

# SAVE:

Testing static analyzers and compilers





## **Intro**

About the problem and us

01

## **SAVE**

Ideas and approaches

02

## **CI/CD and Cloud**

Let's make it look like enterprise

03

## **Conclusion**

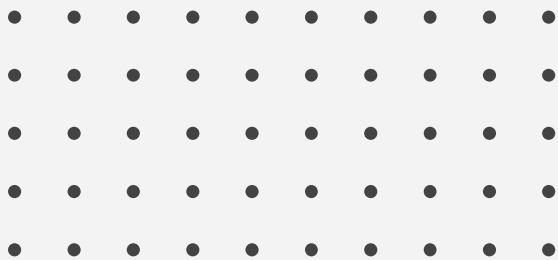
Let's summarize

04

# 01

## Intro

About the problem and us



# Problem

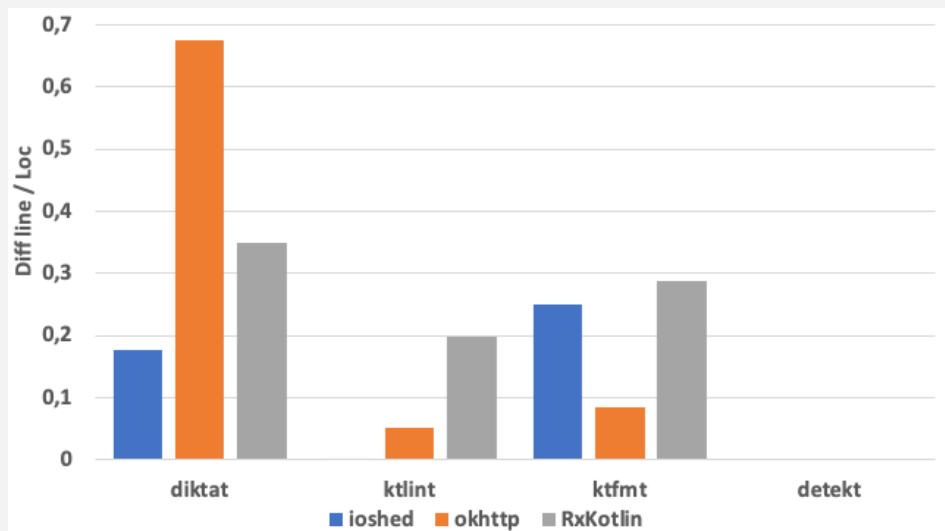
We are creating and using:

- Vendor tools and Internal Systems for Static Analysis

We see the lack of tools and frameworks in open-source for:

- Measuring of capabilities, functional testing
- Quick and easy regression testing
- Standardization and benchmarking

# Need some evaluation and comparison mechanism



A figure from our paper for ISSRE conference comparing code fixers for Kotlin

For example:

We created static analyzer for Kotlin:  
<https://github.com/cqfn/diKTat>

We integrated to:  
<https://github.com/diffplug/spotless>

**But what if we want to find and evaluate existing tool instead of creating a brand new one?**

# Our inspiration and existing works

lit – LLVM Integrated Tester



spec.org - Standard Performance  
Evaluation Corporation



MISRA - Motor Industry Software  
Reliability Association



Industrial Compilers and Static Analyzers





# 350,000+

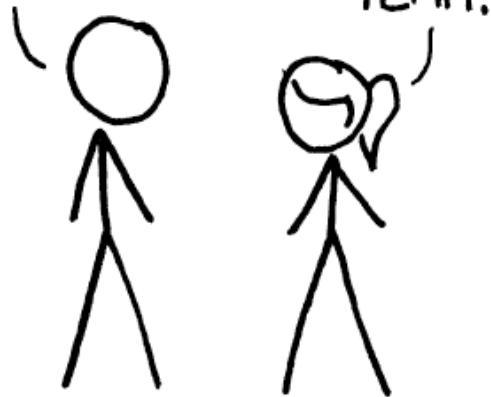
This is how many functional tests are there in GCC. LLVM and Clang have mostly same huge test packages. Industrial compilers, like ICC, can have 1 mln+ tests

# HOW STANDARDS PROLIFERATE:

(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)

SITUATION:  
THERE ARE  
14 COMPETING  
STANDARDS.

14?! RIDICULOUS!  
WE NEED TO DEVELOP  
ONE UNIVERSAL STANDARD  
THAT COVERS EVERYONE'S  
USE CASES.



SOON:

SITUATION:  
THERE ARE  
15 COMPETING  
STANDARDS.



Abstract geometric lines in a dark gray color. One line starts from the top center, goes down and left to a point, then goes down and right to the bottom center. Another line starts from the bottom center and goes up and right towards the right edge. A third line starts from the top center and goes down and right towards the right edge. These lines create a large, irregular polygonal shape on the left side of the slide.

# 02

## SAVE

Static Analysis Verification and Evaluation,  
CLI application, core framework

# How to close these gaps

<https://github.com/cqfn/save>

01

Native CLI application that can  
process readable tests with  
different plugins

Some specific CI/CD service, that  
has standard benchmarks and  
executes tests on flexible cluster

02

<https://github.com/cqfn/save-cloud>

3 active contributors with the most active @petertrr



# Concept points

What did we want (and still want) to achieve?

01

**Native** application

**Kotlin Native Multiplatform**

Win/Linux/macOS

02

**Plugins** and reporters for test logic  
(processing)

Plugins and reporters should have  
a common interface

03

**Configuration** mechanism

Hierarchical inheritance  
Logic via config DSL (~~TOML~~)

<https://github.com/cqfn/save>

# Config and default plugins

save.toml (<https://github.com/akuleshov7/ktoml>)

[general]

```
tags = ["documentation", "custom tag", "other tags"]
description = "Test for diktat - linter and formatter for Kotlin"
suiteName = "warnings"
execCmd = "java -jar ktlint -R diktat.jar"
expectedWarningsPattern = "// ;warn:(.+):(\\d+): (.+)"
```

Common section with the main info

[warn]

```
testNameSuffix = "Test.kt"
actualWarningsPattern="(\\w+\\.+):(\\d+):(\\d+): (\\[.*\\].*)$"
exactWarningsMatchHasColumn = true
warningTextHasLine = true
```

Plugins (test execution logic). Optional

[fix]

```
execFlags = "-F"
```

# Default plugins

## [FIX] plugin

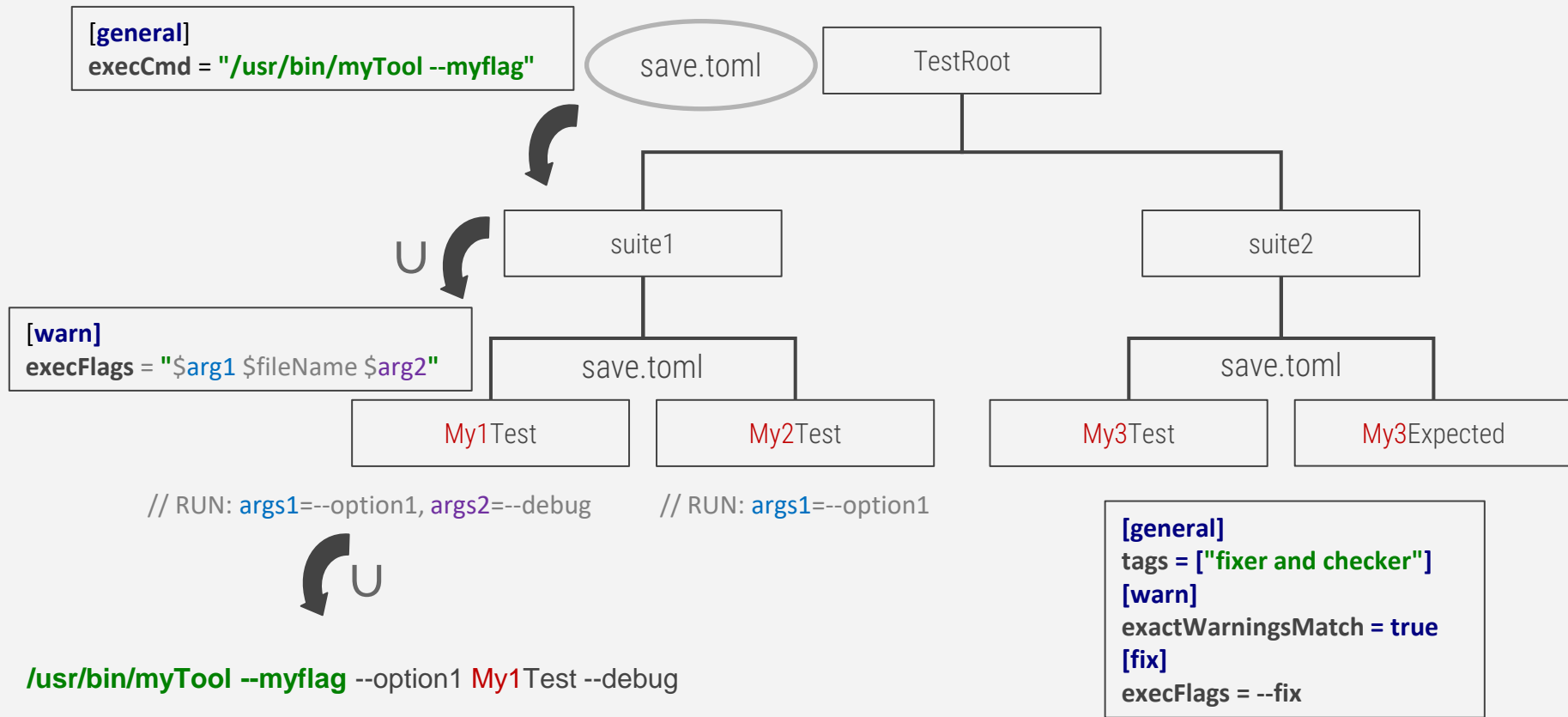
1. **Execute** tested tool that should be tested on the test resource with **Test** suffix in the name
2. **Compare** the result with the resource with **Expected** suffix in the name

## [WARN] plugin

1. **Execute** tested tool that should be tested on the test resource with **Test** suffix in the name
2. Map and **Compare** the output with special metadata

```
// ;warn:1:7: Class name should be in an uppercase format
// ;warn:3:13: Method B() should follow camel-case convention
class a {
    // ;warn:2:13: Single symbol variables are not informative
    // ;warn:2:14: Trailing semicolon is redundant in Kotlin
    val b: String;
    fun B(): String {}
    fun setB(): String {}
}
```

# Less code duplication with configs



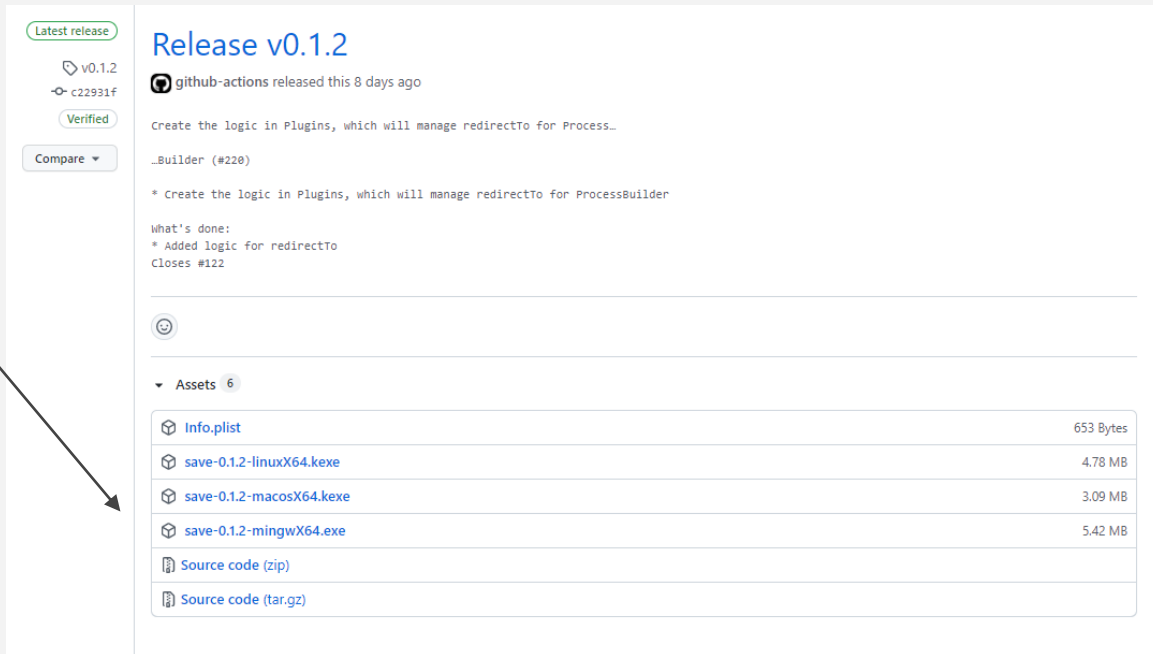
# In-file test setup

<https://github.com/cqfn/save/tree/main/examples/kotlin-diktat>

```
// RUN: args2=--debug  
enum class EnumValueSnakeCaseTest {  
    // ;warn:10:5: [ENUM_VALUE] enum values should be in selected UPPER_CASE format: paSC_SAI_I  
    paSC_SAI_I,  
}
```

# Setup and run

1. Download SAVE executable (native binary) from GitHub. No additional SDK runtime required; everything runs out of the box.
2. Execute  
\$ ./save --help  
\$ ./save test/root/location
3. Get results in your **favourite** format: JSON/PLAIN/YML



Latest release

v0.1.2  
c22931f  
Verified  
Compare ▾

## Release v0.1.2

github-actions released this 8 days ago

Create the logic in Plugins, which will manage redirectTo for Process...

\_Builder (#228)

\* Create the logic in Plugins, which will manage redirectTo for ProcessBuilder

What's done:  
\* Added logic for redirectTo  
Closes #122

Assets 6

Info.plist	653 Bytes
save-0.1.2-linuxX64.kexe	4.78 MB
save-0.1.2-macosX64.kexe	3.09 MB
save-0.1.2-mingwX64.exe	5.42 MB
Source code (zip)	
Source code (tar.gz)	





# 03

## **SAVE CI/CD in Cloud**

Let's make it look like enterprise with SAVE-cloud service



# Concept points

What did we want (and still want) to achieve?

01

Parallel execution of a large number of tests

Flexible execution of several SAVE-cli frameworks in **Docker** containers

02

User-friendly **dashboard** with test results

Historical results, regression testing

03

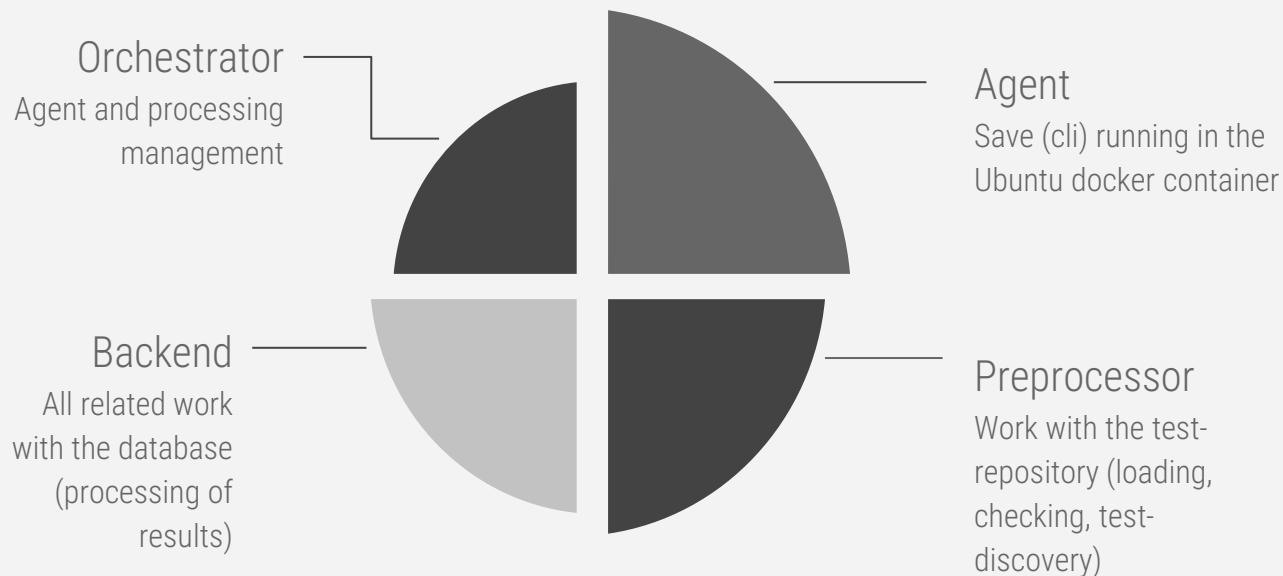
Benchmarking and Comparisson

Standard test suites, benchmarks, comparison of community-made static analyzers

<https://github.com/cqfn/save-cloud>



# Save-cloud main components



All components are using **JVM and Kotlin**

# Technology stack

	Technology stack
<b>Orchestrator</b>	Docker, Prometheus, Graphana, testcontainers
<b>Agent</b>	Kotlin Native, KTOR (communication via heartbeats)
<b>Preprocessor</b>	Spring Boot 2.5, Spring Security
<b>Backend</b>	Spring Boot 2.5 WebFlux (Project Reactor stack), Hibernate/JPA, Mysql, Liquibase

# Upload your custom git test repository or select existing suites

SAVE / cqfn / save-examples

User Name 

Project save-examples

UPLOAD TYPES

Upload project as Git

Upload project as binary file

RUN TEST

Git url:

Path to property file:

INFORMATION

Name: save-examples

Description: Examples for diktat analyzer from SAVE project

[Latest execution](#)

[Execution history](#)

Select files:

✕ ktlint (uploaded at 2021-09-01T09:18:14.369, size 52897 KiB)

✕ diktat.jar (uploaded at 2021-09-01T09:17:48.285, size 6218 KiB)

Select a file from existing

⬆️ Upload files:

SDK:

SDK's version:

Run tests now

# Simply get test results with the history

SAVE / cqfn / save-examples / history / 50

User Name 

Project version: 9ec6c44a2771cf62a600ea6754254a6685c6c592

Status: FINISHED

Rerun execution

#	Start time	Status	Test file path
0	2021-09-01T09:20:02Z	FAILED	fix/smoke/src/main/kotlin/org/cqfn/save/Example1Expected.kt
1	2021-09-01T09:20:02Z	FAILED	fix_and_warn/smoke/src/main/kotlin/org/cqfn/save/Example1Expected.kt
2	2021-09-01T09:20:02Z	PASSED	warn/chapter1/EnumValueSnakeCaseTest.kt
3	2021-09-01T09:20:02Z	PASSED	warn/chapter1/GenericFunctionTest.kt
4	2021-09-01T09:20:02Z	PASSED	warn-dir/chapter1/EnumValueSnakeCaseTest.kt
5	2021-09-01T09:20:02Z	PASSED	warn-dir/chapter1/GenericFunctionTest.kt
6	2021-09-01T09:20:02Z	PASSED	warn-dir/chapter1/SmallTest.kt
7	2021-09-01T09:20:02Z	FAILED	warn-dir/chapter2/GenericFunctionTest.kt
8	2021-09-01T09:20:02Z	FAILED	warn-dir/chapter3/GenericFunctionTest.kt



04

# Conclusion

Let's summarize





# Where SAVE could be used



## Static analyzers and auto-fixers

- Checking warnings in the code
- Checking auto-fixed code



## Compilers and their parts

- Testing generated IR and generated final code
- Expected behaviour of compiled program
- Warnings and errors in the front-end (parser)