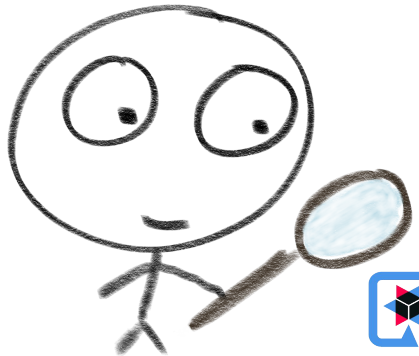
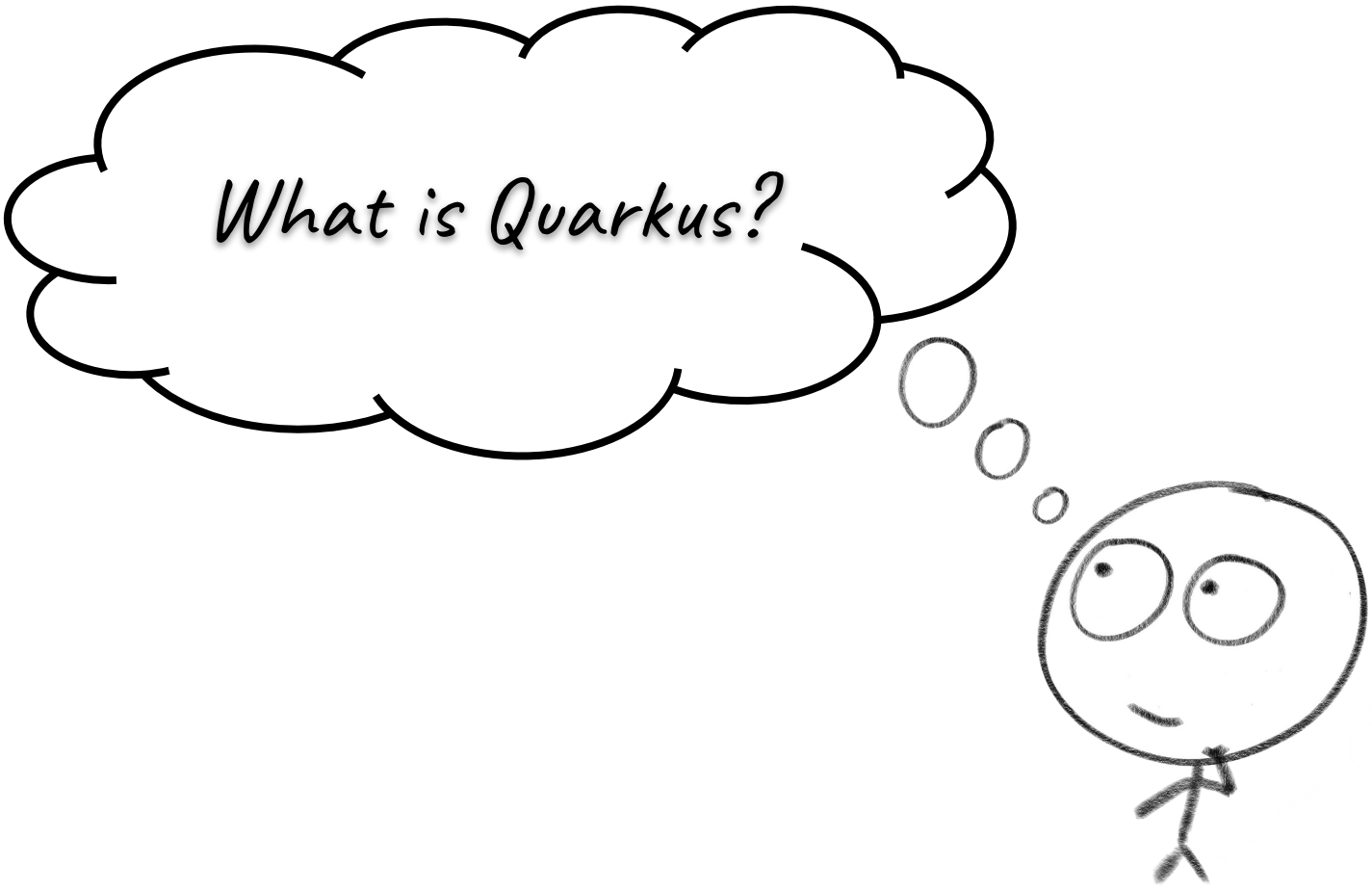
QUARKUS

Exploring Quarkus Native: Choices and Implementation

Foivos Zakkak, Red Hat

@zakkak
@foivoszakkak



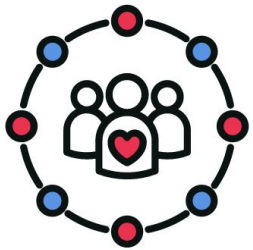

What is Quarkus?

An Open Source stack to write Java apps

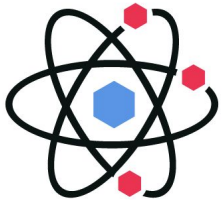
Developer Joy

Kubernetes-native

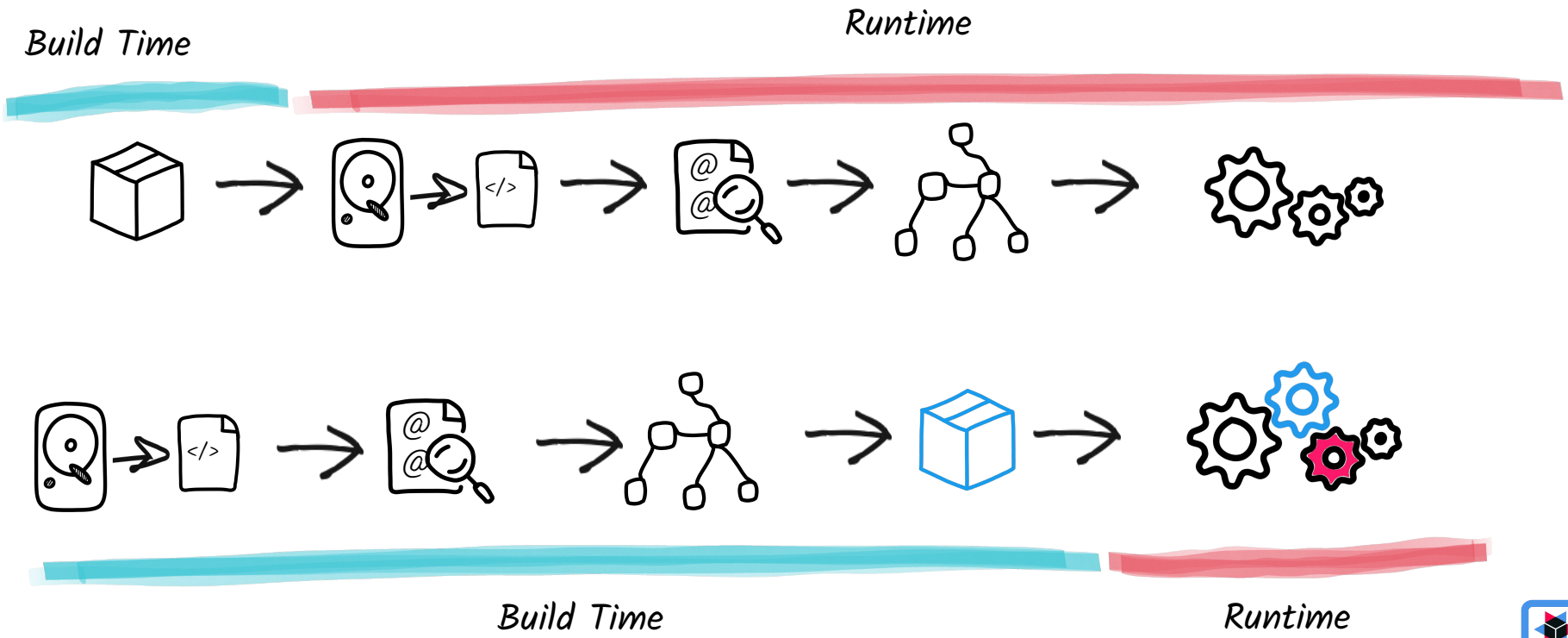
Best of Breed
Libraries and Standards



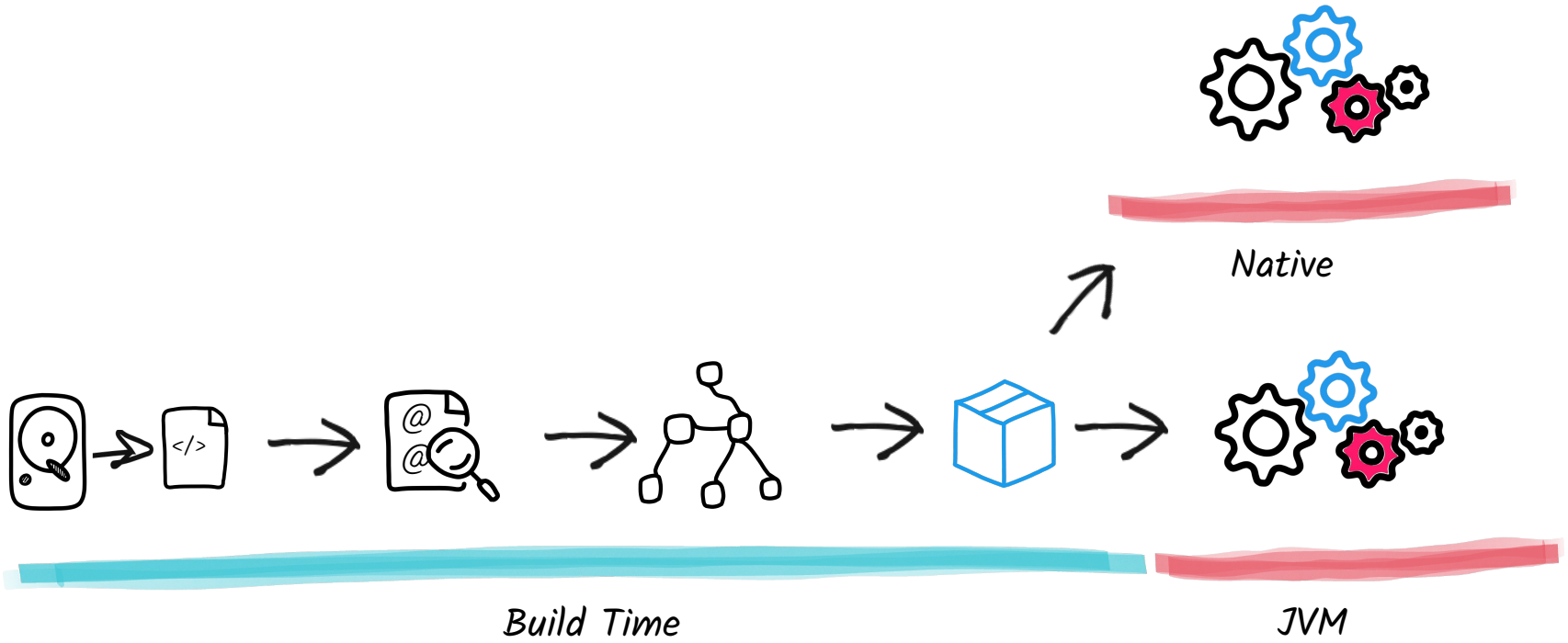
Imperative and reactive
code



The Traditional vs. Quarkus Ways



The Quarkus Way enables Native compilation




Why Native?

Pros

Faster startup

Close to peak performance from the beginning

Small standalone binary

Smaller memory footprint (RSS)

Pros and Cons

Faster startup

Close to peak performance from the beginning

Small standalone binary

Smaller memory footprint (RSS)

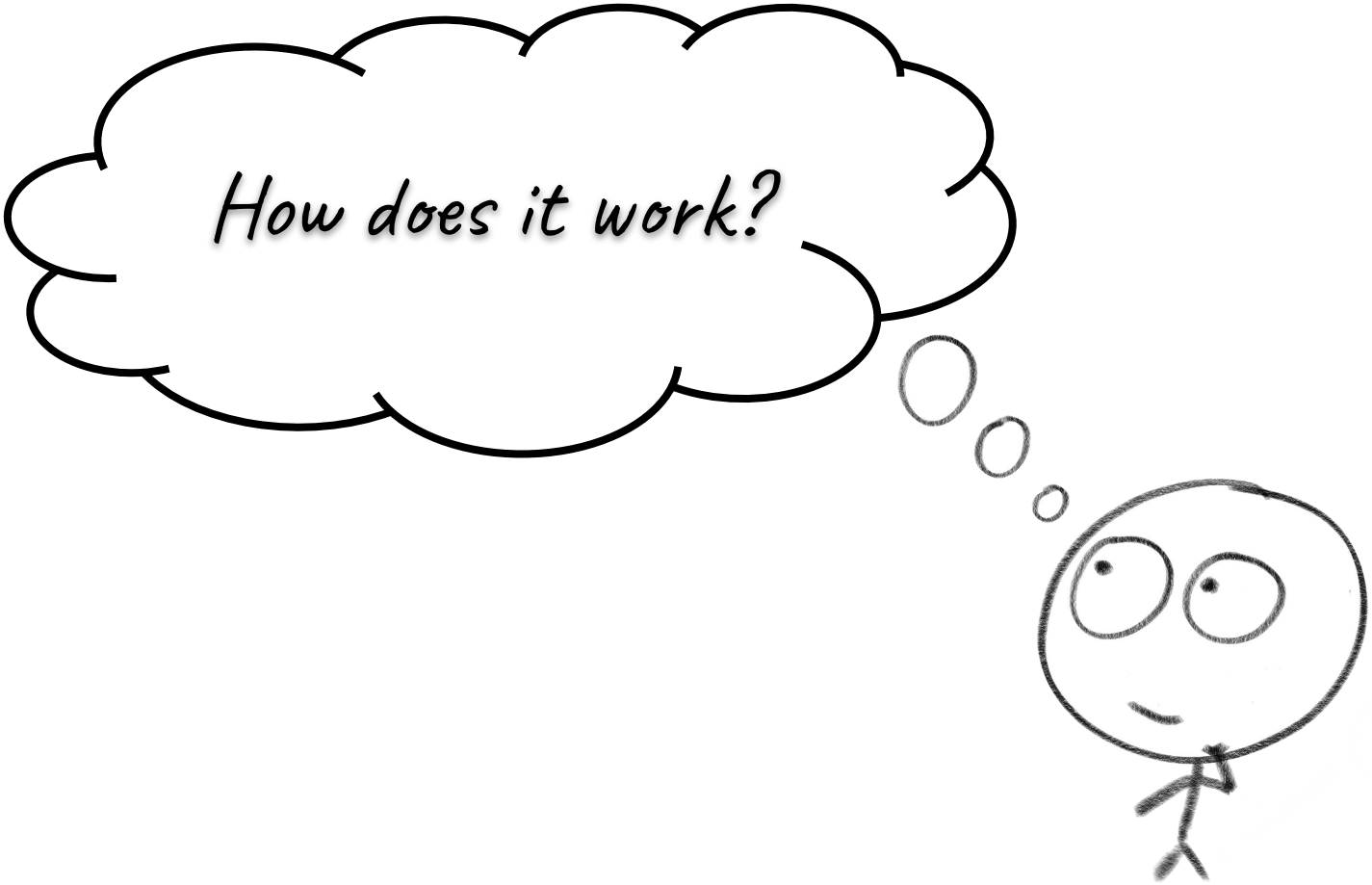
Slower development cycle

Lower peak performance

Security patches require recompilation

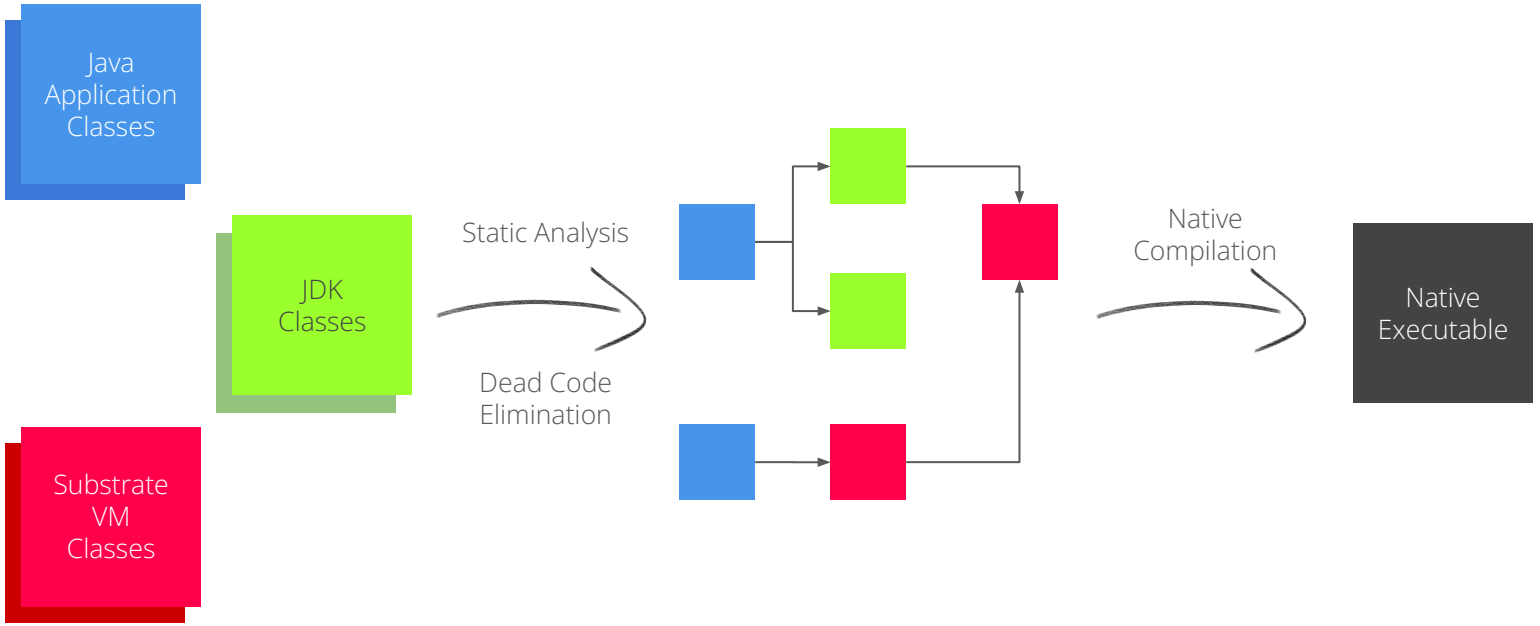
Not portable

Lacks behind in terms of tooling support


How does it work?


GraalVM™

AOTC - GraalVM native image - Dead code elimination



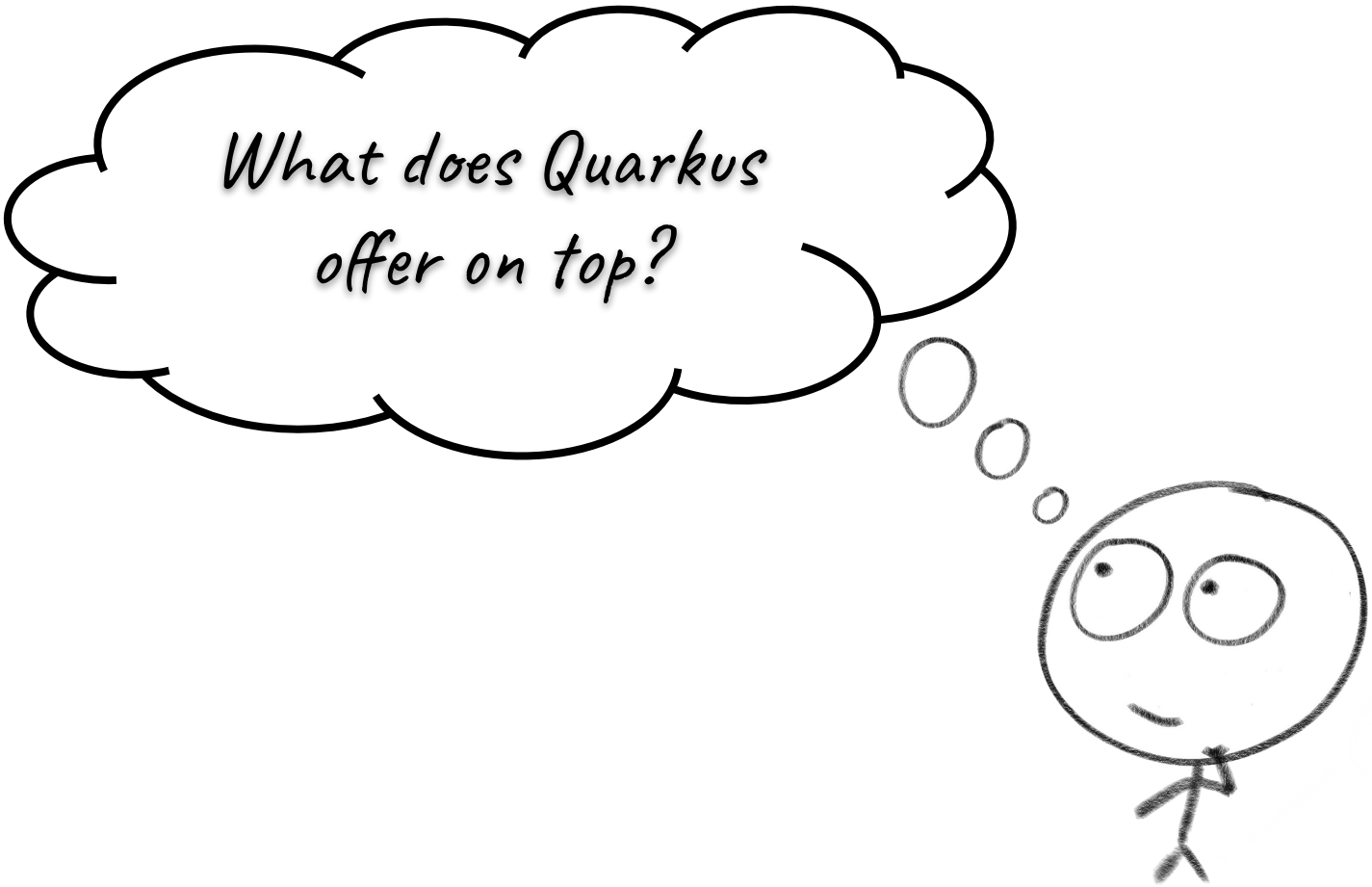
The Dark Side

Not supported

- Dynamic classloading
- InvokeDynamic & Method handles
- Finalizer
- Security manager
- JVMTI, JMX, native VM Interfaces

OK with caveats in usage

- Reflection (manual list)
- Dynamic proxy (manual list)
- JNI (manual list)
- Static initializers (eager)
- Lambda, Threads (OK, pfff!)
- References (similar)


What does Quarkus
offer on top?

Developing native applications for GraalVM's native-image can be painful

Quarkus enables Java developers to easily
use the most popular frameworks and standards
directly on GraalVM's native-image without any hassle

How does Quarkus native help developers?

Drives the gathering of metadata needed by GraalVM

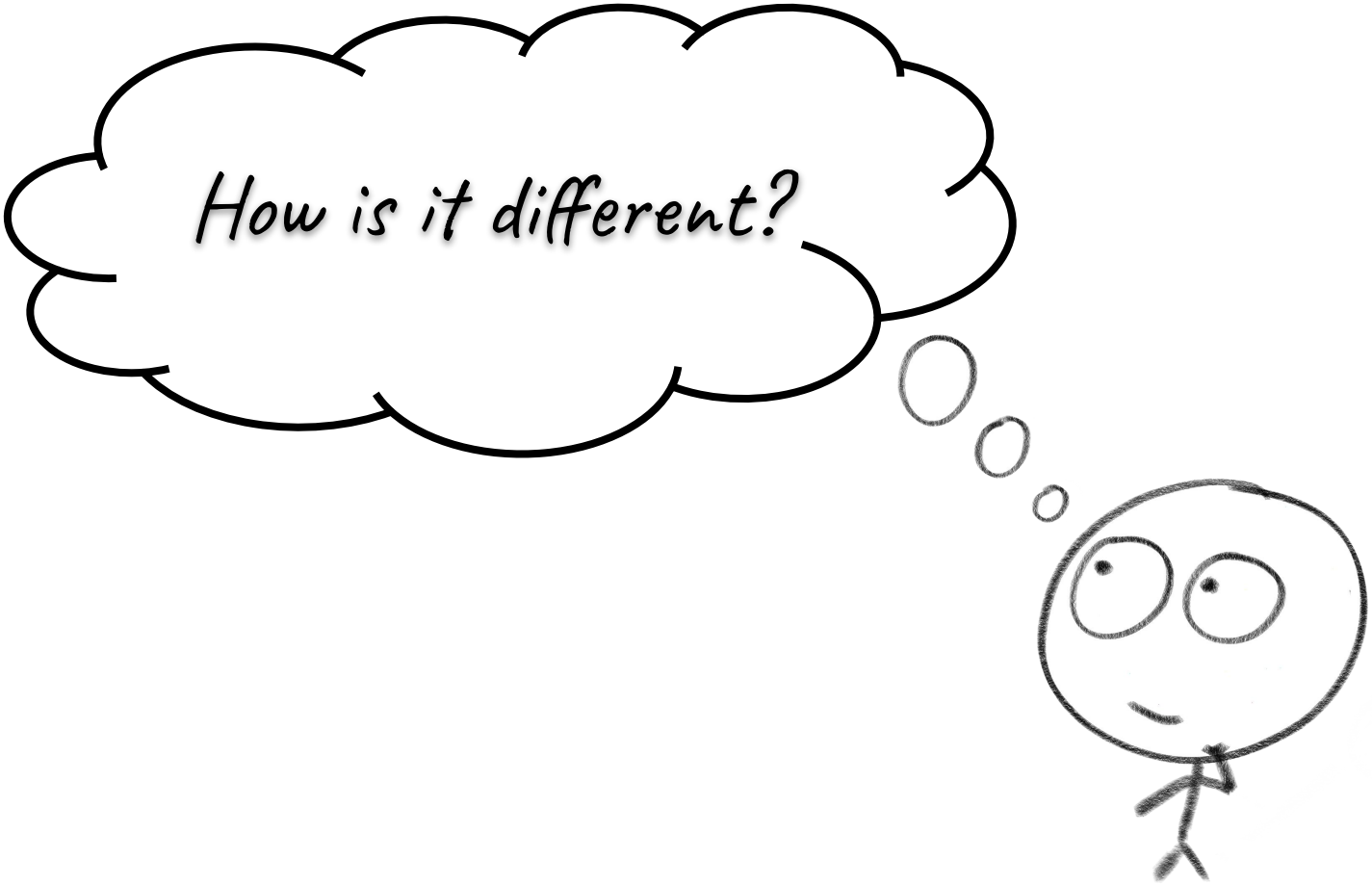
Most of the ecosystem already supported on GraalVM

- based on framework knowledge
- Classes using reflection, resources, etc
- No need for agent + prerun, long JSON metadata or manual command lines

Minimizes dependencies

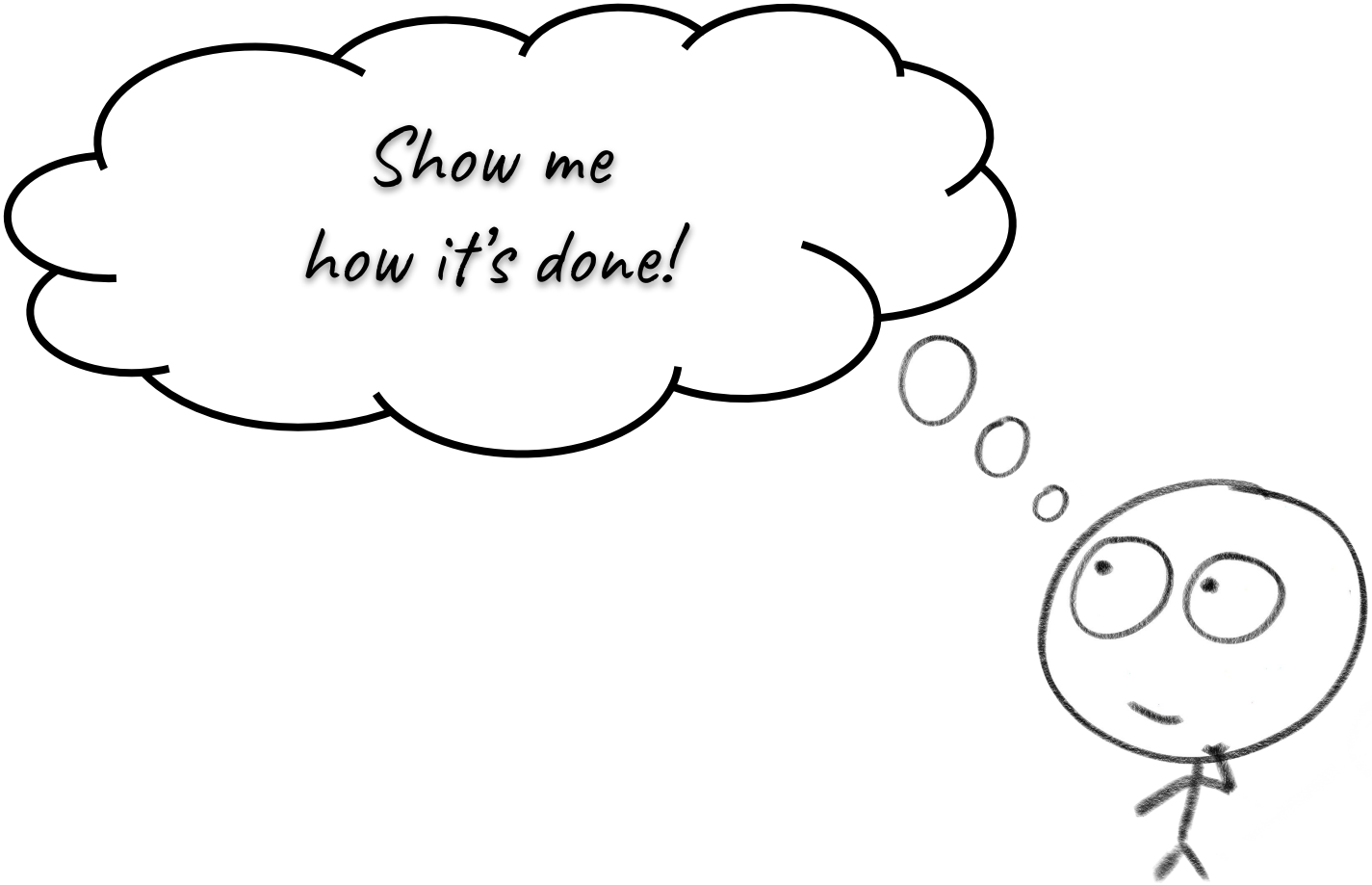
Helps static analysis for dead code elimination

Quarkus annotations, APIs, and configuration properties allow for further configuration


How is it different?

Quarkus Native defaults

- Build time initialization of all classes (where possible)
  - Re-initialize when necessary (e.g. random seeds, platform specific values, etc.)
  - Reset fields to null to prevent pulling in undesired state or classes
- Doesn't allow incomplete classpaths (`--link-at-build-time`)
  - No unexpected runtime failures due to ClassNotFoundException
- Mandrel: Downstream distribution of GraalVM CE basen on Eclipse Temurin OpenJDK builds and specifically tailored to Quarkus (maintained by Red Hat)


Show me
how it's done!

Implementation

- GraalVM native-image JSON configuration generation
- Code substitutions (`com.oracle.svm.core.annotate.Substitute`)
- Code generation
- Default configuration overrides and parameterization of native builds

JSON configuration files generation

Automatically generates:

- `jni-config.json`
- `proxy-config.json`
- `reflect-config.json`
- `resource-config.json`
- `serialization-config.json`

Handled by `io.quarkus.deployment.steps.NativeImage*ConfigStep`

Based on present build items
`io.quarkus.deployment.builditem.nativeimage.*BuildItem`

- Typically provided by the core framework and its extensions

Substitutions: Statistics in Quarkus core

303 method substitutions and 32 field re-computations in 208 classes

- To assist in dead code elimination
- To make code compatible with build time initialization

Substitutions: Example

```
@TargetClass(className = "io.netty.handler.codec.compression.ZstdEncoder",
             onlyWith = IsZstdAbsent.class)
public static final class ZstdEncoderFactorySubstitution {

   @Substitute
   protected ByteBuf allocateBuffer(ChannelHandlerContext ctx,
                                    ByteBuf msg,
                                    boolean preferDirect) throws Exception {
       throw new UnsupportedOperationException();
   }
```

Substitutions: Example

```
public static class IsZstdAbsent implements BooleanSupplier {

    @Override
    public boolean getAsBoolean() {
        try {
            Class.forName("com.github.luben.zstd.Zstd");
            return false;
        } catch (Exception e) {
            return true;
        }
    }

}
```

Substitutions: Recompute Field Reset Example

```
@TargetClass(className = "org.bouncycastle.math.ec.ECPoint",
             onlyWith = BouncyCastleCryptoFips.class)
final class Target_org_bouncycastle_math_ec_ECPoint {
   @Alias //
   @RecomputeFieldValue(kind = RecomputeFieldValue.Kind.Reset) //
   private static SecureRandom testRandom;
}
```

Substitutions: Recompute Field FromAlias Example

```
@TargetClass(className = "io.netty.handler.ssl.OpenSsl")
final class Target_io_netty_handler_ssl_OpenSsl {

    @Alias
    @RecomputeFieldValue(kind = Kind.FromAlias)
    private static Throwable UNAVAILABILITY_CAUSE =
        new RuntimeException("OpenSsl unsupported on Quarkus");

...

    @Substitute
    public static boolean isAvailable() {
        return false;
    }

...
```

Feature Generation

Handled by `io.quarkus.deployment.steps.NativeImageFeatureStep`

Uses `io.quarkus.gizmo` for bytecode generation

Necessary to register/configure anything that's not possible through JSON config

Feature Generation - GraalVM defaults override

```
import org.graalvm.nativeimage.hosted.RuntimeClassInitialization;

private static final MethodDescriptor BUILD_TIME_INITIALIZATION =
    ofMethod(RuntimeClassInitialization.class,
             "initializeAtBuildTime", void.class, String[].class);

...

overallCatch.invokeStaticMethod(BUILD_TIME_INITIALIZATION,
       // empty string means initialize everything
       overallCatch.marshalAsArray(String.class, overallCatch.load("")));

...
```

GraalVM defaults override and parameterization

`native-image` invocations driven by
`io.quarkus.deployment.pkg.steps.NativeImageBuildStep`

GraalVM defaults override: Example

```
addExperimentalVMOption(nativeImageArgs, "-H:+AllowFoldMethods");
if (nativeConfig.headless()) {
   nativeImageArgs.add("-J-Djava.awt.headless=true"); // Default
}
if (nativeConfig.enableFallbackImages()) {
   nativeImageArgs.add("--auto-fallback");
} else {
   nativeImageArgs.add("--no-fallback"); // Default
}
if (!classpathIsBroken) {
   nativeImageArgs.add("--link-at-build-time"); // Default
}
```

QUARKUS

https://quarkus.io

https://quarkusio.zulipchat.com

https://youtube.com/quarkusio

@quarkusio

https://github.com/quarkusio/quarkus

Acknowledgement:
Quarkus participates in the EU funded project AERO with project number 101092850.

Funded by
the European Union