
WE HATE CODE
The joy of maintaining dead code

ABOUT ME.

@hansolo_

Gerrit Grunwald | Developer Advocate | Azul

azul

WHO LOVES TO MAINTAIN LEGACY CODE...?

SOME STATS...

4 HOURS A WEEK...

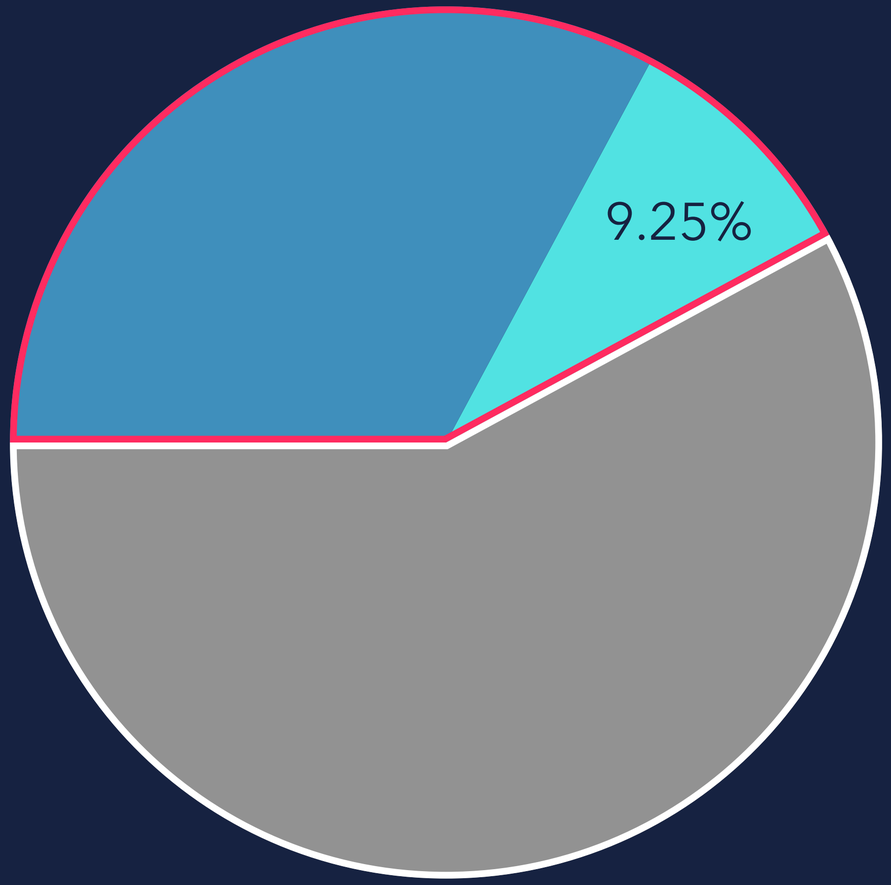
azul

...EQUATES TO $85 BILLION ANNUALLY!

Sources: Evans Data Corp, CIA Factbook, Stripe research

azul

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

Sources: Evans Data Corp, CIA Factbook, Stripe research

azul

MORE INFO...

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

azul

AND WHAT DOES IT COST YOU...?

azul

BAD CODE

WHAT DOES IT COST YOU...?

- Time
- Productivity
- Agility
- Money
- Reputation

TWO TYPES OF BAD CODE...

TWO TYPES OF BAD CODE...

LEGACY CODE

DEAD CODE

azul

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 STILL USING IT...?

WHY STILL USING IT...?

BECAUSE...

- It works

WHY STILL USING IT...?

BECAUSE...

- It works
- 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

- Refactoring

azul

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

- You need to know what the code does

azul

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

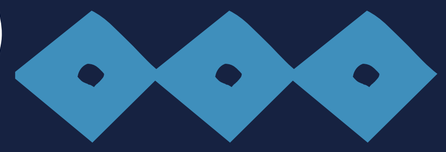
- You need to know what the code does
- It's like a new project with all it's work (and cost) involved
- You need to know the requirements

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 REMAINS...



azul

IS IT USED AT ALL...?

azul

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

azul

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

WHERE DOES IT COME 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 REMOVE 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

azul

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"

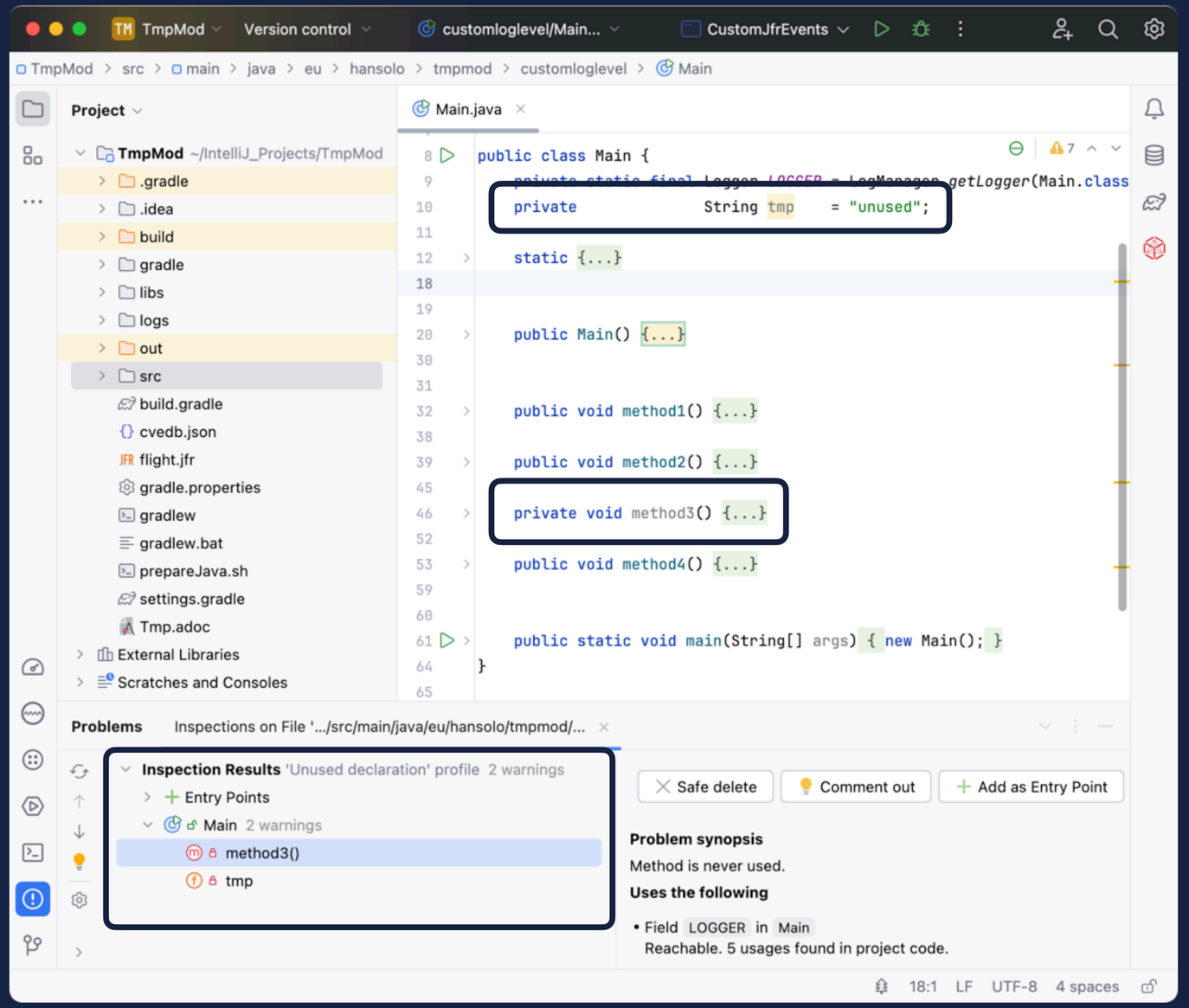
DEAD CODE

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



azul

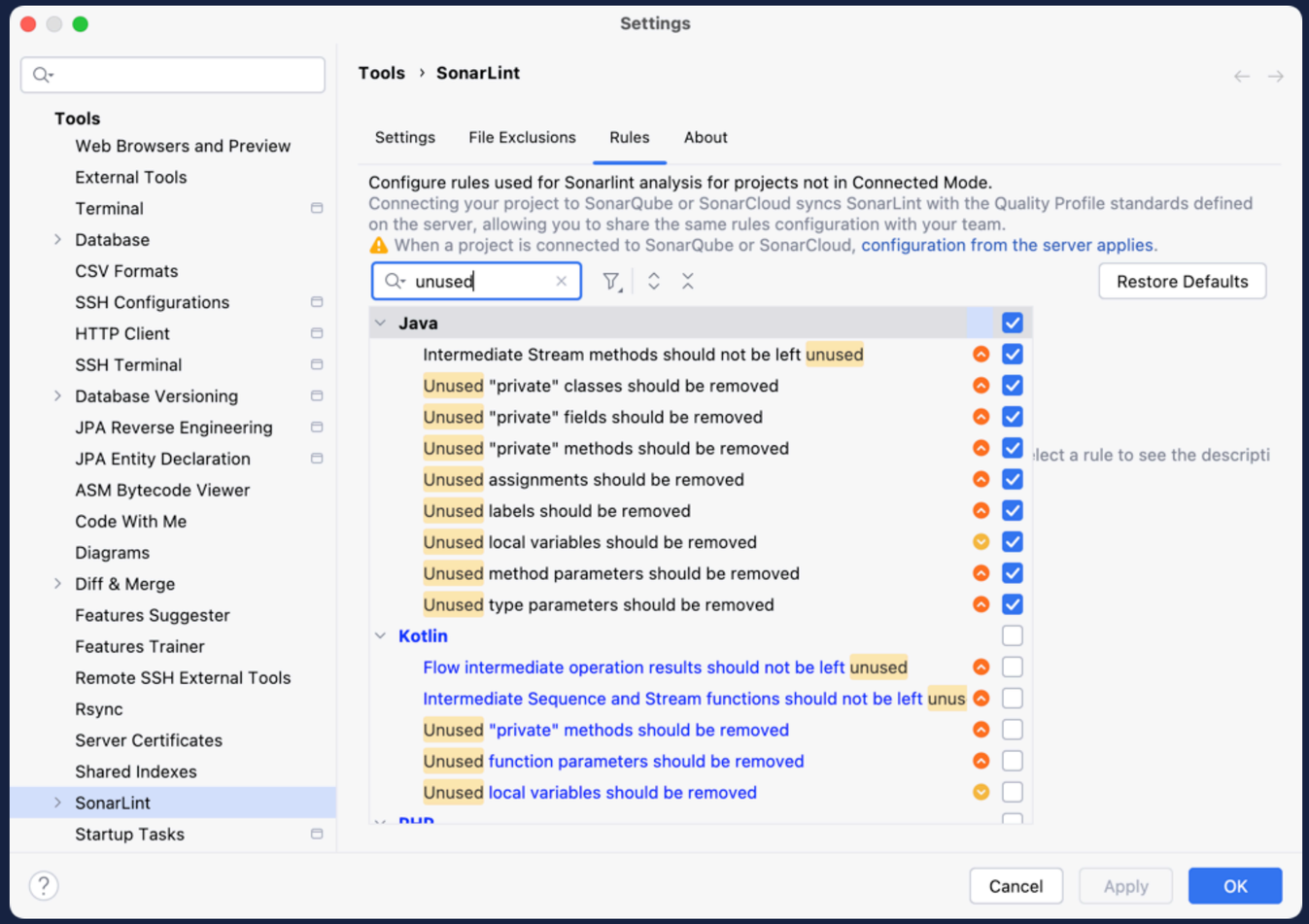
HOW TO REMOVE IT...?

POSSIBLE SOLUTIONS...

- IDE's e.g. Intellij IDEA
- Sonarqube ide

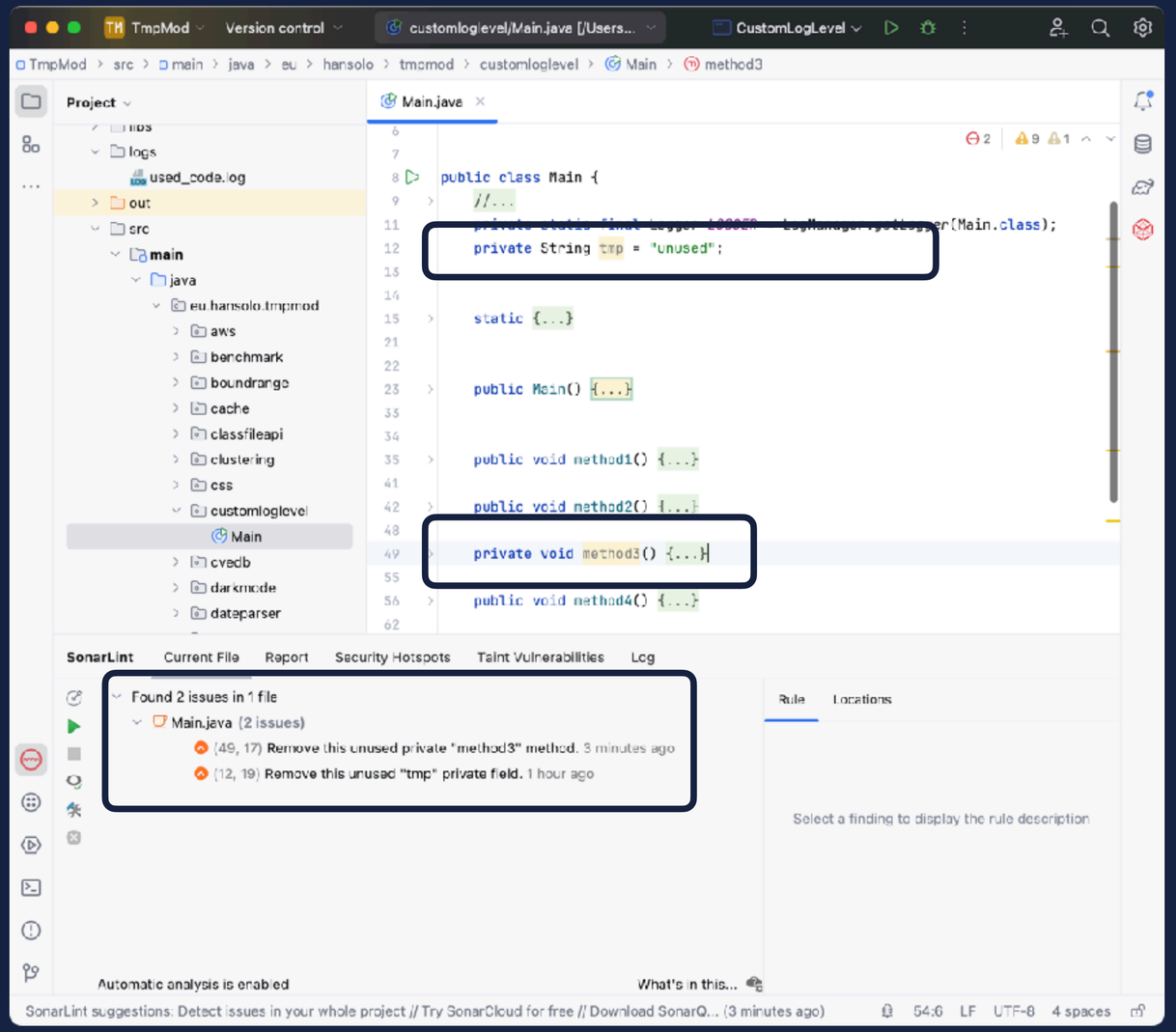
DEAD CODE

SONARQUBE IDE



DEAD CODE

SONARQUBE IDE



azul

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 REWRITE

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 REWRITE

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
```

azul

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

BUT WHAT ABOUT...

ZOMBIE

CODE

MAINTAINED CODE...

azul


...NEVER
EXECUTED

HOW TO FIND IT...?

azul

ONLY IN A RUNNING SYSTEM...

ZOMBIE CODE

HOW TO FIND IT...?

- Tombstone logging using static initialisers and custom log levels

ZOMBIE CODE

LOG4J CONFIGURATION

```xml
<?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
<?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;
    }
}
```

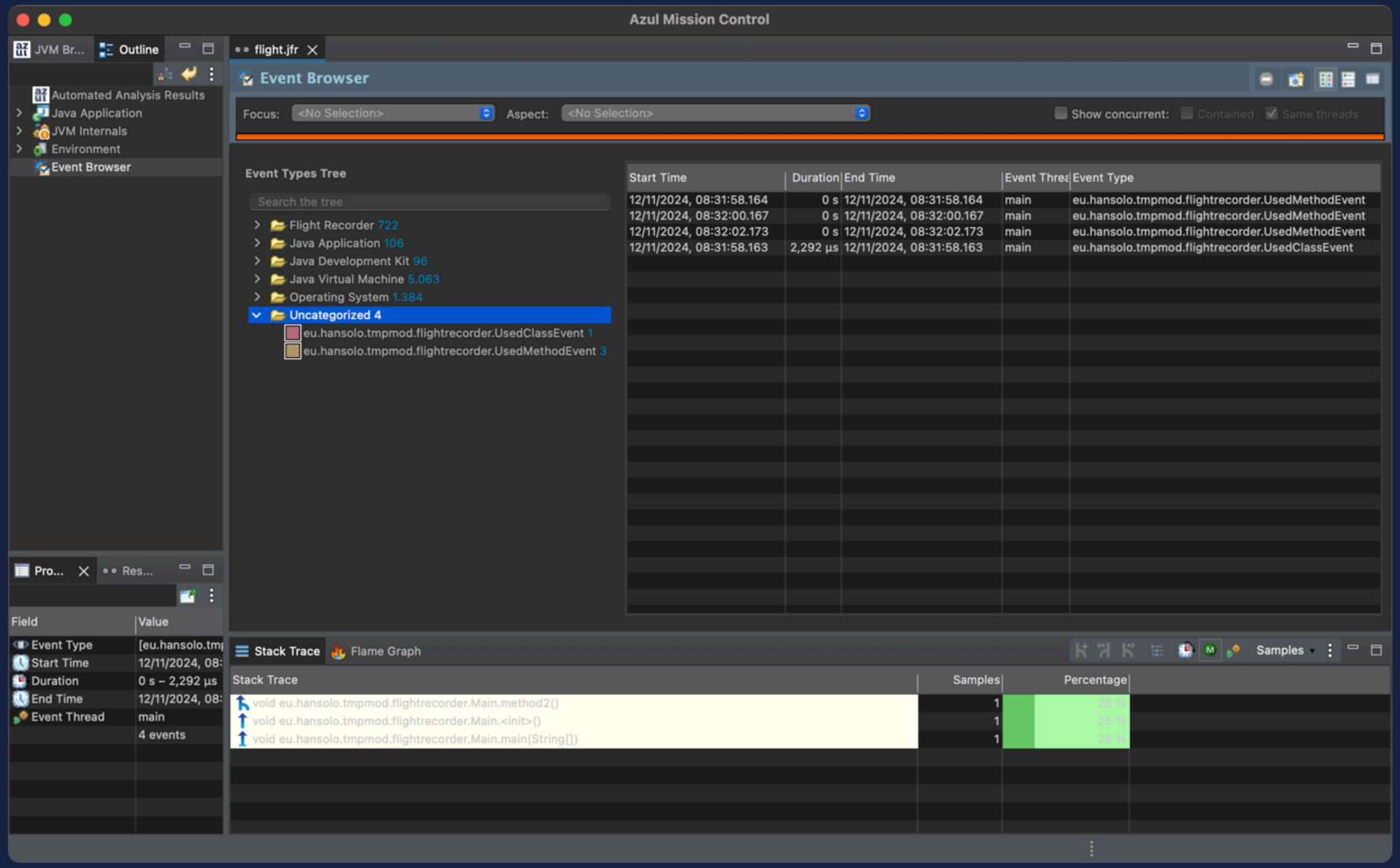
ZOMBIE CODE

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();
    }
}
```

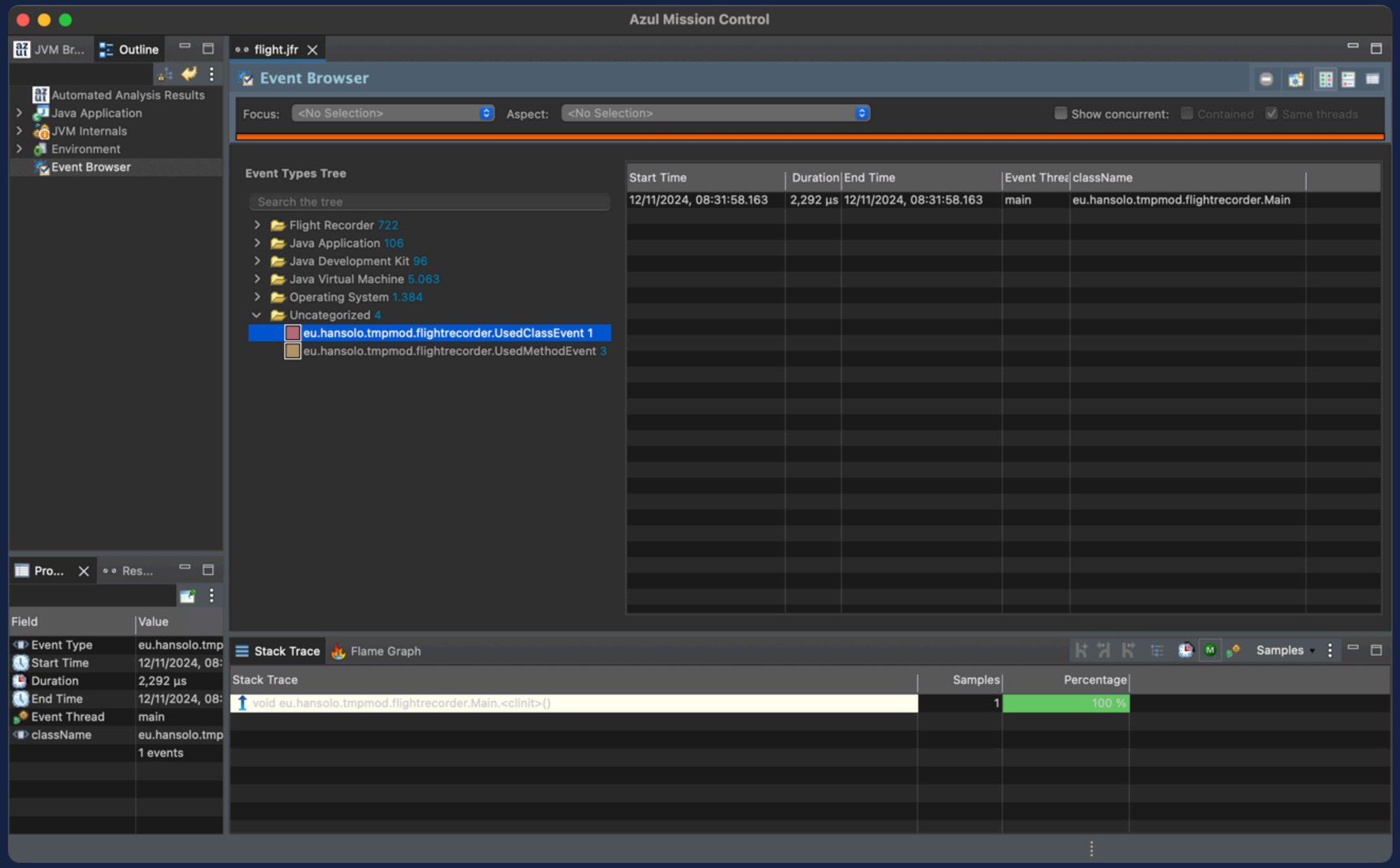
ZOMBIE CODE

MISSION CONTROL



ZOMBIE CODE

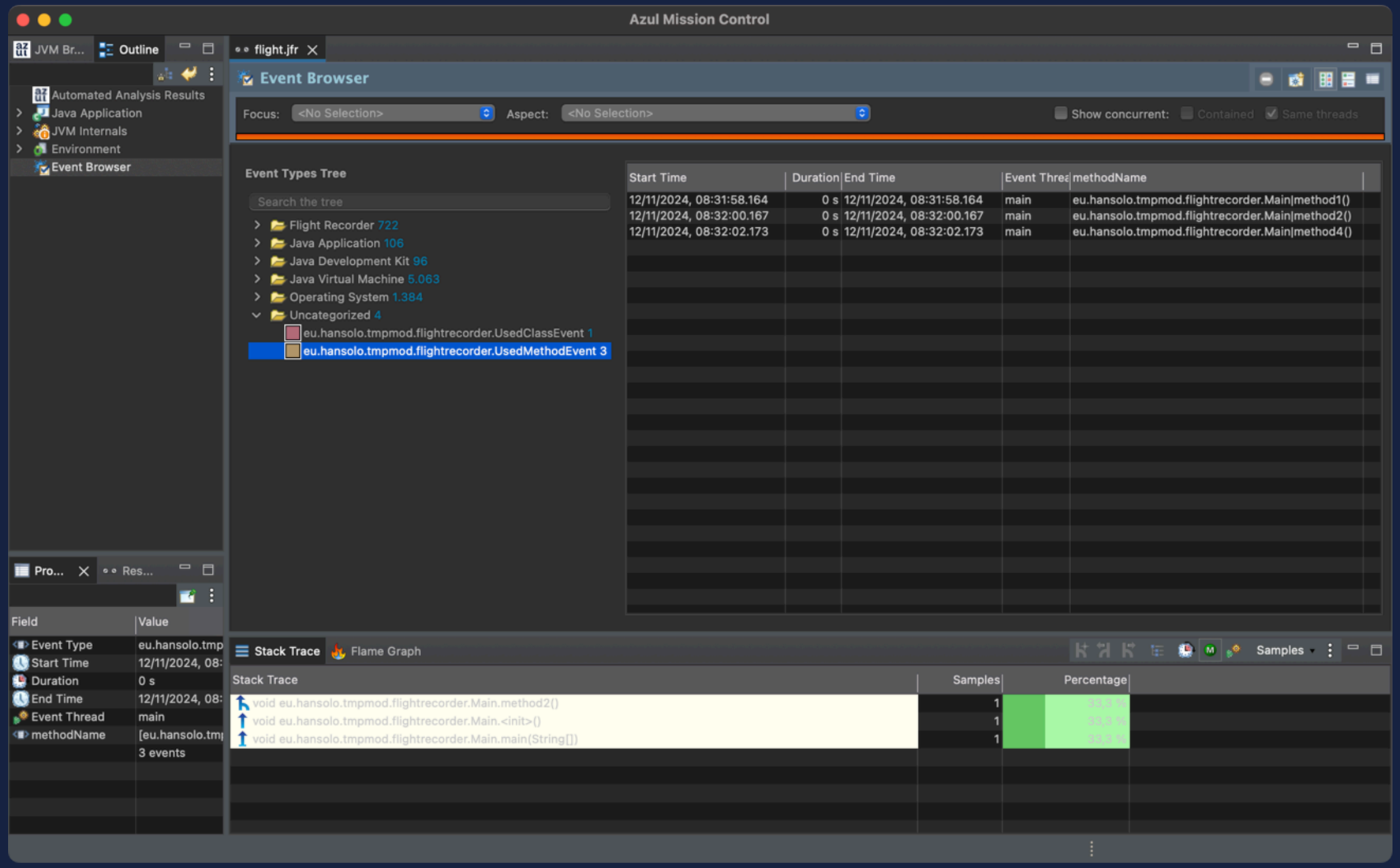
MISSION CONTROL



azul

ZOMBIE CODE

MISSION CONTROL



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

BUT A LOT OF WORK!

OTHER SOLUTIONS...

OTHER SOLUTIONS...

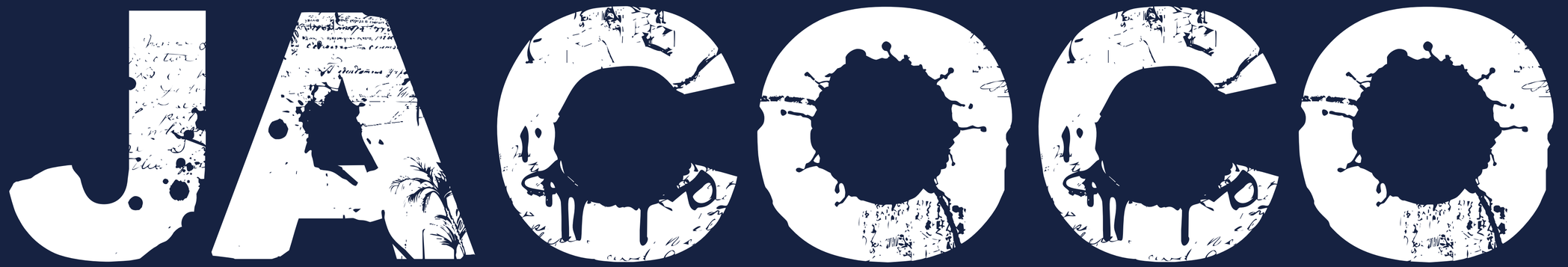
HOW TO FIND ZOMBIE CODE...?

- Java Code Coverage (JaCoCo)

OTHER SOLUTIONS...

HOW TO FIND ZOMBIE CODE...?

- Java Code Coverage (JaCoCo)
- Azul Intelligence Cloud



Java Code Coverage

JACOCO

WHAT IS IT...?

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

JACOCO

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

JACOCO

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

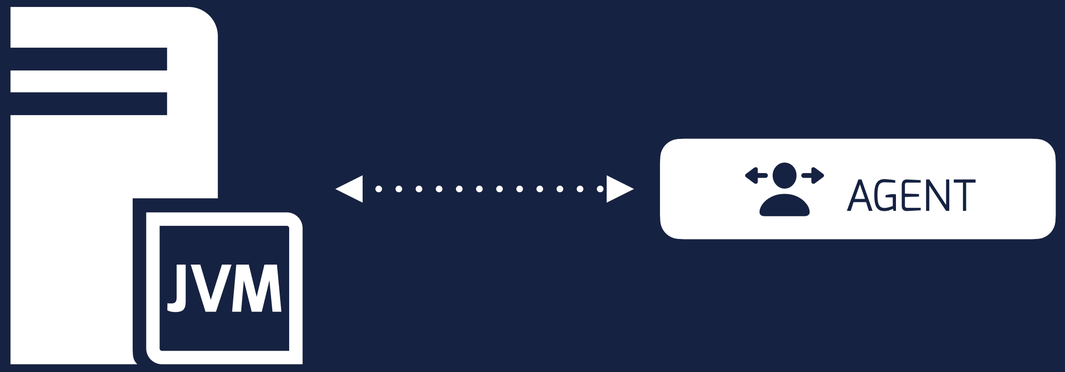
HOW DOES IT WORK...?

JACOCO

JVM runs application

HOW DOES IT WORK...?

JACOCO



JVM runs application

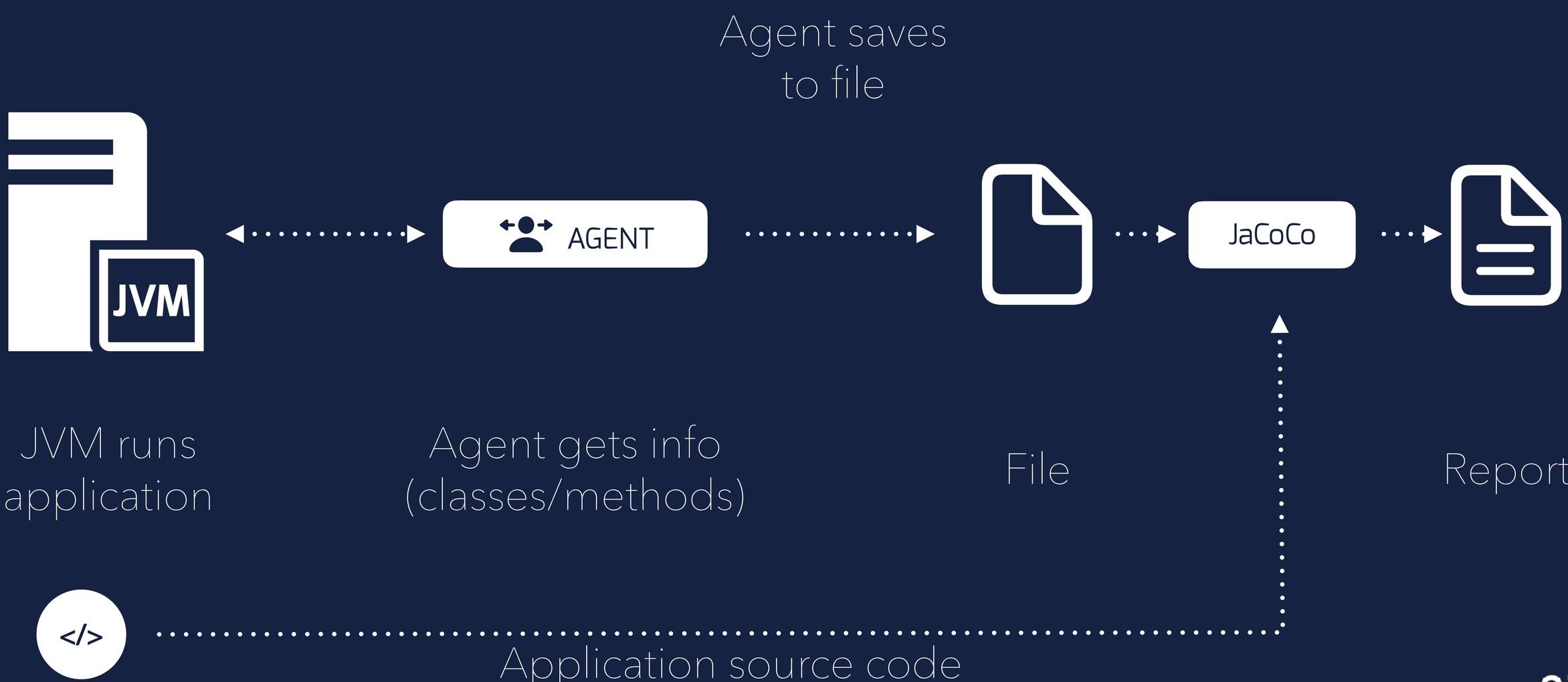
Agent gets info (classes/methods)

HOW DOES IT WORK...?

JACOCO

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

azul

WHERE TO GET IT...?

JACOCO

https://github.com/jacoco/jacoco

azul

INTELLIGENCE CLOUD

azul

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...?

azul

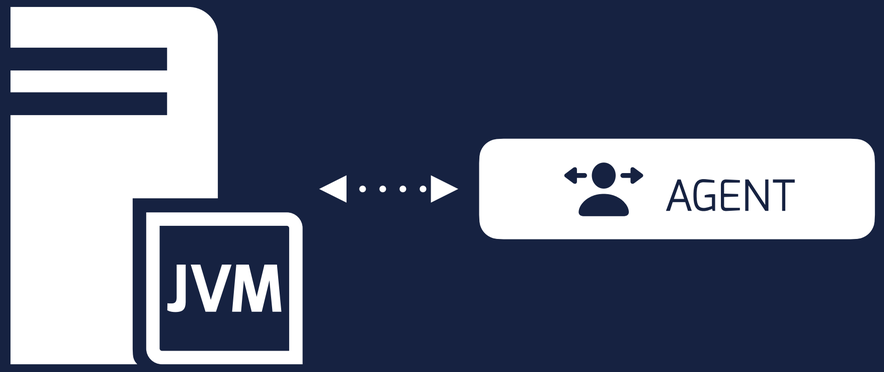
INTELLIGENCE CLOUD

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 ?

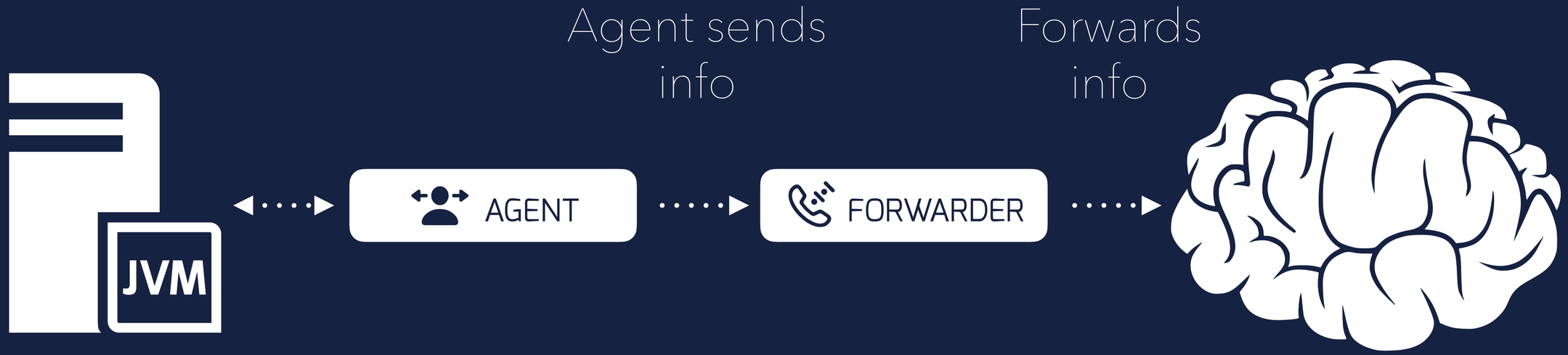
JVM runs application

Agent gets info (classes/methods)

Forwarder runs in infrastructure

INTELLIGENCE CLOUD

HOW DOES IT WORK ?



JVM runs application

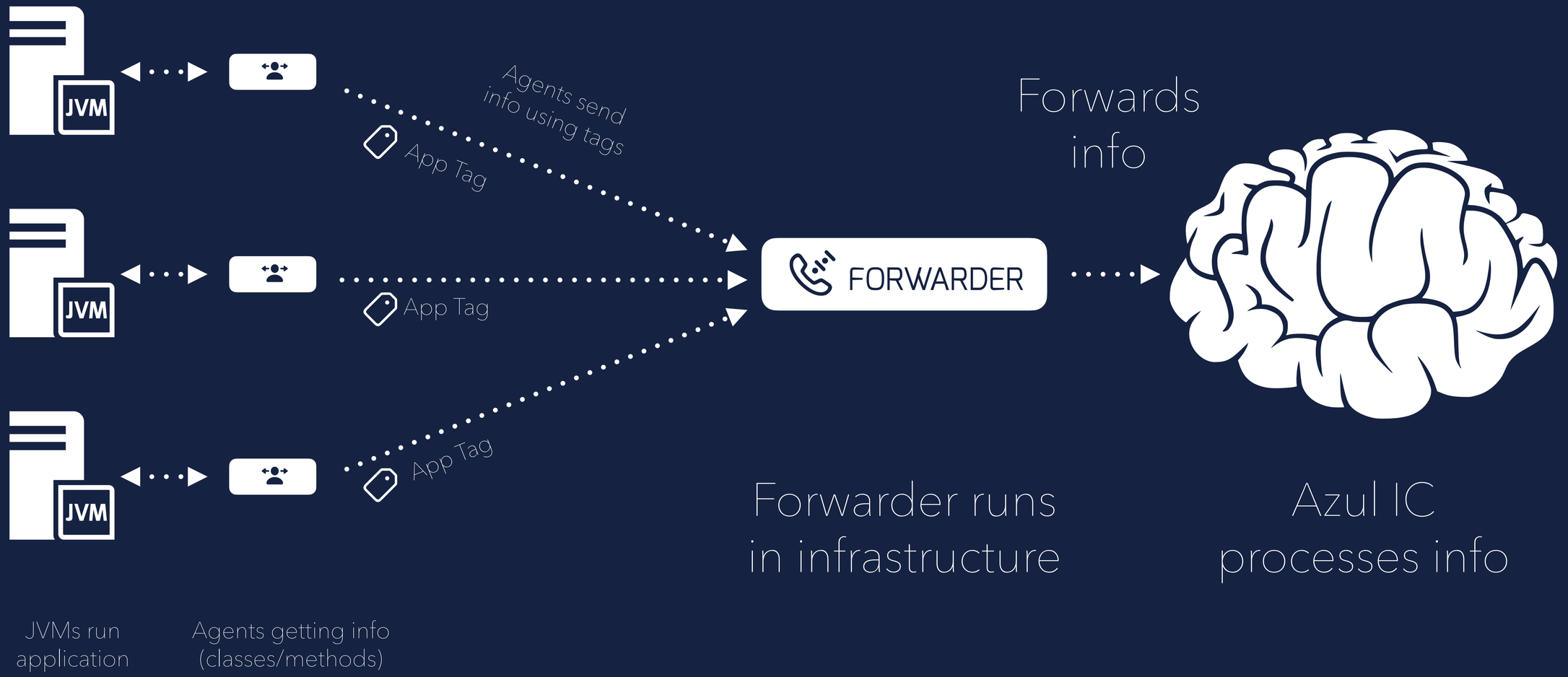
Agent gets info (classes/methods)

Forwarder runs in infrastructure

Azul IC processes info

INTELLIGENCE CLOUD

HOW DOES IT WORK ?



azul

INTELLIGENCE CLOUD

FIND ZOMBIE CODE...

azul

INTELLIGENCE CLOUD

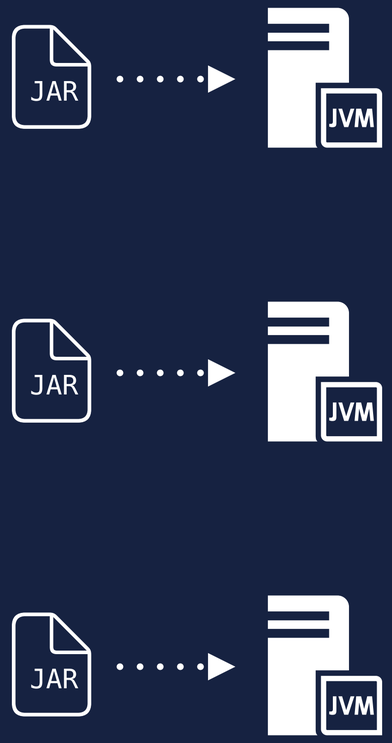
FINDING ZOMBIE CODE...

Application

azul

INTELLIGENCE CLOUD

FINDING ZOMBIE CODE...



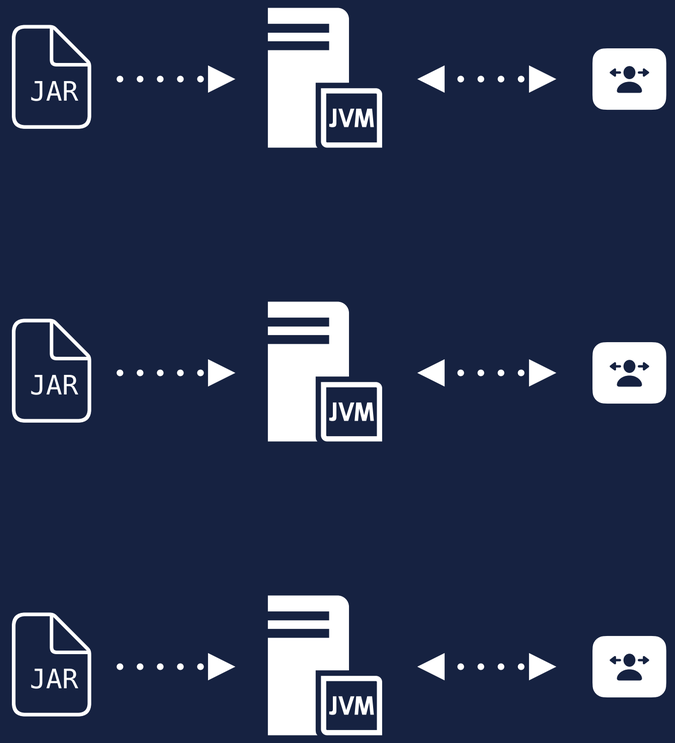
Application

JVMs run Application

azul

INTELLIGENCE CLOUD

FINDING ZOMBIE CODE...



Application

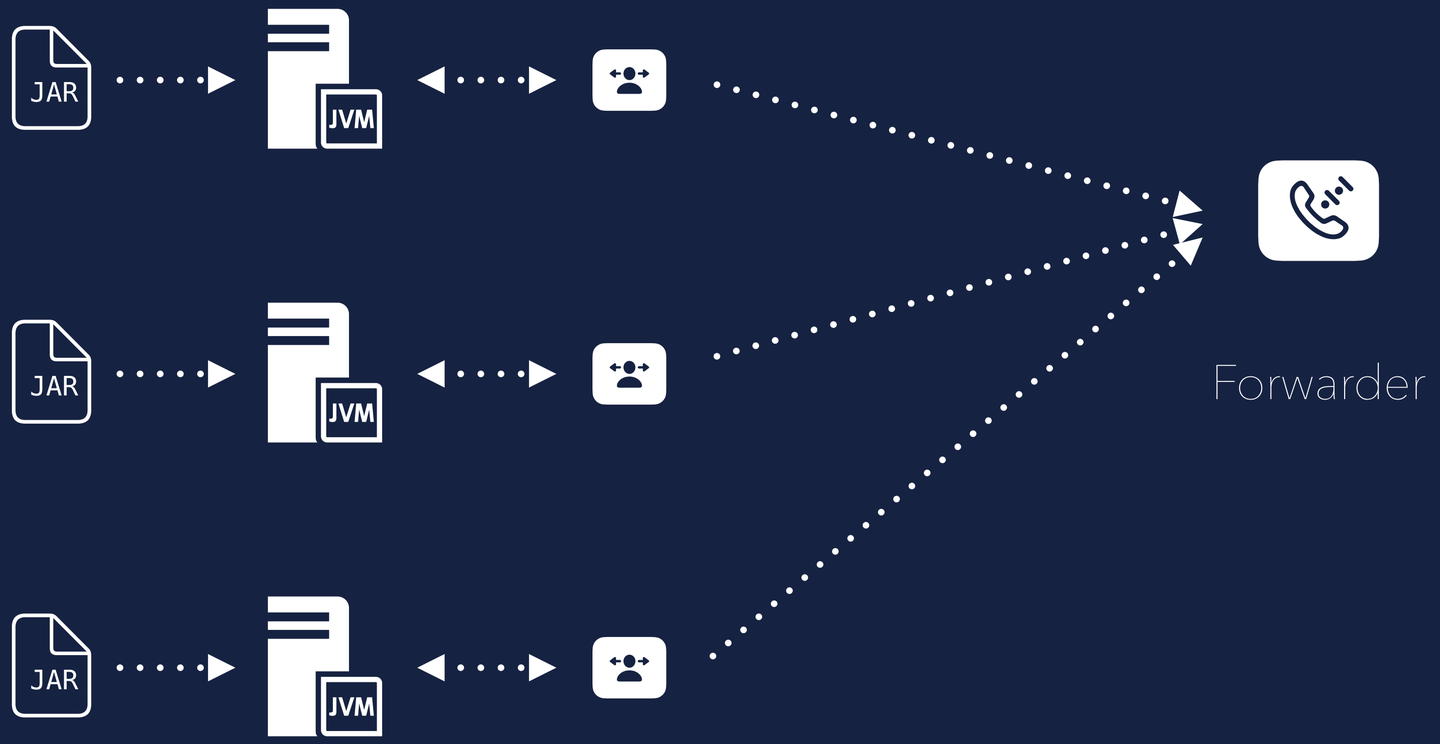
JVMs run Application

Agents

azul

INTELLIGENCE CLOUD

FINDING ZOMBIE CODE...



Application

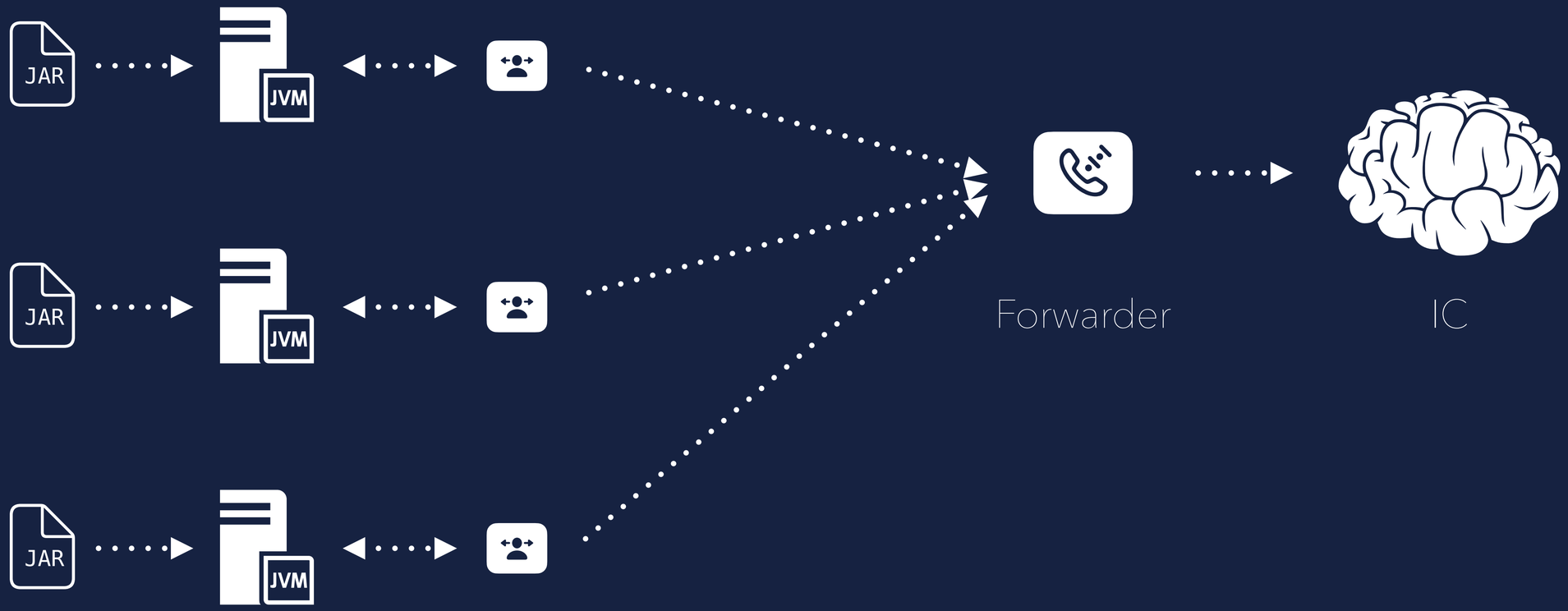
JVMs run Application

Agents

azul

INTELLIGENCE CLOUD

FINDING ZOMBIE CODE...



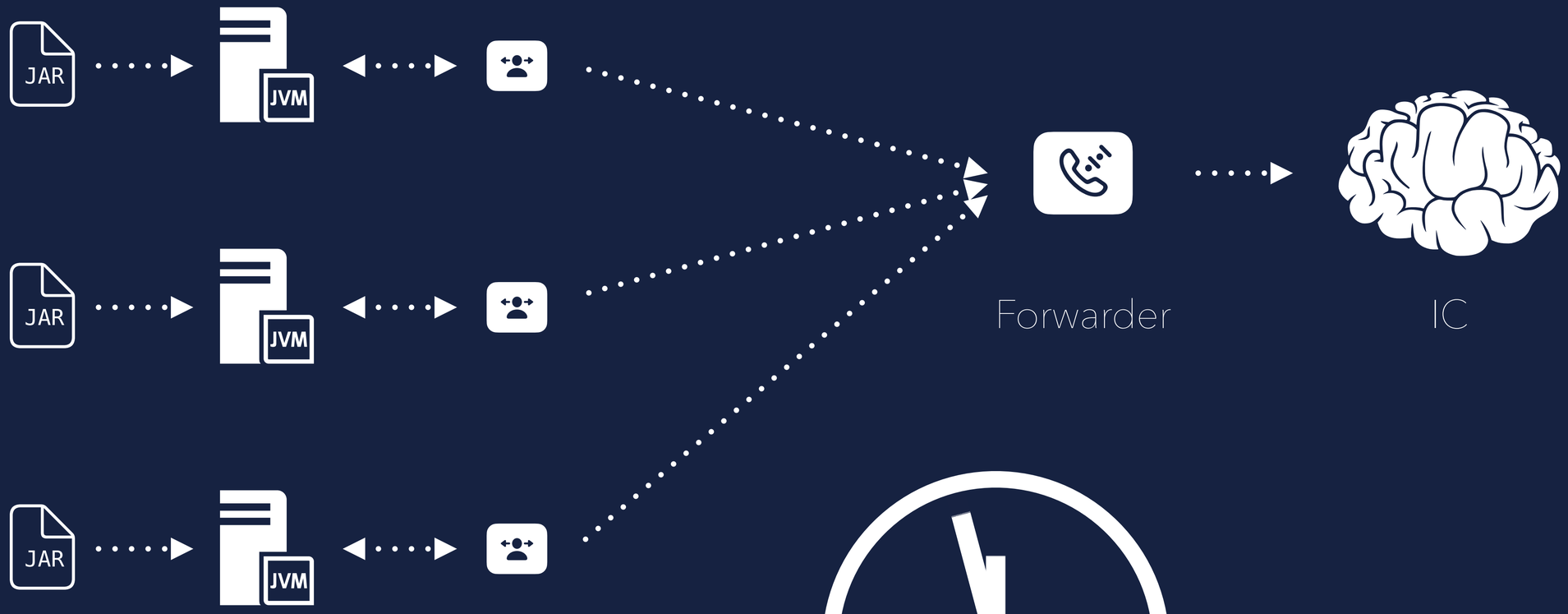
Application

JVMs run Application

Agents

INTELLIGENCE CLOUD

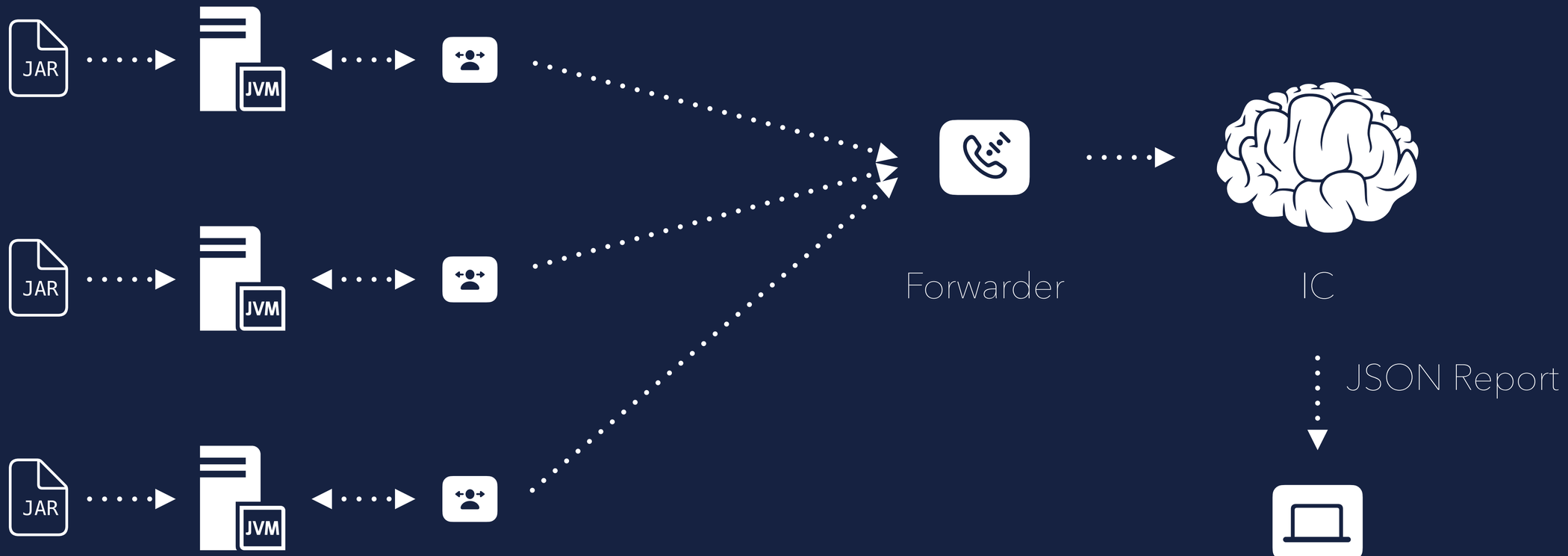
FINDING ZOMBIE CODE...



RUN IT FOR SOME TIME...

INTELLIGENCE CLOUD

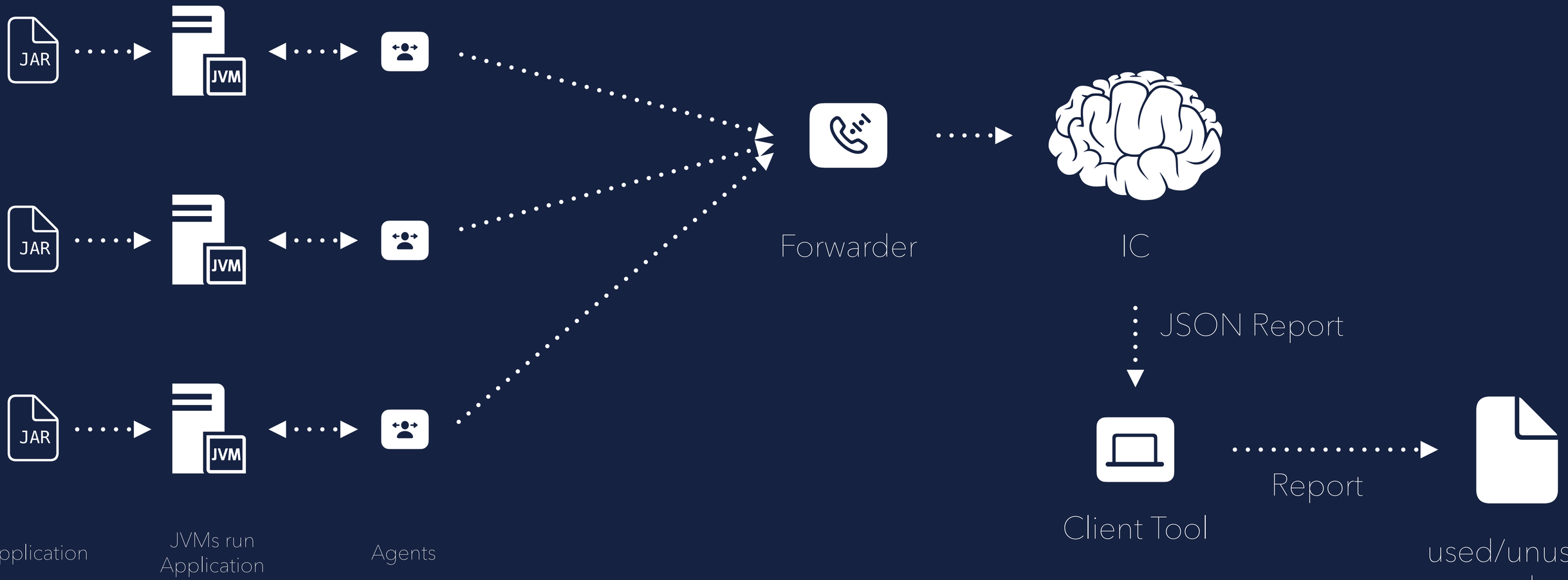
FINDING ZOMBIE CODE...



azul

INTELLIGENCE CLOUD

FINDING ZOMBIE CODE...



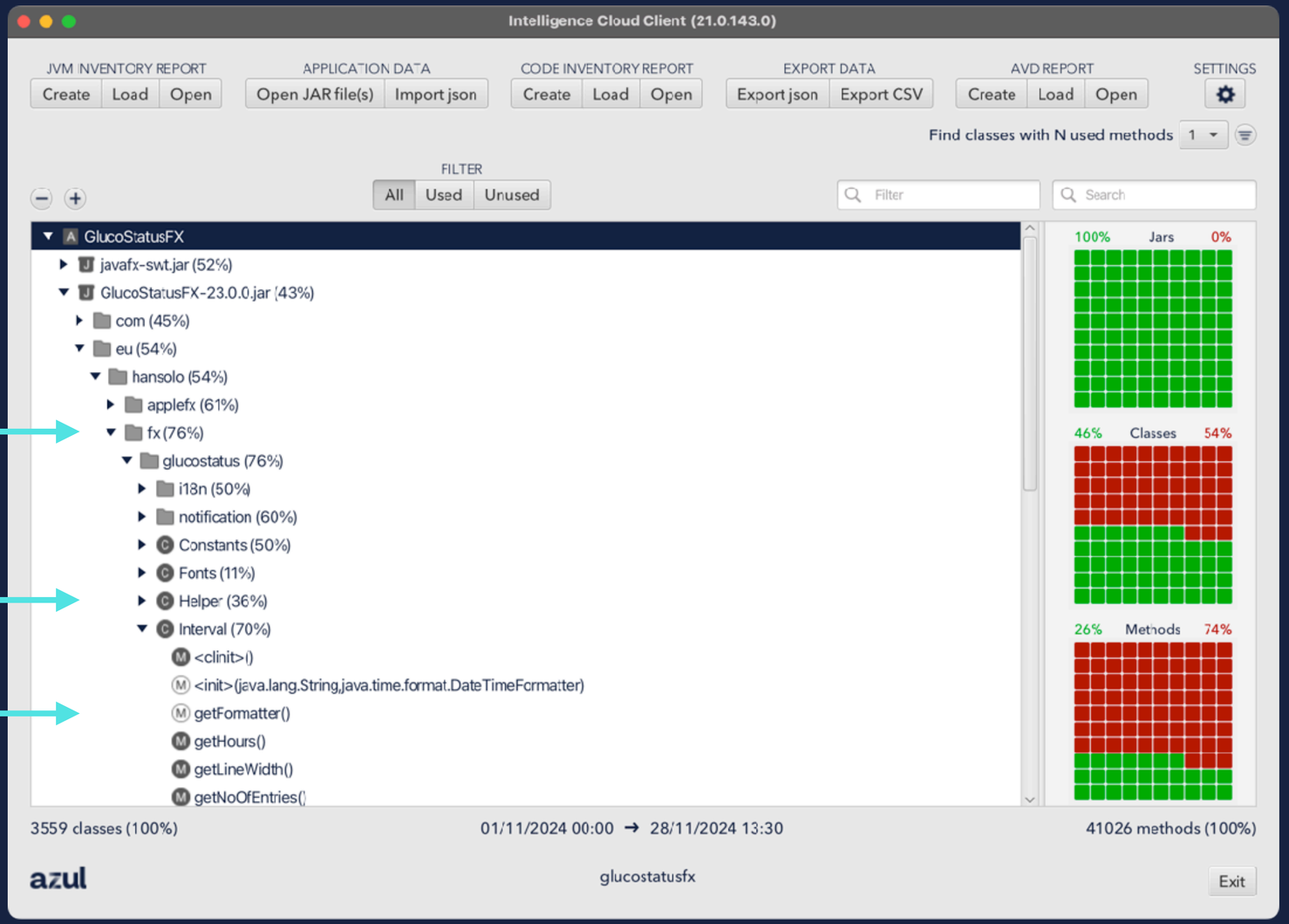
INTELLIGENCE CLOUD

FINDING ZOMBIE CODE...

76% of package used

36% of class used

Unused method



INTELLIGENCE CLOUD

WHAT YOU GET...

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

MORE INFO...

https://www.azul.com/products/intelligence-cloud/

azul

CONCLUSION

azul

CONCLUSION

- Zombie code is real, but hard to find

azul

CONCLUSION

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

azul

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

azul