Continuous Integration

Prof. Sven Apel

Universität des Saarlandes

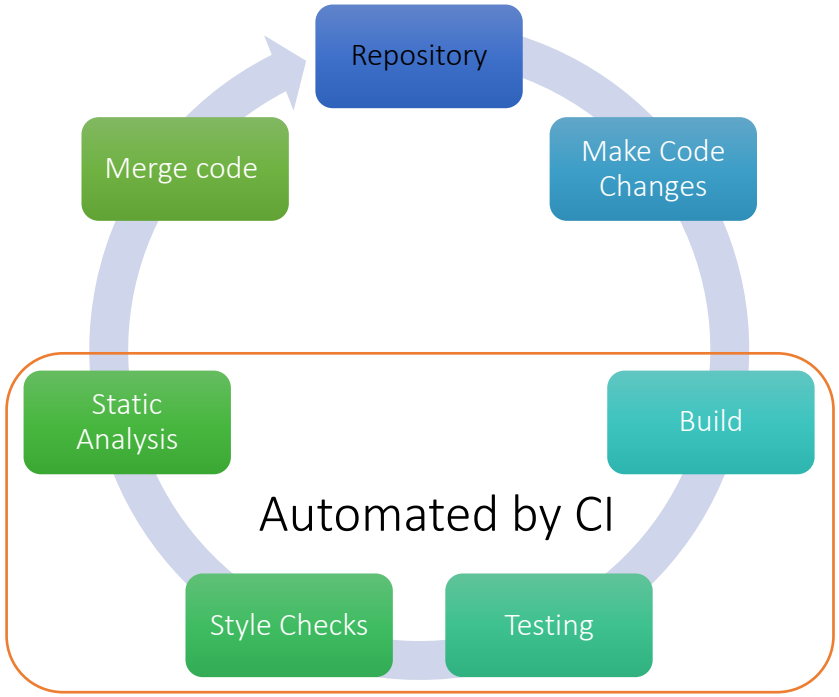
Continuous Integration

Continuous Integration (CI) is a software development practice where developers integrate code changes into a shared repository frequently. These changes are built, tested, and checked automatically before integration.

Goals:

- Enabling *rapid development* and *higher code quality*
- Catching *bugs* as early as possible
- Ensuring that the current state of the codebase *works*

Building blocks



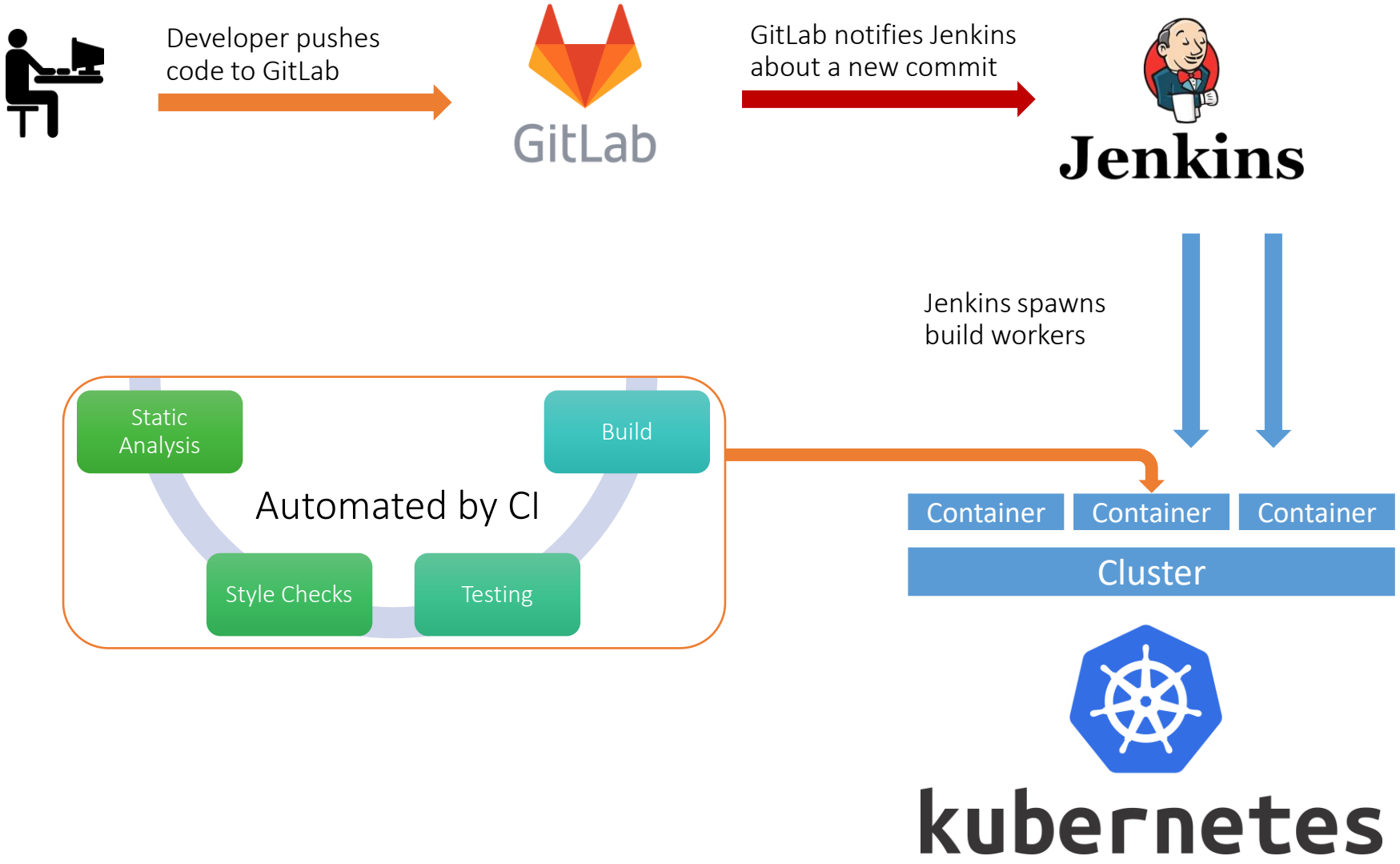
Build: the software project is built automatically.

Testing: tests are run automatically.

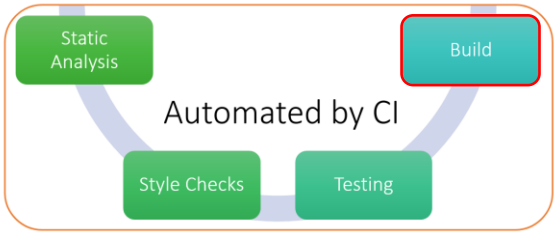
Style Checks: configured style checks enforce project style guidelines.

Static Analysis: static-analysis tools check for bugs.

What is the CI setup?



Build



What is a build system?

On a high level a build system describes how source code is transformed into executable binaries.

Why do we need a build system?

Build systems encapsulate the knowledge of *how software is built*:

- Removes burden to know details about the build process from developers.
- With versioning, allows one to track changes in how the software was build.
- Allows automation of the build process.

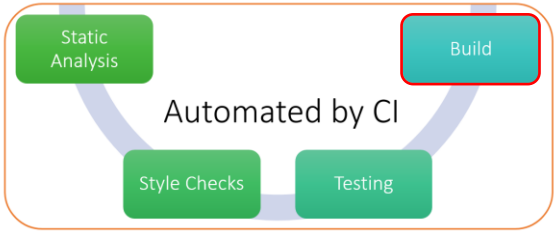
Build systems handle internal and external *dependency management*.

Can automatically *speed up* the building of software by using computing infrastructure in the background.

- Parallel builds -> distributing the building onto multiple machines
- Caching and updating build artefacts, libraries, and binaries used by other teams/developers (e.g., Nexus)

Build systems handle *tool integration*.

Build



Build systems for Java:

Gradle

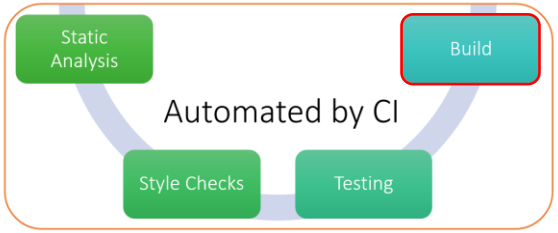
Maven

Ant

Maven™

<APACHE ANT>

Build



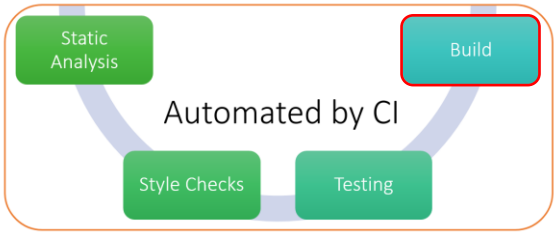
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Project

```
VLExample ~/IdeaProjects/
  .gradle
  .idea
  build
    classes
    generated
    libs
    reports
      checkstyle
        main.html
      spotbugs
        main.html
    tmp
  config
    checkstyle.xml
    spotbugs_exclude.xml
    spotbugs_include.xml
  gradle
  src
    main
      java
        de.unisaarland.se.so
          Test
      resources
    test
      java
      resources
  build.gradle
  gradlew
  gradlew.bat
  settings.gradle
External Libraries
Scratches and Consoles
```

```
import com.github.spotbugs.SpotBugsTask

plugins {
    id 'java'
    id 'checkstyle'
    id "com.github.spotbugs" version "2.0.0"
}

group 'de.unisaarland.se.sopra'
version '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.12'

    checkstyle "com.puppycrawl.tools:checkstyle:8.23"

    spotbugs "com.github.spotbugs:spotbugs:3.1.12"
    spotbugsPlugins "com.mebigfatguy.sb-contrib:sb-contrib:7.4.6"
}

sourceSets {
    main {
        java {
            srcDir 'src/main/java'
        }
        resources {
            srcDir 'src/main/resources'
        }
    }
}

checkstyle {
    toolVersion = "8.23"
    ignoreFailures = true
}

tasks.withType(Checkstyle) {
    reports {
        xml.enabled false
        html.enabled true
    }
    configFile file("$projectDir/config/checkstyle.xml")
}

spotbugs {
    toolVersion = "3.1.12"
    ignoreFailures = true
    effort = "max"
    includeFilter = file("$projectDir/config/spotbugs_include.xml")
    excludeFilter = file("$projectDir/config/spotbugs_exclude.xml")
}
```

Build



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

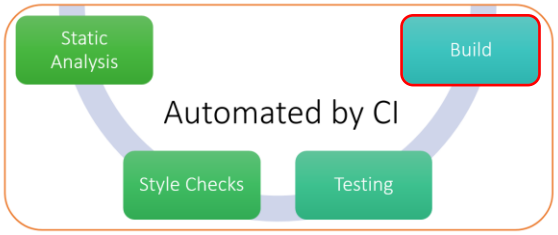
Project

```
VLExample ~/IdeaProjects/
.gradle
.idea
build
  classes
  generated
  libs
  reports
    checkstyle
      main.html
    spotbugs
      main.html
  tmp
config
  checkstyle.xml
  spotbugs_exclude.xml
  spotbugs_include.xml
gradle
src
  main
    java
      de.unisaarland.se.so
        Test
    resources
  test
    java
    resources
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
```

```
import com.github.spotbugs.SpotBugsTask

plugins {
    id 'java'
    id 'checkstyle'
    id "com.github.spotbugs" version "2.0.0"
}

group 'de.unisaarland.se.sopra'
version '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.1
    checkstyle "com.puppycrawl.tools:checkstyle:8.23"
    spotbugs "com.github.spotbugs:spotbugs:3.1.12"
    spotbugsPlugins "com.mebigfatguy.sb-contrib:sb-contrib:7
}

sourceSets {
    main {
        java {
            srcDir 'src/main/java'
        }
        resources {
            srcDir 'src/main/resources'
        }
    }
}

checkstyle {
    toolVersion = "8.23"
    ignoreFailures = true
}

tasks.withType(Checkstyle) {
    reports {
        xml.enabled false
        html.enabled true
    }
    configFile file("$projectDir/config/checkstyle.xml")
}

spotbugs {
    toolVersion = "3.1.12"
    ignoreFailures = true
    effort = "max"
    includeFilter = file("$projectDir/config/spotbugs_include.xml")
    excludeFilter = file("$projectDir/config/spotbugs_exclude.xml")
}
```


```
dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.12'

    checkstyle "com.puppycrawl.tools:checkstyle:8.23"

    spotbugs "com.github.spotbugs:spotbugs:3.1.12"
    spotbugsPlugins "com.mebigfatguy.sb-contrib:sb-contrib:7.4.6"
}

sourceSets {
    main {
        java {
            srcDir 'src/main/java'
        }
        resources {
            srcDir 'src/main/resources'
        }
    }
}
```

Build



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Project

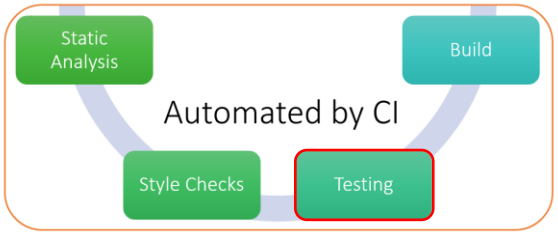
```
VLExample ~/IdeaProjects/
.gradle
.idea
build
  classes
  generated
  libs
  reports
    checkstyle
      main.html
    spotbugs
      main.html
  tmp
config
  checkstyle.xml
  spotbugs_exclude.xml
  spotbugs_include.xml
gradle
src
  main
    java
      de.unisaarland.se.so
        Test
    resources
  test
    java
    resources
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
```

```groovy
import com.github.spotbugs.SpotBugsTask

plugins {
    id 'java'
    id 'checkstyle'
    id "com.github.spotbugs" version "2.0.0"
}

group 'de.unisaarland.se.sopra'
version '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.12'

    checkstyle "com.puppycrawl.tools:checkstyle:8.23"

    spotbugs "com.github.spotbugs:spotbugs:3.1.12"
    spotbugsPlugins "com.mebigfatguy.sb-contrib:sb-contrib:7.4.6"
}

sourceSets {
    main {
        java {
            srcDir 'src/main/java'
        }
        resources {
            srcDir 'src/main/resources'
        }
    }
}
```

```groovy
checkstyle {
    toolVersion = "8.23"
    ignoreFailures = true
}

tasks.withType(Checkstyle) {
    reports {
        xml.enabled false
        html.enabled true
    }
    configFile file("$projectDir/config/checkstyle.xml")
}

spotbugs {
    toolVersion = "3.1.12"
    ignoreFailures = true
    effort = "max"
    includeFilter = file("$projectDir/config/spotbugs_include.xml")
    excludeFilter = file("$projectDir/config/spotbugs_exclude.xml")
}
```

```groovy
checkstyle {
    toolVersion = "8.23"
    ignoreFailures = true
}

tasks.withType(Checkstyle) {
    reports {
        xml.enabled false
        html.enabled true
    }
    configFile file("$projectDir/config/checkstyle.xml")
}

tasks.withType(SpotBugsTask) {
    reports {
        xml.enabled = false
        html.enabled = true
    }
    pluginClasspath = project.configurations.spotbugsPlugins
    classpath = sourceSets.main.output + configurations.compile
}
```

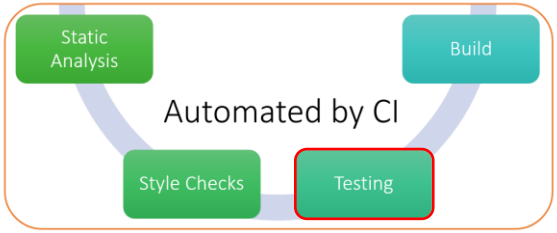
Testing



Important aspects of testing, regarding CI:

- Automate all tests
- Make tests deterministic and independent
- Split up tests regarding different sizes/requirements
  - Small tests <- easy and fast to run
  - Medium sized tests <- take up more resources
  - Large scale tests <- require larger setups/time

Testing

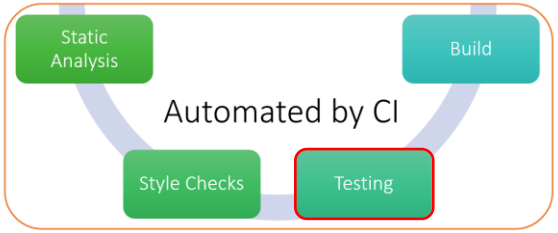



```
int calculateTheAnswerToEverything(int Question) {
    if (Question == 42) {
        return 1337;
    }
    return 42;
}
```

Generate
- Constructor
- toString()
- Override Methods... Ctrl+O
- Test...
- Copyright

```
package de.unisaarland.se.sopra;

import ...

public class ExampleClassTest {
    private ExampleClass testInstance;

    @Before
    public void setUp() throws Exception {
        testInstance = new ExampleClass();
    }

    @Test
    public void calculateTheAnswerToEverything() {
        assertEquals( expected: 42, testInstance.calculateTheAnswerToEverything( Question: 4));
        assertEquals( expected: 42, testInstance.calculateTheAnswerToEverything( Question: 13));
        assertEquals( expected: 42, testInstance.calculateTheAnswerToEverything( Question: 2020));
    }

    @Test
    public void calculateTheAnswerToEverythingSpecialInput() {
        assertEquals( expected: 21, testInstance.calculateTheAnswerToEverything( Question: 42));
    }
}
```

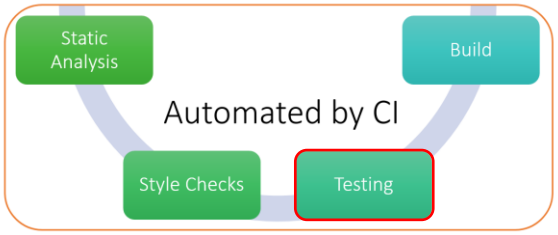
Testing




```java
public class ExampleClassTest {
    private ExampleClass testInstance;

    @Before
    public void setUp() throws Exception {
        testInstance = new ExampleClass();
    }

    @Test
    public void calculateTheAnswerToEverything() {
        assertEquals( expected: 42, testInstance.calculateTheAnswerToEverything( Question: 4));
        assertEquals( expected: 42, testInstance.calculateTheAnswerToEverything( Question: 13));
        assertEquals( expected: 42, testInstance.calculateTheAnswerToEverything( Question: 2020));
    }

    @Test
    public void calculateTheAnswerToEverythingSpecialInput() {
        assertEquals( expected: 21, testInstance.calculateTheAnswerToEverything( Question: 42));
    }
}
```

```
Tests failed: 1, passed: 1 of 2 tests – 1 ms

Test Results 1 ms
de.unisaarland.se.sopra.ExampleClassTest 1 ms
calculateTheAnswerToEverythingSpecialInput 1 ms

Testing started at 5:47 PM ...

> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE
> Task :compileTestJava
> Task :processTestResources NO-SOURCE
> Task :testClasses
> Task :test FAILED

expected:<21> but was:<1337>
Expected :21
Actual   :1337
<Click to see difference>

java.lang.AssertionError Create breakpoint : expected:<21> but was:<1337> <1 internal call>
    at org.junit.Assert.failNotEquals(Assert.java:834) <2 internal calls>
    at de.unisaarland.se.sopra.ExampleClassTest.calculateTheAnswerToEverythingSpecialInput(ExampleClassTest.java:25) <47 internal calls>
    at java.base/java.lang.Thread.run(Thread.java:834)
```

Testing



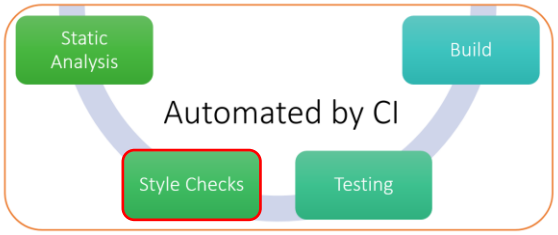
If test execution is automated through the build system, CI workers can run the test suite and report back to the user.

```
> ./gradlew test

> Task :test FAILED

de.unisaarland.se.sopra.ExampleClassTest > calculateTheAnswerToEverythingSpecialInput FAILED
    java.lang.AssertionError at ExampleClassTest.java:25

2 tests completed, 1 failed

FAILURE: Build failed with an exception.
```

After pushing the fixed code, the CI confirms that everything works now.

```
> ./gradlew test

BUILD SUCCESSFUL in 372ms
3 actionable tasks: 3 up-to-date
```

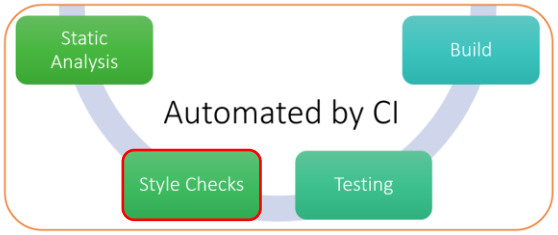
Style Checks



Code is more often read than written! So, optimize for the reader and make your code easier to read and understand.

```
    void doStuff(
            boolean ExtraWork
  )      {
                    System.out.println("Doing stuff...");
        returnStuff();
        if( ExtraWork )
              System.out.println("Doing more stuff...");
System.out.println("This makes me cry...");}
```

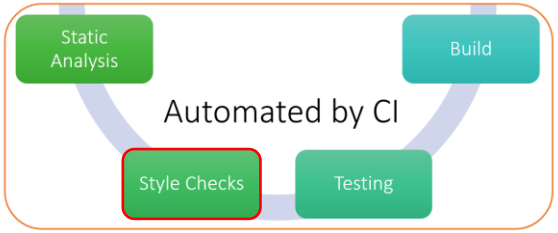
Style Checks



Code is more often read than written! So, optimize for the reader and make your code easier to read and understand.

```
void doStuff(boolean ExtraWork) {
    System.out.println("Doing stuff...");
    returnStuff();

    if (ExtraWork) {
        System.out.println("Doing more stuff...");
    }

    System.out.println("Easier to understand :)");
}
```

Style Checks

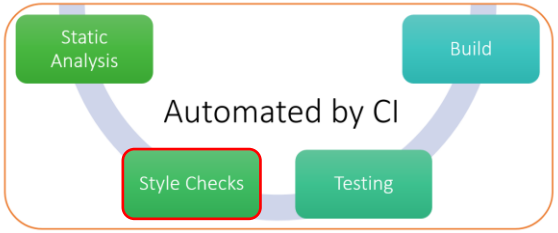


Why do we need coding style guides and rules?

Style guides are guiding principles to:

- make code *consistent*
- make code *easier to read*, *understand*, and *change*
- make codebases *easier to learn*
- helps to *avoid error-prone* and *surprising* constructs

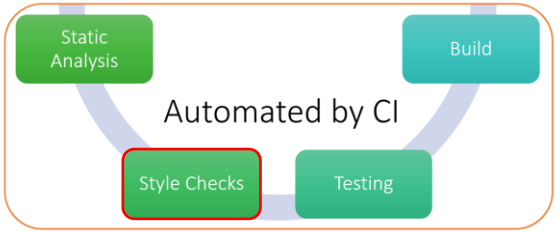
Style Checks



However, style guides also have a costs and should, therefore, not be arbitrary:

- *slowing down* developers
- making it *harder to integrate* new hires into a team
- varying style guides can *confuse* developers

Style Checks

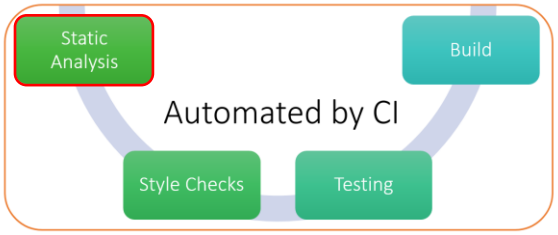


How should we run style checks?

Style checks should be automated, preferably through the build system.

This allows developers to run them *locally* in their IDEs but also enables *CI workers* to *enforce* them before merging new code into the codebase.

Static Analysis

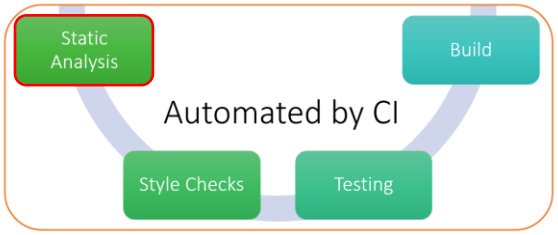


In static analysis, we use programs to analyze the source code without executing.

The goals of running static analysis are:

- Detecting *bugs*, *antipatterns*, and other problems before the new code is integrated into the codebase.
- To *educate* developers about best practices or antipatterns.
- Enable *automatic fixers* to generate code changes that clean up the codebase to reduce technical dept.

Static Analysis



Summary

Warning Type	Number
Dodgy code Warnings	1
Total	**1**

Dodgy code Warnings

Code	Warning
RV	Return value of returnStuff() ignored, but method has no side effect Bug type RV_RETURN_VALUE_IGNORED_NO_SIDE_EFFECT (click for details) In class de.unisaarland.se.sopra.Test In method de.unisaarland.se.sopra.Test.doStuff() Called method de.unisaarland.se.sopra.Test.returnStuff() At Test.java:[line 13]

Details

RV_RETURN_VALUE_IGNORED_NO_SIDE_EFFECT: Return value of method without side effect is ignored

This code calls a method and ignores the return value. However our analysis shows that the method (including its implementations in subclasses if any) does not produce any effect other than return value. Thus this call can be removed.

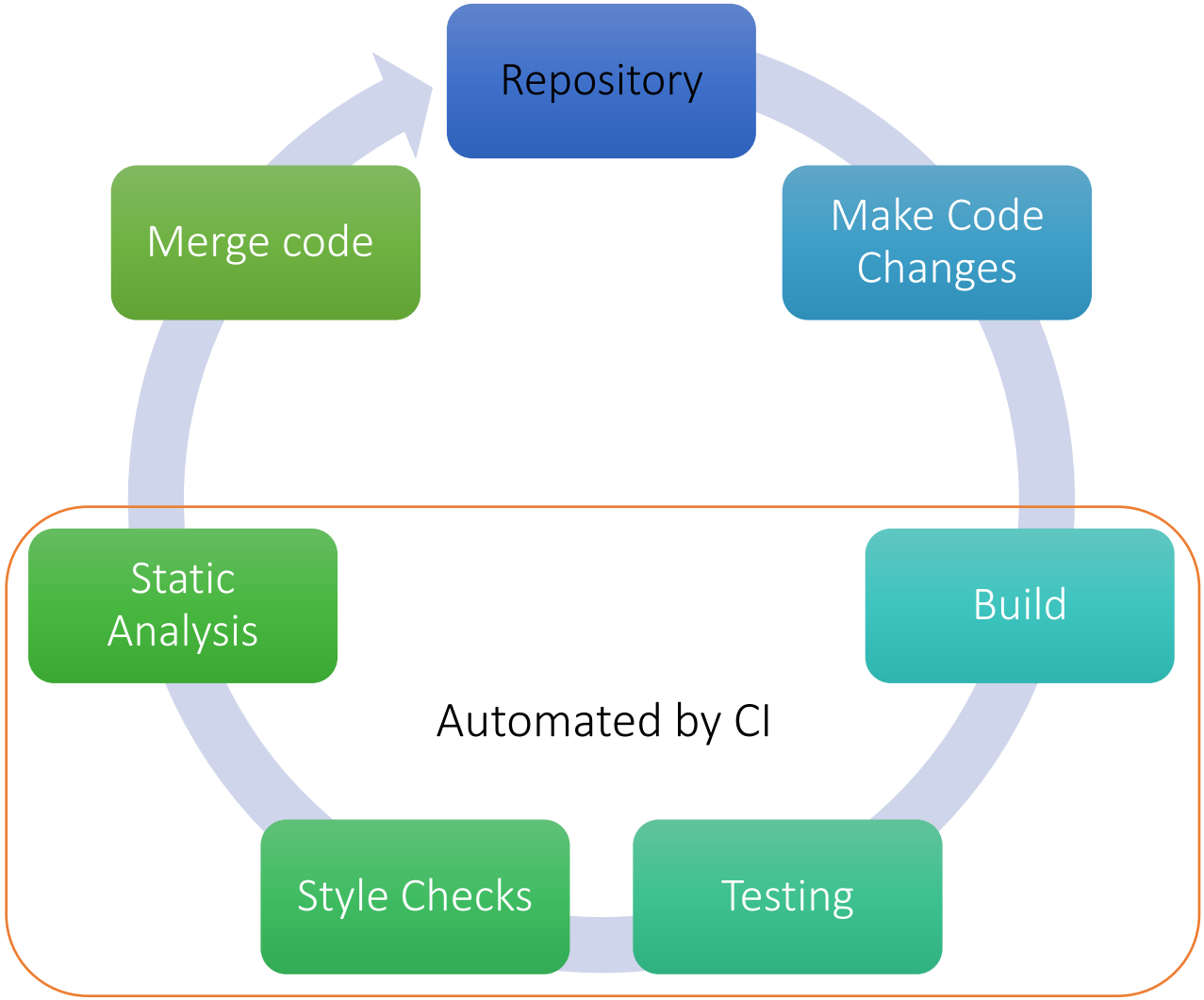
We are trying to reduce the false positives as much as possible, but in some cases this warning might be wrong. Common false-positive cases include:

- The method is designed to be overridden and produce a side effect in other projects which are out of the scope of the analysis.

- The method is called to trigger the class loading which may have a side effect.

- The method is called just to get some exception.

If you feel that our assumption is incorrect, you can use a @CheckReturnValue annotation to instruct SpotBugs that ignoring the return value of this method is acceptable.


Repository
Make Code Changes
Build
Testing
Style Checks
Static Analysis
Merge code
Automated by CI