



The  
joy of maintaining dead code

# ABOUT ME.



Gerrit Grunwald | Developer Advocate | Azul

WHO LOVES TO  
MANITAIN  
LEGACY  
CURE?

SOMEONE  
STARTS

4 HOURS

A WEEK

A collage of nine black and white photographs arranged in three rows. The top row shows a close-up of a document with the word 'HARVEY' at the top, followed by a portrait of a man in a military uniform, and a circular emblem featuring a tree and a figure. The middle row shows a close-up of a document with a circular postmark from 'NEW YORK' and a circular emblem featuring a tree and a figure. The bottom row shows a close-up of a document with the word 'ST. CATHERINE' at the top, followed by a portrait of a woman, and a circular emblem featuring a tree and a figure.

A collage of nine black and white photographs arranged in three rows. The top row shows a close-up of a postcard with a decorative border and faint text. The middle row features a postcard with a large ornate frame and a small inset image. The bottom row includes a postcard with a large stylized letter 'P' and the text 'It's Good', another with a portrait of a woman, and a final one with a hand holding a small object.

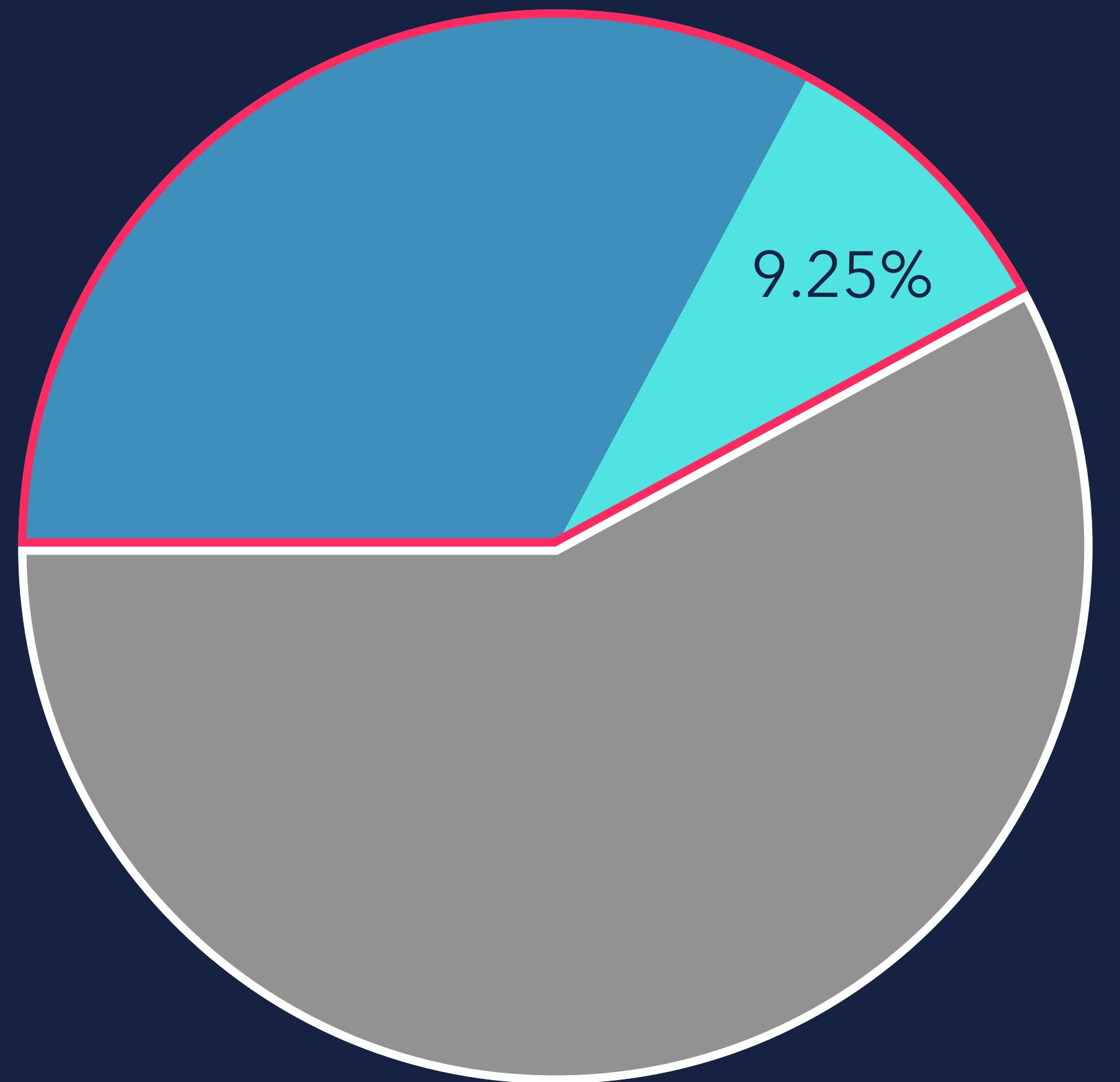
A collage of five black and white photographs. From left to right: 1. A close-up of a dark, textured object, possibly a piece of equipment or a sample, with handwritten text at the bottom: "This Space", "be", and "1". 2. A circular view of the Moon's surface with a prominent crater. 3. A circular view of the Moon's surface with a prominent crater. 4. A circular view of the Moon's surface with a prominent crater. 5. A circular view of the Moon's surface with a prominent crater.

**WE QUALITY IS TO**

**\$85 BILLION**

**ANNUALLY!**

# THE DEVELOPER WORK WEEK



41.1 total hours

Average developer work week

- 17.3 hours  
modifying, debugging,  
refactoring and bad code
- 13.5 hours  
Technical debt
- 3.8 hours  
Bad code



<https://www.pullrequest.com/blog/cost-of-bad-code/>

azul

AND YOU WAIT

YES IT

COST YOU? ?



## WHAT DOES IT COST YOU...?

- ☠ Time
- ☠ Productivity
- ☠ Agility
- ☠ Money
- ☠ Reputation

A collage of five black and white images arranged horizontally. From left to right: 1) A close-up of a hand holding a small, dark, textured object. 2) A decorative floral arrangement in a vase, featuring large, curly leaves and flowers. 3) A postcard with handwritten text and a stamp. The text includes "POST CARD", "PAGE 14", "S C", and "THE PATENTTEVILLE ODEON". 4) A circular portrait of a man with dark hair and a mustache, wearing a suit and tie. 5) A dark, abstract circular shape with irregular edges and some internal texture.

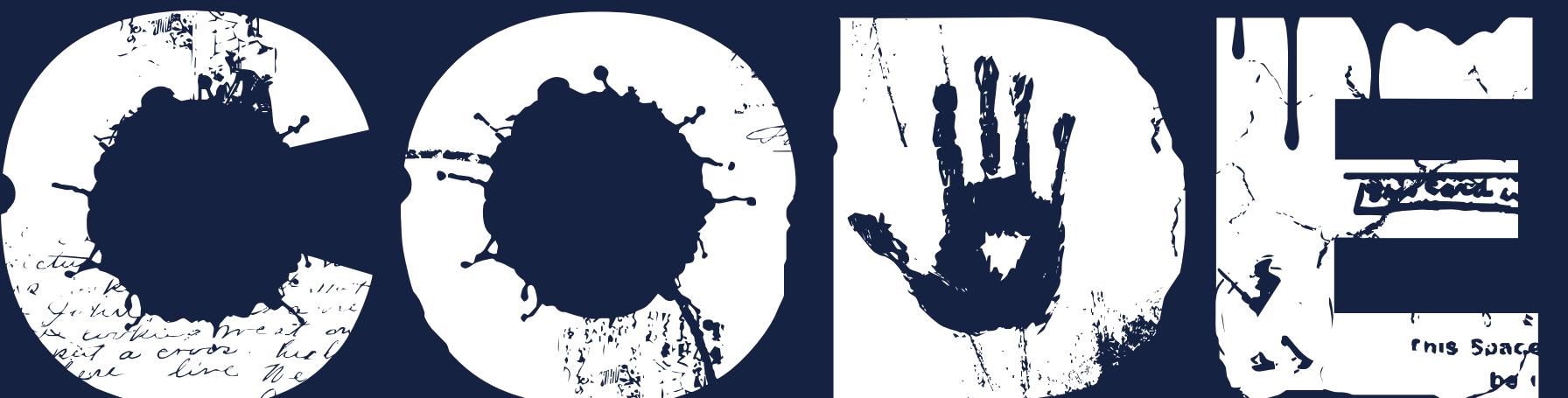
A collage of nine black and white images arranged in a grid. The images include a portrait of a man, a decorative chandelier, a butterfly, a map of New York City, a newspaper clipping from 'The New York Daily News' with the headline 'PUPPETS', a close-up of a hand holding a small object, a circular portrait of a woman, and a close-up of a hand holding a small object.

This strip of film consists of several circular frames arranged horizontally. From left to right, the frames show: 1. A close-up of a hand holding a white card with printed text. The text includes "It's Good", "AND IT'S GOOD FOR", "ST. JOE VALLEY CREAMERY CO.", "FRESH DAIRY PRODUCTS", and "Joseph N". 2. A person's face, possibly a woman, with dark hair and bangs. 3. A hand reaching upwards towards the sky. 4. A large, dark, circular object, possibly a planet or a large animal, with a textured surface and small protrusions. 5. Another view of the same large, dark, circular object. 6. A hand reaching upwards again. 7. A close-up of a hand holding a white card with printed text. The text includes "This Space", "be i", and "not a cross".

azul

TWO TYPES OF BAD CODE...

LEGACY  
CODE



LEGACY  
CODE



# LEGACY CODE



IS CODE THAT...

- ⌚ Was written for no longer supported os, hardware, api's etc.

# LEGACY CODE



IS CODE THAT...

- ⌚ Was written for no longer supported os, hardware, api's etc.
- ⌚ Cannot be tested automatically

# LEGACY CODE



IS CODE THAT...

- ⌚ Was written for no longer supported os, hardware, api's etc.
- ⌚ Cannot be tested automatically
- ⌚ Is no longer developed or maintained

# LEGACY CODE



IS CODE THAT...

- ⌚ Was written for no longer supported os, hardware, api's etc.
- ⌚ Cannot be tested automatically
- ⌚ Is no longer developed or maintained
- ⌚ Lacks documentation

# LEGACY CODE



IS CODE THAT...

- ⌚ Was written for no longer supported os, hardware, api's etc.
- ⌚ Cannot be tested automatically
- ⌚ Is no longer developed or maintained
- ⌚ Lacks documentation
- ⌚ You are afraid to touch

WHY STOOL  
USING OTTOPA

# WHY STILL USING IT...?



BECAUSE...

- ④ It works

# WHY STILL USING IT...?



BECAUSE...

- ① It works
- ② There is no replacement available

# WHY STILL USING IT...?



BECAUSE...

- It works
- There is no replacement available
- Nobody is able to understand it

# WHY STILL USING IT...?



BECAUSE...

- It works
- There is no replacement available
- Nobody is able to understand it
- No one knows if it is still in use

HOW TO  
HANDLE  
IT

# HOW TO HANDLE IT...



## SOLUTIONS TO HANDLE LEGACY CODE

- ① Refactoring

# HOW TO HANDLE IT...



## SOLUTIONS TO HANDLE LEGACY CODE

- ⌚ Refactoring
- ⌚ Rewrite everything

# HOW TO HANDLE IT...



## SOLUTIONS TO HANDLE LEGACY CODE

- ⌚ Refactoring
- ⌚ Rewrite everything
- ⌚ Find out if it is used at all

# REFACTORING



# REFACTORING

## SOLUTION

- ⌚ You need to understand the code first



# REFACTORING

## SOLUTION

- ⌚ You need to understand the code first
- ⌚ You can make use of static code analysis tools

# REFACTORING

## SOLUTION

- 🕒 You need to understand the code first
- 🕒 You can make use of static code analysis tools
- 🕒 CheckStyle, SpotBugs, PMD, SonarQube and others



# REFACTORING

## SOLUTION

- ⌚ You need to understand the code first
- ⌚ You can make use of static code analysis tools
  - ⌚ CheckStyle, SpotBugs, PMD, SonarQube and others
- ⌚ You need the time to do it



**REWRITE**

# REWRITE

## SOLUTION

- ➊ You need to know what the code does



# REWRITE



## SOLUTION

- ⌚ You need to know what the code does
- ⌚ It's like a new project with all it's work (and cost) involved

# REWRITE



## SOLUTION

- ⌚ You need to know what the code does
- ⌚ It's like a new project with all it's work (and cost) involved
- ⌚ You need to know the requirements

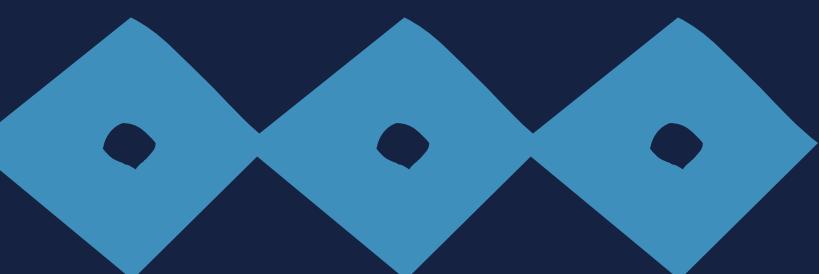
# REWRITE



## SOLUTION

- ⌚ You need to know what the code does
- ⌚ It's like a new project with all it's work (and cost) involved
- ⌚ You need to know the requirements
- ⌚ You need the time to rewrite the code

ONE  
QUESTION  
REMANDS



DS OT USED  
AT ALDOO?



# DEAD CODE...



WHAT IS IT...?

- ☠ Unused Imports, Variables, Methods and Classes

# DEAD CODE...



WHAT IS IT...?

- ☠ Unused Imports, Variables, Methods and Classes
- ☠ Unreachable code

# DEAD CODE...



## WHAT IS IT...?

- ☠ Unused Imports, Variables, Methods and Classes
- ☠ Unreachable code
- ☠ Redundant functions

W H E Y S T A

P R O P R I E T E R Y ?



# WHY IS IT A PROBLEM...?

## REASONS...

- ☠ Reduced Readability and Maintainability

Clutters the codebase, making it harder for developers to read and understand the program's logic



# WHY IS IT A PROBLEM...?

## REASONS...

- ☠ Reduced Readability and Maintainability

Clutters the codebase, making it harder for developers to read and understand the program's logic

- ☠ Increased File Size

Larger source files and compiled binaries, which can impact performance, consume space and increase cost



# WHY IS IT A PROBLEM...?

## REASONS...

- ☠ Reduced Readability and Maintainability

Clutters the codebase, making it harder for developers to read and understand the program's logic

- ☠ Increased File Size

Larger source files and compiled binaries, which can impact performance, consume space and increase cost

- ☠ Confusion for Other Developers

Other developers may spend time trying to understand the purpose of the dead code



# WHY IS IT A PROBLEM...?

## REASONS...

- ☠ Reduced Readability and Maintainability

Clutters the codebase, making it harder for developers to read and understand the program's logic

- ☠ Increased File Size

Larger source files and compiled binaries, which can impact performance, consume space and increase cost

- ☠ Confusion for Other Developers

Other developers may spend time trying to understand the purpose of the dead code

- ☠ Potential for Bugs

Can lead to false positives in code analysis tools and makes debugging harder



# WHY IS IT A PROBLEM...?

## REASONS...

- ☠ Reduced Readability and Maintainability

Clutters the codebase, making it harder for developers to read and understand the program's logic

- ☠ Increased File Size

Larger source files and compiled binaries, which can impact performance, consume space and increase cost

- ☠ Confusion for Other Developers

Other developers may spend time trying to understand the purpose of the dead code

- ☠ Potential for Bugs

Can lead to false positives in code analysis tools and makes debugging harder

- ☠ Increased potential for Security Issues

Adding dependencies always comes with the risk of vulnerable transitive dependencies

WE ARE HERE TO DOES

IT CONVINE

FROM...?

# DEAD CODE...



## WHERE DOES IT COME FROM...?

- ☠ Rapid development and iterations

Strict deadlines can lead developers to leave behind fragments of old code

# DEAD CODE...



## WHERE DOES IT COME FROM...?

- ☠ Rapid development and iterations

Strict deadlines can lead developers to leave behind fragments of old code

- ☠ Hesitance to delete

During debugging phases developers often comment out code rather than removing it

# DEAD CODE...



## WHERE DOES IT COME FROM...?

- ☠ Rapid development and iterations

Strict deadlines can lead developers to leave behind fragments of old code

- ☠ Hesitance to delete

During debugging phases developers often comment out code rather than removing it

- ☠ Incomplete refactoring

Functions/variables may become severed from primary program flow and can persist

# DEAD CODE...



## WHERE DOES IT COME FROM...?

- ☠ Rapid development and iterations

Strict deadlines can lead developers to leave behind fragments of old code

- ☠ Hesitance to delete

During debugging phases developers often comment out code rather than removing it

- ☠ Incomplete refactoring

Functions/variables may become severed from primary program flow and can persist

- ☠ Merging code branches

Can lead to duplicate functions or blocks of code

# DEAD CODE...



## WHERE DOES IT COME FROM...?

- ☠ Rapid development and iterations

Strict deadlines can lead developers to leave behind fragments of old code

- ☠ Hesitance to delete

During debugging phases developers often comment out code rather than removing it

- ☠ Incomplete refactoring

Functions/variables may become severed from primary program flow and can persist

- ☠ Merging code branches

Can lead to duplicate functions or blocks of code

- ☠ Lack of awareness

Large projects make it hard to identify code changes which creates obsolete code sections

HOW TO

REVIEW

IT?



# HOW TO REMOVE IT...?

POSSIBLE SOLUTIONS...

☠ IDE's e.g. IntelliJ IDEA

# DEAD CODE



## DEMO CODE

```
public class Main {  
    private static final String CLASS_NAME = Main.class.getName();  
    private String tmp = "unused";  
  
    static { ... }  
  
    public Main() {...}  
  
    private void method1() { ... }  
    private void method2() { ... }  
    private void method3() { ... } // Unused Method  
    private void method4() { ... }  
  
    public static void main(String[] args) { new Main(); }  
}
```

# DEAD CODE



## INTELLIJ IDEA

- ☠ Select "Code" from menu

# DEAD CODE



## INTELLIJ IDEA

- ☠ Select "Code" from menu

- ☠ Select "Analyze Code"

# DEAD CODE



## INTELLIJ IDEA

- ☠ Select "Code" from menu
- ☠ Select "Analyze Code"
- ☠ Select "Run Inspection by Name"

# DEAD CODE



## INTELLIJ IDEA

- ☠ Select "Code" from menu
- ☠ Select "Analyze Code"
- ☠ Select "Run Inspection by Name"
- ☠ Select "Unused declaration Java|Declaration redundancy"

# FIND UNUSED CODE...



## INTELLIJ IDEA

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under `TmpMod`, including `.gradle`, `.idea`, `build`, `gradle`, `libs`, `logs`, `out`, and `src`.
- Main.java:** The code editor displays the `Main` class with the following code:

```
public class Main {  
    private static final Logger LOGGER = LogManager.getLogger(Main.class);  
    private String tmp = "unused";  
    static {...}  
    public Main() {...}  
    public void method1() {...}  
    public void method2() {...}  
    private void method3() {...} (highlighted)  
    public void method4() {...}  
    public static void main(String[] args) { new Main(); }  
}
```
- Problems View:** Shows inspection results for the file `.../src/main/java/eu/hansolo/tmpmod/Main.java`. The **Inspection Results** section lists two warnings:
  - `method3()` (Method is never used)
  - `tmp` (Reachable. 5 usages found in project code)
- Action Buttons:** Includes `Safe delete`, `Comment out`, and `Add as Entry Point`.
- Bottom Status Bar:** Shows the time as 18:1, file encoding as LF, and other settings.



# HOW TO REMOVE IT...?

## POSSIBLE SOLUTIONS...

- ☠ IDE's e.g. IntelliJ IDEA
- ☠ Sonarqube ide

# DEAD CODE



## SONARQUBE IDE

The screenshot shows the SonarQube IDE's Settings dialog for the SonarLint tool. The left sidebar lists various tools, with 'SonarLint' selected. The main pane shows the 'Rules' tab of the SonarLint settings. A search bar at the top right is set to 'unused'. The Java section contains several rules related to unused code, all of which are checked (indicated by a blue checkmark). The Kotlin section contains fewer rules, also mostly checked. A note at the top states: 'Configure rules used for Sonarlint analysis for projects not in Connected Mode. Connecting your project to SonarQube or SonarCloud syncs SonarLint with the Quality Profile standards defined on the server, allowing you to share the same rules configuration with your team.' A warning message below it says: '⚠ When a project is connected to SonarQube or SonarCloud, configuration from the server applies.'

Tools > SonarLint

Settings File Exclusions Rules About

Configure rules used for Sonarlint analysis for projects not in Connected Mode. Connecting your project to SonarQube or SonarCloud syncs SonarLint with the Quality Profile standards defined on the server, allowing you to share the same rules configuration with your team.

⚠ When a project is connected to SonarQube or SonarCloud, configuration from the server applies.

Q unused

Java

- Intermediate Stream methods should not be left unused
- Unused "private" classes should be removed
- Unused "private" fields should be removed
- Unused "private" methods should be removed
- Unused assignments should be removed
- Unused labels should be removed
- Unused local variables should be removed
- Unused method parameters should be removed
- Unused type parameters should be removed

Kotlin

- Flow intermediate operation results should not be left unused
- Intermediate Sequence and Stream functions should not be left unused
- Unused "private" methods should be removed
- Unused function parameters should be removed
- Unused local variables should be removed

Cancel Apply OK

# DEAD CODE



## SONARQUBE IDE

The screenshot shows the SonarQube IDE interface with the following details:

- Project Structure:** The left sidebar shows a project tree with packages like `eu.hansolo.tmpmod`, `aws`, `benchmark`, `boundrange`, `cache`, `classfileapi`, `clustering`, `css`, `customloglevel`, and `main`.
- Code Editor:** The main editor window displays `Main.java` with the following code:

```
public class Main {  
    //...  
    private static final Logger LOGGER = LogManager.getLogger(Main.class);  
    private String tmp = "unused";  
  
    static {...}  
  
    public Main() {...}  
  
    public void method1() {...}  
  
    public void method2() {...}  
  
    private void method3() {...}  
  
    public void method4() {...}
```
- SonarLint Results:** The bottom left pane shows analysis results:
  - Found 2 issues in 1 file**
  - Main.java (2 issues)**
    - (49, 17) Remove this unused private "method3" method. 3 minutes ago
    - (12, 19) Remove this unused "tmp" private field. 1 hour ago
- Bottom Status Bar:** Shows "Automatic analysis is enabled", "SonarLint suggestions: Detect issues in your whole project // Try SonarCloud for free // Download SonarQ... (3 minutes ago)", and various file and encoding settings.



# WHERE TO GET IT...?

SONARQUBE IDE



<https://sonarsource/products/sonarlint>

azul



# HOW TO REMOVE IT...?

## POSSIBLE SOLUTIONS...

- ☠ IDE's e.g. IntelliJ IDEA

- ☠ Sonarqube ide

- ☠ OpenRewrite

# OPEN REWRITER



WHAT IS IT...?

- ☠ A community driven open source project

# OPEN REWRITE



## WHAT IS IT...?

- ☠ A community driven open source project
- ☠ Enables developers to effectively eliminate technical debt



# OPEN REWRITE

## WHAT IS IT...?

- ☠ A community driven open source project
- ☠ Enables developers to effectively eliminate technical debt
- ☠ Contains an auto-refactoring engine



# OPEN REWRITER

## WHAT IS IT...?

- ☠ A community driven open source project
- ☠ Enables developers to effectively eliminate technical debt
- ☠ Contains an auto-refactoring engine
- ☠ Has 'Recipes' to clean up code  
(e.g. to remove unused variables, methods etc.)

# DEAD CODE



## DEMO CODE

```
public class Main {  
    private static final String CLASS_NAME = Main.class.getName();  
    private String tmp = "unused";  
  
    static { ... }  
  
    public Main() {...}  
  
    private void method1() { ... }  
    private void method2() { ... }  
    private void method3() { ... } // Unused Method  
    private void method4() { ... }  
  
    public static void main(String[] args) { new Main(); }  
}
```

# DEAD CODE



## ADD RECIPE TO PROJECT POM FILE

```
...
<build>
  <plugins>
    <plugin>
      <groupId>org.openrewrite.maven</groupId>
      <artifactId>rewrite-maven-plugin</artifactId>
      <version>6.0.0</version>
      <configuration>
        <exportDatatables>true</exportDatatables>
        <activeRecipes>
          <recipe>org.openrewrite.staticanalysis.RemoveUnusedPrivateMethods</recipe>
        </activeRecipes>
      </configuration>
      <dependencies>
        <dependency>
          <groupId>org.openrewrite.recipe</groupId>
          <artifactId>rewrite-static-analysis</artifactId>
          <version>2.0.0</version>
        </dependency>
      </dependencies>
    </plugin>
  </plugins>
</build>
...
```

# DEAD CODE



## EXECUTE MVN TASK

mvn rewrite:run

# DEAD CODE



## DEMO CODE

```
public class Main {  
    private static final String CLASS_NAME = Main.class.getName();  
    private String tmp = "unused";  
  
    static { ... }  
  
    public Main() {...}  
  
    private void method1() { ... }  
    private void method2() { ... }  
    private void method4() { ... }  
  
    public static void main(String[] args) { new Main(); }  
}
```

# WHERE TO GET IT...?



OPEN REWRITE

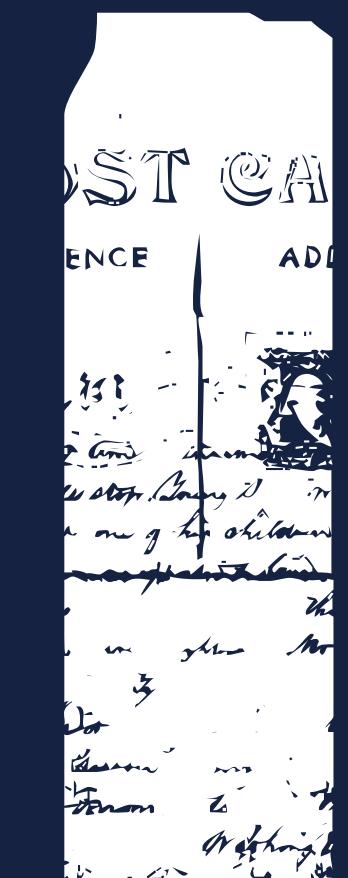
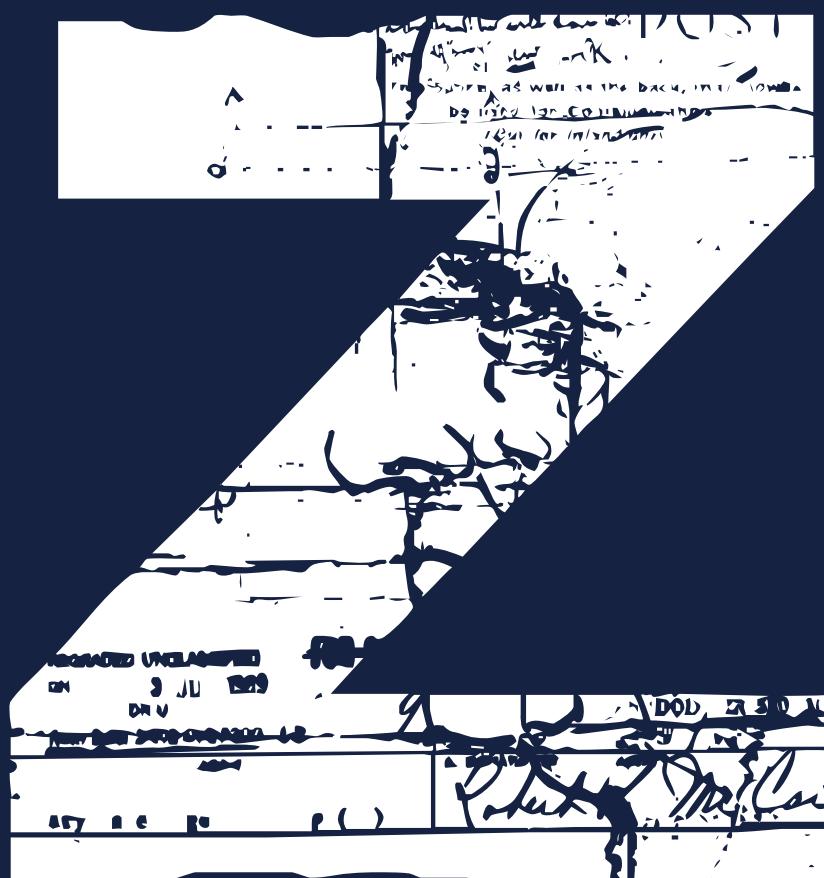


<https://github.com/openrewrite>

azul

BOUTIQUE

ABOUT



azul

M A N T A N E D U

C O M

EXHIBITION  
NEW YORK  
EXHIBITION  
NEW YORK  
EXHIBITION  
NEW YORK  
EXHIBITION  
NEW YORK

HOW TO FIND IT?

CONVYING  
ROUTINING  
SYSTEM

# ZOMBIE CODE



## HOW TO FIND IT...?

- ☠ Tombstone logging using static initialisers and custom log levels

# ZOMBIE CODE



## LOG4J CONFIGURATION

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn">

    <CustomLevels>
        <CustomLevel name="USED_CLASS" intLevel="550" />
        <CustomLevel name="USED_METHOD" intLevel="551" />
    </CustomLevels>

</Configuration>
```



# ZOMBIE CODE

## LOG4J CONFIGURATION

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn">

    <CustomLevels>
        <CustomLevel name="USED_CLASS" intLevel="550" />
        <CustomLevel name="USED_METHOD" intLevel="551" />
    </CustomLevels>

    <Appenders>
        <Console name="console" target="SYSTEM_OUT">
            <PatternLayout pattern="%d %-7level %logger{36} %msg%n"/>
        </Console>
        <File name="logfile" fileName="logs/used_code.log">
            <PatternLayout pattern="%d %-7level %logger{36} %msg%n"/>
        </File>
    </Appenders>

</Configuration>
```

# ZOMBIE CODE



## LOG4J CONFIGURATION

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn">

    <CustomLevels>
        <CustomLevel name="USED_CLASS" intLevel="550" />
        <CustomLevel name="USED_METHOD" intLevel="551" />
    </CustomLevels>

    <Appenders>
        <Console name="console" target="SYSTEM_OUT">
            <PatternLayout pattern="%d %-7level %logger{36} %msg%n"/>
        </Console>
        <File name="logfile" fileName="logs/used_code.log">
            <PatternLayout pattern="%d %-7level %logger{36} %msg%n"/>
        </File>
    </Appenders>

    <Loggers>
        <Root level="used_method"> <!-- Can be used to define class or method level logging -->
            <AppenderRef ref="console" level="used_method" />
            <AppenderRef ref="logfile" level="used_method" />
        </Root>
    </Loggers>
</Configuration>
```

# ZOMBIE CODE



## DEMO CODE

```
public class Main {  
    private static final Logger LOGGER = LogManager.getLogger(Main.class);
```

# ZOMBIE CODE



## DEMO CODE

```
public class Main {  
    private static final Logger LOGGER = LogManager.getLogger(Main.class);  
  
    static {  
        LOGGER.log(Level.valueOf("USED_CLASS"), "");  
    }  
}
```

# ZOMBIE CODE



## DEMO CODE

```
public class Main {  
    private static final Logger LOGGER = LogManager.getLogger(Main.class);  
  
    static {  
        LOGGER.log(Level.valueOf("USED_CLASS"), "");  
    }  
  
    public Main() {  
        for (int i = 0 ; i < 3 ; i++) {  
            switch (i) {  
                case 0 -> method1();  
                case 1 -> method2();  
                case 2 -> method4();  
            }  
            try { new Thread().sleep(3000); } catch (InterruptedException e) {}  
        }  
    }  
}
```

# ZOMBIE CODE



## DEMO CODE

```
public class Main {  
    private static final Logger LOGGER = LogManager.getLogger(Main.class);  
  
    static {  
        LOGGER.log(Level.valueOf("USED_CLASS"), "");  
    }  
  
    public Main() {...}  
  
    public void method1() { LOGGER.log(Level.valueOf("USED_METHOD"), "method1()"); }  
    public void method2() { LOGGER.log(Level.valueOf("USED_METHOD"), "method2()"); }  
    public void method3() { LOGGER.log(Level.valueOf("USED_METHOD"), "method3()"); } // Unused method  
    public void method4() { LOGGER.log(Level.valueOf("USED_METHOD"), "method4()"); }  
  
    public static void main(String[] args) {  
        new Main();  
    }  
}
```

# ZOMBIE CODE



## LOGFILE CONTENT OF DEMOCODE

```
2024-11-11 11:27:06,068 USED_CLASS eu.hansolo.tmpmod.customloglevel.Main
2024-11-11 11:27:06,069 USED_METHOD eu.hansolo.tmpmod.customloglevel.Main method1()
2024-11-11 11:27:11,075 USED_METHOD eu.hansolo.tmpmod.customloglevel.Main method2()
2024-11-11 11:27:16,084 USED_METHOD eu.hansolo.tmpmod.customloglevel.Main method4()
```

No output from method3()

# ZOMBIE CODE



## TOMBSTONE LOGGING PROBLEMS

- ☠ You only get the used classes and methods
- ☠ You need to find a way to find the unused classes and methods

# ZOMBIE CODE



## HOW TO FIND IT...?

- ☠ Tombstone logging using static initialisers and custom log levels
- ☠ Java Flight Recorder and Mission Control



# ZOMBIE CODE

## CUSTOM JFR EVENTS

```
public class UsedClassEvent extends jdk.jfr.Event {  
    final String className;  
  
    public UsedClassEvent(final String className) {  
        this.className = className;  
    }  
}
```



# ZOMBIE CODE

## CUSTOM JFR EVENTS

```
public class UsedClassEvent extends jdk.jfr.Event {  
    final String className;  
  
    public UsedClassEvent(final String className) {  
        this.className = className;  
    }  
}
```

```
public class UsedMethodEvent extends jdk.jfr.Event {  
    final String methodName;  
  
    public UsedMethodEvent(final String methodName) {  
        this.methodName = methodName;  
    }  
}
```

# ZOMBIE CODE



## DEMO CODE

```
public class Main {  
    private static final String CLASS_NAME = Main.class.getName();  
  
    static {  
        new UsedClassEvent(CLASS_NAME).commit();  
    }  
  
    public Main() {...}  
  
    public void method1() { new UsedMethodEvent(String.join("|", CLASS_NAME, "method1()")).commit(); }  
    public void method2() { new UsedMethodEvent(String.join("|", CLASS_NAME, "method2()")).commit(); }  
    public void method3() { new UsedMethodEvent(String.join("|", CLASS_NAME, "method3()")).commit(); }  
    public void method4() { new UsedMethodEvent(String.join("|", CLASS_NAME, "method4()")).commit(); }  
  
    public static void main(String[] args) {  
        new Main();  
    }  
}
```

# ZOMBIE CODE



## MISSION CONTROL

Azul Mission Control

JVM Br... Outline Event Browser

Automated Analysis Results

- > Java Application
- > JVM Internals
- > Environment
- > Event Browser

Event Types Tree

Search the tree

- > Flight Recorder 722
- > Java Application 106
- > Java Development Kit 96
- > Java Virtual Machine 5.063
- > Operating System 1.384
- > Uncategorized 4
  - eu.hansolo.tmpmod.flightracer.UsedClassEvent 1
  - eu.hansolo.tmpmod.flightracer.UsedMethodEvent 3

Start Time Duration End Time Event Thread Event Type

Start Time	Duration	End Time	Event Thread	Event Type
12/11/2024, 08:31:58.164	0 s	12/11/2024, 08:31:58.164	main	eu.hansolo.tmpmod.flightracer.UsedMethodEvent
12/11/2024, 08:32:00.167	0 s	12/11/2024, 08:32:00.167	main	eu.hansolo.tmpmod.flightracer.UsedMethodEvent
12/11/2024, 08:32:02.173	0 s	12/11/2024, 08:32:02.173	main	eu.hansolo.tmpmod.flightracer.UsedMethodEvent
12/11/2024, 08:31:58.163	2,292 µs	12/11/2024, 08:31:58.163	main	eu.hansolo.tmpmod.flightracer.UsedClassEvent

Pro... Res... Stack Trace Flame Graph Samples

Field Value

Field	Value
Event Type	[eu.hansolo.tmpmod.flightracer.UsedMethodEvent]
Start Time	12/11/2024, 08:31:58.164
Duration	0 s - 2,292 µs
End Time	12/11/2024, 08:31:58.164
Event Thread	main
4 events	

Stack Trace

Stack Trace	Samples	Percentage
void eu.hansolo.tmpmod.flightracer.Main.method2()	1	25 %
void eu.hansolo.tmpmod.flightracer.Main.<init>()	1	25 %
void eu.hansolo.tmpmod.flightracer.Main.main(String[])	1	25 %

# ZOMBIE CODE



## MISSION CONTROL

The screenshot shows the Azul Mission Control interface with the following details:

- Title Bar:** Azul Mission Control
- Toolbar:** Includes icons for JVM Browser, Outline, Event Browser, and other tools.
- Event Browser:** Focus: <No Selection>, Aspect: <No Selection>. Shows an event tree with categories like Flight Recorder, Java Application, Java Development Kit, Java Virtual Machine, Operating System, and Uncategorized. A specific event is selected: `eu.hansolo.tmpmod.flightracer.UsedClassEvent 1`.
- Table View:** Displays event details:

Start Time	Duration	End Time	Event Thread	className
12/11/2024, 08:31:58.163	2,292 µs	12/11/2024, 08:31:58.163	main	eu.hansolo.tmpmod.flightracer.Main
- Stack Trace:** Shows a single sample with 100% percentage, corresponding to the selected event:

Stack Trace	Samples	Percentage
void eu.hansolo.tmpmod.flightracer.Main.<clinit>()	1	100 %
- Bottom Navigation:** Includes icons for Samples, Flame Graph, and other navigation options.

# ZOMBIE CODE



## MISSION CONTROL

Azul Mission Control

JVM Br... Outline Event Browser

Automated Analysis Results

- > Java Application
- > JVM Internals
- > Environment
- Event Browser

Focus: <No Selection> Aspect: <No Selection>

Show concurrent: Contained Same threads

Event Types Tree

- Search the tree
- > Flight Recorder 722
- > Java Application 106
- > Java Development Kit 96
- > Java Virtual Machine 5.063
- > Operating System 1.384
- > Uncategorized 4
  - eu.hansolo.tmpmod.flightracer.UsedClassEvent 1
  - eu.hansolo.tmpmod.flightracer.UsedMethodEvent 3

Start Time	Duration	End Time	Event Thread	methodName
12/11/2024, 08:31:58.164	0 s	12/11/2024, 08:31:58.164	main	eu.hansolo.tmpmod.flightracer.Main method1()
12/11/2024, 08:32:00.167	0 s	12/11/2024, 08:32:00.167	main	eu.hansolo.tmpmod.flightracer.Main method2()
12/11/2024, 08:32:02.173	0 s	12/11/2024, 08:32:02.173	main	eu.hansolo.tmpmod.flightracer.Main method4()

Pro... Res... Field Value

Field	Value
Event Type	eu.hansolo.tmp
Start Time	12/11/2024, 08:
Duration	0 s
End Time	12/11/2024, 08:
Event Thread	main
methodName	[eu.hansolo.tm 3 events

Stack Trace Flame Graph

Stack Trace

Stack Trace	Samples	Percentage
void eu.hansolo.tmpmod.flightracer.Main.method2()	1	33,3 %
void eu.hansolo.tmpmod.flightracer.Main.<init>()	1	33,3 %
void eu.hansolo.tmpmod.flightracer.Main.main(String[])	1	33,3 %

Samples

# ZOMBIE CODE



## JFR + MISSION CONTROL PROBLEMS

- ☠ You only get the used classes and methods
- ☠ You need to find a way to find the unused classes and methods

GOALS

azul

**BUT A LOT OF  
WORK!**

OUT OF THE PARK

SOLUTIONS

# OTHER SOLUTIONS...

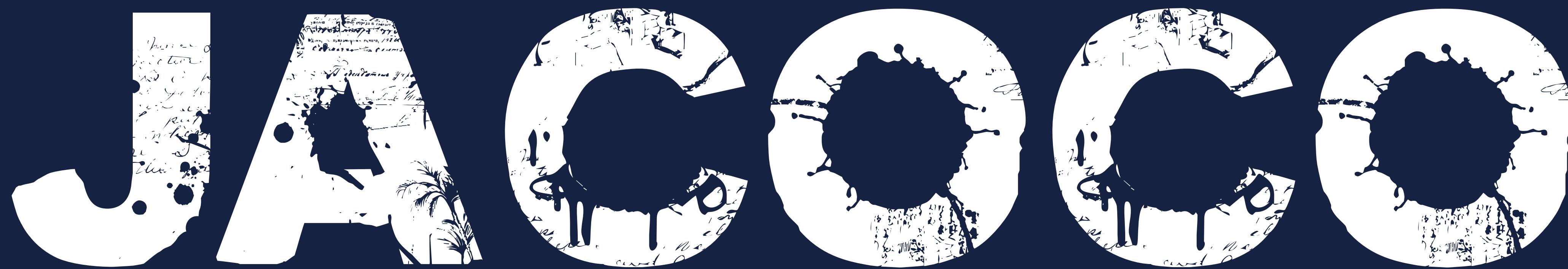
## HOW TO FIND ZOMBIE CODE...?

- ☠ Java Code Coverage (JaCoCo)

# OTHER SOLUTIONS...

## HOW TO FIND ZOMBIE CODE...?

- ☠ Java Code Coverage (JaCoCo)
- ☠ Azul Intelligence Cloud



Java Code Coverage



## WHAT IS IT...?

- ☠ A free tool to report code coverage on automated test suites



## WHAT IS IT...?

- ☠ A free tool to report code coverage on automated test suites
- ☠ Collects coverage metrics through a Java agent when the class loader loads classes



## WHAT IS IT...?

- ☠ A free tool to report code coverage on automated test suites
- ☠ Collects coverage metrics through a Java agent when the class loader loads classes
- ☠ Usually used in development



## WHAT IS IT...?

- ☠ A free tool to report code coverage on automated test suites
- ☠ Collects coverage metrics through a Java agent when the class loader loads classes
- ☠ Usually used in development
- ☠ Can be used to detect dead code

# JACOCO



FIND YOUR ZONE IMPERFECT

The word "FIND" is composed of two torn pieces of paper with a postcard-like background. The first piece shows a handprint, and the second piece shows a stamp.

The word "YOUR" is a circular graphic featuring a dark, central splash or hole surrounded by a textured, torn paper border.

The word "ZONE" is composed of two torn pieces of paper with a postcard-like background. The first piece shows a handprint, and the second piece shows a stamp.

The word "IMPERFECT" is composed of two torn pieces of paper with a postcard-like background. The first piece shows a handprint, and the second piece shows a stamp.

azul

# HOW DOES IT WORK...?



JACOCO



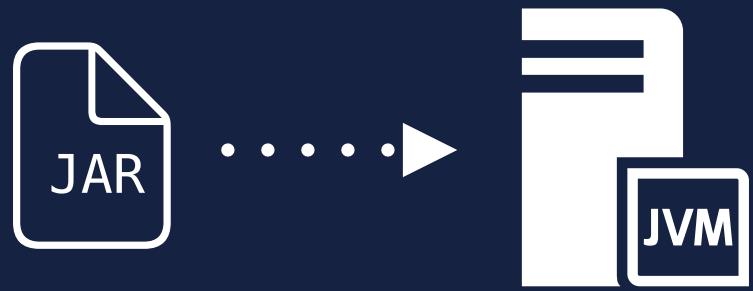
Application

azul

# HOW DOES IT WORK...?



JACOCO



Application

JVMs run  
Application

# HOW DOES IT WORK...?



## JACOCO



Application

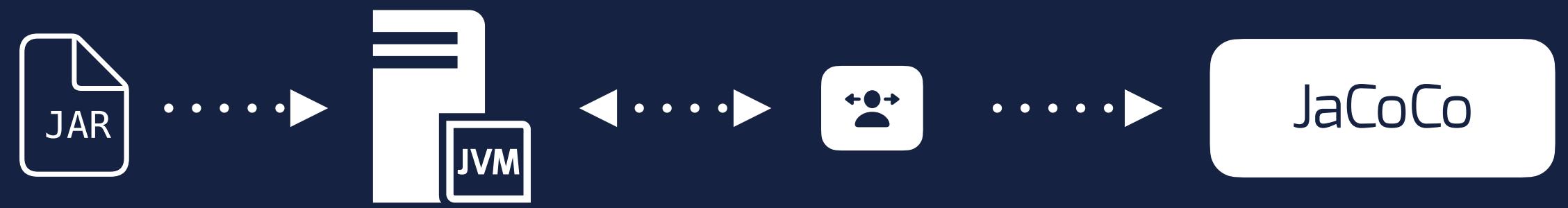
JVMs run  
Application

Agent  
configured for  
spec. pkgs to  
reduce overhead

# HOW DOES IT WORK...?



## JACOCO



Application

JVMs run  
Application

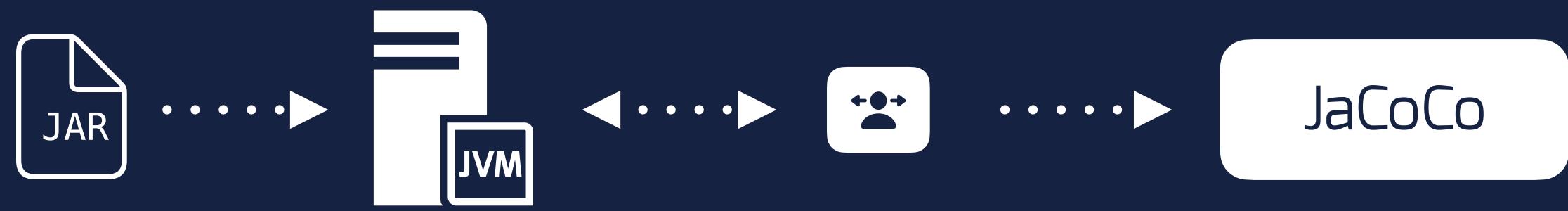
Agent

azul

# HOW DOES IT WORK...?



## JACOCO



Application

JVMs run  
Application

Agent



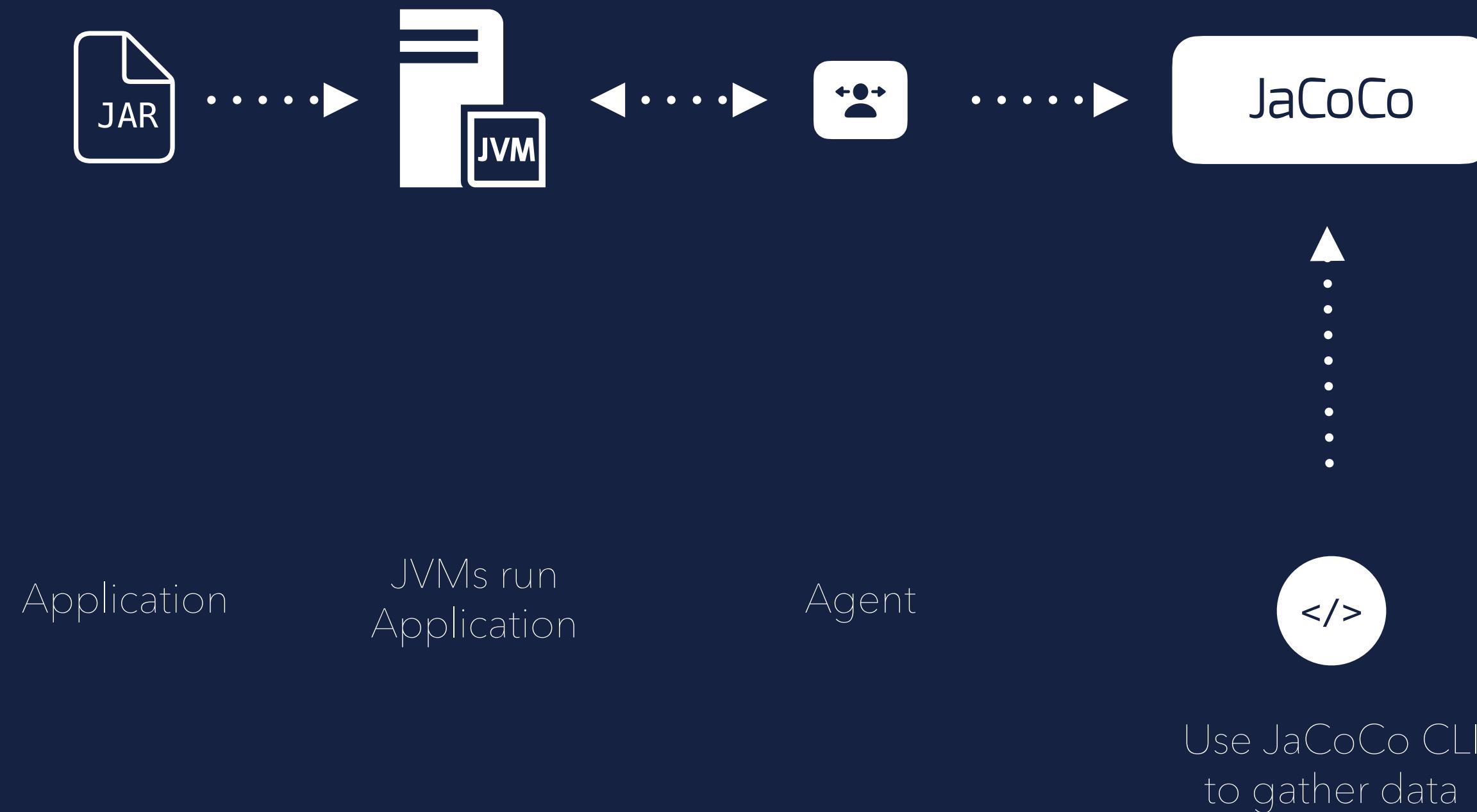
RUN IT FOR SOME TIME...

azul

# HOW DOES IT WORK...?



## JACOCO



# HOW DOES IT WORK...?



## JACOCO



Application

JVMs run  
Application

Agent

Data  
File

azul

# HOW DOES IT WORK...?



## JACOCO



Application

JVMs run  
Application

Agent

</>  
Pass .class files  
and .java files  
to JaCoCo

Data  
File

azul

# HOW DOES IT WORK...?



## JACOCO



Application

JVMs run  
Application

Agent

Data  
File

Report

azul

# HOW DOES IT WORK...?



## JACOCO



# INTERESTING BLOGPOST



BY PICNIC

- ☠ Run JaCoCo in production
- ☠ Run it within Kubernetes
- ☠ Use JaCoCo CLI and scripts to aggregate data
- ☠ Create an html report



<https://tinyurl.com/4f6m3u5x>

# WHERE TO GET IT...?



JACOCO



<https://github.com/jacoco/jacoco>

azul

ANTITERROR  
INTELLIGENCE  
CLOUD



# INTELLIGENCE CLOUD



## WHAT IS IT ?

- Cloud service, collecting data from Java Virtual Machine(s)

# INTELLIGENCE CLOUD



## WHAT IS IT ?

- Cloud service, collecting data from Java Virtual Machine(s)
- Collect information by using an Agent for any OpenJDK JVM

# INTELLIGENCE CLOUD



## WHAT IS IT ?

- Cloud service, collecting data from Java Virtual Machine(s)
- Collect information by using an Agent for any OpenJDK JVM
- Offers information about used/unused code in production

# INTELLIGENCE CLOUD



FIND ZONE IMPACT

CODE DUE

# INTELLIGENCE CLOUD



## FINDING ZOMBIE CODE...



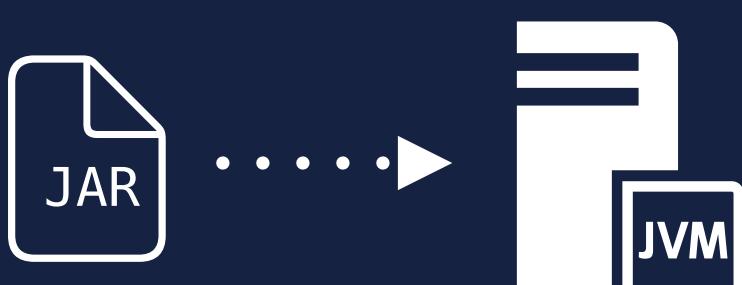
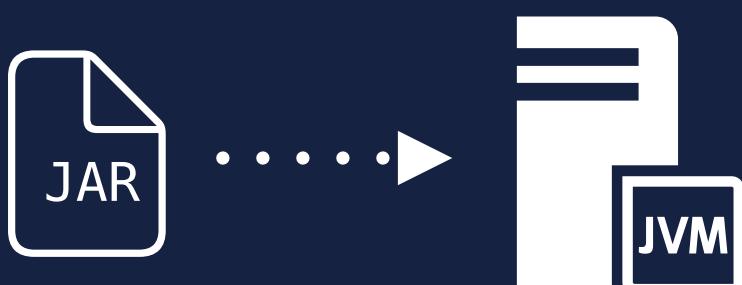
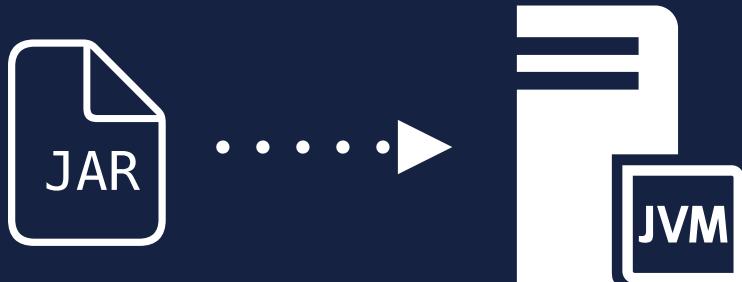
JAR

Application

# INTELLIGENCE CLOUD



## FINDING ZOMBIE CODE...



Application

JVMs run  
Application

# INTELLIGENCE CLOUD



## FINDING ZOMBIE CODE...



Application

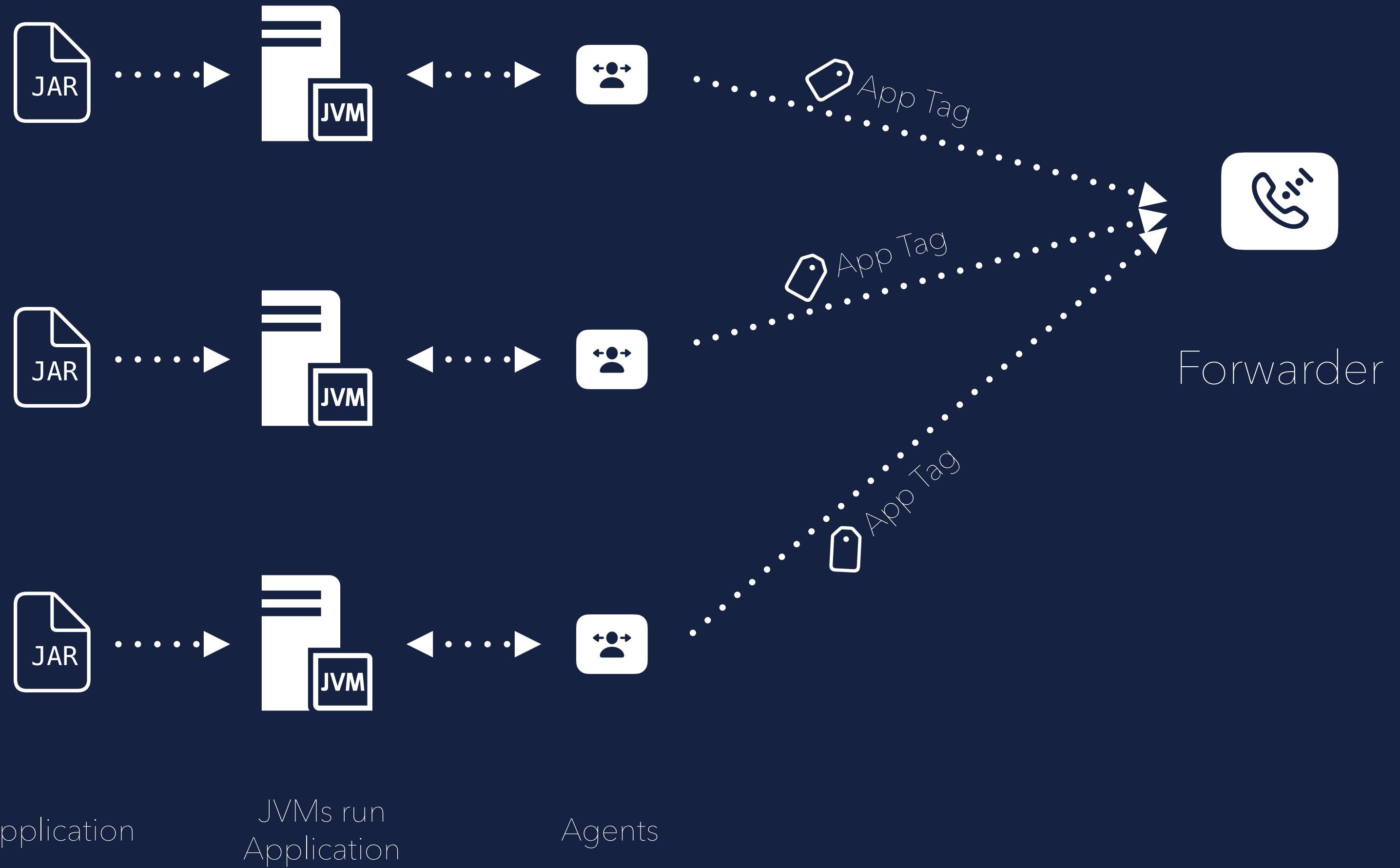
JVMs run  
Application

Agents

# INTELLIGENCE CLOUD



## FINDING ZOMBIE CODE...



# INTELLIGENCE CLOUD



## FINDING ZOMBIE CODE...



# INTELLIGENCE CLOUD



## FINDING ZOMBIE CODE...



# INTELLIGENCE CLOUD



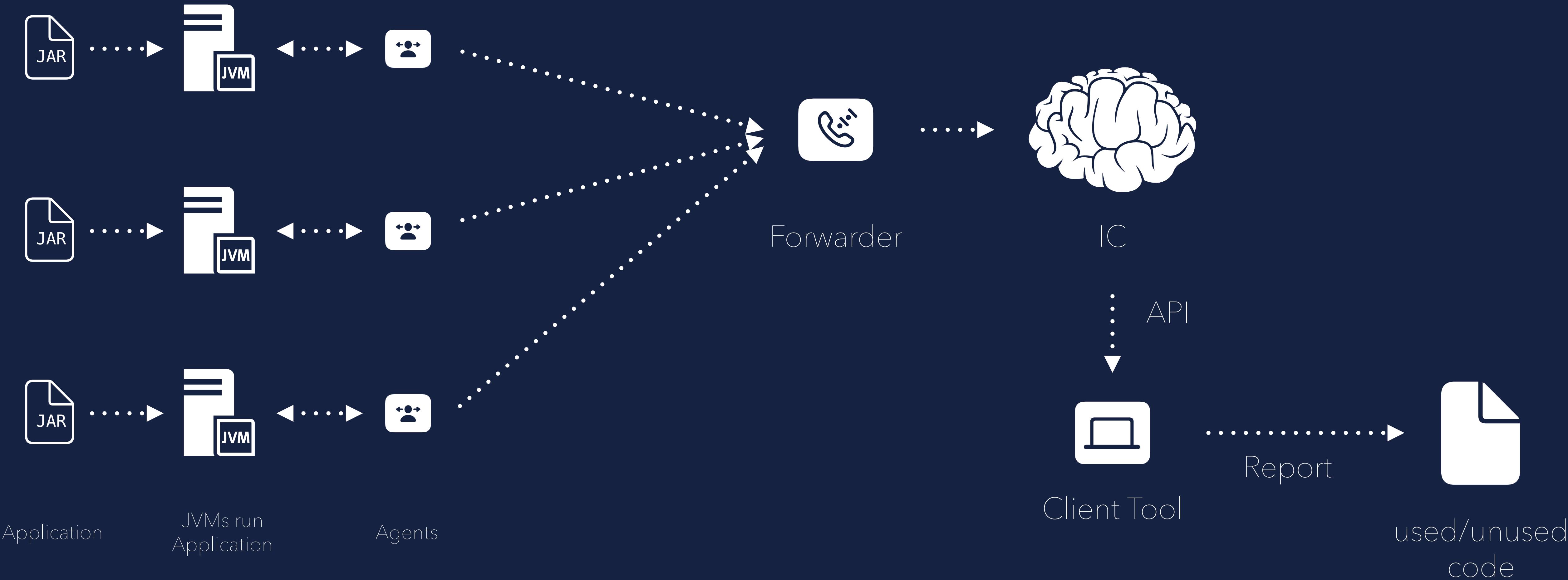
## FINDING ZOMBIE CODE...



# INTELLIGENCE CLOUD



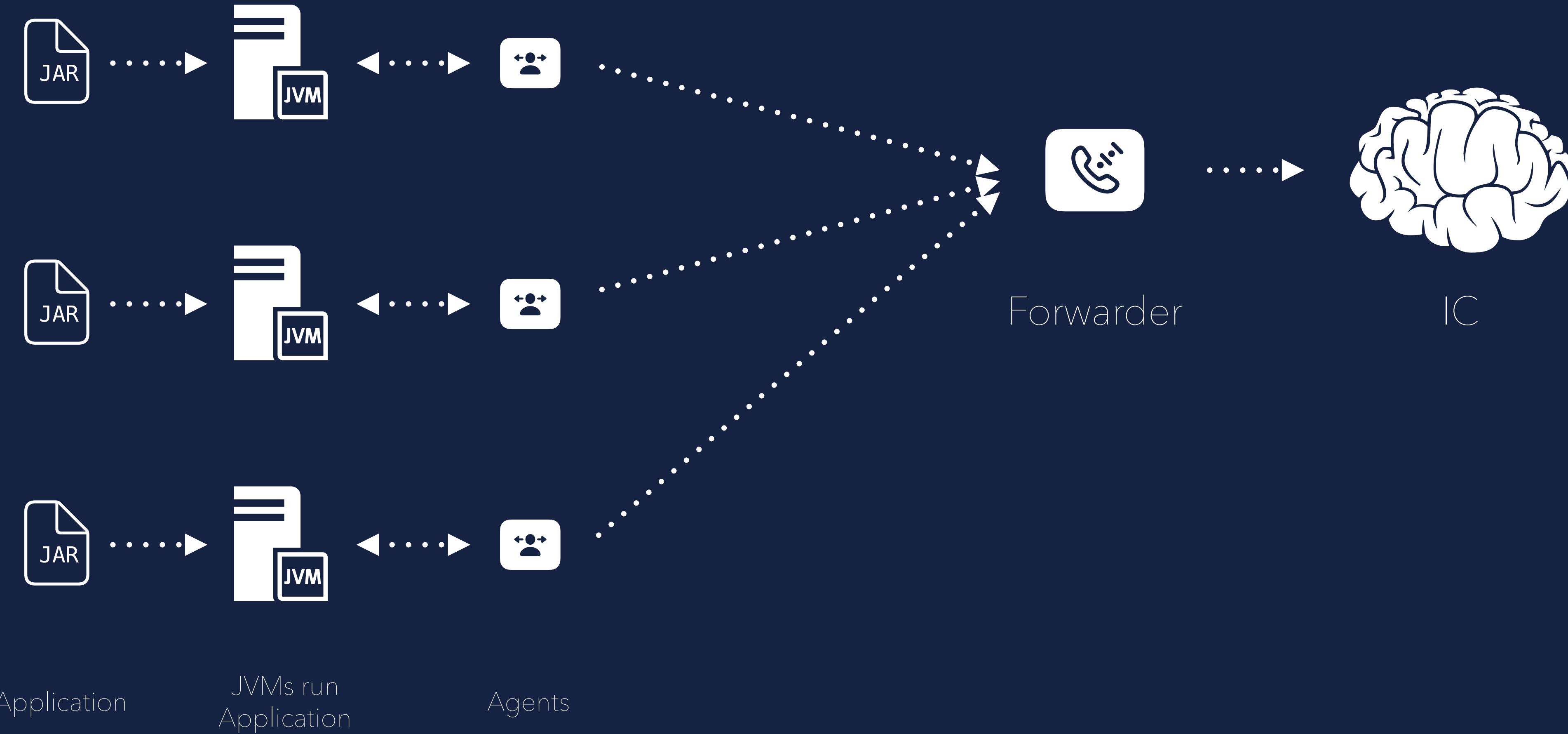
## FINDING ZOMBIE CODE...



# INTELLIGENCE CLOUD



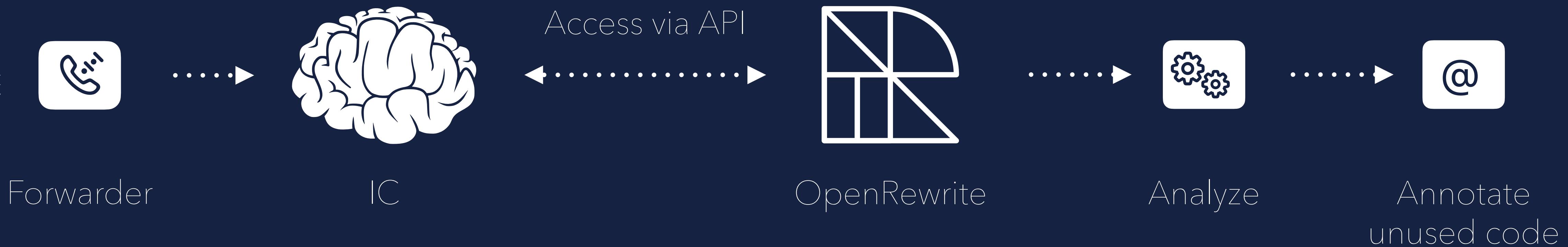
## FINDING ZOMBIE CODE...



# INTELLIGENCE CLOUD



## FINDING ZOMBIE CODE...



### Step 1

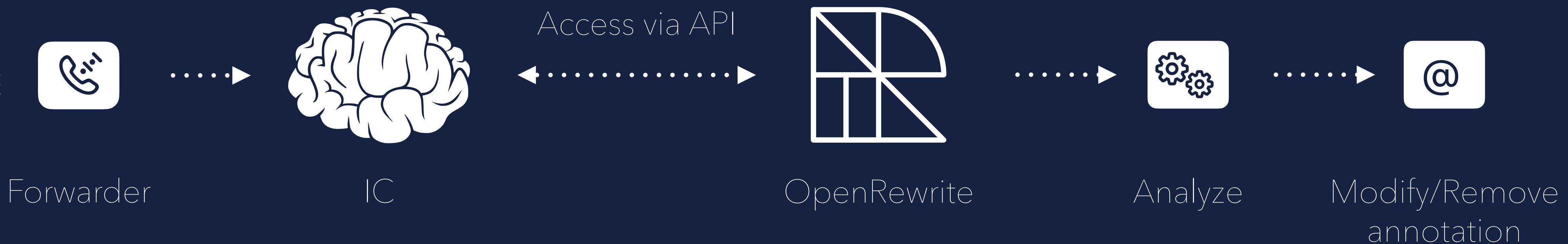
(e.g. run after 6 months)

```
@Deprecated(since="codeInventory_20250101, forRemoval=false)
```

# INTELLIGENCE CLOUD



## FINDING ZOMBIE CODE...



## Step 2

(e.g. run after 3-6 months)

`@Deprecated(since="codeInventory_20250101, forRemoval=true)`

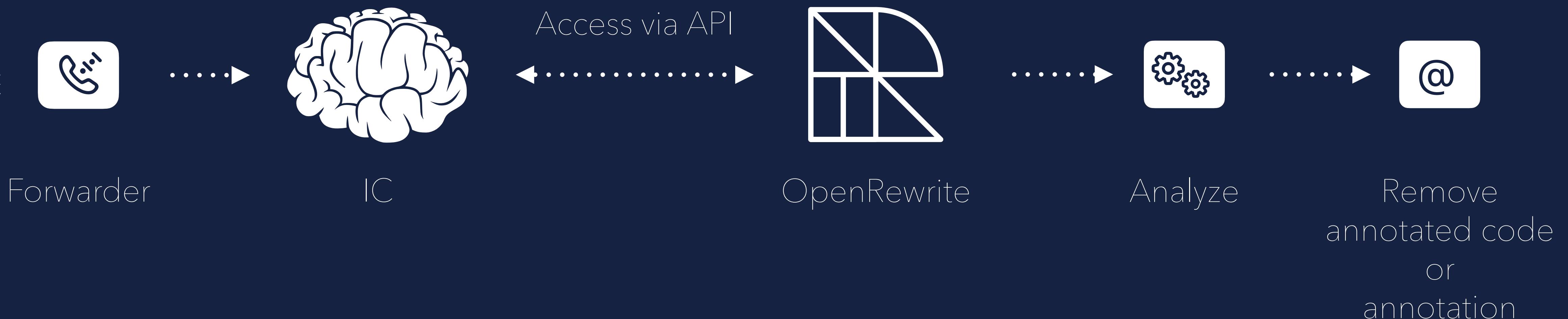
or

remove annotation if code was called in between

# INTELLIGENCE CLOUD



## FINDING ZOMBIE CODE...



### Step 3

(e.g. run after 3 months)

Remove annotated code  
or  
remove annotation if code was called in between

# INTELLIGENCE CLOUD



## WHAT'S NEW

## YOU OUGHT



azul



# WHAT YOU GET...

## AZUL INTELLIGENCE CLOUD

- ☠ Ability to find used/unused code in your running application



# WHAT YOU GET...

## AZUL INTELLIGENCE CLOUD

- ☠ Ability to find used/unused code in your running application
- ☠ Create reports at any time for specified time frames



# WHAT YOU GET...

## AZUL INTELLIGENCE CLOUD

- ☠ Ability to find used/unused code in your running application
- ☠ Create reports at any time for specified time frames
- ☠ Run app on multiple JVMs at the same time using tags



# WHAT YOU GET...

## AZUL INTELLIGENCE CLOUD

- ☠ Ability to find used/unused code in your running application
- ☠ Create reports at any time for specified time frames
- ☠ Run app on multiple JVMs at the same time using tags
- ☠ Offers an API for integration with your own tools



# WHAT YOU GET...

## AZUL INTELLIGENCE CLOUD

- ☠ Ability to find used/unused code in your running application
- ☠ Create reports at any time for specified time frames
- ☠ Run app on multiple JVMs at the same time using tags
- ☠ Offers an API for integration with your own tools
- ☠ Annotate unused code automatically using OpenRewrite



# WHAT YOU GET...

## AZUL INTELLIGENCE CLOUD

- ☠ Ability to find used/unused code in your running application
- ☠ Create reports at any time for specified time frames
- ☠ Run app on multiple JVMs at the same time using tags
- ☠ Offers an API for integration with your own tools
- ☠ Annotate unused code automatically using OpenRewrite
- ☠ Remove unused code automatically using OpenRewrite

# WHERE TO GET IT...?



## AZUL INTELLIGENCE CLOUD



[https://www\\_azul\\_com\\_products\\_intelligence-cloud/](https://www_azul_com_products_intelligence-cloud/)

MORE INFO

azul

CONCUSSION



# CONCLUSION

- ☠ Zombie code is real, but hard to find



# CONCLUSION

- ☠ Zombie code is real, but hard to find
- ☠ Monitoring takes time



# CONCLUSION

- ☠️ Zombie code is real, but hard to find
- ☠️ Monitoring takes time
- ☠️ Setup and reporting is challenging



# CONCLUSION

- ☠️ Zombie code is real, but hard to find
- ☠️ Monitoring takes time
- ☠️ Setup and reporting is challenging
- ☠️ Document your code, refactor regularly and communicate



# CONCLUSION

- ☠️ Zombie code is real, but hard to find
- ☠️ Monitoring takes time
- ☠️ Setup and reporting is challenging
- ☠️ Document your code, refactor regularly and communicate
- ☠️ Tools like JaCoCo or OpenRewrite + Intelligence Cloud can help

THANK

YOU