



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

DOUGH FAVOURITE
WITH THE
COUPON

WITH THE
COUPON

COUPON

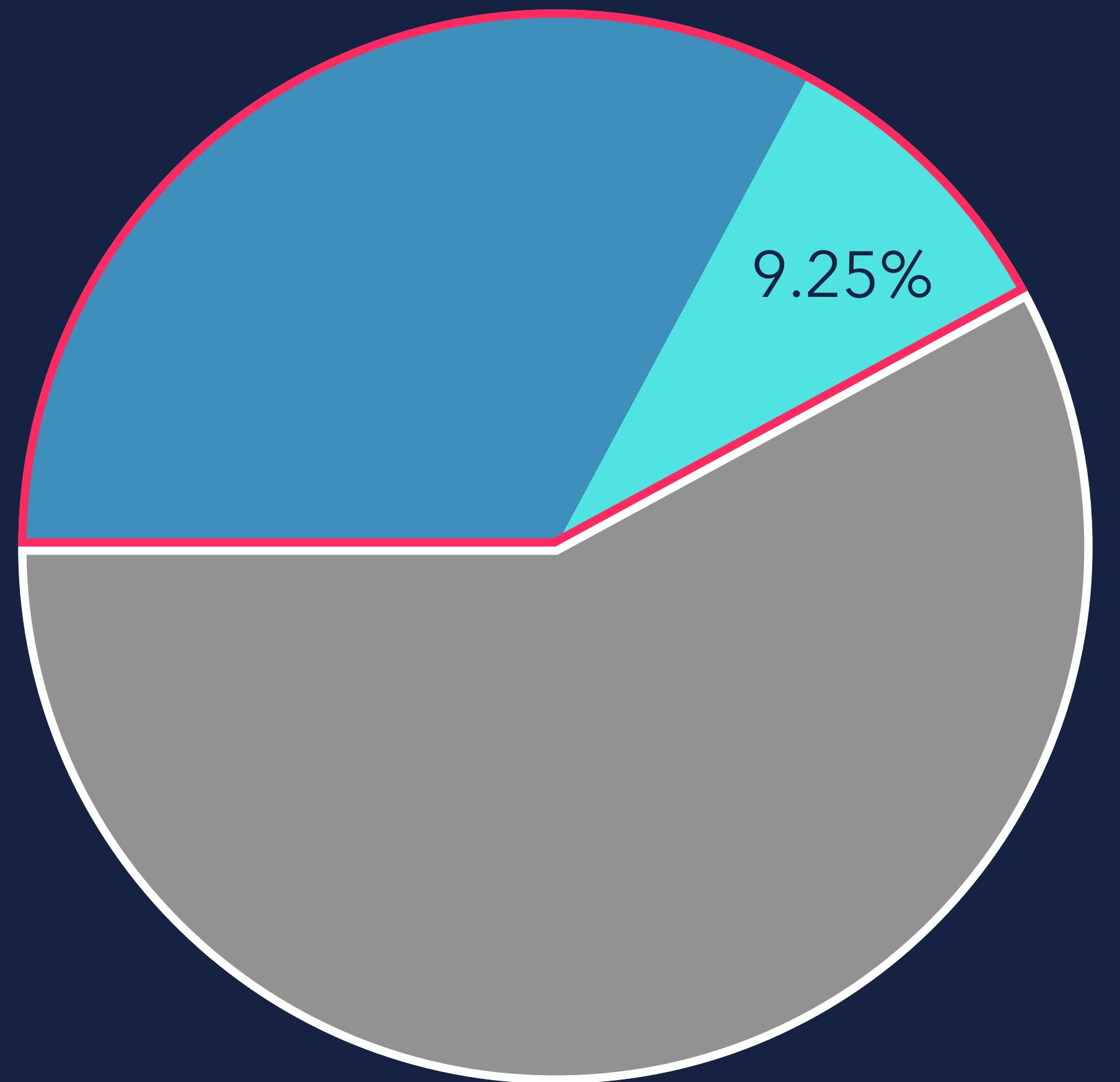
azul

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

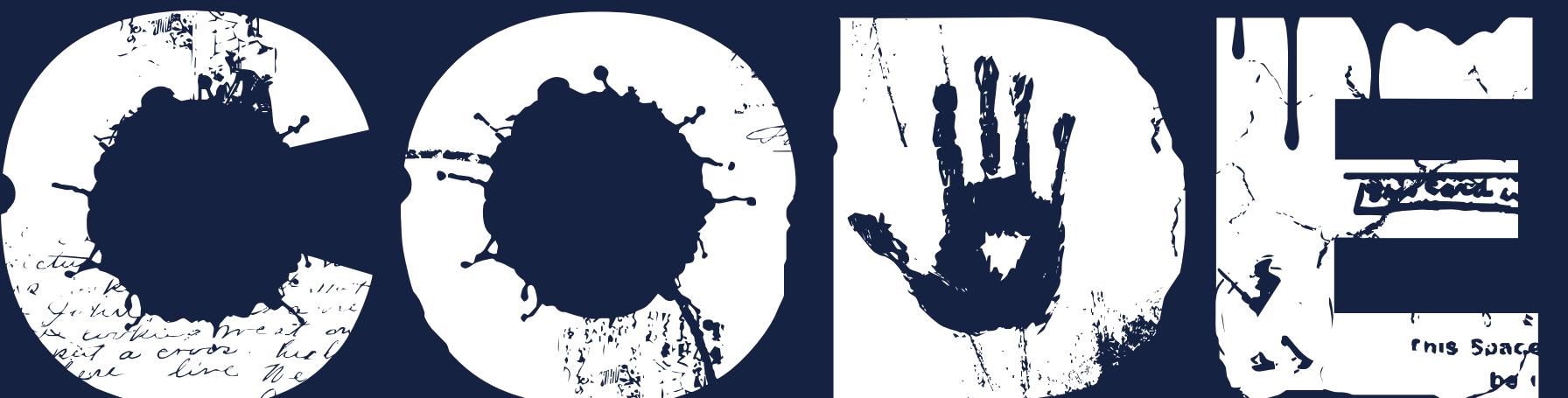
azul

Two types of
paper come
in two types of
ink.

It's Good
And It's
Good For
St. Joe Valley
Creamery Co.
Fresh Dairy
Products
Distributed by The Olden Company
Denton, Texas
Barrett, P.T.
Joseph N.
K & S
DAIRY

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

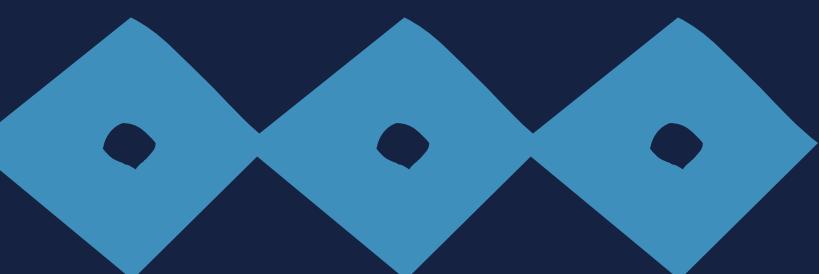
REWRITE



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.

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

Restore Defaults

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

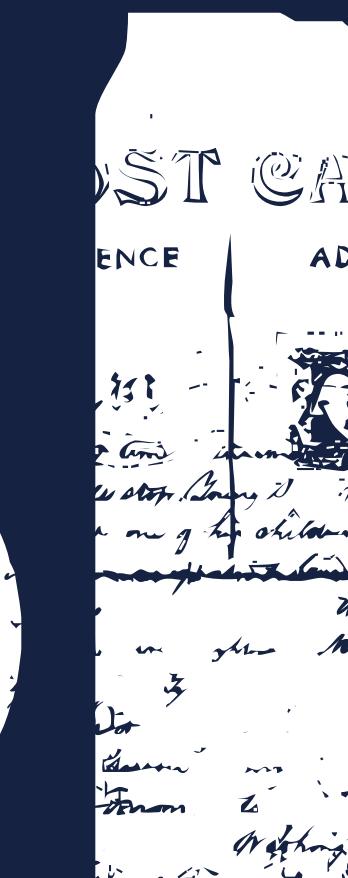
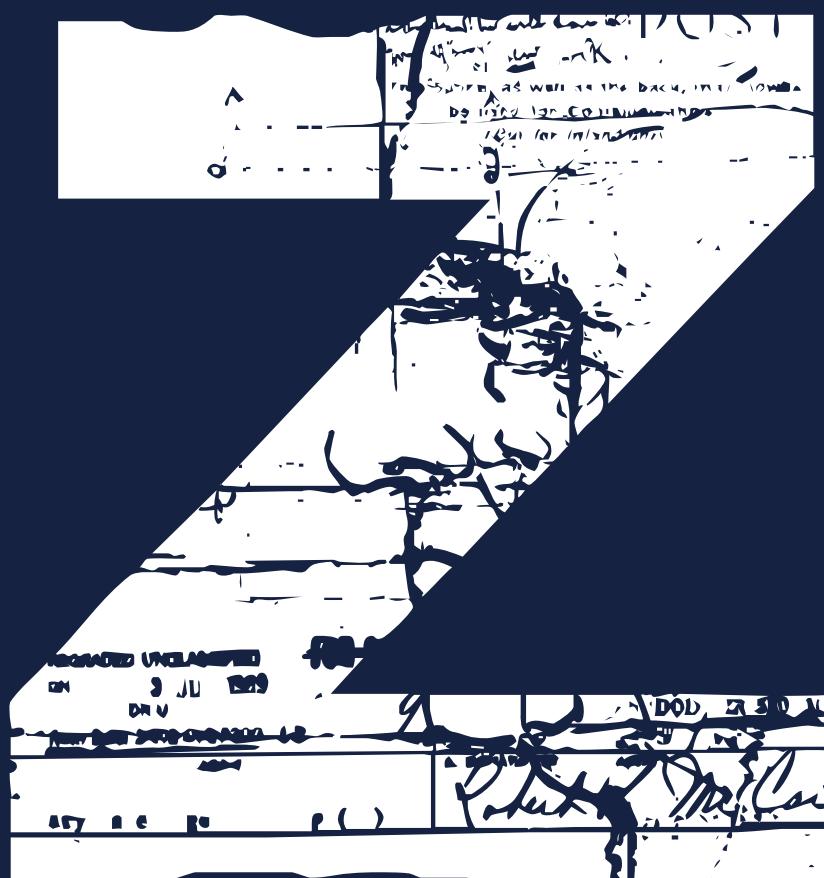


<https://github.com/openrewrite>

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

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

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 Samples Percentage

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 %

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...

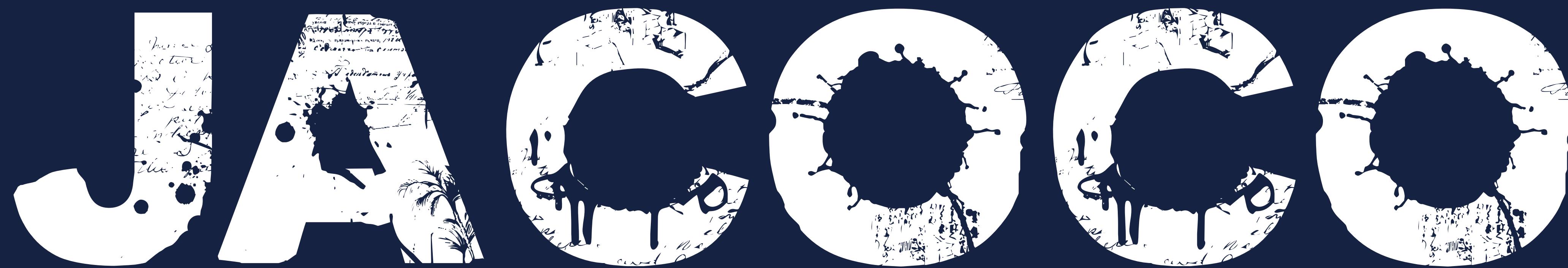
HOW TO FIND ZOMBIE CODE...?

- ☠ 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
- ☠ Can be used to detect dead code

JACOCO



HOW DOES

IT WORK?

azul

HOW DOES IT WORK...?



JACOCO

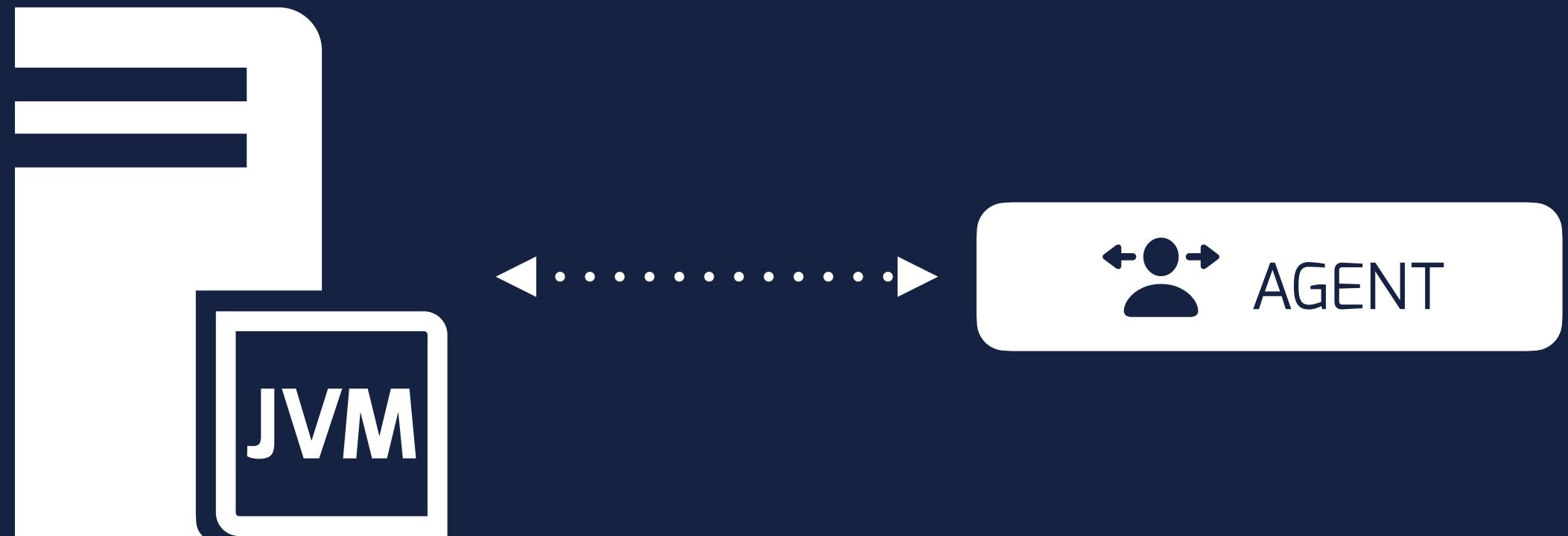


JVM runs
application

HOW DOES IT WORK...?



JACOCO



JVM runs
application

Agent gets info
(classes/methods)

HOW DOES IT WORK...?



JACOCO



JVM runs
application

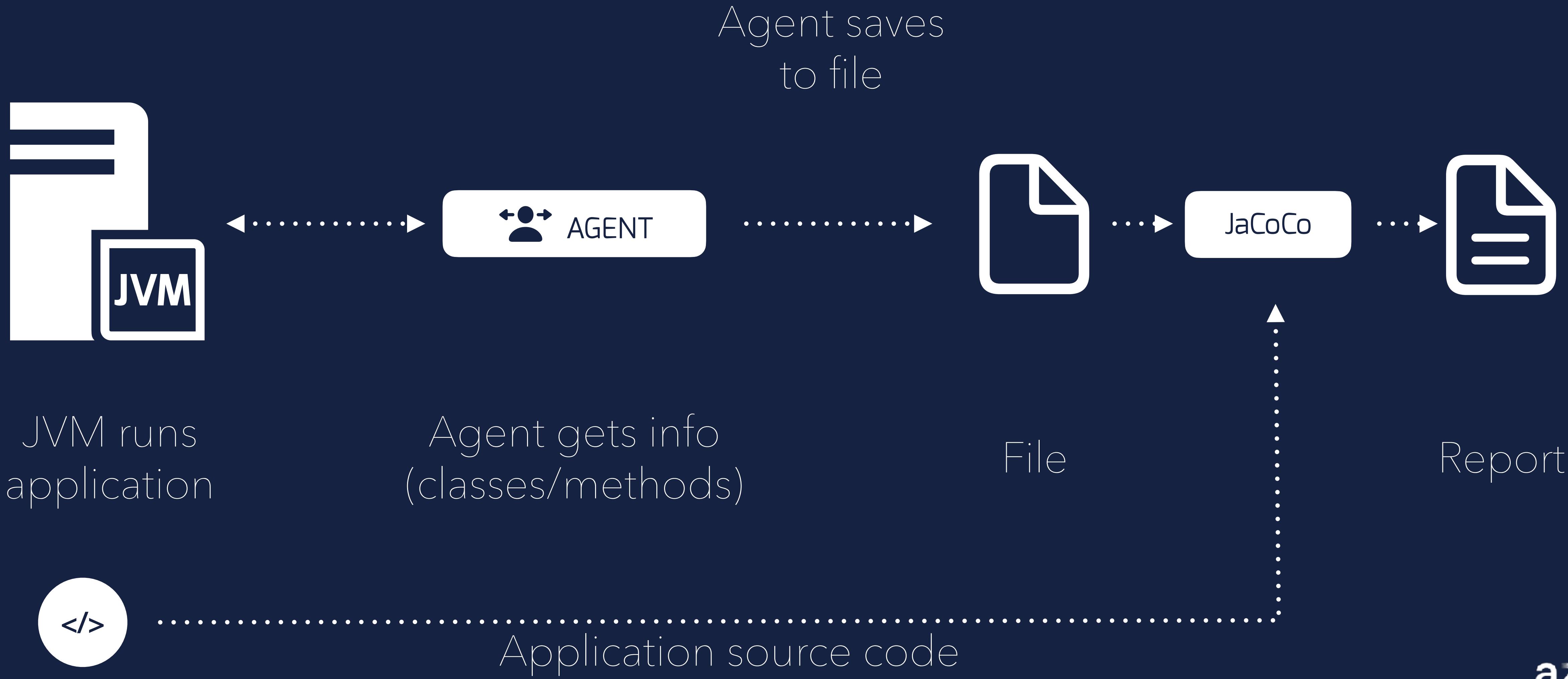
Agent gets info
(classes/methods)

Agent saves
to file

HOW DOES IT WORK...?



JACOCO



azul

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)
- Directly from Azul JVMs or by using an Agent for any other JVM

INTELLIGENCE CLOUD



WHAT IS IT ?

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

INTELLIGENCE CLOUD



WHAT IS IT ?

- Cloud service, collecting data from Java Virtual Machine(s)
- Directly from Azul JVMs or by using an Agent for any other JVM
- Offers information about used/unused code in production
- Vulnerability scanning in production

INTELLIGENCE CLOUD



WHAT IS IT ?

- Cloud service, collecting data from Java Virtual Machine(s)
- Directly from Azul JVMs or by using an Agent for any other JVM
- Offers information about used/unused code in production
- Vulnerability scanning in production
- Information about used Java Virtual Machine(s)

INTELLIGENCE CLOUD



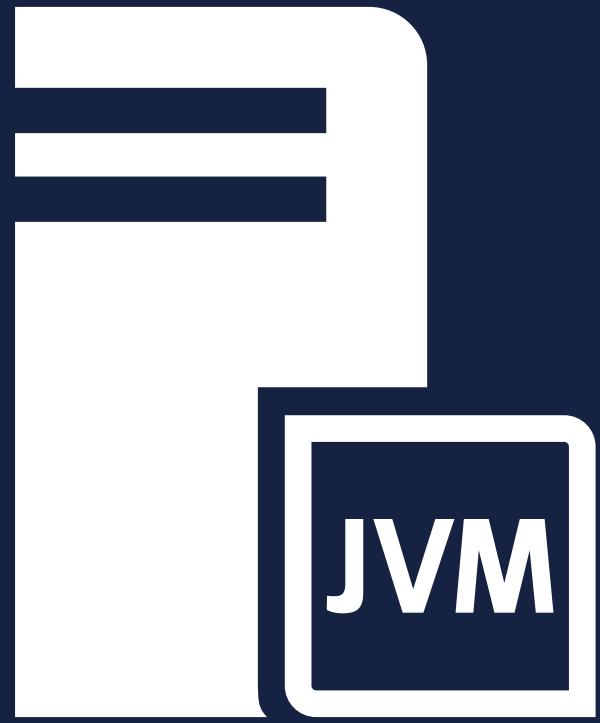
HOW DOES

IT WORK?

INTELLIGENCE CLOUD



HOW DOES IT WORK ?

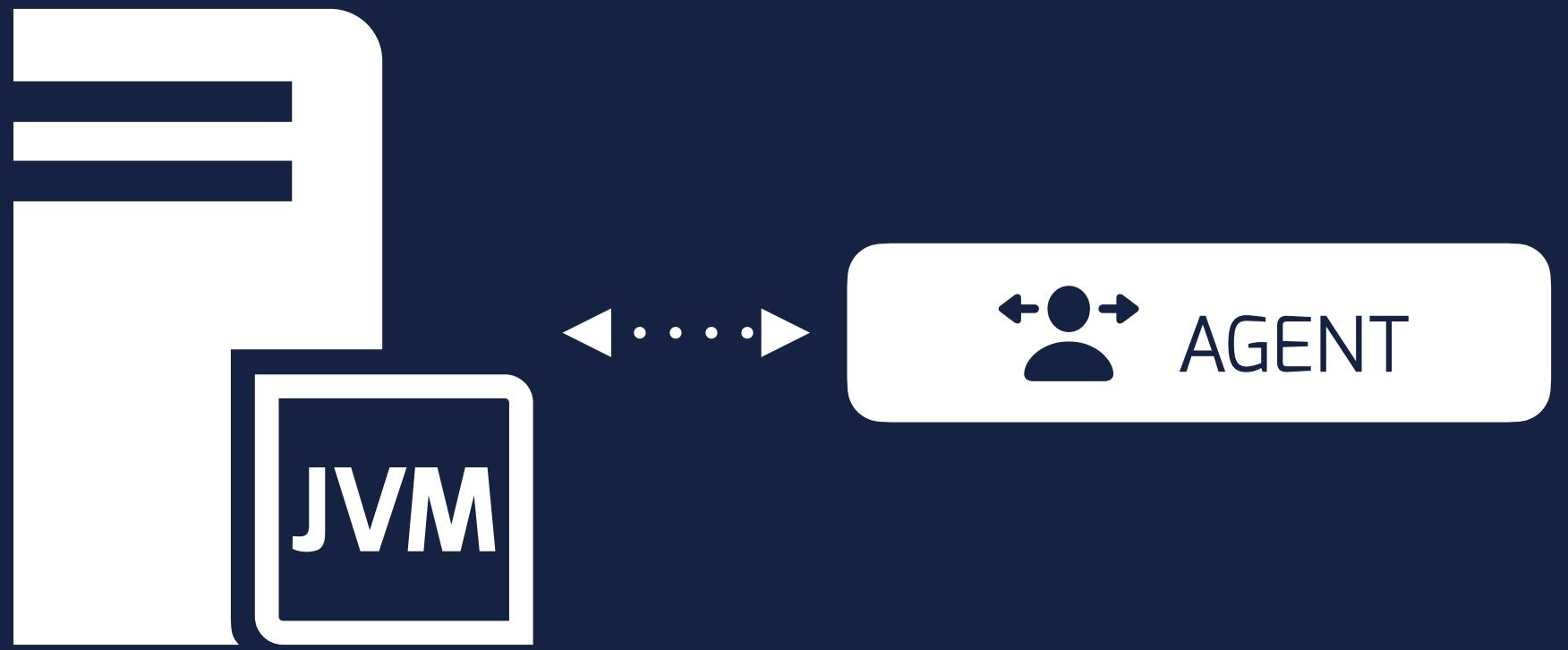


JVM runs
application

INTELLIGENCE CLOUD



HOW DOES IT WORK ?



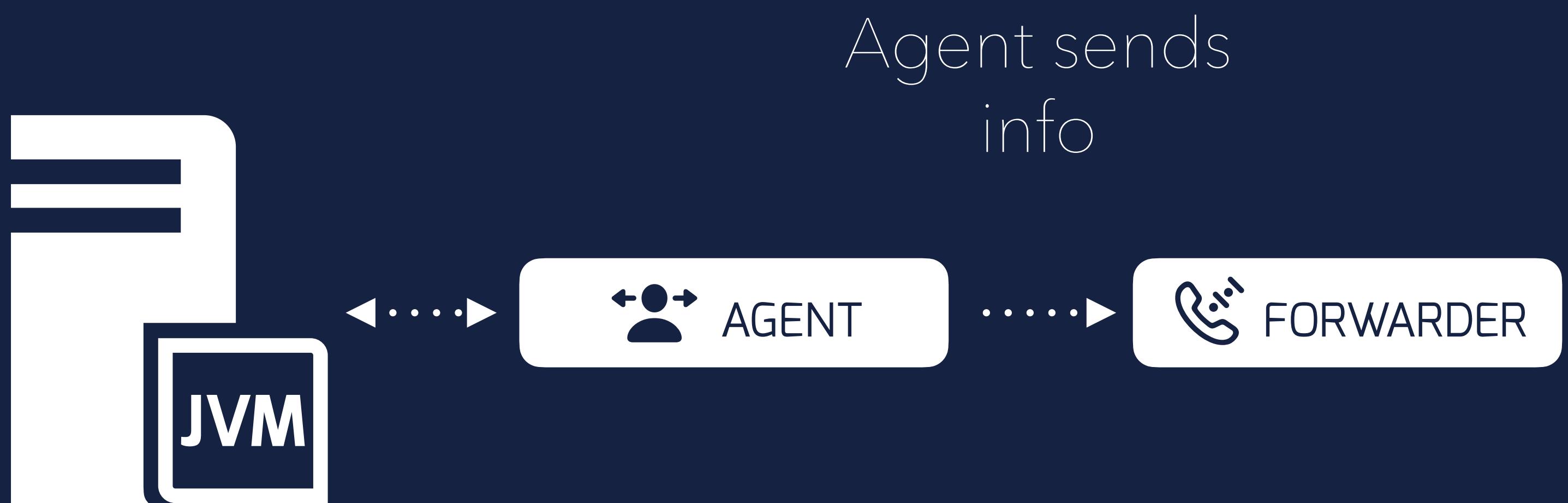
JVM runs
application

Agent gets info
(classes/methods)

INTELLIGENCE CLOUD



HOW DOES IT WORK ?



JVM runs
application

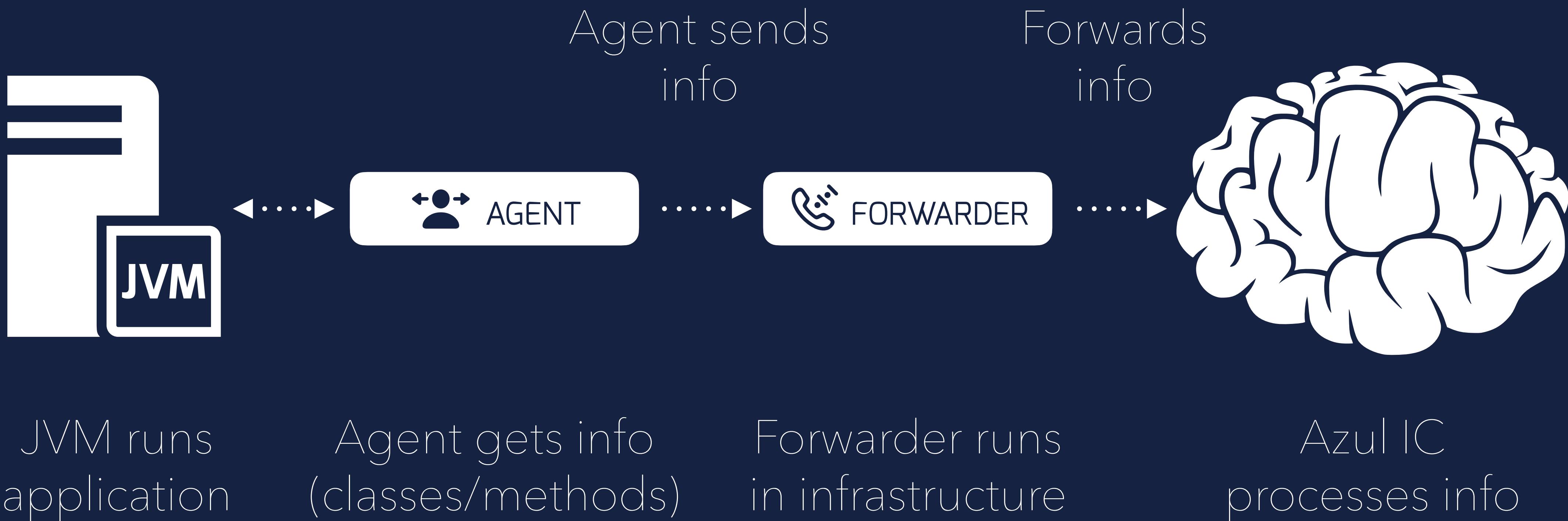
Agent gets info
(classes/methods)

Forwarder runs
in infrastructure

INTELLIGENCE CLOUD



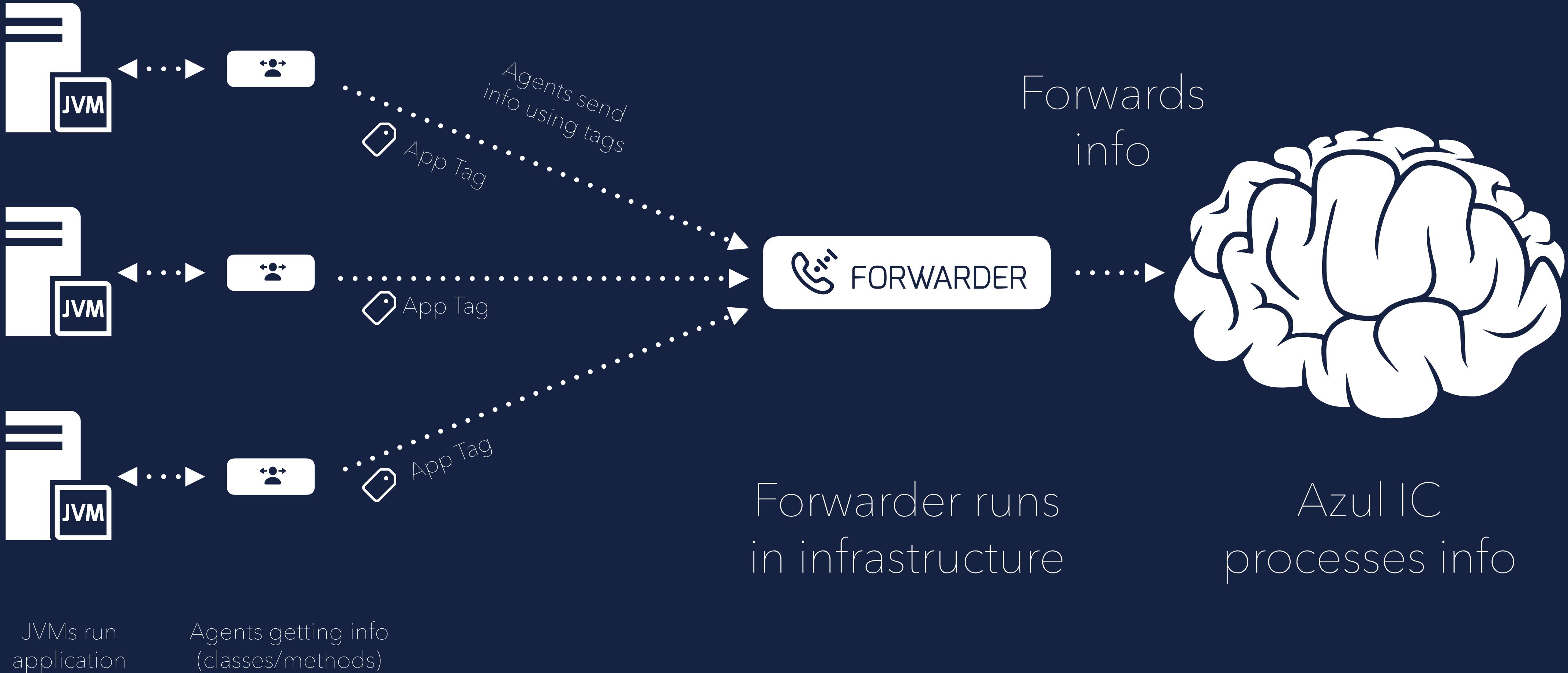
HOW DOES IT WORK ?



INTELLIGENCE CLOUD



HOW DOES IT WORK ?



INTELLIGENCE CLOUD



FIND ZONE IMPACT

CODE DUE

INTELLIGENCE CLOUD



FINDING ZOMBIE CODE...



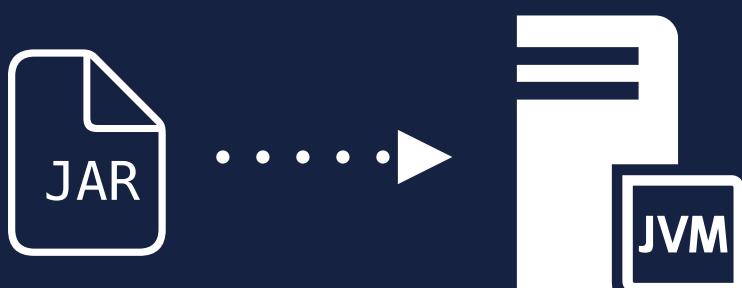
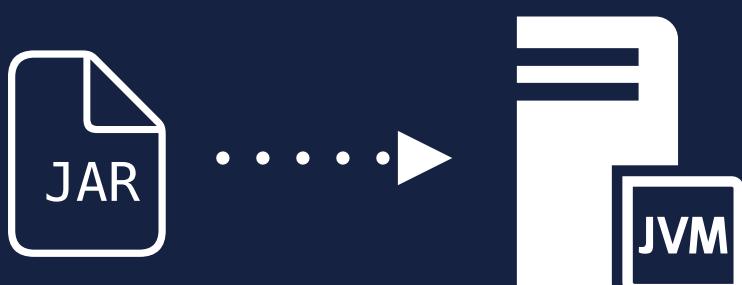
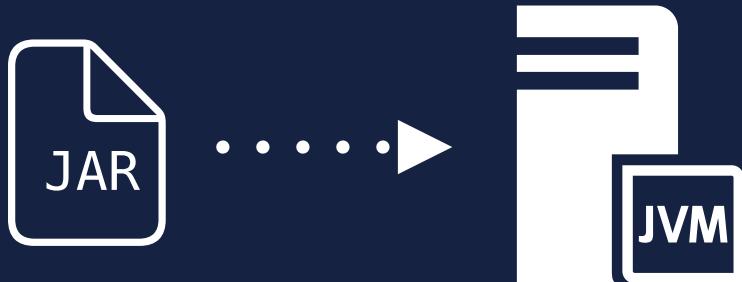
JAR

Application

INTELLIGENCE CLOUD



FINDING ZOMBIE CODE...



Application

JVMs run
Application

INTELLIGENCE CLOUD



FINDING ZOMBIE CODE...



Application

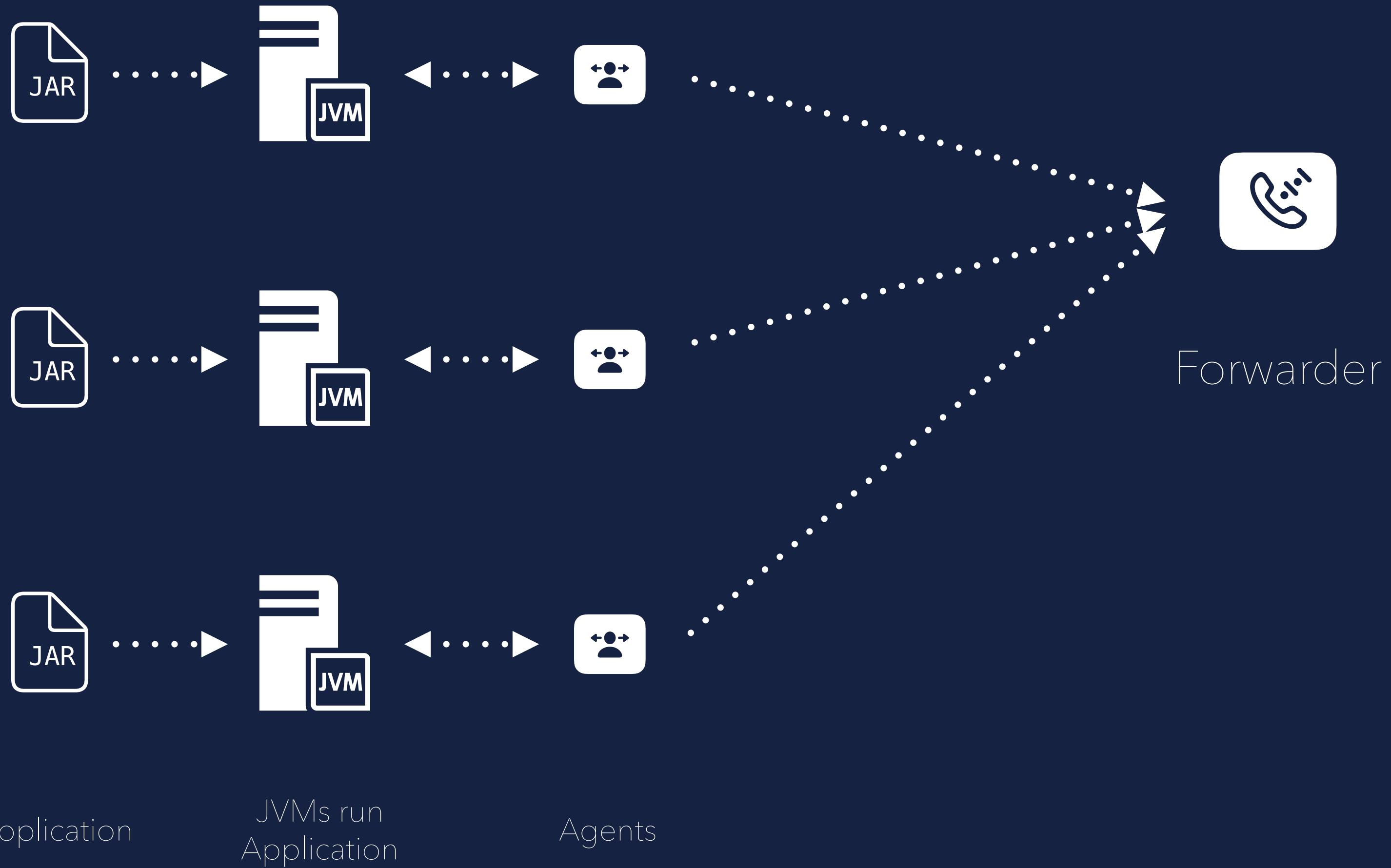
JVMs run
Application

Agents

INTELLIGENCE CLOUD



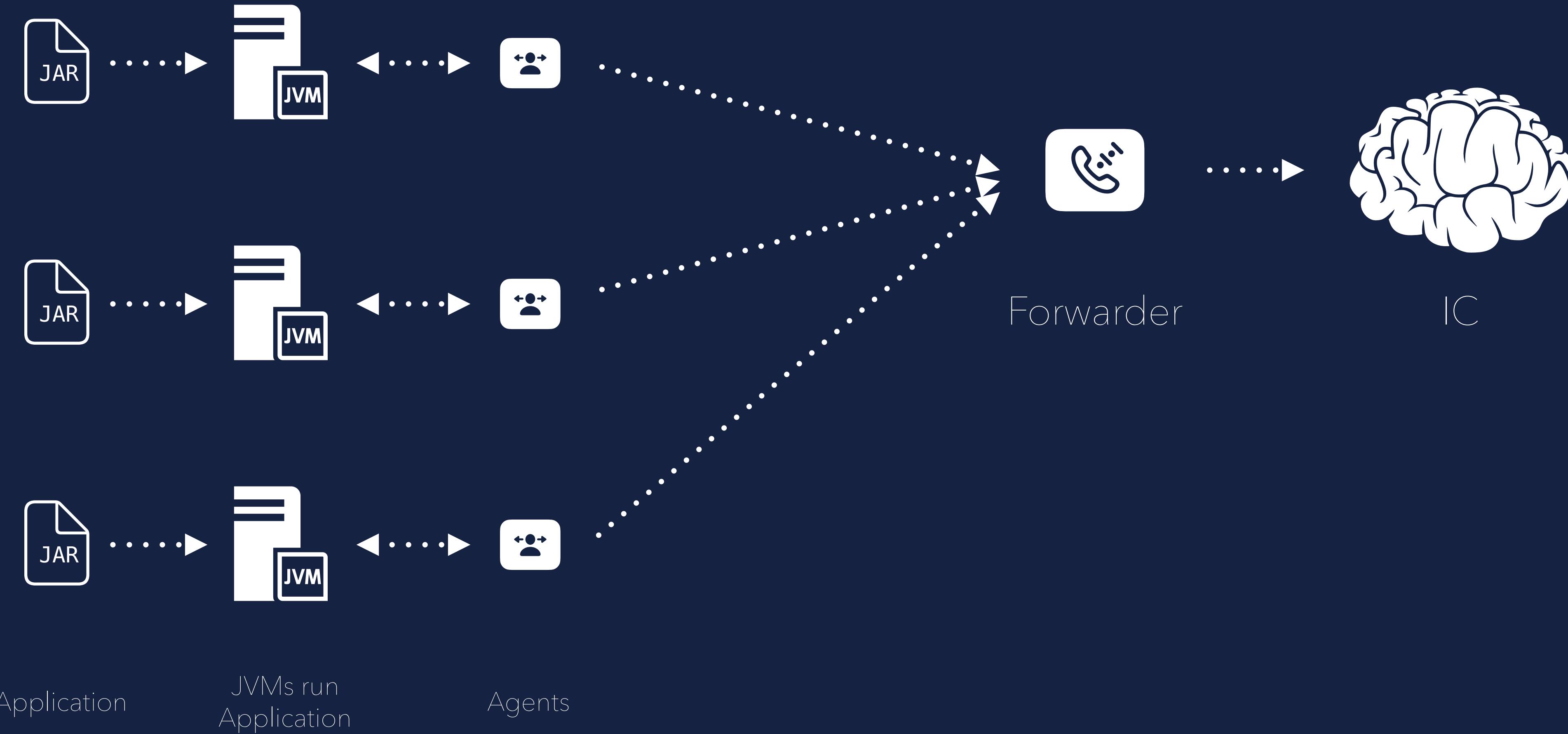
FINDING ZOMBIE CODE...



INTELLIGENCE CLOUD



FINDING ZOMBIE CODE...



INTELLIGENCE CLOUD



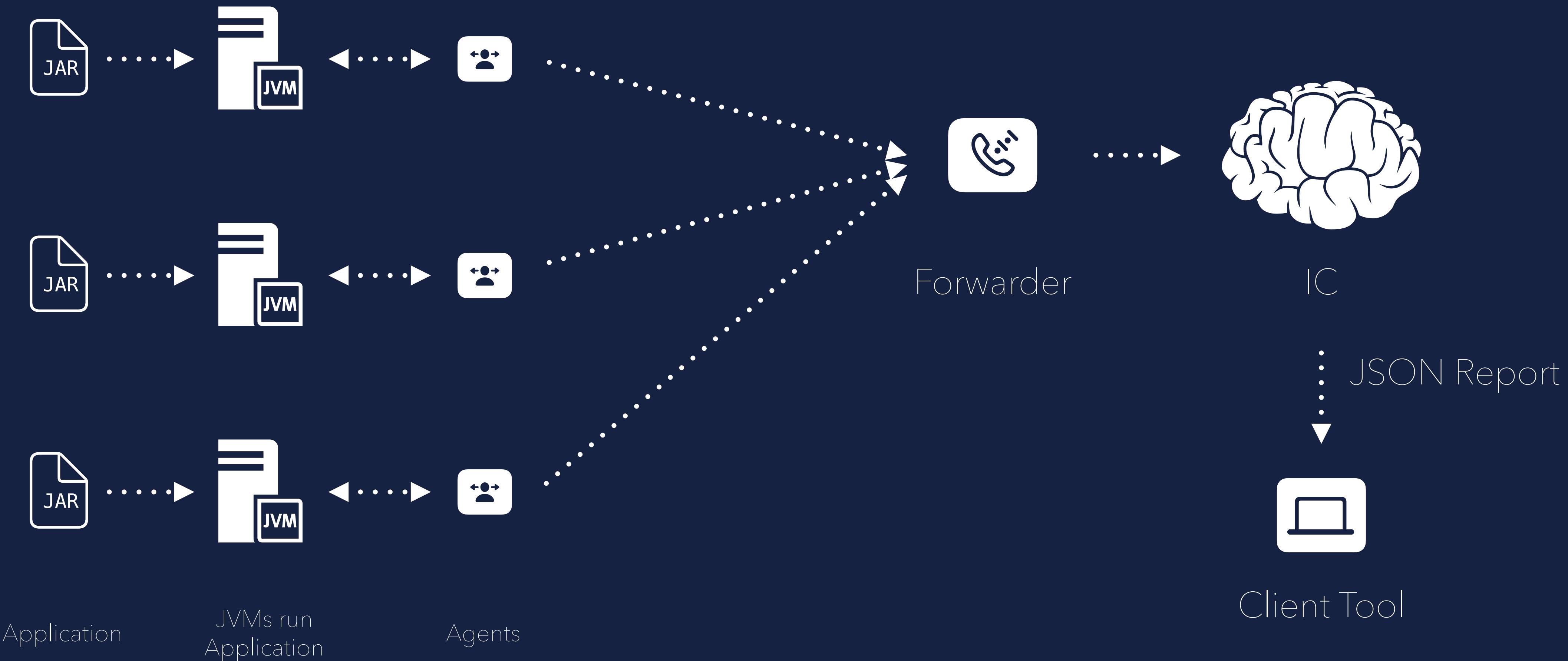
FINDING ZOMBIE CODE...



INTELLIGENCE CLOUD



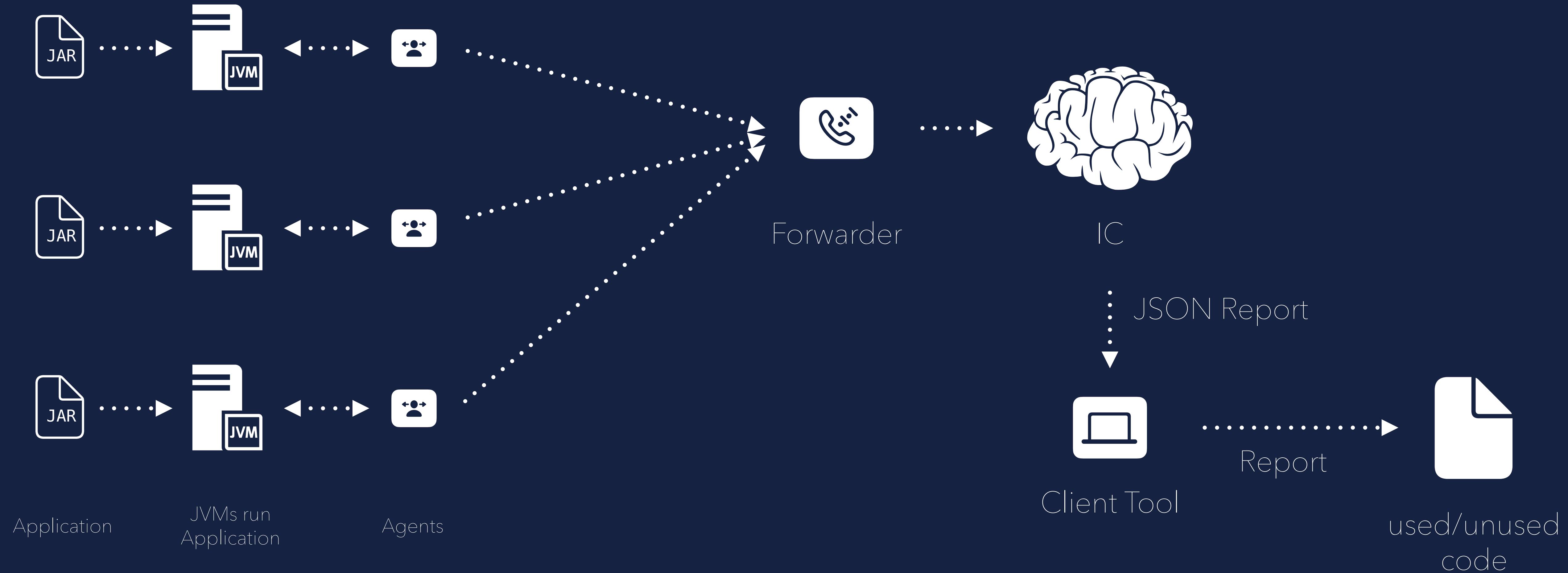
FINDING ZOMBIE CODE...



INTELLIGENCE CLOUD



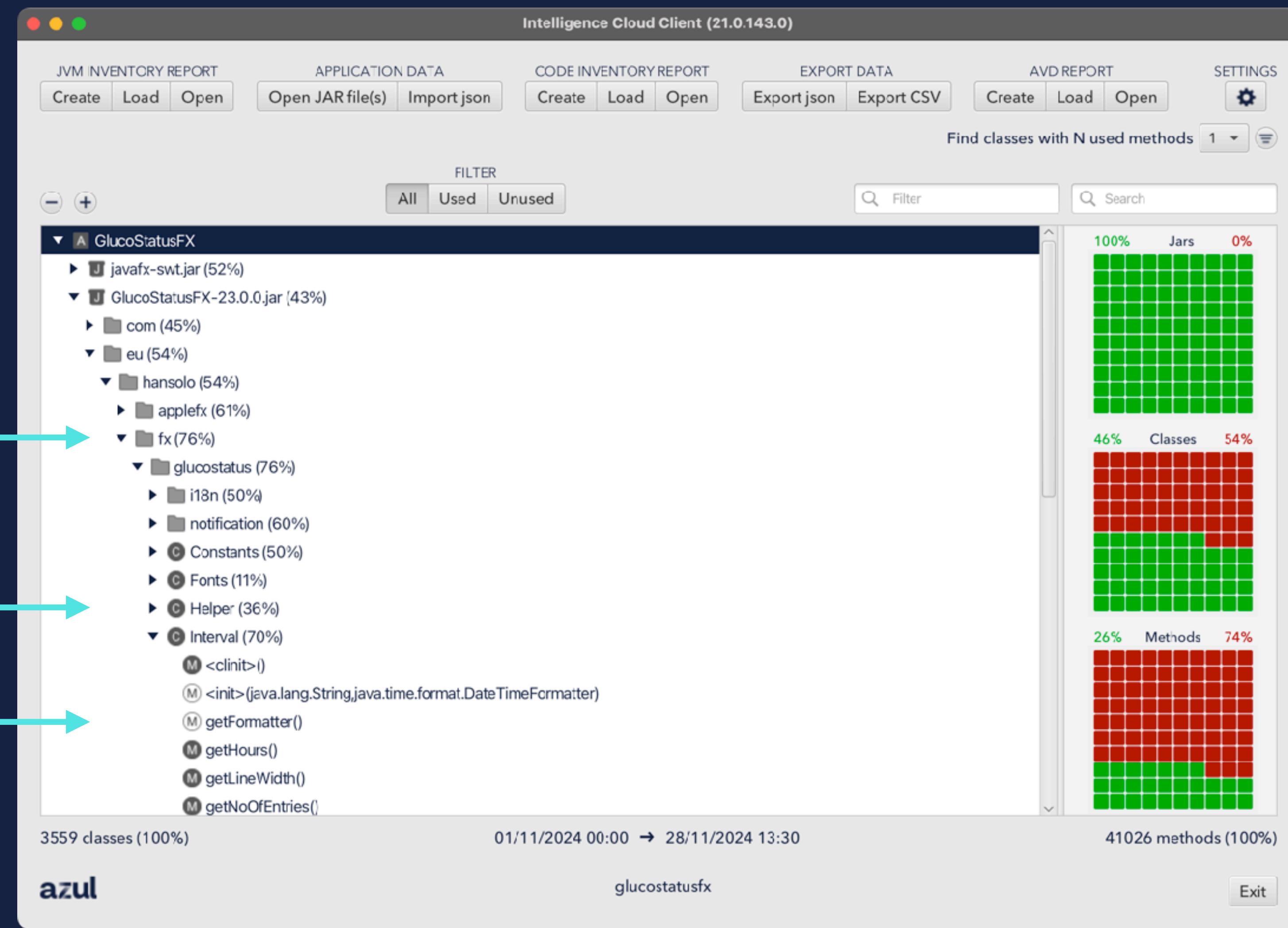
FINDING ZOMBIE CODE...



MURDER IN THE INGENIE CLOUD



FINDING ZOMBIE CODE...



INTELLIGENCE CLOUD



WHAT?

YOU'D GIVE



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
- ☠ Method level resolution



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
- ☠ Method level resolution
- ☠ 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
- ☠ Method level resolution
- ☠ Run app on multiple JVMs at the same time using tags
- ☠ Offers an API for easy integration in 3rd party tools

WHERE TO GET IT...?



AZUL 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 Azul Intelligence Cloud can help

THANK

YOU