Boosting Developer Productivity Through Better Modularity

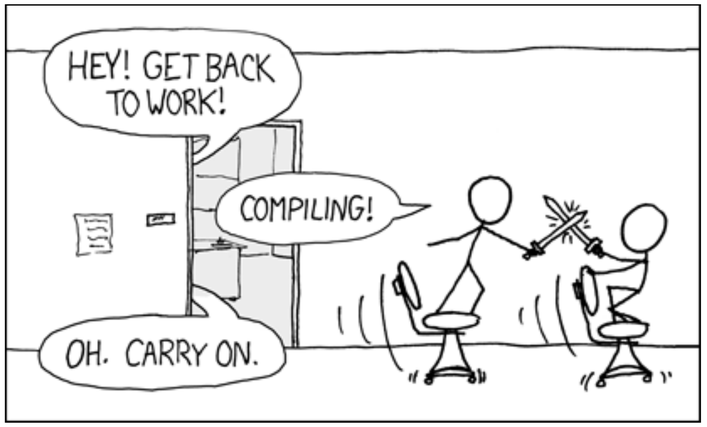
Jendrik Johannes

github.com/jjohannes

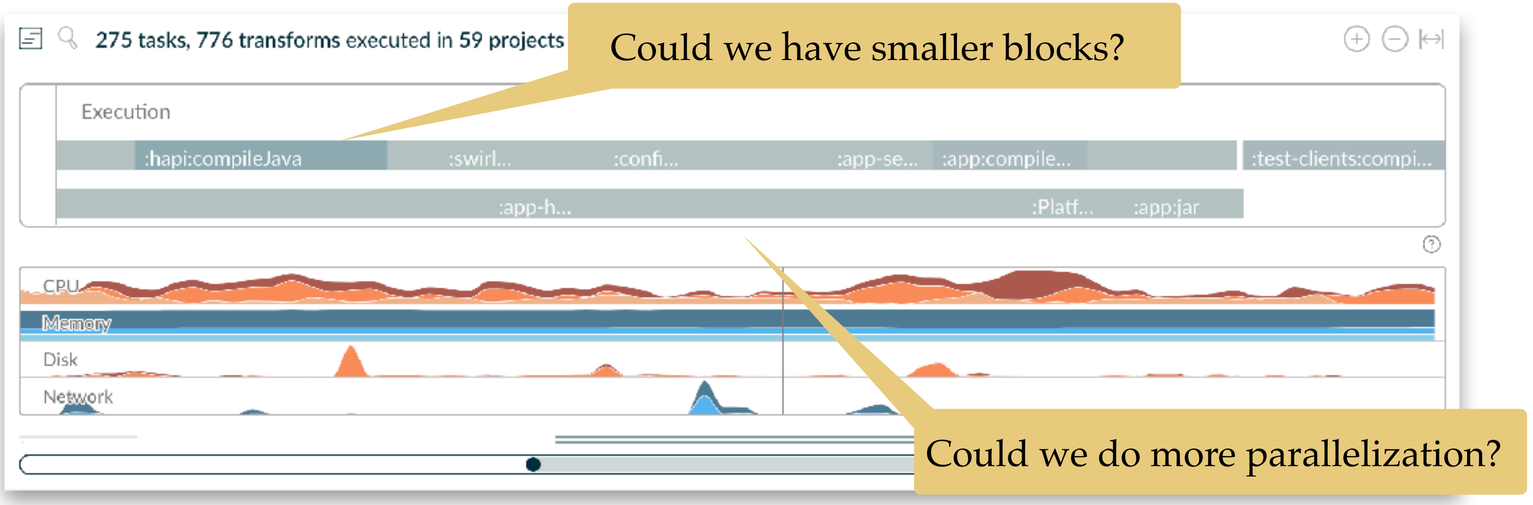
DPE Summit 2024

What makes us productive?

…Build Performance?



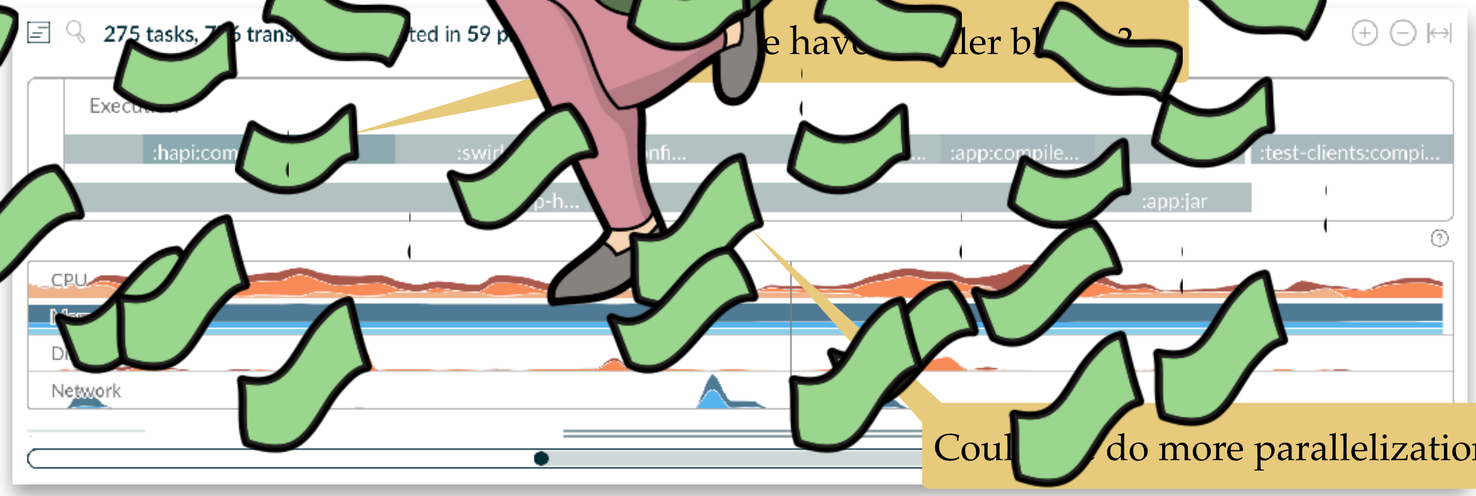
xkcd.com/303/



What makes us productive?

…Build Perfo[illegible]ance?

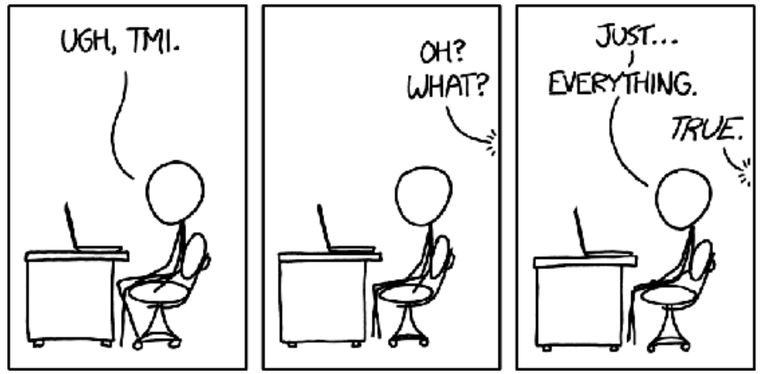
xkcd.com/303/



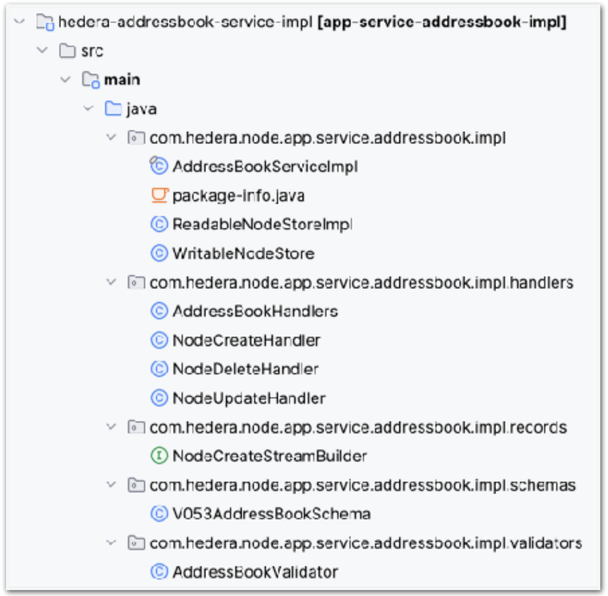
Coul[illegible] do more parallelization?

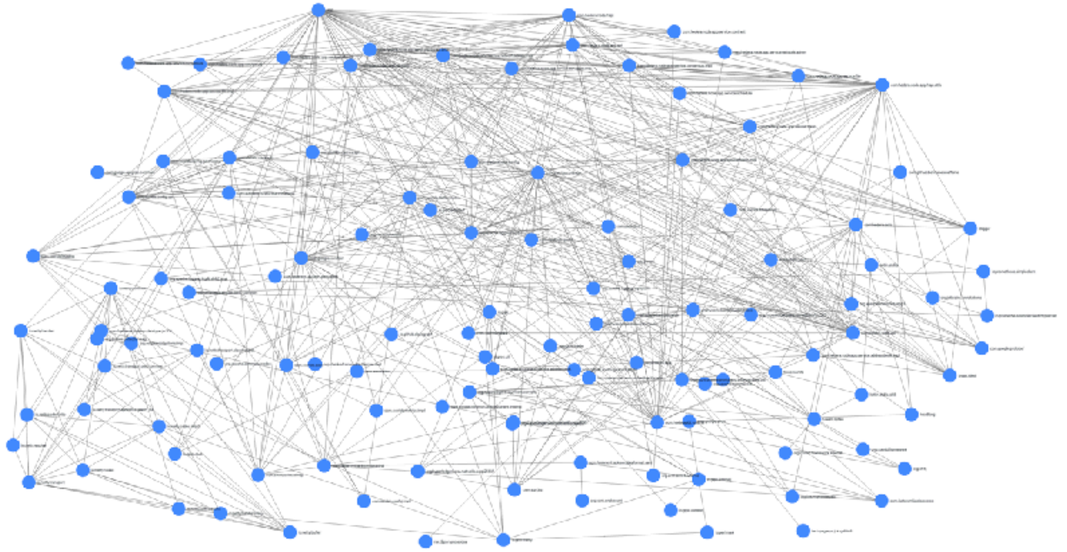
What makes us productive?

…Comprehensibility?



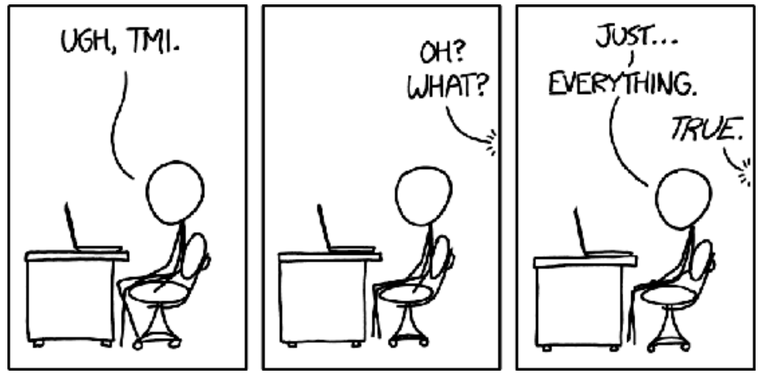
xkcd.com/1369/



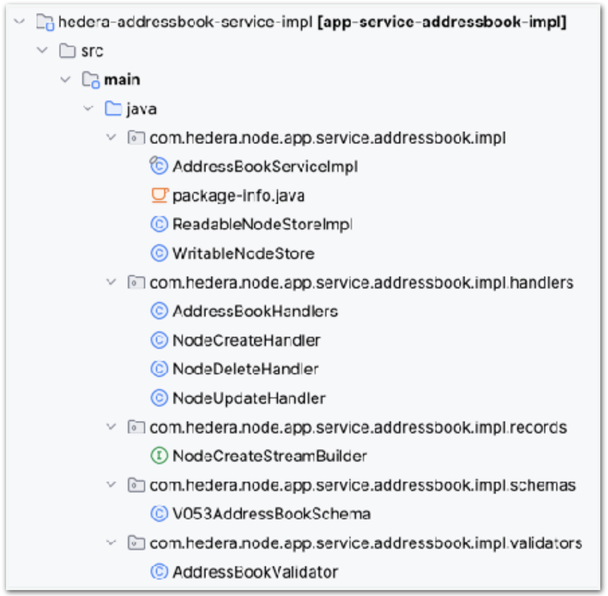


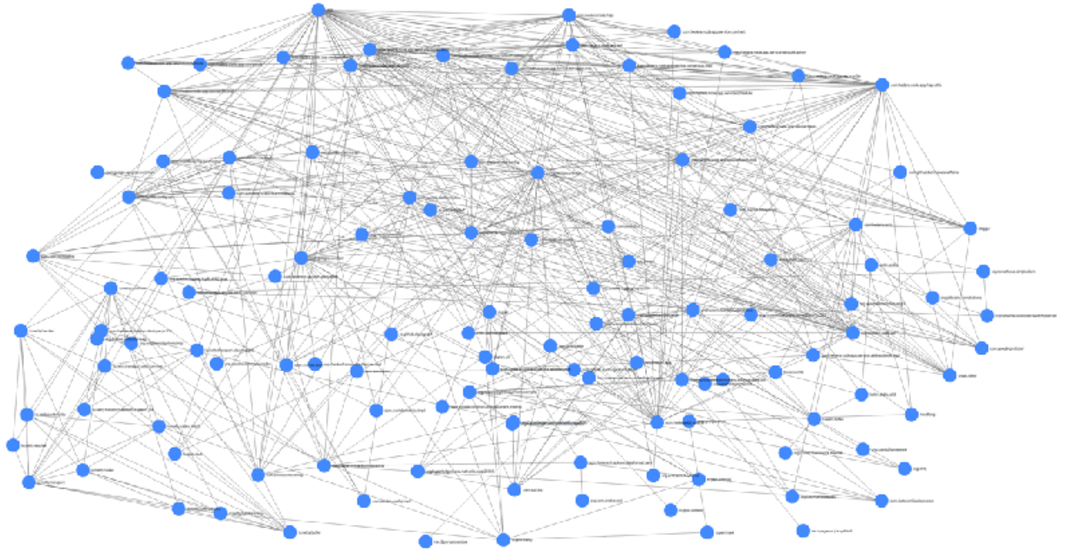
What makes us productive?

…Comprehensibility?

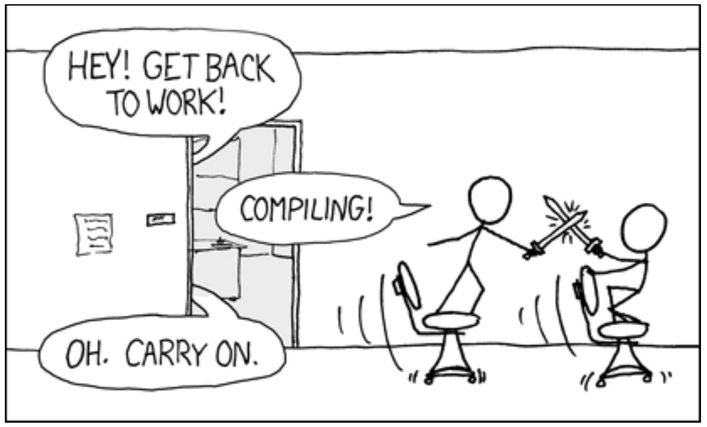


xkcd.com/1369/

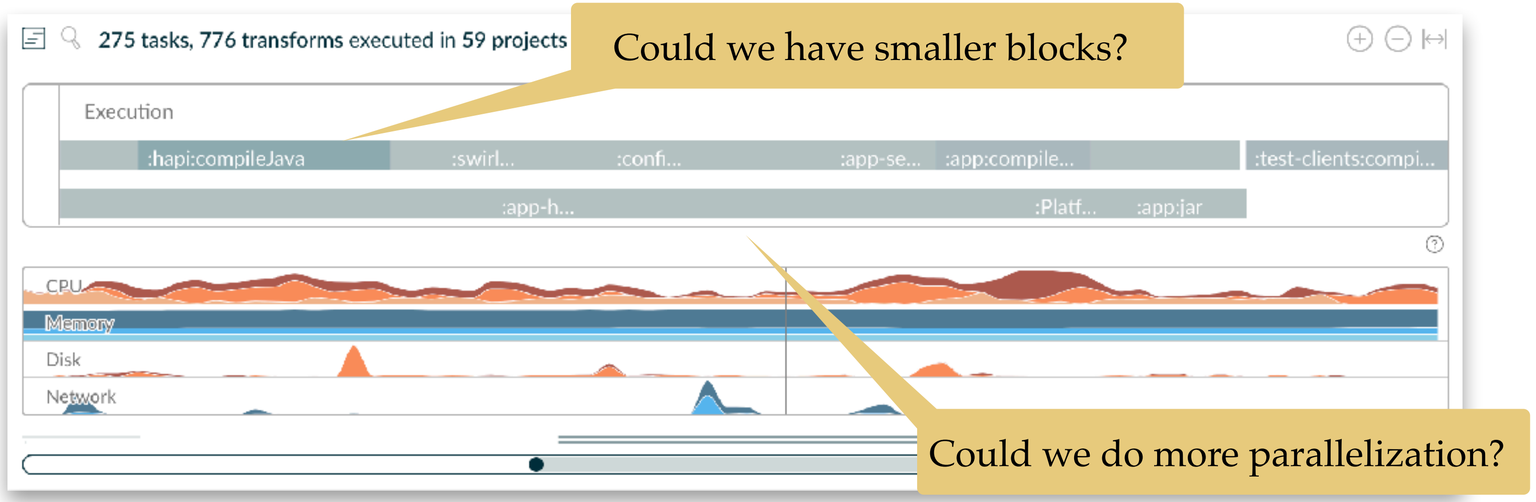


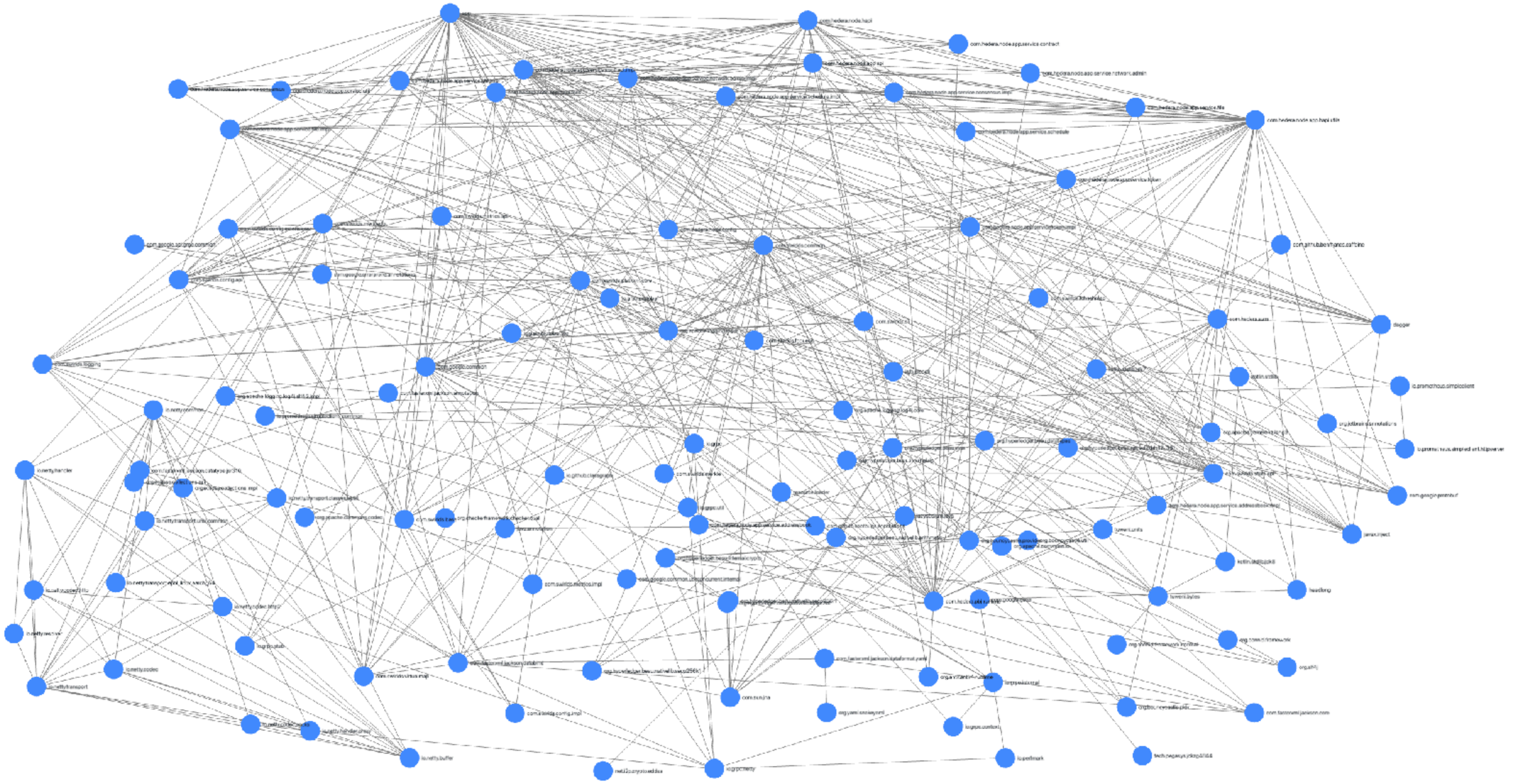


…Build Performance?



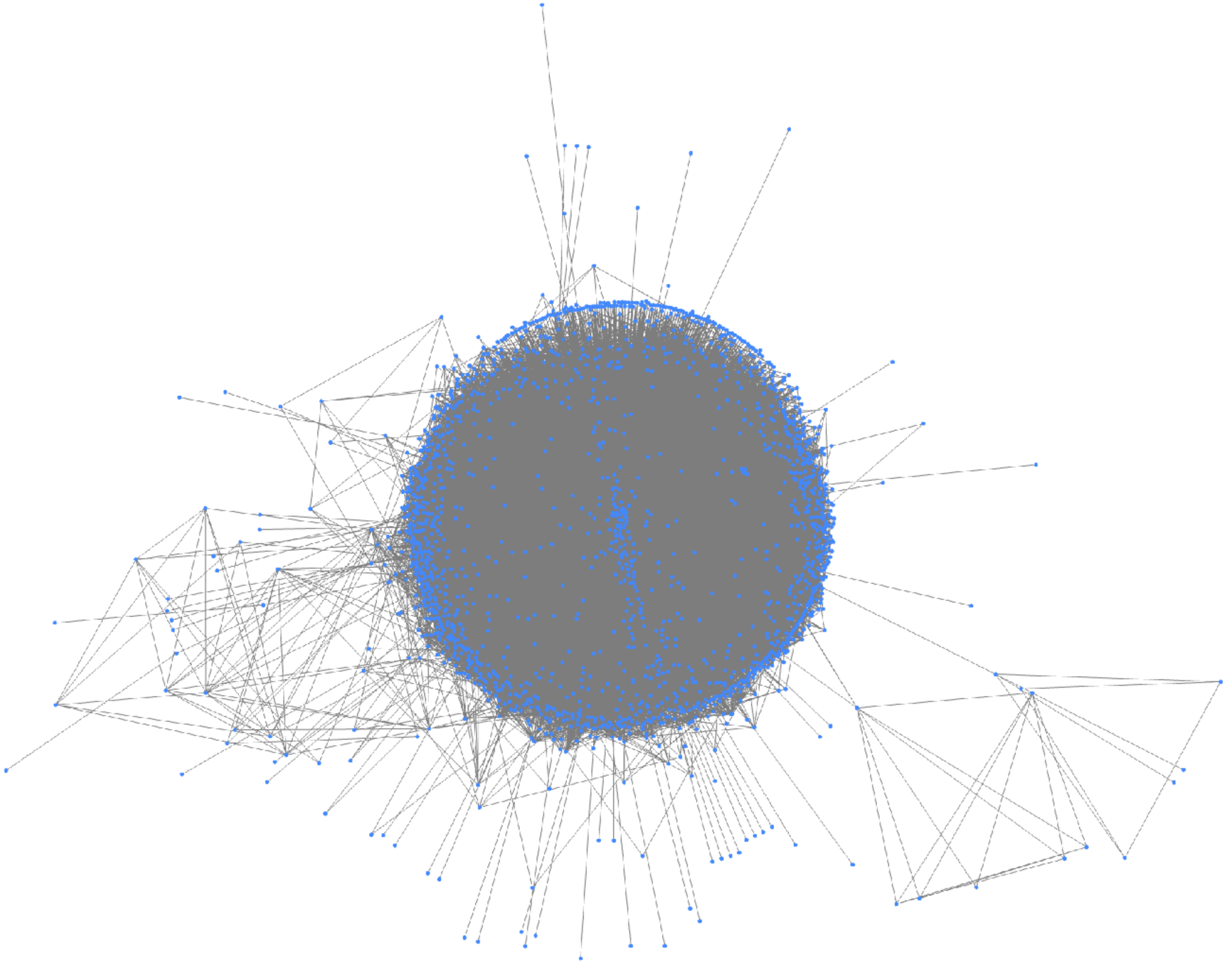
xkcd.com/303/





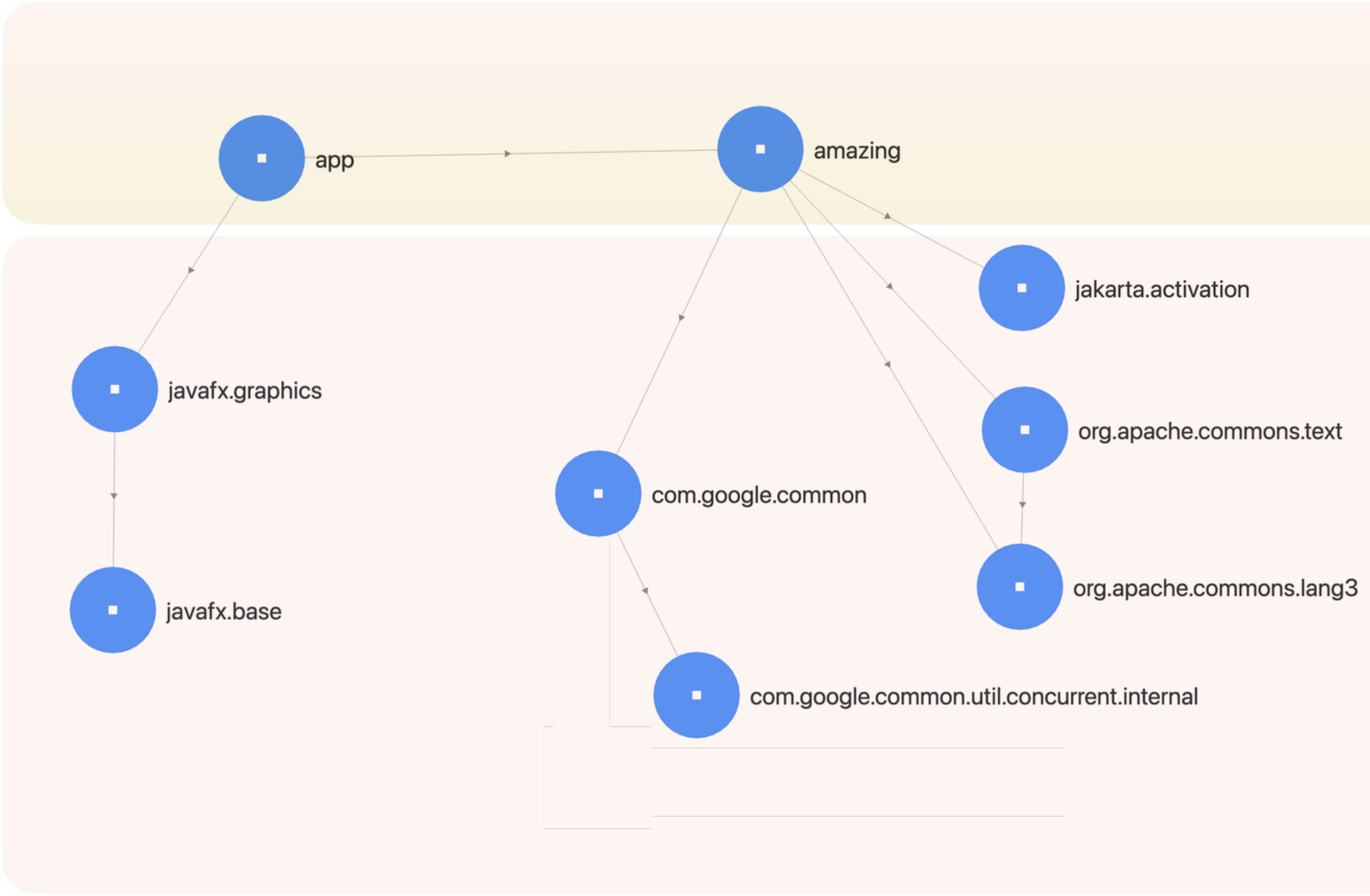
github.com/hashgraph/hedera-services

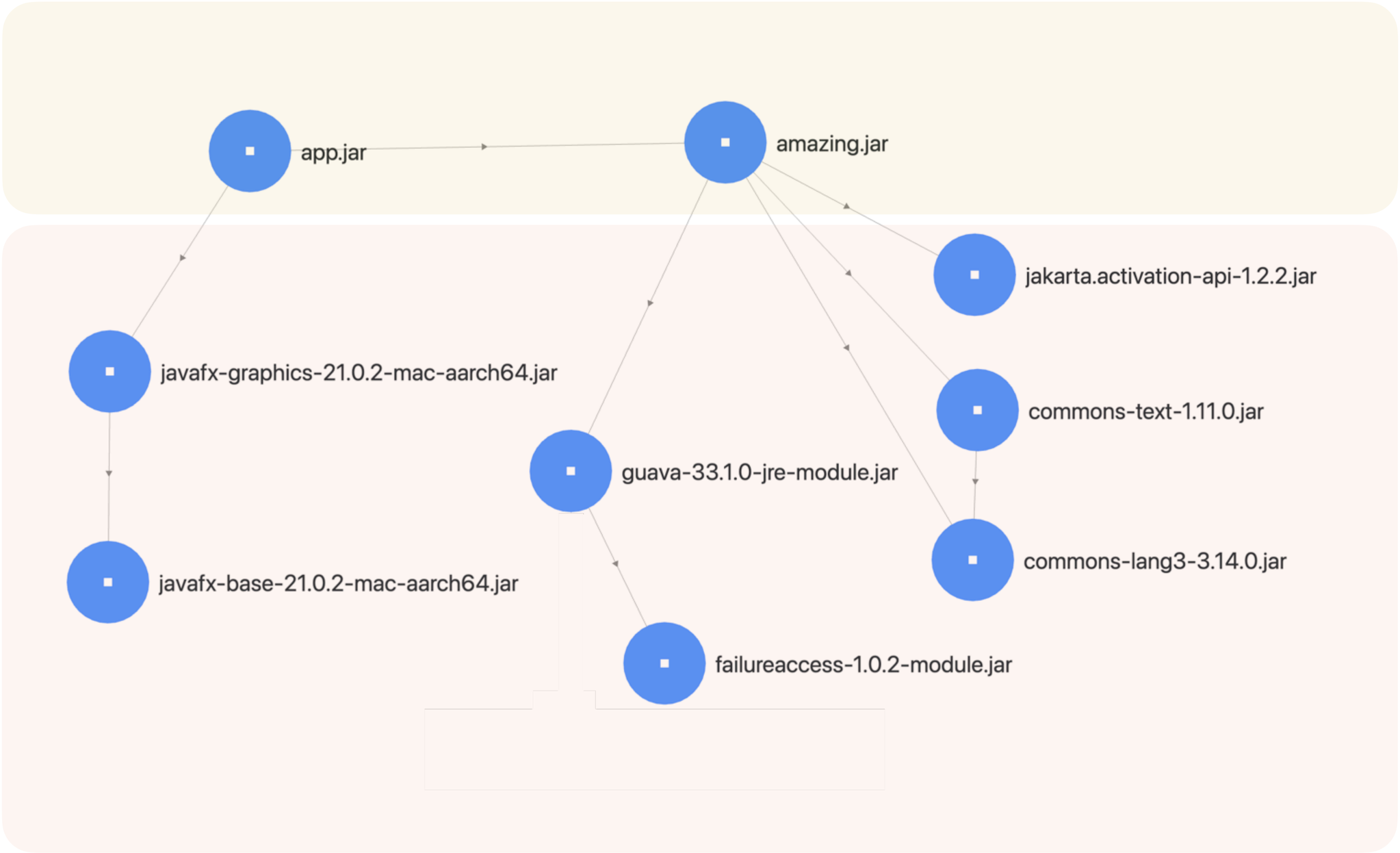
poloclub.github.io/argo-graph-lite/#d5b95db8-a363-46a5-8757-a846e116fa2b



Part 1

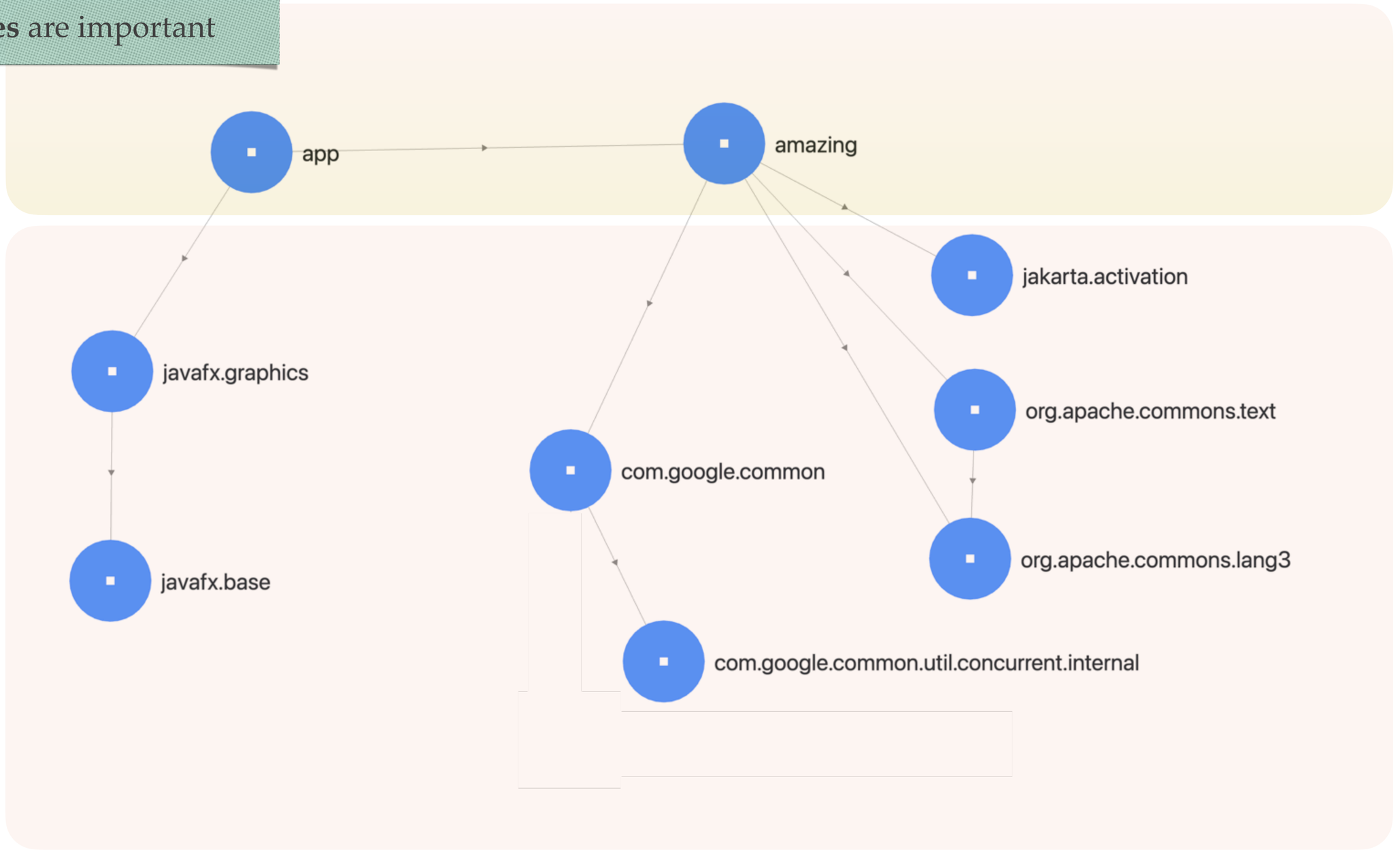
Complexity in **Module Definition**


app
amazing
javafx.graphics
jakarta.activation
org.apache.commons.text
com.google.common
javafx.base
org.apache.commons.lang3
com.google.common.util.concurrent.internal


app.jar
amazing.jar
jakarta.activation-api-1.2.2.jar
javafx-graphics-21.0.2-mac-aarch64.jar
commons-text-1.11.0.jar
guava-33.1.0-jre-module.jar
javafx-base-21.0.2-mac-aarch64.jar
commons-lang3-3.14.0.jar
failureaccess-1.0.2-module.jar

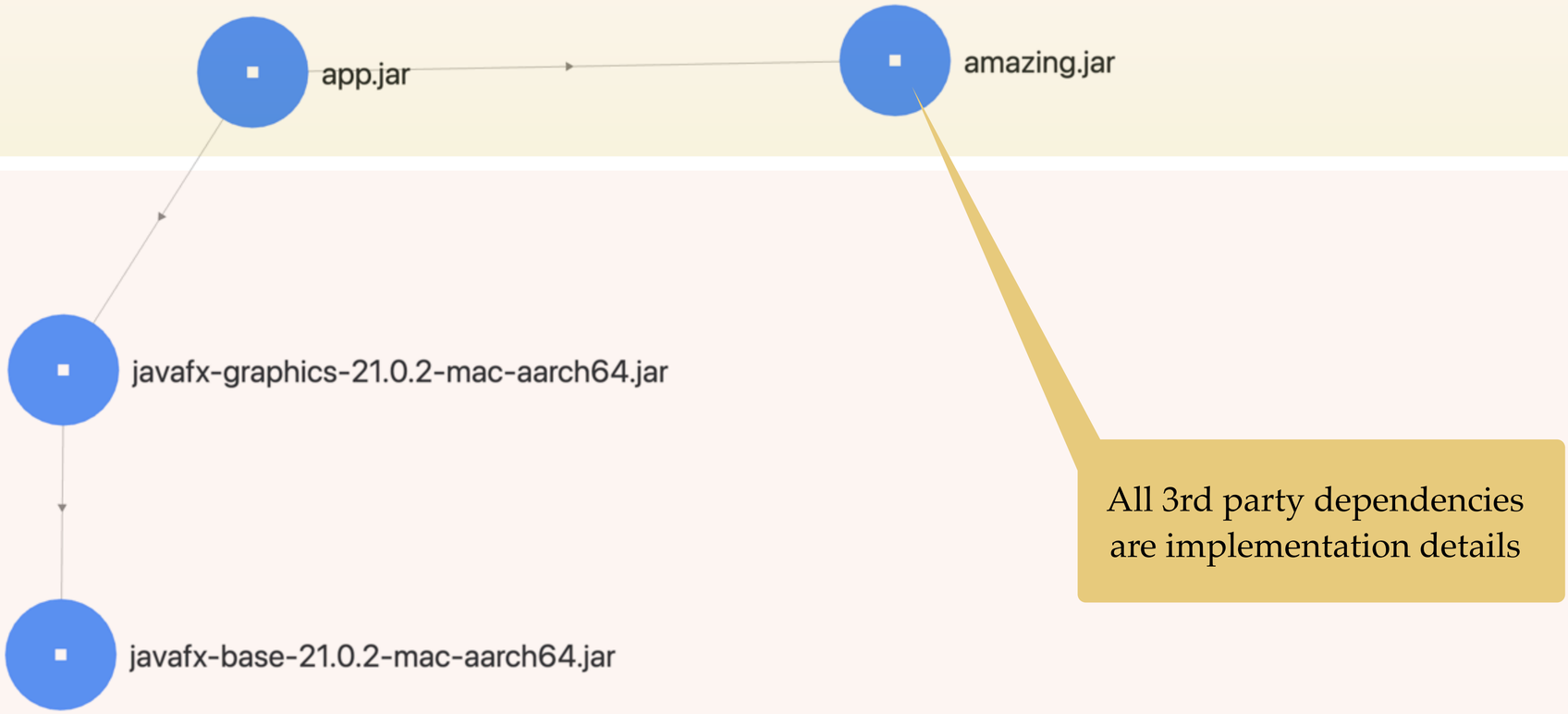
Essential Complexity #1

Scopes are important



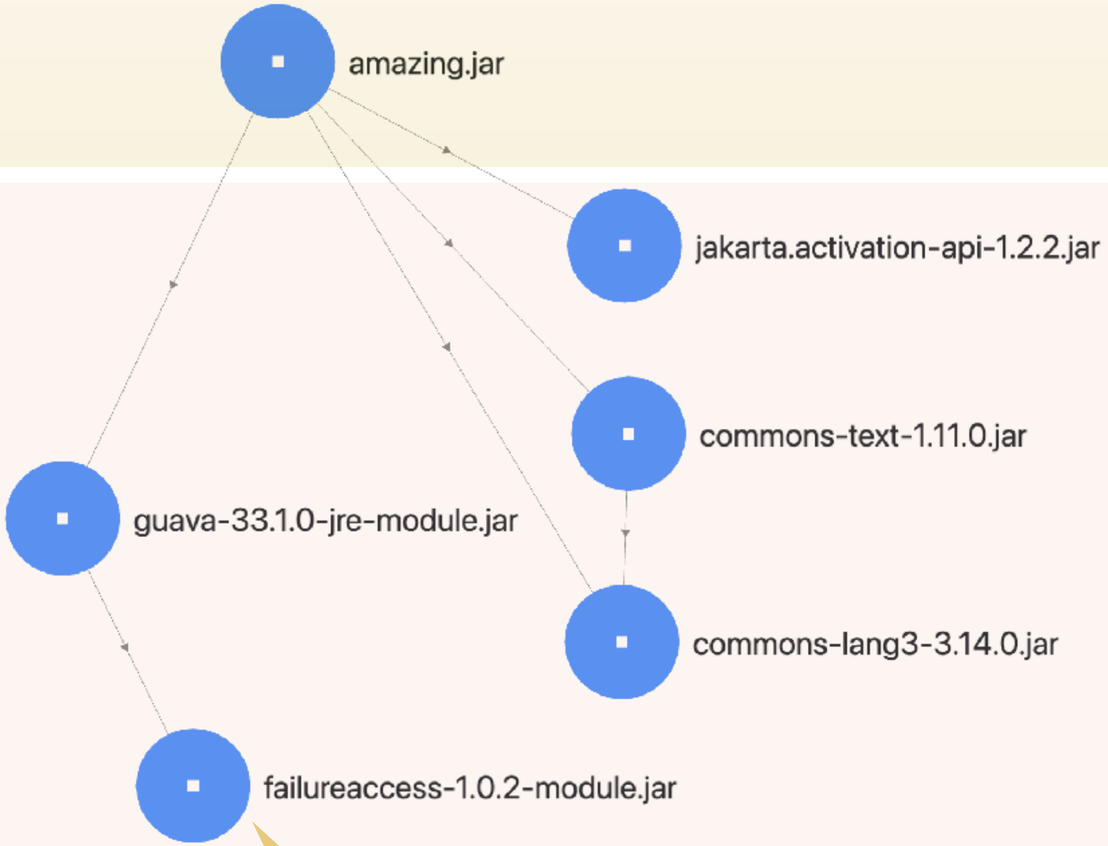
Essential Complexity #1

Scopes are important



Essential Complexity #1

Scopes are important

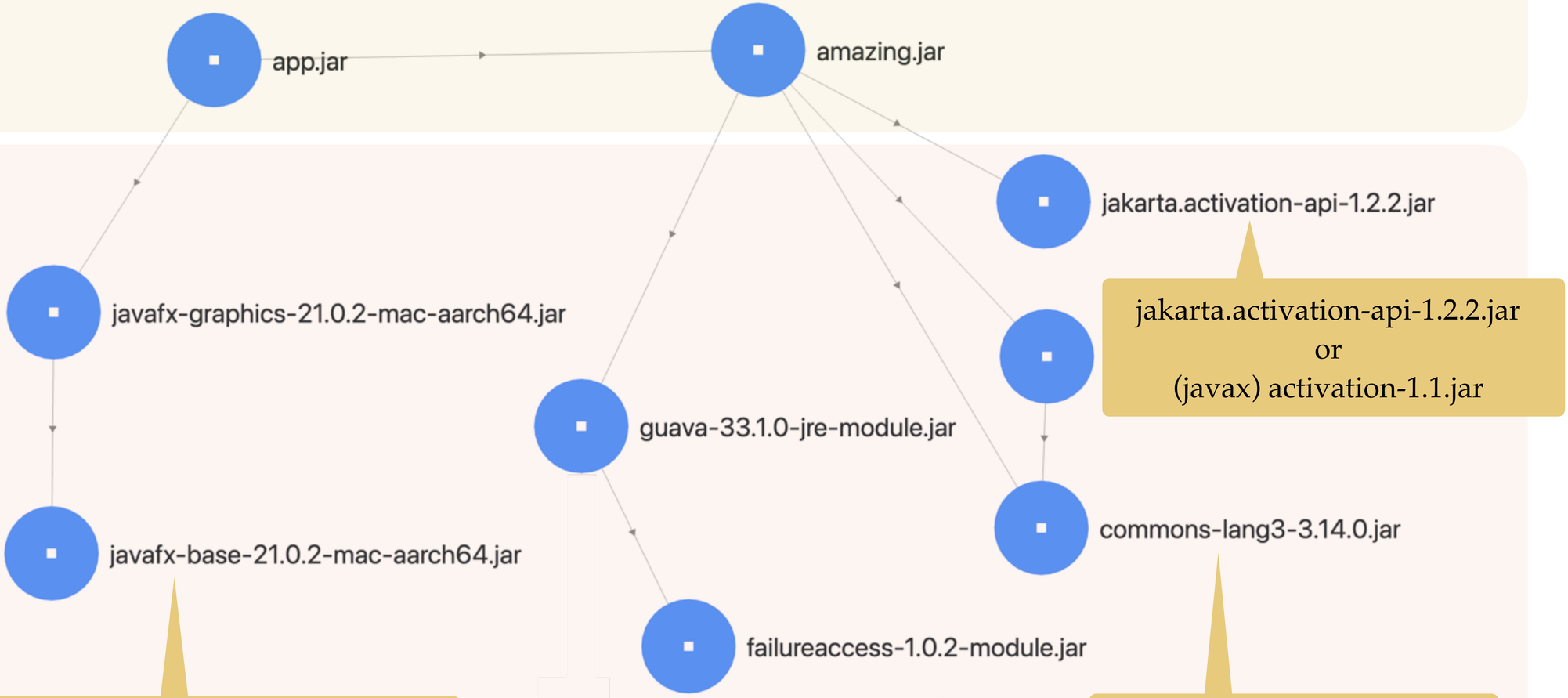


Essential Complexity #1

Scopes are important

Essential Complexity #2

Multiple **Versions/Variants** of 3rd party Modules



jakarta.activation-api-1.2.2.jar
or
(javax) activation-1.1.jar

javafx-base-21.0.2-**mac-aarch64**.jar
or
javafx-base-21.0.2-**win**.jar

commons-lang3-**3.14.0**.jar
or
commons-lang3-**3.15.0**.jar

Essential Complexity #1

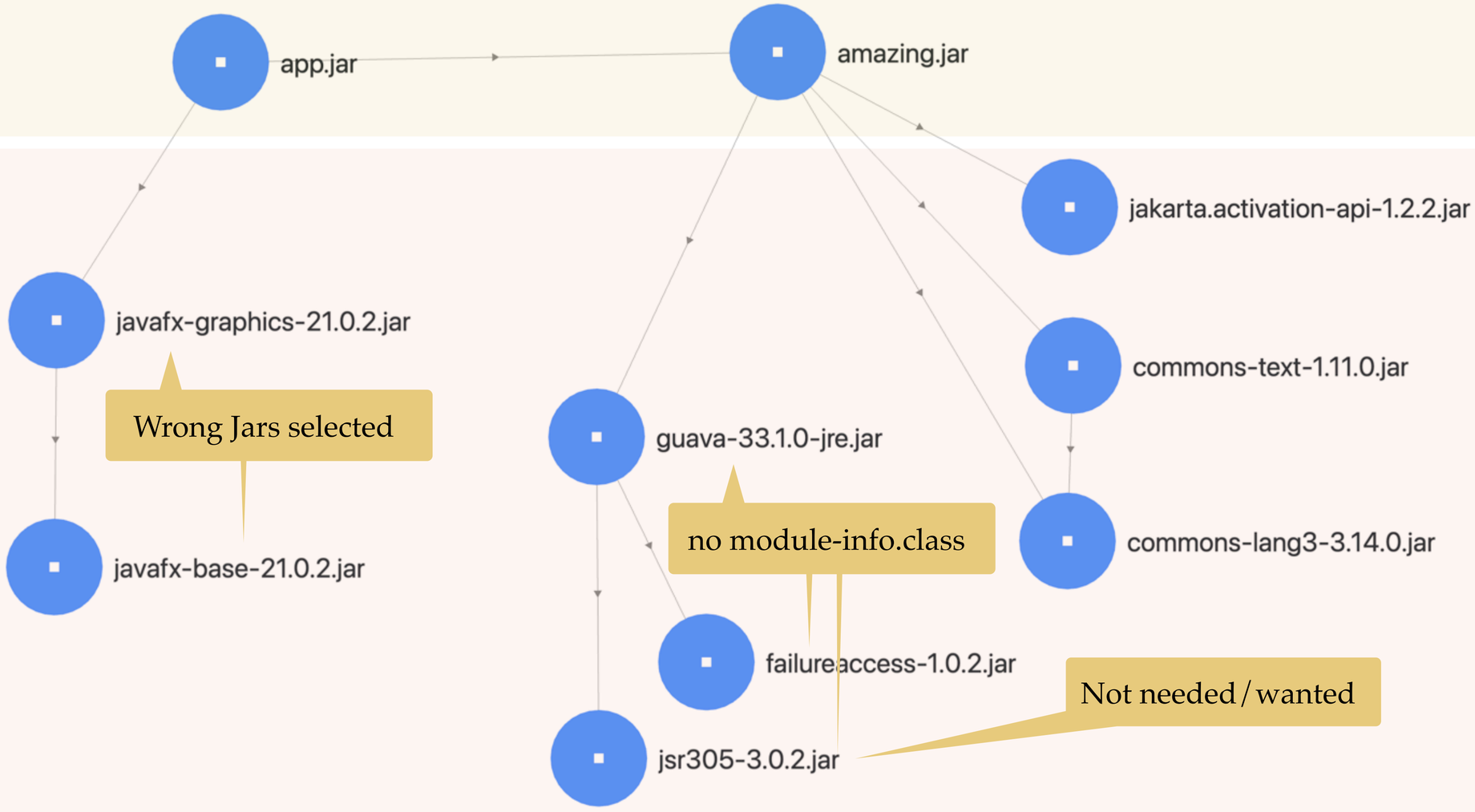
Scopes are important

Essential Complexity #2

Multiple **Versions/Variants** of 3rd party Modules

"Essential" Complexity #3

The world is **not perfect**



Part 2

Accidental Complexity
introduced through
insufficient Tooling and Knowledge

github.com/hibernate/hibernate-orm/blob/main/hibernate-core/hibernate-core.gradle

~300 Lines total
~20 Lines Module Definition

```
30    dependencies {
31        api jakartaLibs.jpa
32        api jakartaLibs.jta
33
34        implementation libs.hcann
35        implementation libs.jandex
36        implementation libs.classmate
37        implementation libs.byteBuddy
38
39        implementation jakartaLibs.jaxbApi
40        implementation jakartaLibs.jaxb
41        implementation jakartaLibs.inject
42
43        implementation libs.antlrRuntime
```

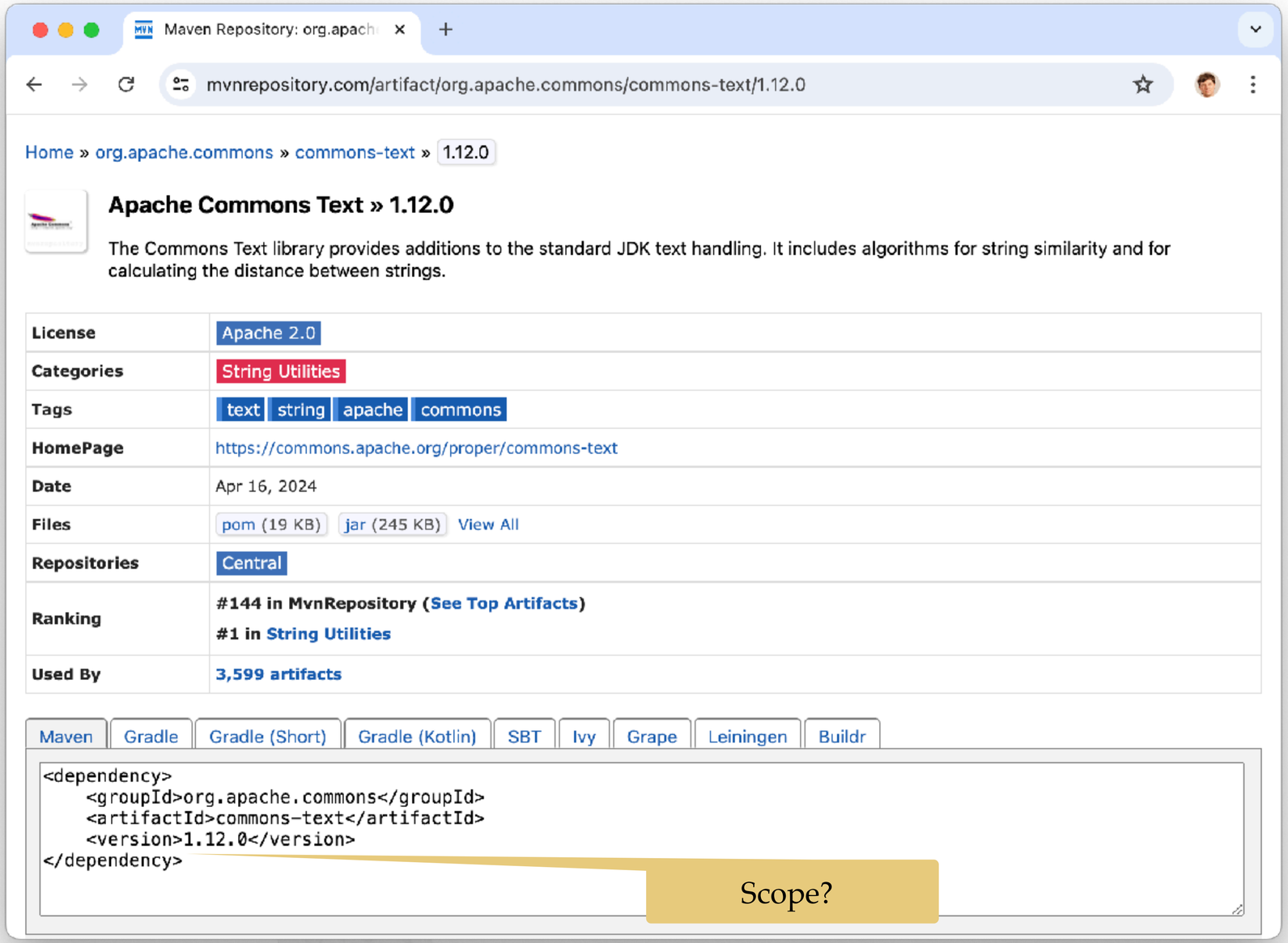
```
30      <dependencies>
31
32        <dependency>
33          <groupId>org.slf4j</groupId>
34          <artifactId>slf4j-api</artifactId>
35        </dependency>
36
37        <dependency>
38          <groupId>org.apache.hadoop</groupId>
39          <artifactId>hadoop-auth</artifactId>
40        </dependency>
41
42        <dependency>
43          <groupId>org.apache.hadoop</groupId>
44          <artifactId>hadoop-annotations</artifactId>
45        </dependency>
```

~300 Lines total
~100 Line Module Definition

github.com/apache/hadoop/blob/trunk/hadoop-common-project/hadoop-registry/pom.xml

Accidental Complexity #0

Module definition mixed with other build concerns

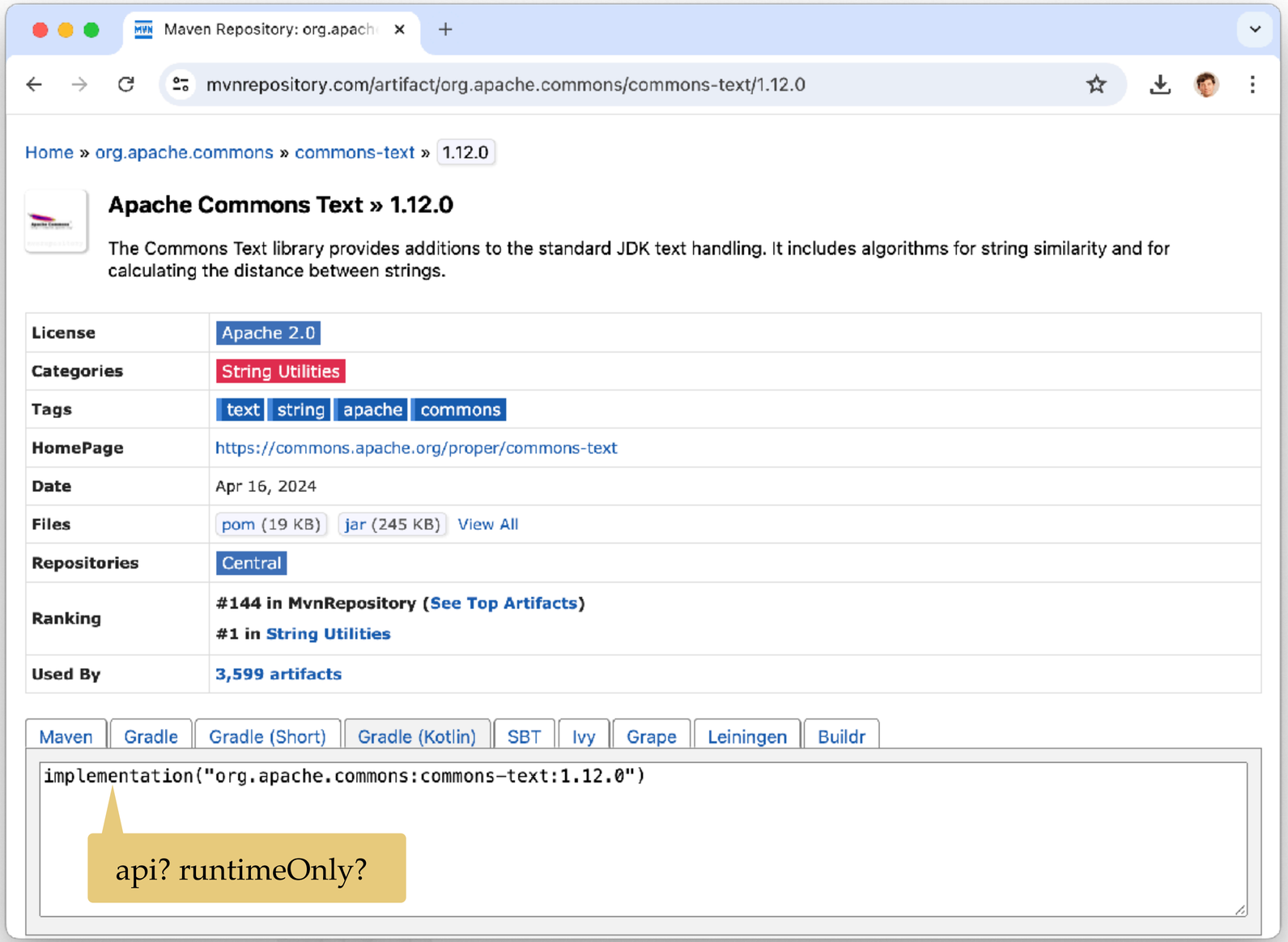


Accidental Complexity #1

Module definition mixed with other build concerns

Accidental Complexity #2

Scopes not cared about (enough)



Accidental Complexity #0

Module definition mixed with other build concerns

Accidental Complexity #1

Scopes not cared about (enough)

repo1.maven.org/maven2/org/openjfx/javafx-graphics/21.0.2/

```
org/openjfx/javafx-graphics/21.0.2/
├── javafx-graphics-21.0.2.pom
├── javafx-graphics-21.0.2.module
├── javafx-graphics-21.0.2.jar
├── javafx-graphics-21.0.2-linux.jar
│   └── module-info.class
├── javafx-graphics-21.0.2-mac.jar
│   └── module-info.class
└── javafx-graphics-21.0.2-win.jar
    └── module-info.class
```

No rich metadata beyond POM 4.0.0 👎

Empty Jar to get Maven to do a OS-based selection
-> such hacks shouldn't be needed
-> does not work for Gradle 👎

Java Module Descriptor
(Java Module System / JPMS)
"Java standard notation" 👍

Accidental Complexity #0
Module definition mixed with other build concerns

Accidental Complexity #1
Scopes not cared about (enough)

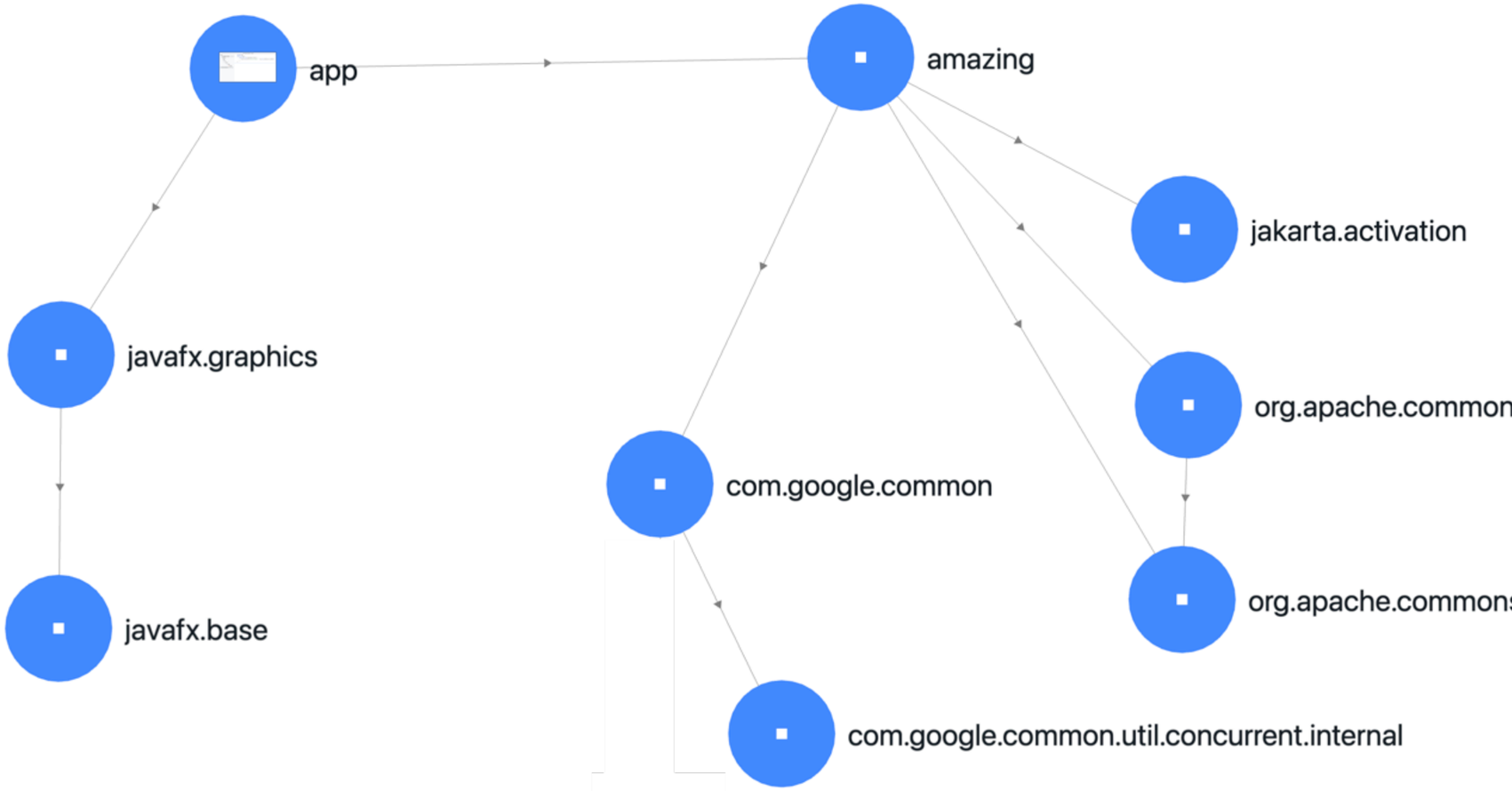
Accidental Complexity #2
Multiple Version/Variants not cared about (enough)

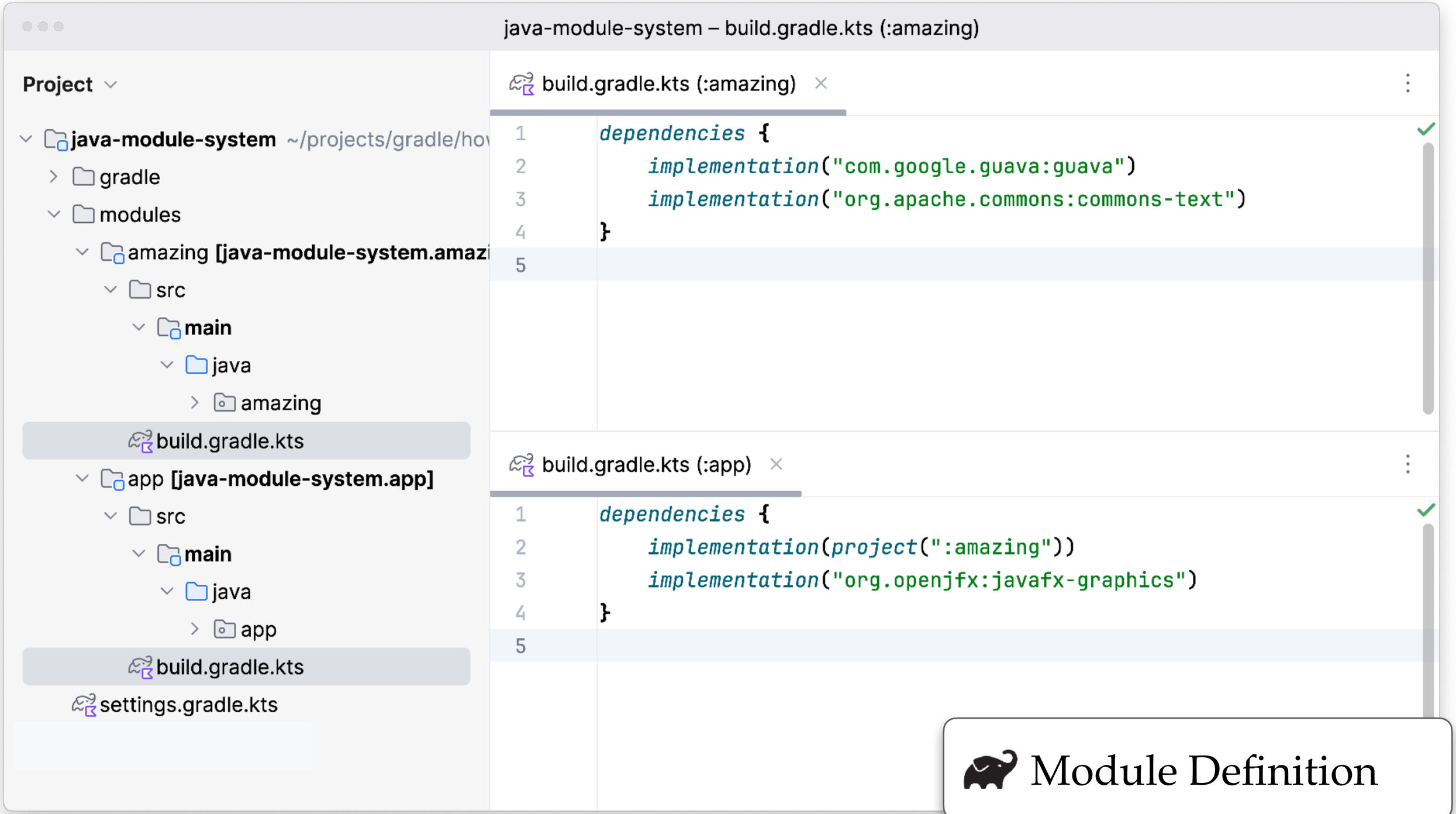
Accidental Complexity #3
Metadata of 3rd party modules is incomplete or wrong

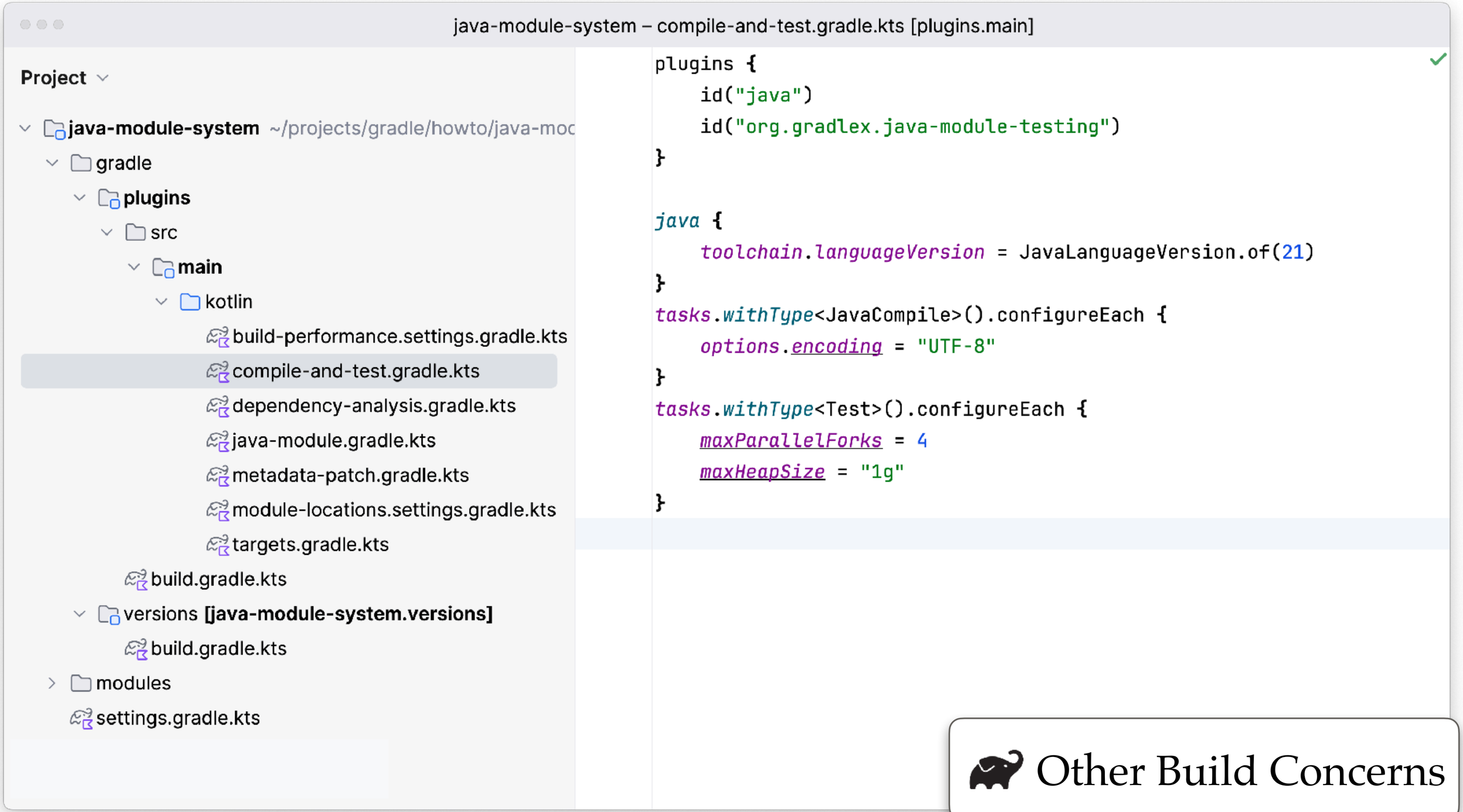
Part 3

How to do

Better?

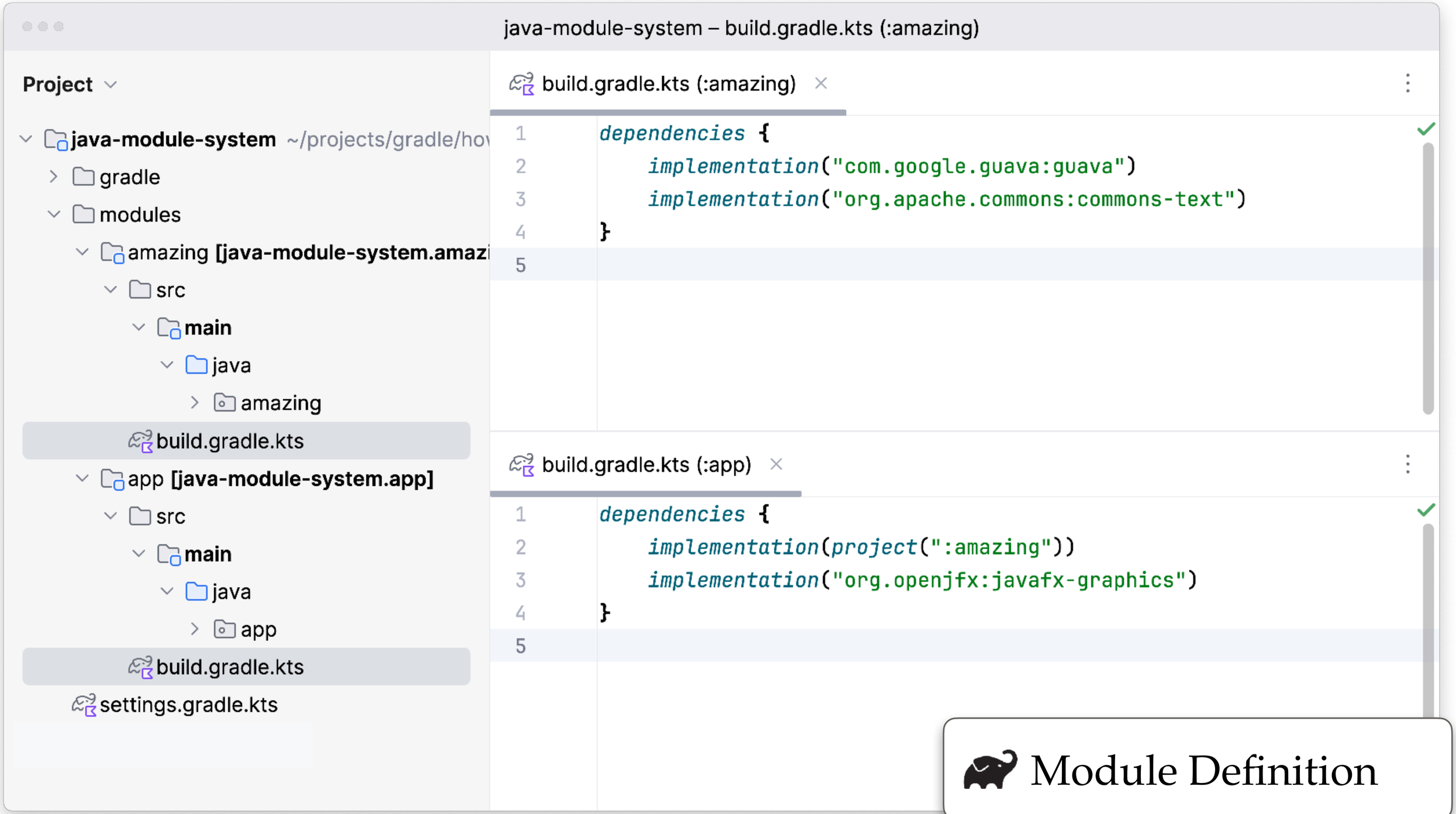

app
amazing
jakarta.activation
javafx.graphics
org.apache.common
com.google.common
javafx.base
org.apache.commons
com.google.common.util.concurrent.internal

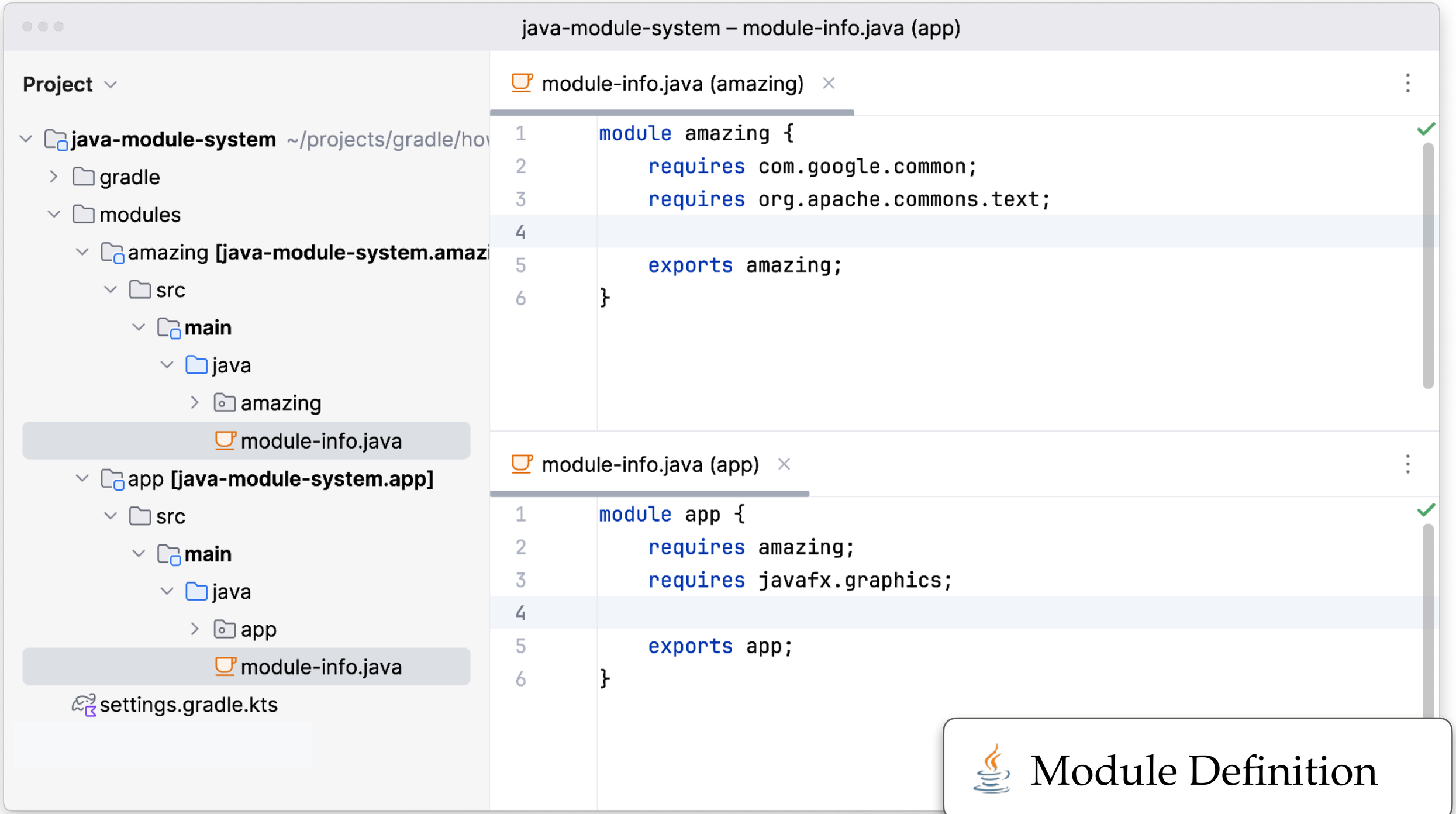

java-module-system – build.gradle.kts (:amazing)
Project
java-module-system ~/projects/gradle/hov
gradle
modules
amazing [java-module-system.amazi
src
main
java
amazing
build.gradle.kts
app [java-module-system.app]
src
main
java
app
build.gradle.kts
settings.gradle.kts
build.gradle.kts (:amazing)
1 dependencies {
2 implementation("com.google.guava:guava")
3 implementation("org.apache.commons:commons-text")
4 }
5
build.gradle.kts (:app)
1 dependencies {
2 implementation(project(":amazing"))
3 implementation("org.openjfx:javafx-graphics")
4 }
5
Module Definition

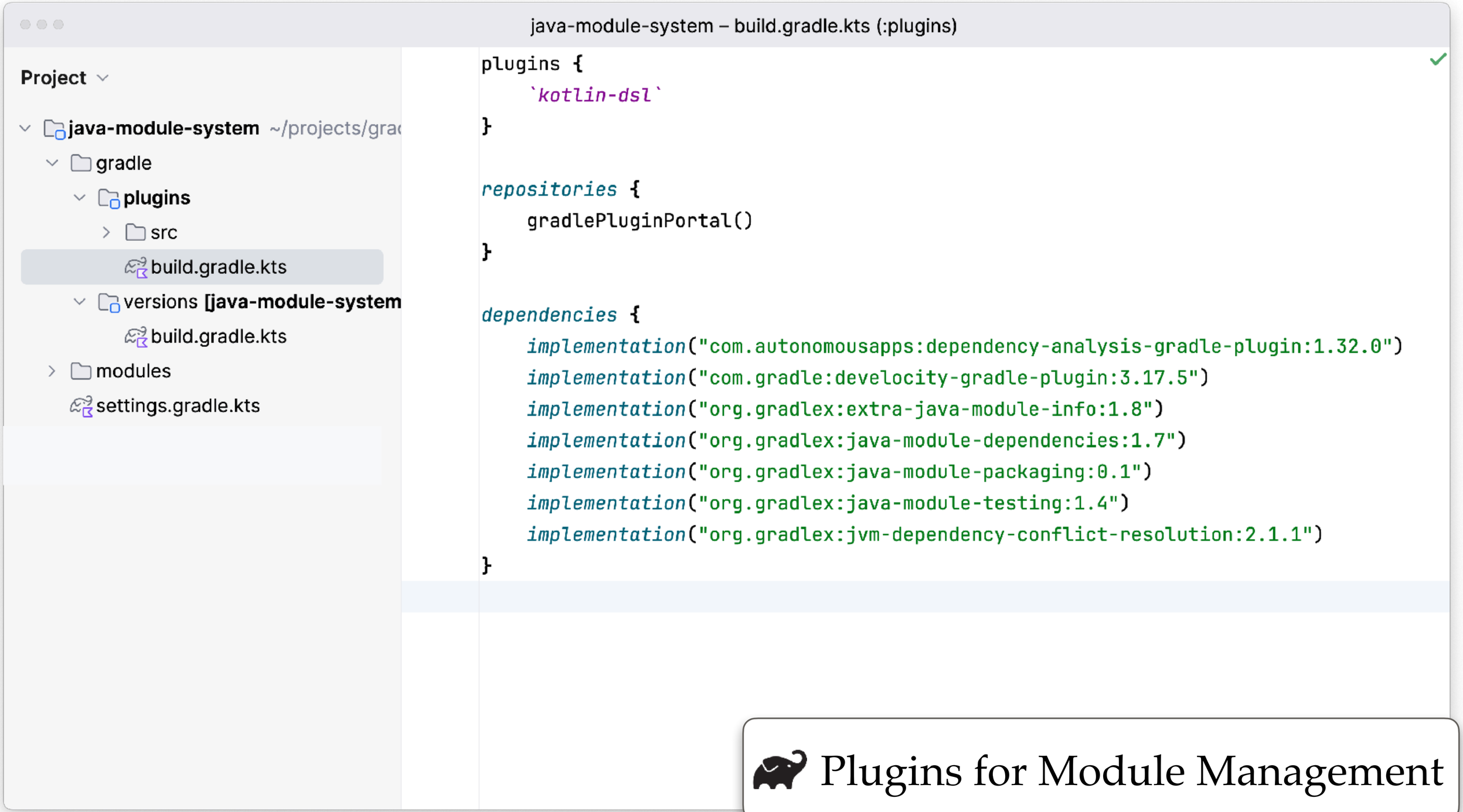


Other Build Concerns


java-module-system – metadata-patch.gradle.kts [plugins.main]
Project
java-module-system ~/projects/gradle/howto/
gradle
plugins
src
main
kotlin
build-performance.settings.gr
compile-and-test.gradle.kts
dependency-analysis.gradle.k
java-module.gradle.kts
metadata-patch.gradle.kts
module-locations.settings.gra
targets.gradle.kts
build.gradle.kts
versions [java-module-system.versions]
build.gradle.kts
modules
settings.gradle.kts
jvmDependencyConflicts { // Patch Maven/Gradle metadata (*.pom / *.module)
patch {
module("com.google.guava:guava") {
removeDependency("com.google.code.findbugs:jsr305")
removeDependency("org.checkerframework:checker-qual")
removeDependency("com.google.errorprone:error_prone_annotations")
}
listOf("base", "graphics", "controls").forEach { jfxModule ->
module("org.openjfx:javafx-$jfxModule") {
addTargetPlatformVariant("linux", OperatingSystemFamily.LINUX
addTargetPlatformVariant("mac", OperatingSystemFamily.MACOS,
addTargetPlatformVariant("mac-aarch64", OperatingSystemFamily
addTargetPlatformVariant("win", OperatingSystemFamily.WINDOWS
}
}
}
}
Module Patching


java-module-system – build.gradle.kts (:amazing)
Project
java-module-system ~/projects/gradle/hov
gradle
modules
amazing [java-module-system.amazi
src
main
java
amazing
build.gradle.kts
app [java-module-system.app]
src
main
java
app
build.gradle.kts
settings.gradle.kts
build.gradle.kts (:amazing)
1 dependencies {
2 implementation("com.google.guava:guava")
3 implementation("org.apache.commons:commons-text")
4 }
5
build.gradle.kts (:app)
1 dependencies {
2 implementation(project(":amazing"))
3 implementation("org.openjfx:javafx-graphics")
4 }
5
Module Definition


java-module-system – module-info.java (app)
Project
java-module-system ~/projects/gradle/hov
gradle
modules
amazing [java-module-system.amazi
src
main
java
amazing
module-info.java
app [java-module-system.app]
src
main
java
app
module-info.java
settings.gradle.kts
module-info.java (amazing)
1 module amazing {
2 requires com.google.common;
3 requires org.apache.commons.text;
4
5 exports amazing;
6 }
module-info.java (app)
1 module app {
2 requires amazing;
3 requires javafx.graphics;
4
5 exports app;
6 }
Module Definition


java-module-system – build.gradle.kts (:plugins)
Project
java-module-system ~/projects/grad
gradle
plugins
src
build.gradle.kts
versions [java-module-system
build.gradle.kts
modules
settings.gradle.kts
plugins {
`kotlin-dsl`
}
repositories {
gradlePluginPortal()
}
dependencies {
implementation("com.autonomousapps:dependency-analysis-gradle-plugin:1.32.0")
implementation("com.gradle:develocity-gradle-plugin:3.17.5")
implementation("org.gradlex:extra-java-module-info:1.8")
implementation("org.gradlex:java-module-dependencies:1.7")
implementation("org.gradlex:java-module-packaging:0.1")
implementation("org.gradlex:java-module-testing:1.4")
implementation("org.gradlex:jvm-dependency-conflict-resolution:2.1.1")
}
Plugins for Module Management

Summary

Essential Complexity #1

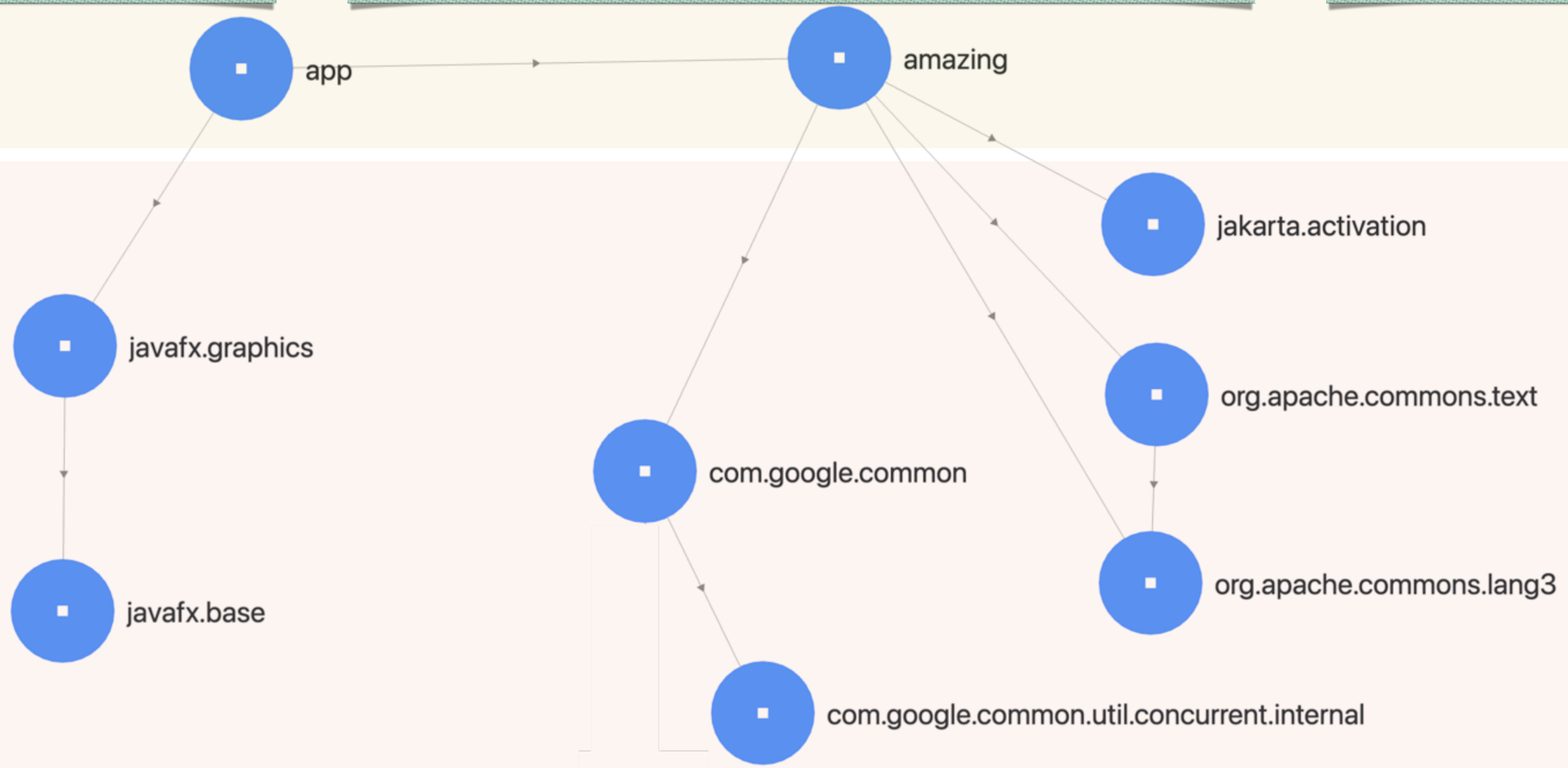
Scopes are important

Essential Complexity #2

Multiple **Versions/Variants** of 3rd party Modules

"Essential" Complexity #3

The world is **not perfect**



Dedicated places for dependency definitions

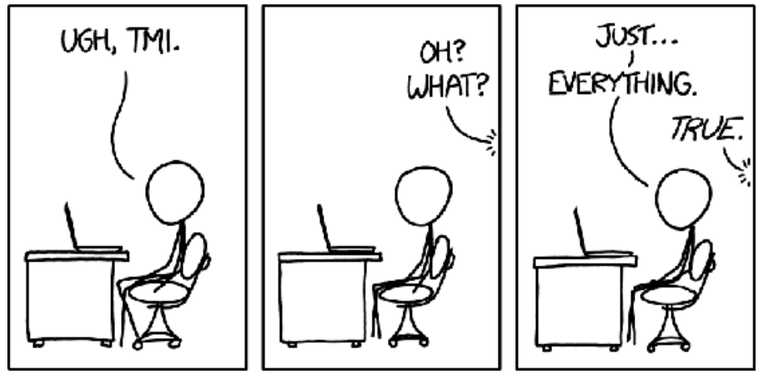
Unified syntax for scope definition

Variant-aware conflict detection and resolution (as provided in Gradle)

Central patch file(s) to enrich an fix metadata

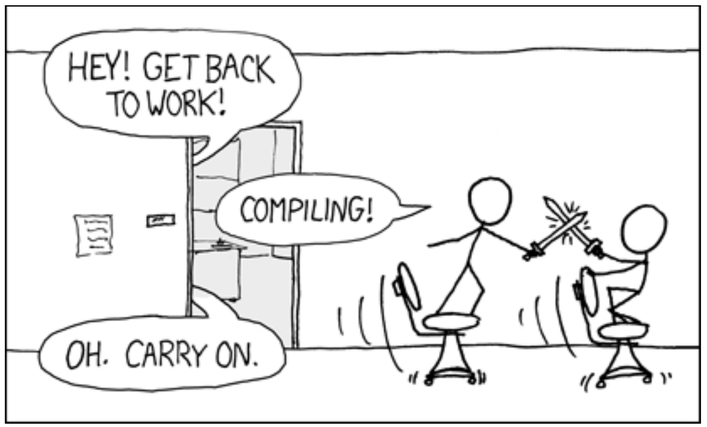
What makes us (more) productive?

…Comprehensibility?



xkcd.com/1369/

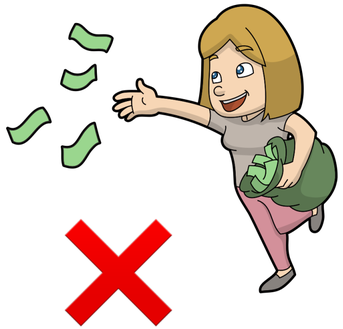
…Build Performance?



xkcd.com/303/

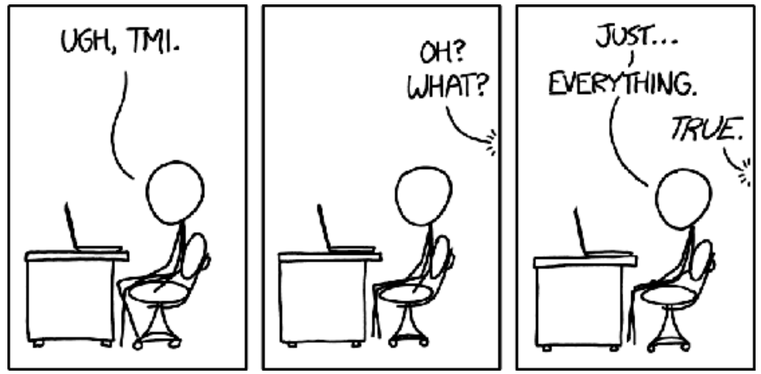
With infinite speed/memory/money

✅



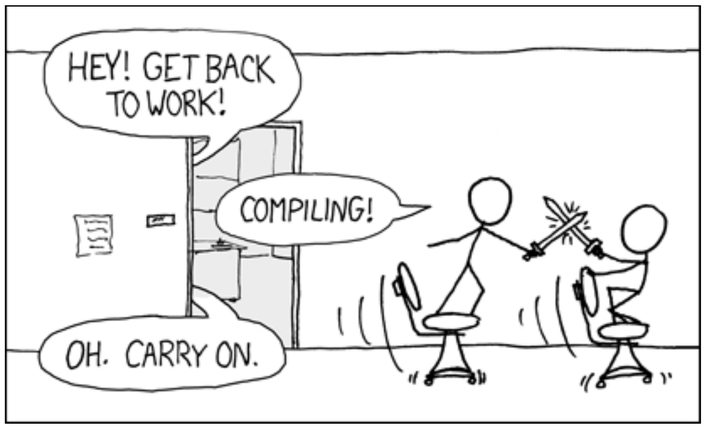
What makes us (more) productive?

…Comprehensibility?

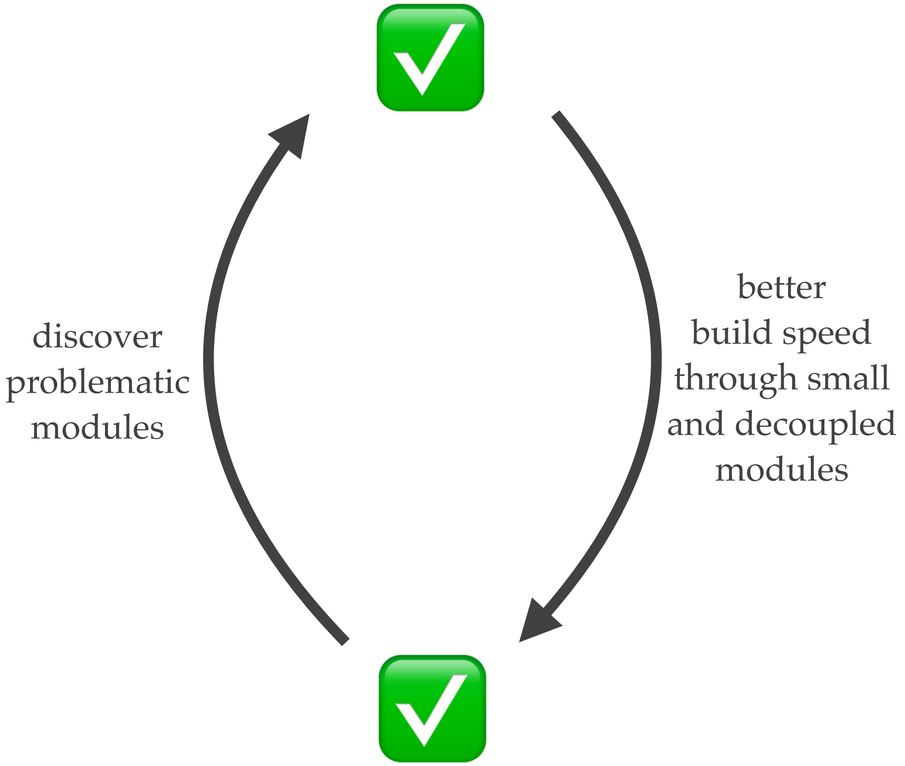


xkcd.com/1369/

…Build Performance?



xkcd.com/303/



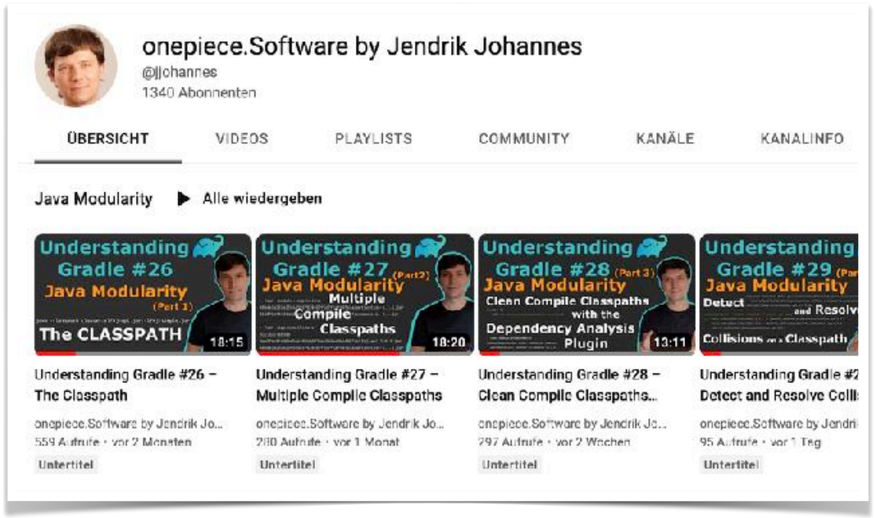
Current Focus of off-the-shelf DPE Tools*

★☆☆☆☆

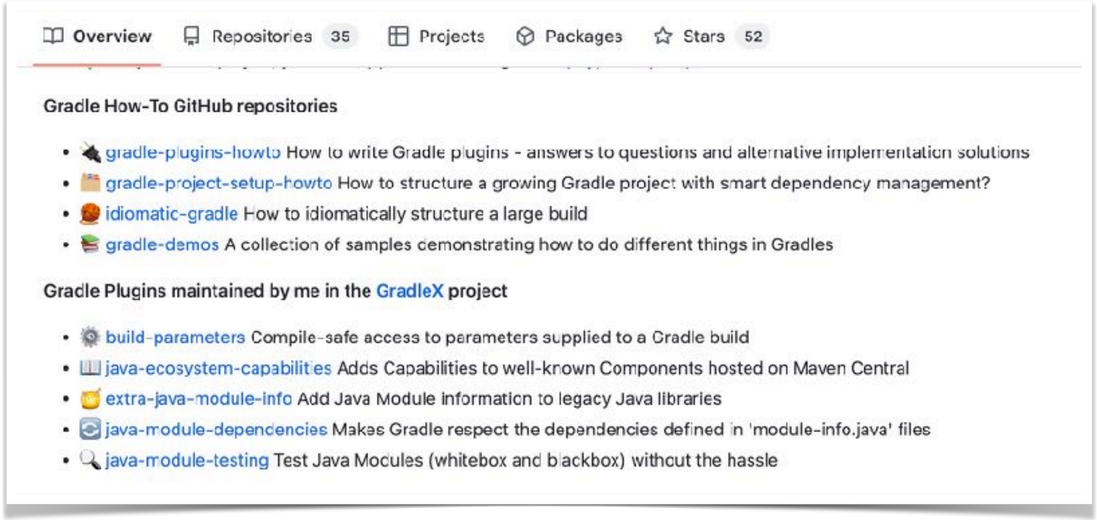
★★★☆☆

*subjective impression

youtube.com/@jjohannes



github.com/jjohannes



github.com/jjohannes/java-module-system
github.com/jjohannes/gradle-project-setup-howto
github.com/hashgraph/hedera-services

stay in touch
mastodon.social/@jendrik
linkedin.com/in/jendrikjohannes
jendrik@onepiece.software

training and consulting
onepiece.software

GradleX Project
gradlex.org