode=cpp \  
  --master=ru --personal=./dict list; done \  
| sort | uniq
```

Проверка правописания

Проверка правописания

- Храните пользовательский словарь в проекте

Проверка правописания

- Храните пользовательский словарь в проекте
- Quality Gate: **не должно быть незнакомых спелчекеру слов.**

Упавшая проверка

Stage Logs (Spellcheck)

☑ Shell Script -- true -- (self time 263ms)

☑ Print Message -- The following words are probaly misspelled: -- (self time 5ms)

☑ Print Message -- вфыва длфювфыа фыв фыжв ывдыфоа ыффыва -- (self time 4ms)

вфыва

длфювфыа

фыв

фыжв

ывдыфоа

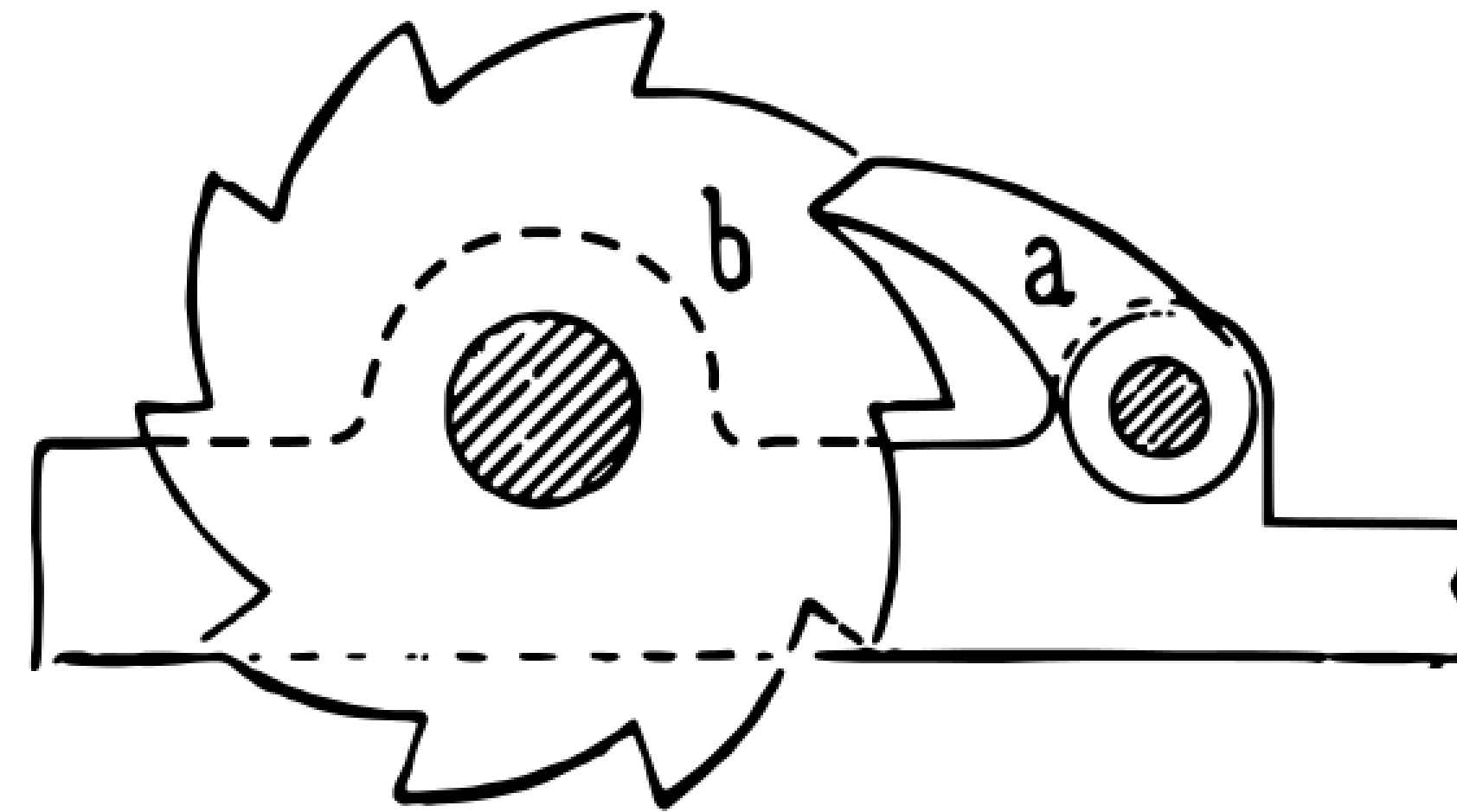
ыффыва

Error signal (self time 5ms)

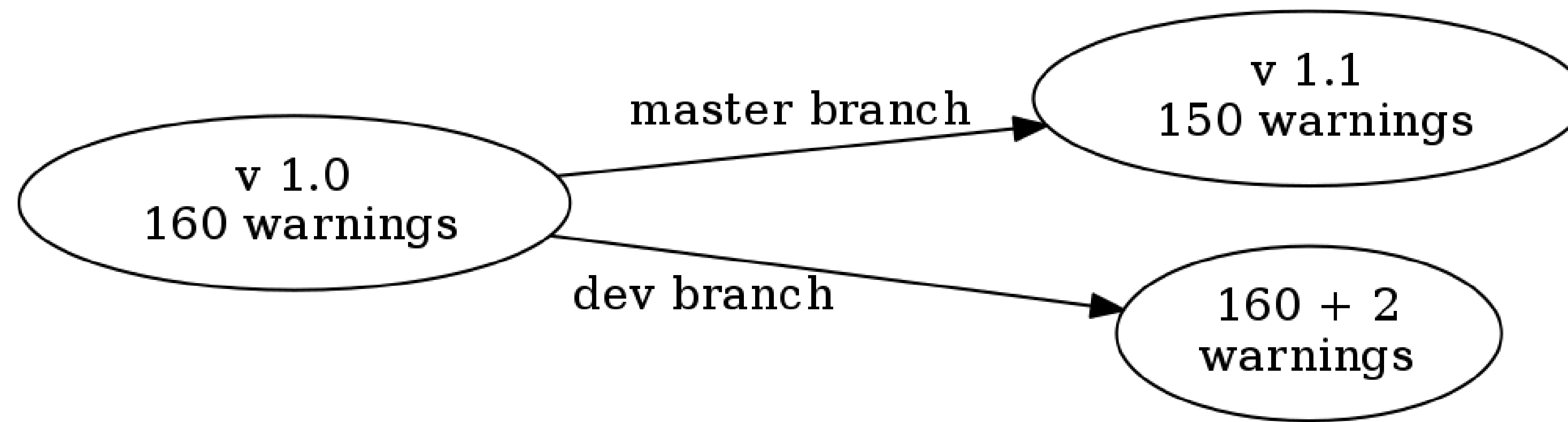
Проверка правописания

Вывод: spellchecker может быть частью пайплайна

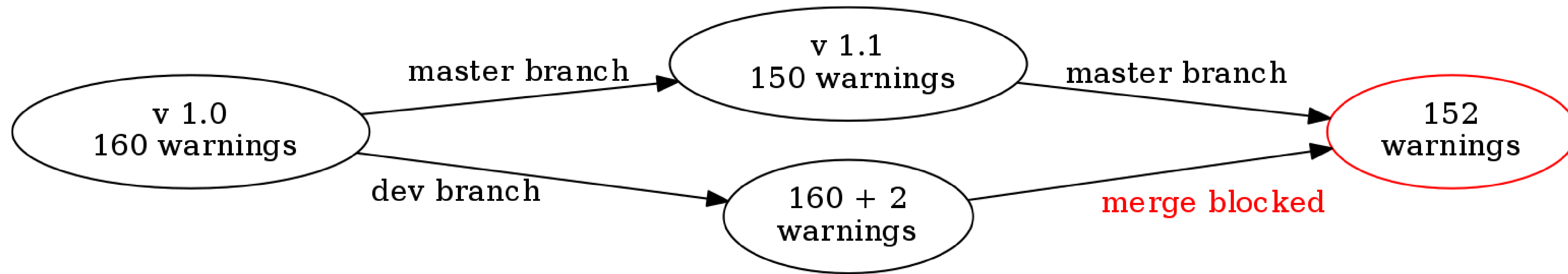
Храповик



Принцип работы



Принцип работы



Упавшая проверка

Pull

Full project



Number of flake8 warnings 1597 is greater than previous 1596.

Stage View

Average stage times:

Clone

7s

Static analysis

23s

#1

Mar 30

03:24

No Changes



7s

23s

816ms

failed

Error signal

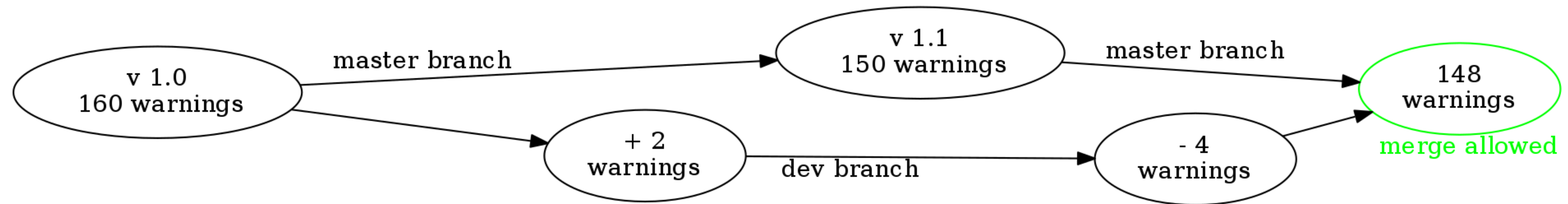
Number of flake8 warnings 1597 is greater than previous 1596.

See stage logs for more detail.

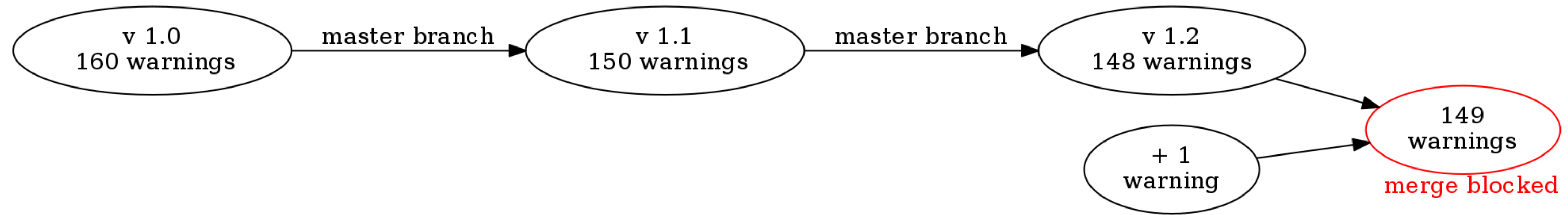
Logs

Error(s)

Принцип работы



Принцип работы



Много модулей/инструментов

Вид метаданных:

```
# warnings.yml  
celestasql:  
  checkstyle: 434  
  spotbugs: 45  
celestacore:  
  checkstyle: 206  
  spotbugs: 13  
celestamavenplugin:  
  checkstyle: 19  
  spotbugs: 0  
celestainit:  
  checkstyle: 0  
  spotbugs: 0
```

Упавшая проверка

```
= celesta-sql.checkstyle: 399->399
= celesta-sql.findbugs: 41->41
= celesta-core.checkstyle: 185->185
= celesta-core.findbugs: 13->13
+ celesta-maven-plugin.checkstyle: 19->20
= celesta-maven-plugin.findbugs: 0->0
= celesta-system-services.checkstyle: 18->18
= celesta-system-services.findbugs: 0->0
= dbschemasync.checkstyle: 3->3
= dbschemasync.findbugs: 0->0
= celesta-unit.checkstyle: 0->0
= celesta-unit.findbugs: 0->0
```

Error signal (self time 3ms)

Как это реализовано у нас

- Jenkins scripted pipeline
- Jenkins shared libraries in Groovy
- JFrog Artifactory для хранения метаданных о сборках

Парсинг XML-вывода анализаторов

```
<checkstyle>
  <file name="...">
    <error line="1" severity="..." message="..." />
  </file>
  ...
</checkstyle>
private Map countModule(prefix) {
  def count = [:]
  def f = new File("${prefix}/target/checkstyle-result.xml")
  if (f.exists()) {
    def checkstyle = new XmlSlurper().parseText(f.text)
    count.put("checkstyle", checkstyle.file.error.size())
  }
  ...
  count
}
```

Скачиваем данные о последней сборке

```
def server = Artifactory.server 'ART'
def downloadSpec = """
    {"files": [
      {
        "pattern": "warn/${project}/*/warnings.yml",
        "build": "${project} :: dev/LATEST",
        "target": "previous.yml",
        "flat": "true"
      }
    ]}
  """
server.download spec: downloadSpec
oldWarnings = readYaml file: 'previous.yml'
```

Шаг храповика

```
stage ('Ratcheting') {  
  def warningsMap = countWarnings()  
  writeYaml file: 'target/warnings.yml', data: warningsMap  
  compareWarningMaps oldWarnings, warningsMap  
}
```

Jenkins Warnings NG Plugin

Собирает и читает отчёты всех известных анализаторов

```
def checkstyle
  = scanForIssues tool: checkStyle(pattern: '**/cs.xml')

def spotbugs
  = scanForIssues tool: spotBugs(pattern: '**/spotbugs.xml')

def idea
  = scanForIssues tool: ideaInspection(pattern: 'target/idea_inspections/*')

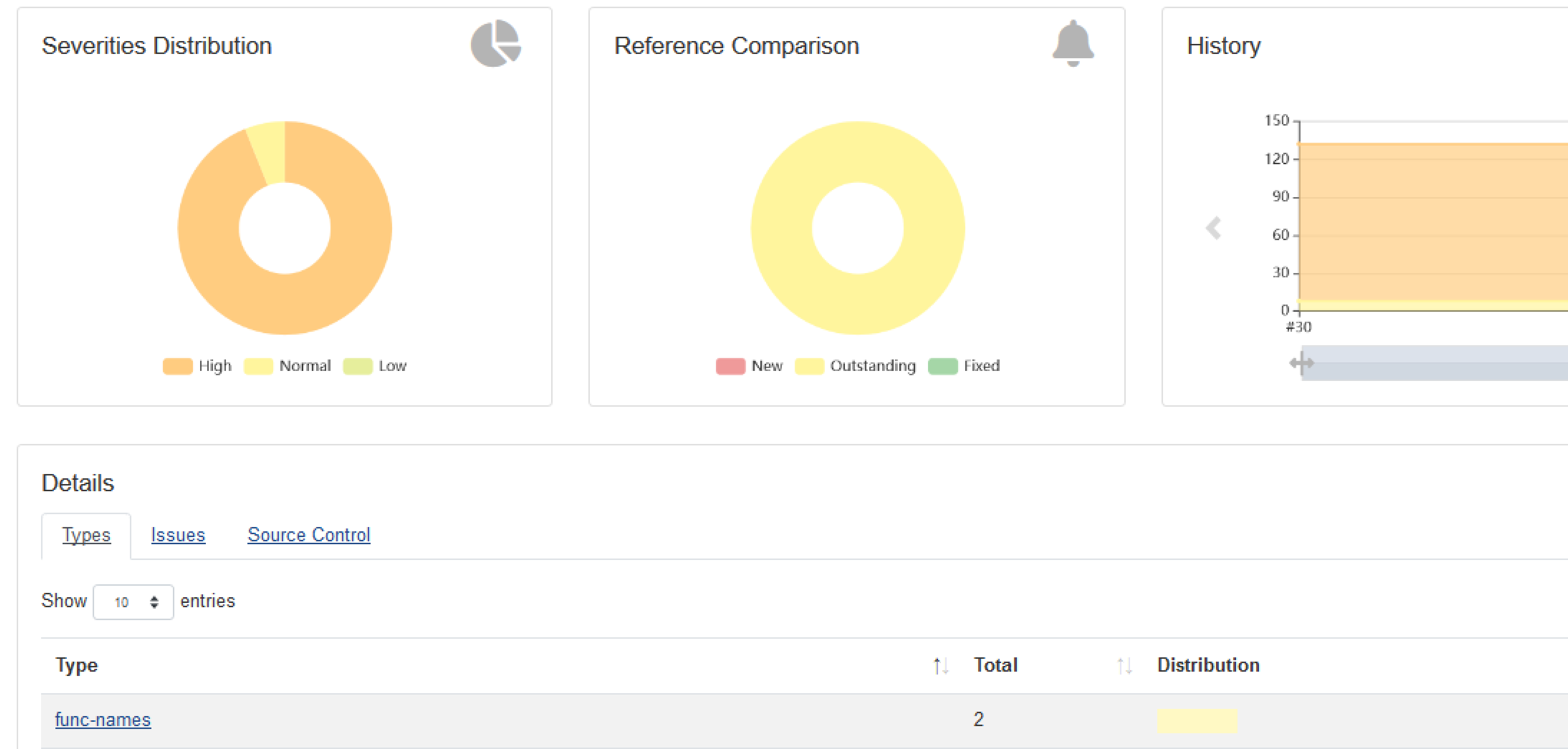
def eslint
  = scanForIssues tool: esLint(pattern: '**/eslint.xml')

publishIssues issues: [checkstyle, spotbugs, idea, eslint]
. . .
```

Jenkins Warnings NG Plugin

Красиво отображает

ESLint Warnings



Jenkins Warnings NG Plugin

Можно программировать Quality Gates, в т. ч. в виде разницы с reference build:

```
recordIssues tool: java(pattern: '*.log'),  
             qualityGates: [[threshold: 1,  
                             type: 'TOTAL',  
                             unstable: true]]
```

Храповик: ожидание



Храповик: реальность



Случай из практики

Кто здесь видит проблему?

```
#.travis.yml
```

```
install:
```

- pip install yamllint
- pip install ansible-lint

```
script:
```

```
  . . .
```

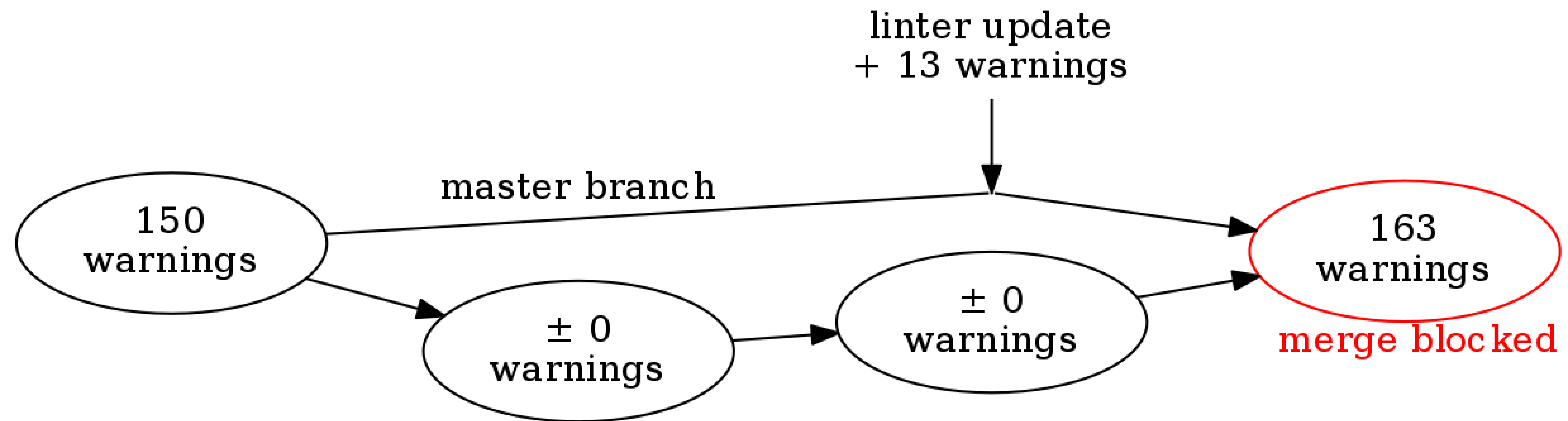
```
# Check YAML validity
```

- yamllint -c yamllint.yml .

```
# Ansible code static analysis
```

- ansible-lint . . .
- ansible-lint . . .
- ansible-lint . . .

Невоспроизводимая сборка



При замене фильтров бывает грязно!



Фиксируем версии всего!

```
#.travis.yml
```

```
install:
```

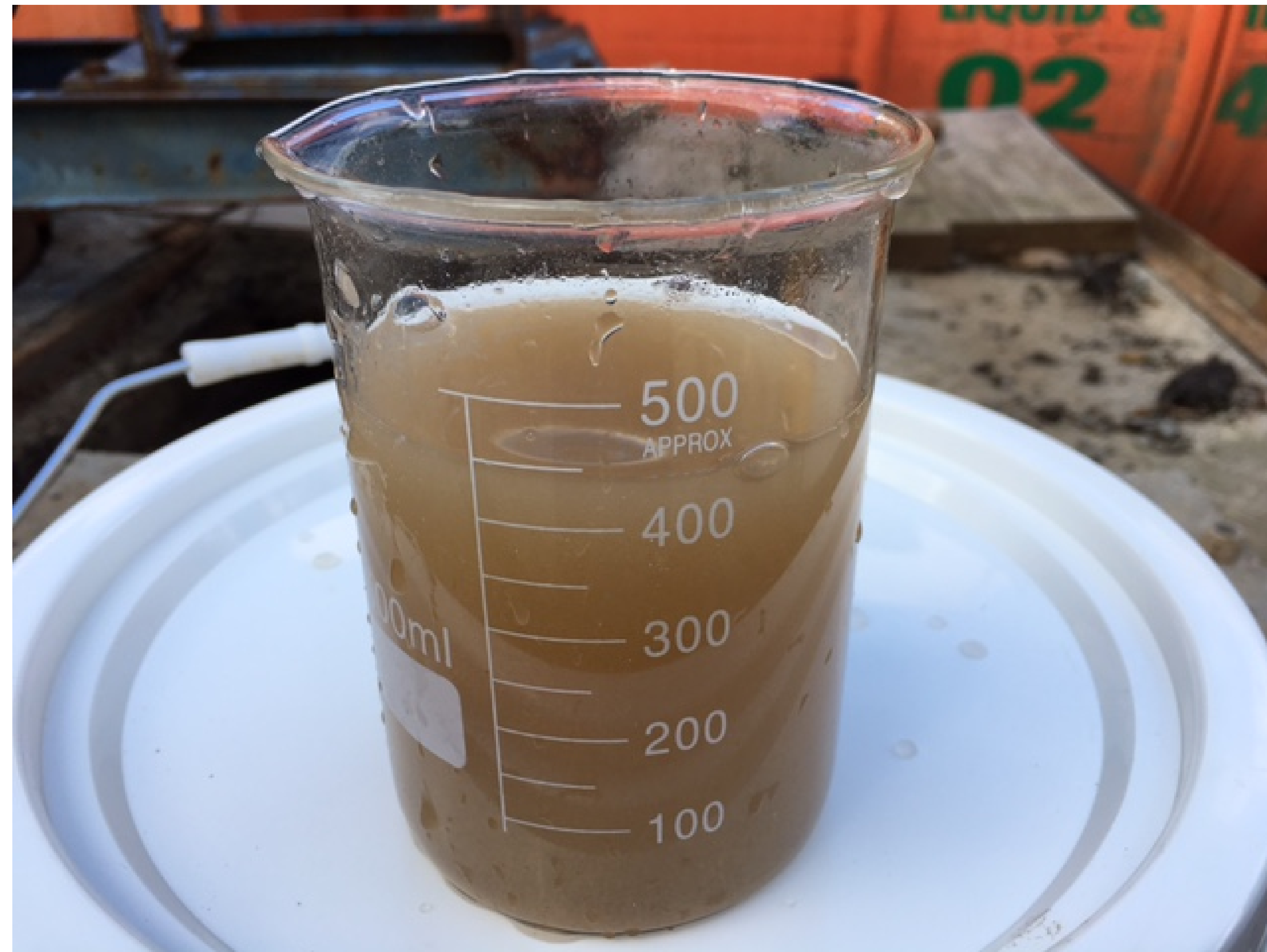
- pip install yamllint==1.13.0
- pip install ansible-lint==3.5.1

Выводы

Статанализ разнообразен



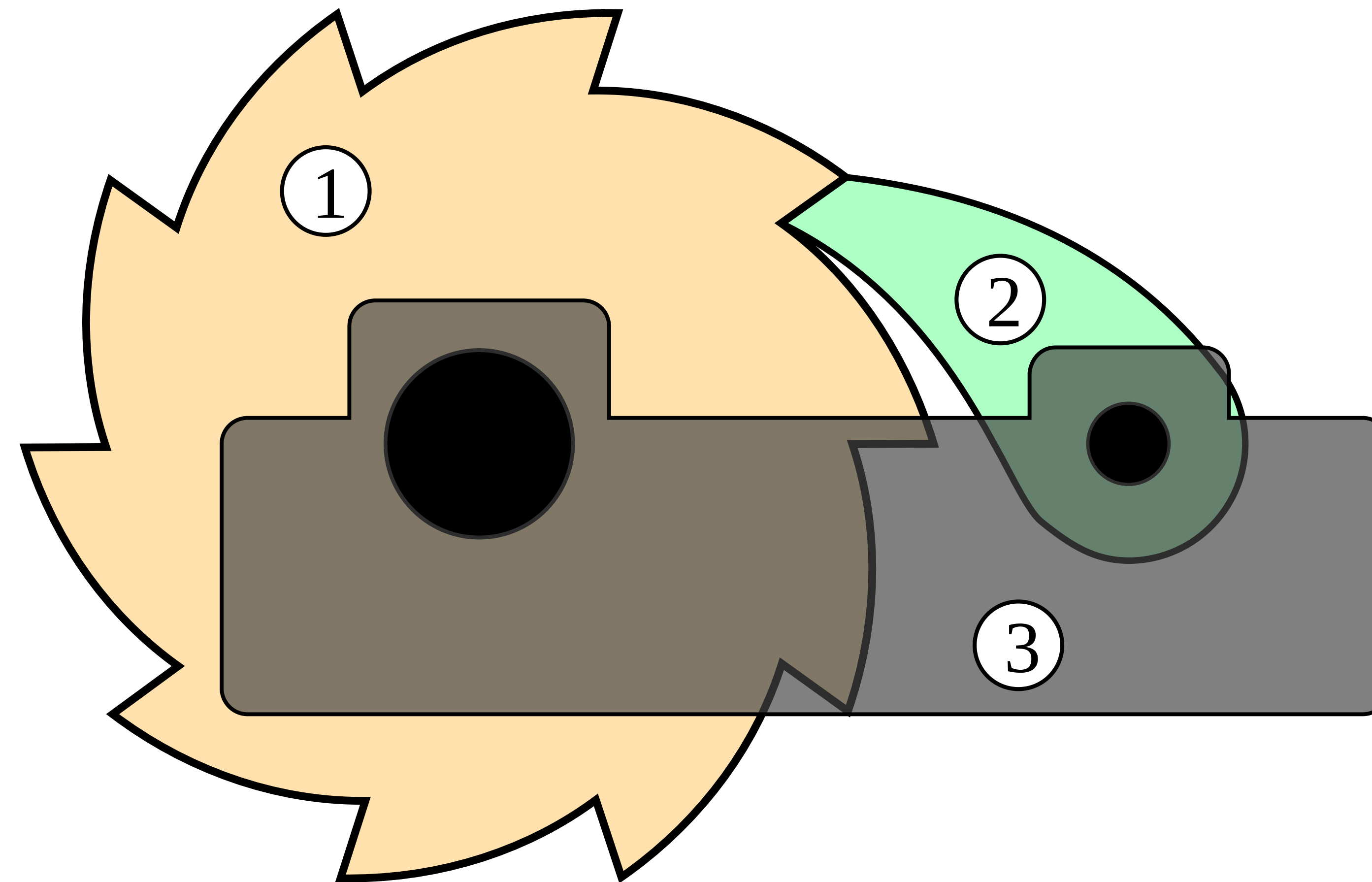
Статанализ бесполезен при нерегулярном применении



Фильтр грубой очистки ставится в начале каскада



Используйте храповик





Не забывайте про повторяемость сборок



Ссылки

- **Humble, Jez; Farley, David (2011)**. Continuous Delivery: reliable software releases through build, test, and deployment automation.
- **Иван Пономарев** [Внедряйте статический анализ в процесс, а не ищите с его помощью баги](#)
- **Иван Пономарев** [Запускаем инспекции IntelliJ IDEA на Jenkins](#)
- **Алексей Кудрявцев** [Анализ программ: как понять, что ты хороший программист](#)

На этом всё!

-  @inponomarev
-  ponomarev@corchestra.ru
- **Спасибо!**

