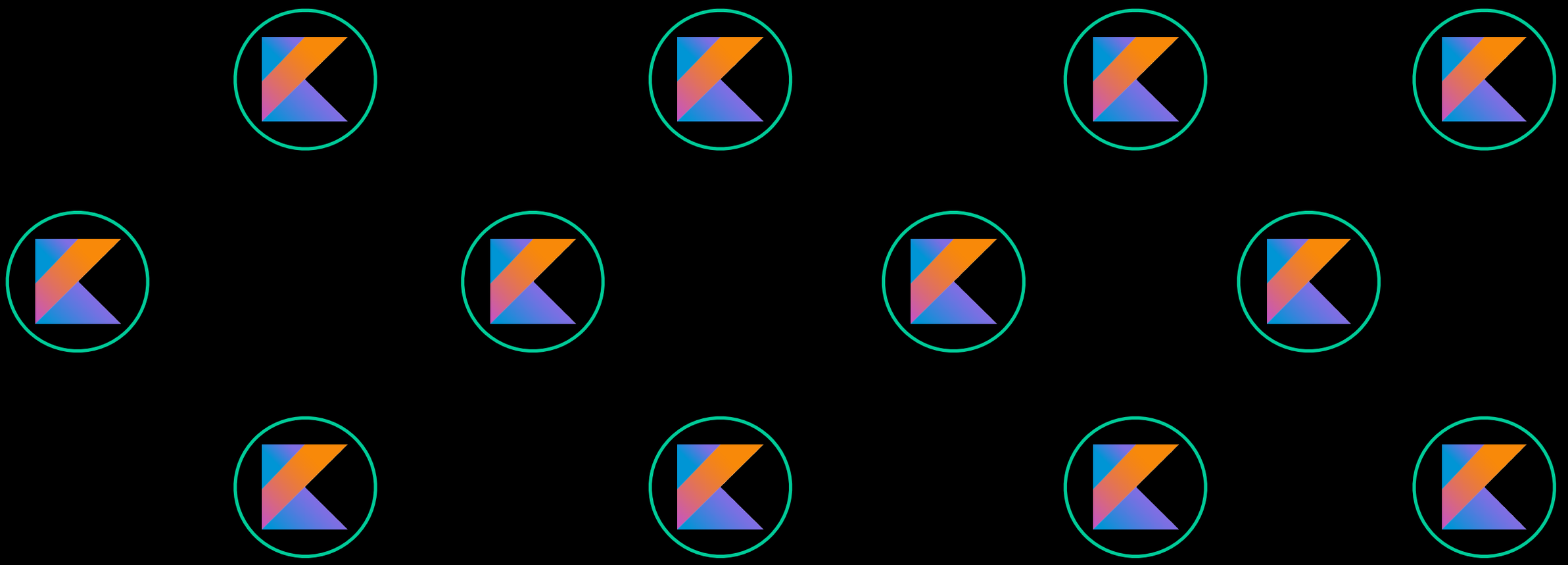
Introduction to Kotlin Microservices



Presenter: David Lucas

LSE

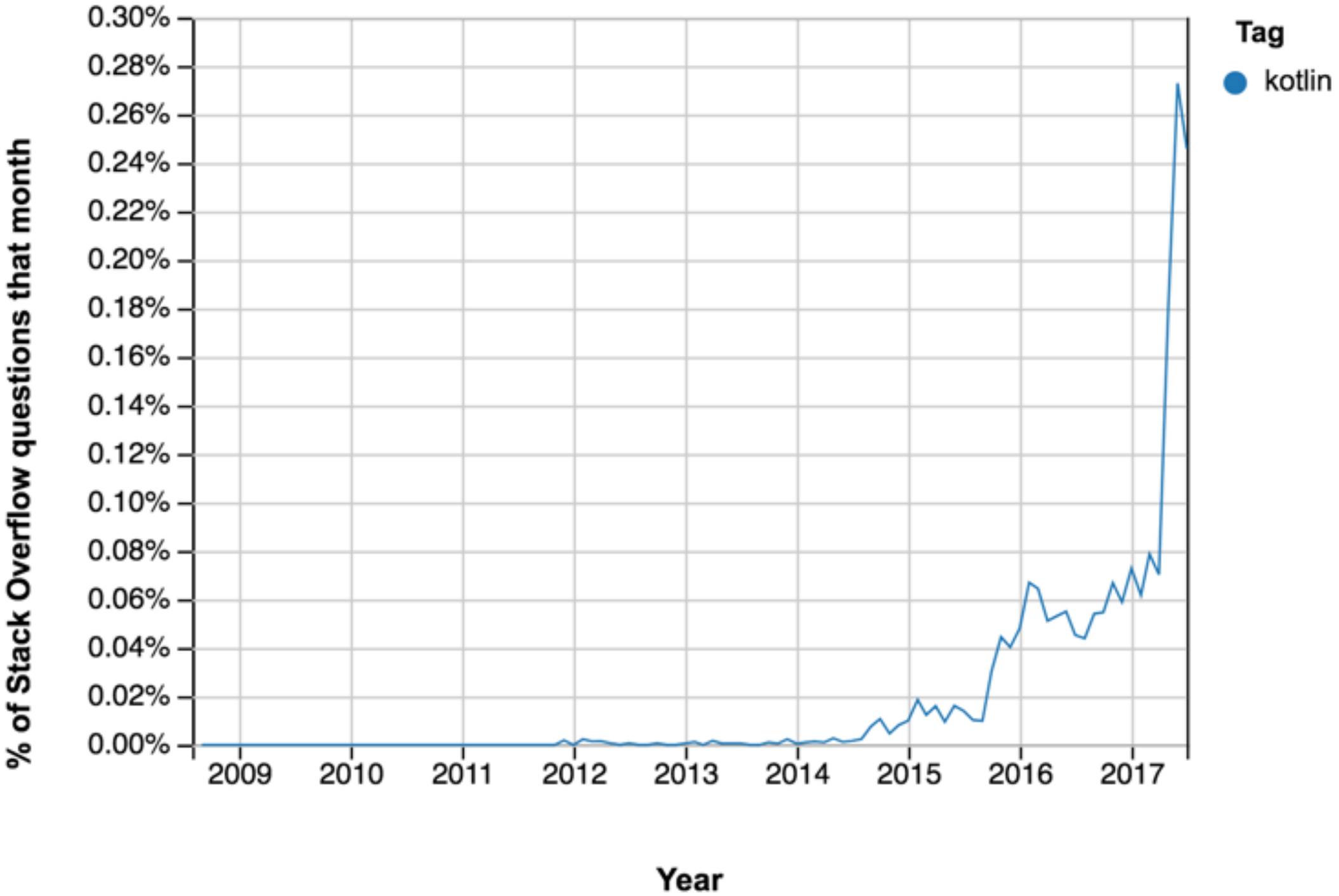
Who am I ?

- Over 25 years in software industry
- Working with Java since 1998
- IntelliJ / JetBrains introduced me to Kotlin
- Google made me realize …
  Kotlin is now the new Java
- Now I am a Kotlin Enthusiast

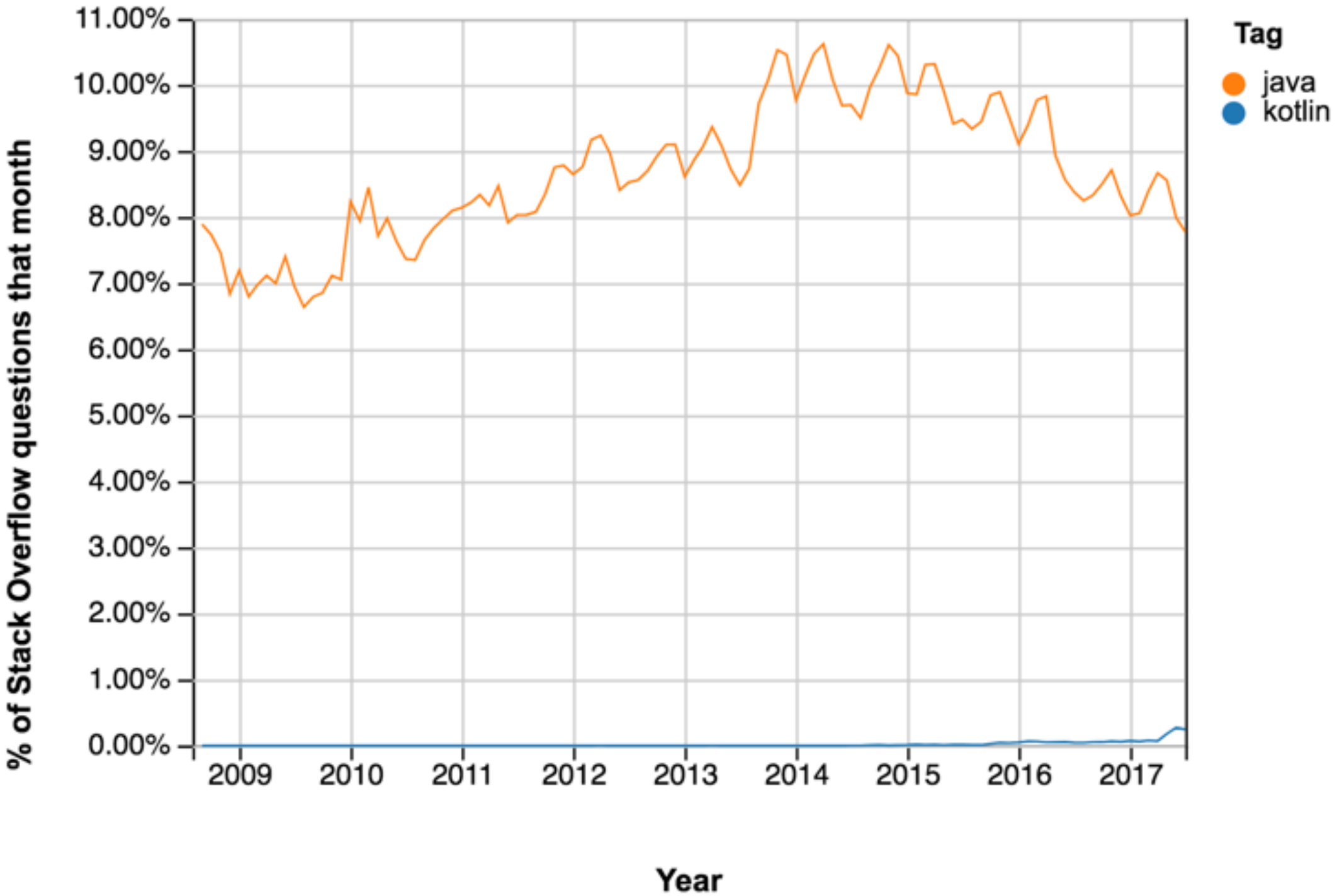
David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com
@DavidDLucas

L S E

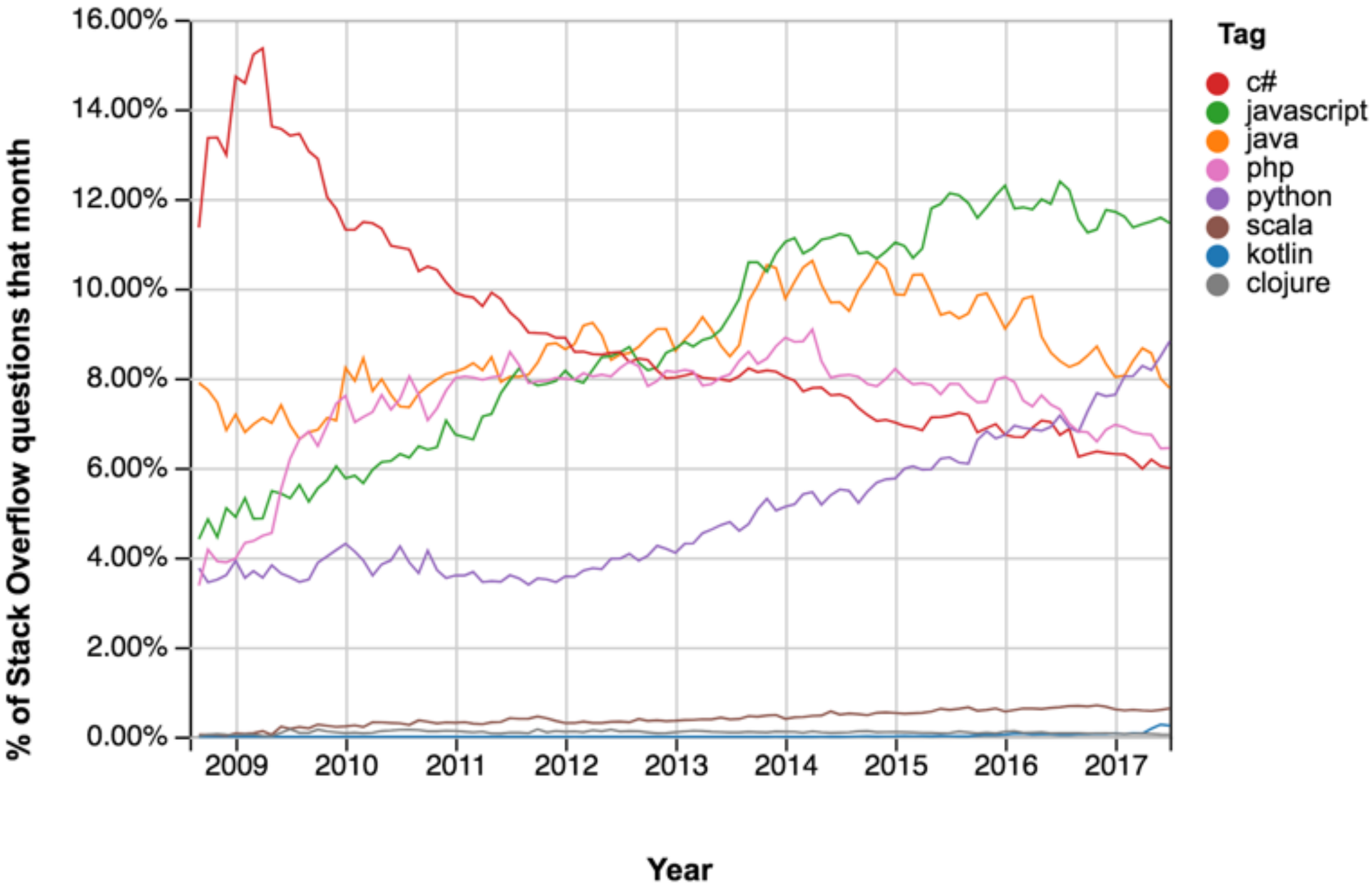
Kotlin Stats



Kotlin Stats



Kotlin Stats



Considering Kotlin

- Leverage Kotlin's strongly typed language and concise expressive code style
- Interoperability with Java VM libraries
- Easy to use abstraction
- Co-routines help scale concurrent requests
- Migration can be gradual from Java to Kotlin

Considering Kotlin

- Great IDE Support (IntelliJ, Eclipse)
- Learning Curve minimized by re-use:
  - Maven
  - Gradle
  - JVM
- Training and documentation
  - Koans / Katas
  - Samples
  - Android
  - Spring

Considering Kotlin (sync vs async)

- Synchronous

```
fun main(args: Array<String>) {
    var intList = listOf<Int>(1,2,3,4,5,6,7,8,9,10)
    var strList = convertToString(intList)
    print("strList = $strList")
}

fun log(msg: String) = println("[${Thread.currentThread().name}] $msg")

fun convertToString(list : List<Any>) : List<String> {
    return list.map {
        log("+ converting $it")
        it.toString()
    }.toList()
}
```

Considering Kotlin (sync vs async)

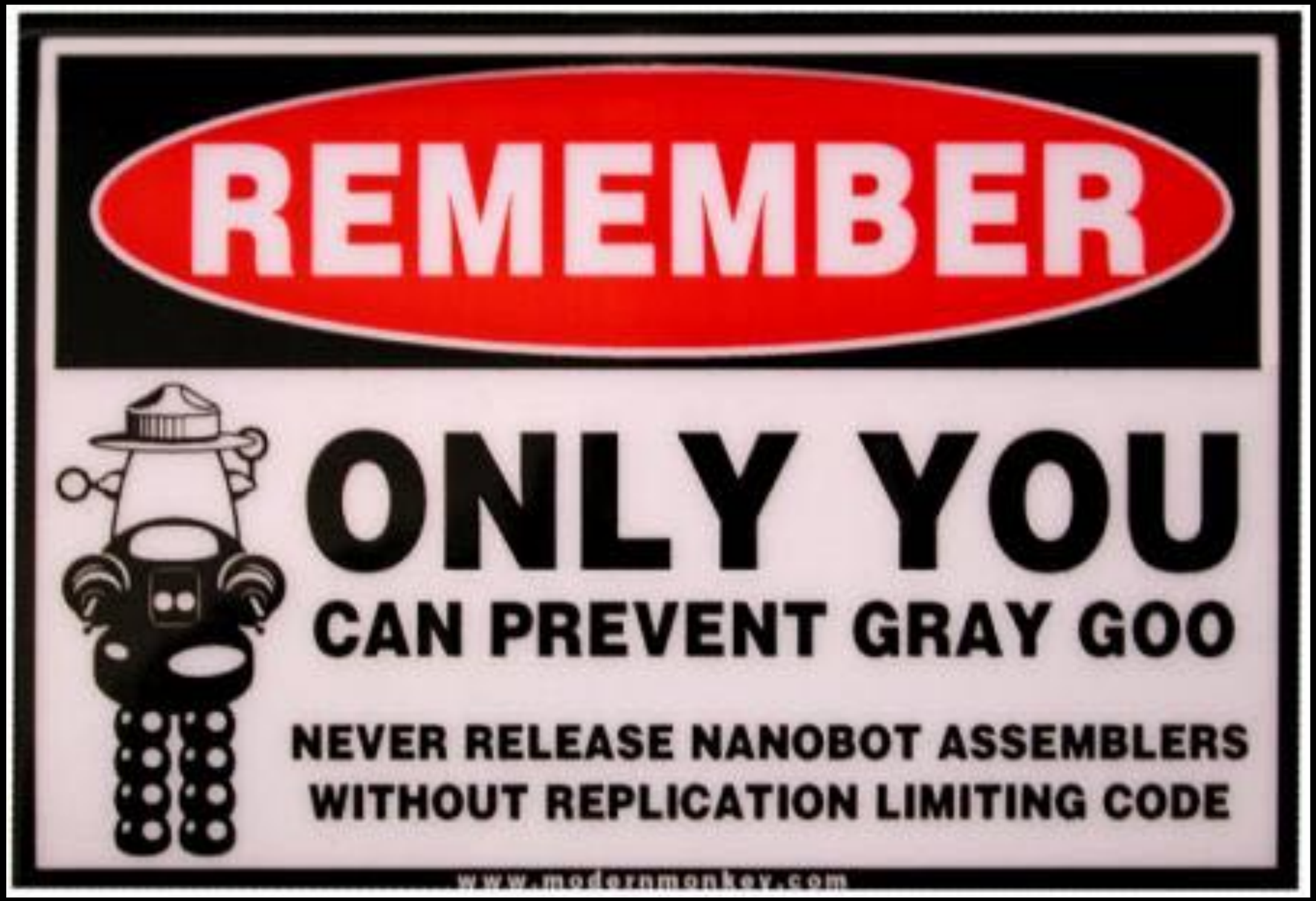
- Coroutine Style (kotlinx)

```
//Kotlin 1.1.4
fun main(args: Array<String>) = runBlocking<Unit> {
    var intList = listOf<Int>(1,2,3,4,5,6,7,8,9,10)
    var strList = convertToString(intList).map { it.await() }
    print("strList = $strList")
}

fun log(msg: String) = println("[${Thread.currentThread().name}] $msg")
fun convertToString(list : List<Any>) : List<Deferred<String>> {
    return list.map {
        async{ processSomething(it) } //asynchronous
    }.toList()
}

suspend fun processSomething(item : Any) : String {
    log("+ converting $item")
    return item.toString()
}
```

Considering Kotlin (sync vs async)

- Spring Reactive Style (RxJava)

```
accept(TEXT_HTML).nest {
    GET("/hello", {
        ServerResponse.ok().body(Mono.just(Message("Hello World")))
    })
}

// Mono returns a single item asynchronously
// Flux returns a stream of items asynchronously
```


REMEMBER
ONLY YOU
CAN PREVENT GRAY GOO
NEVER RELEASE NANOBOT ASSEMBLERS
WITHOUT REPLICATION LIMITING CODE
www.modernmanlove.com

L S E

Microservices

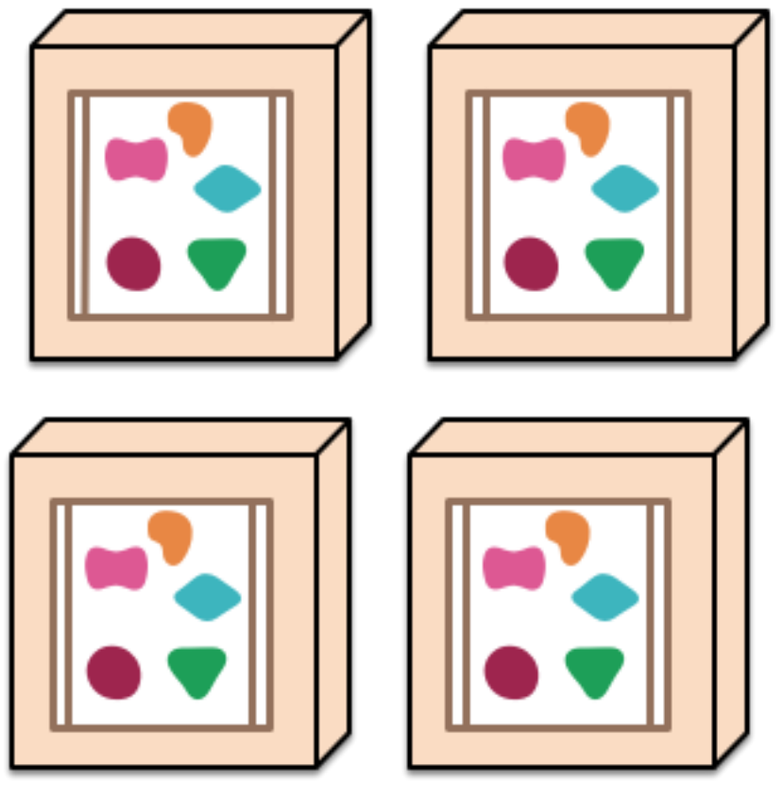
...the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

— Martin Fowler

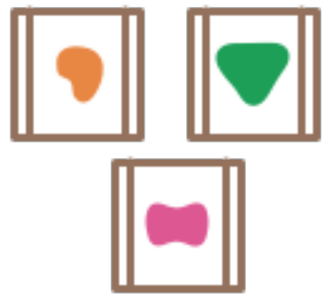
Microservices

A monolithic application puts all its functionality into a single process...

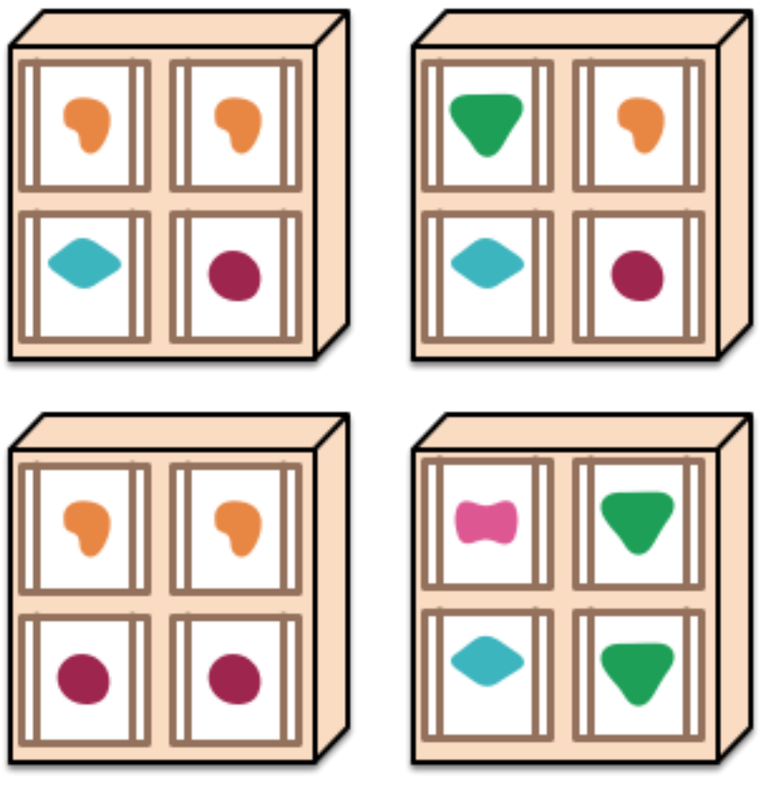
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



https://martinfowler.com/articles/microservices.html

Microservices

- Key is about …
  - scaling
  - owning resources

- What is a Polylith?
  Cluster of Monoliths
  (everything replicated per instance)

- What is a Macroservice?
  System of microservices
  (individually replicated)

BTW: Unix got it right !

cat data.csv | **grep** “LOGIN“ | **tr** -d’,’ -f1-3 > login.csv

<stream> | <process> | <sink>

- simple
- each module does one thing well
- leverages other capabilities

Microservices

12 Factors (https://12factor.net)

1. **One codebase tracked in revision control, many deploys**
2. **Explicitly declare and isolate dependencies**
3. Store config in the environment
4. Treat backing services as attached resources
5. **Strictly separate build and run stages**
6. **Execute the app as one or more stateless processes**
7. **Export services via port binding**
8. **Scale out via the process model**
9. **Maximize robustness with fast startup and graceful shutdown**
10. **Keep development, staging, and production as similar as possible**
11. Treat logs as event streams
12. Run admin/management tasks as one-off processes

Microservice Types

- Application (frontend)
- Business (backend)
- Infrastructure (config, discovery, route)

- Event
  (stream, transform, process, analyze, store)
- Task / Job
  (one time or scheduled)

Kotlin Server Side (Examples)

- Ktor (100% Kotlin web framework)
  Coroutines (async thread-like)
  http://ktor.io

- Vert.x (non-blocking event driven toolkit)
  http://vertx.io

- Spring WebFlux (start.spring.io)
  Reactive (RxJava)
  https://docs.spring.io/spring-framework/docs/5.0.x/kdoc-api/spring-framework/

Kotlin Server Side (Honorable Mention)

- Hexagon (simple microservice framework)
  http://hexagonkt.com

- Javalin (simple REST library)
  (Java and Kotlin)
  https://github.com/tipsy/javalin

- kottpd (simple REST framework, pure Kotlin)
  https://github.com/gimlet2/kottpd

Kotlin Server Side: Ktor

- Ktor (web framework, all Kotlin)
  Coroutines (async threads)
  http://ktor.io
- HTTP WebVerbs
- HTML Builder for Responses
- Integrates with Coroutines
- Verify basic
- Future All Native Solution

Kotlin Server Side: Ktor

```
get("/rx") {
    log("/rx")
    val start = System.currentTimeMillis()
    kotlinx.coroutines.experimental.run(CommonPool) {
        call.handleLongCalculation(start)
    }
}
```

Kotlin Server Side: Ktor

DEMO

Kotlin Server Side: Vert.x

- **Vert.x**: event driven asynchronous framework
- Un-opinionated (unlike Spring Boot)
- Actor Model (Verticles)
- Fluent API (chaining method calls)
- Reactive
  - Always Responsive
  - Message Driven
- Polyglot (JVM languages), originally Node.x, taking on Node.js
- Clustering via Event Bus
- Supports HTTP 2.x (non-blocking I/O)

Kotlin Server Side: Vertx

- http://vertx.io/docs/vertx-core/kotlin/
- Comes with some examples
  - web (gradle)
  - kotlintest (gradle)
  - coroutines (maven)

```
// Build Vert.x Web router
val router = Router.router(vertx)
router.get("/movie/:id").coroutineHandler { ctx ->
getMovie(ctx) }
router.post("/rateMovie/:id").coroutineHandler { ctx ->
rateMovie(ctx) }
router.get("/getRating/:id").coroutineHandler { ctx ->
getRating(ctx) }
```

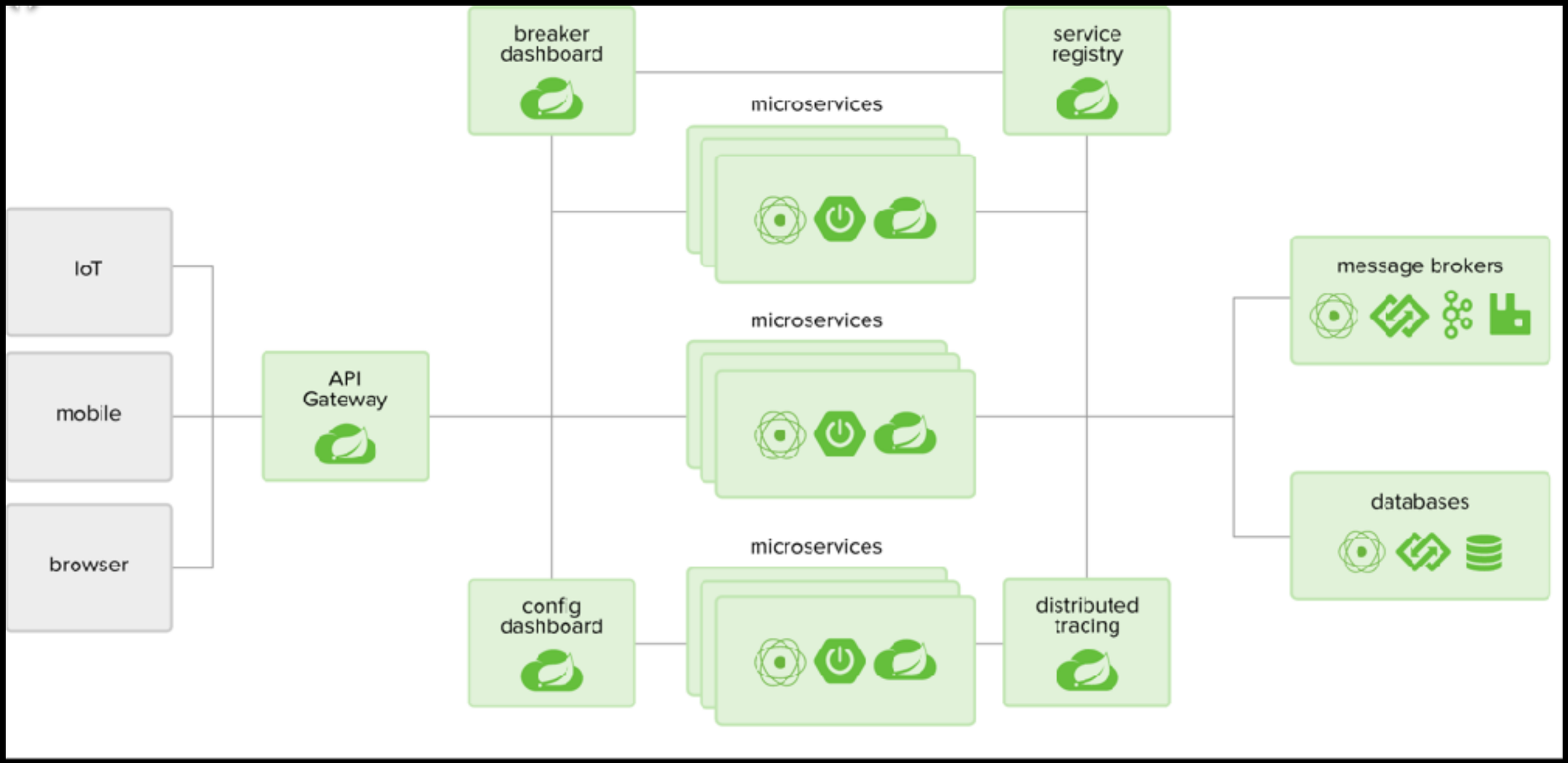
Kotlin Server Side: Vertx

DEMO

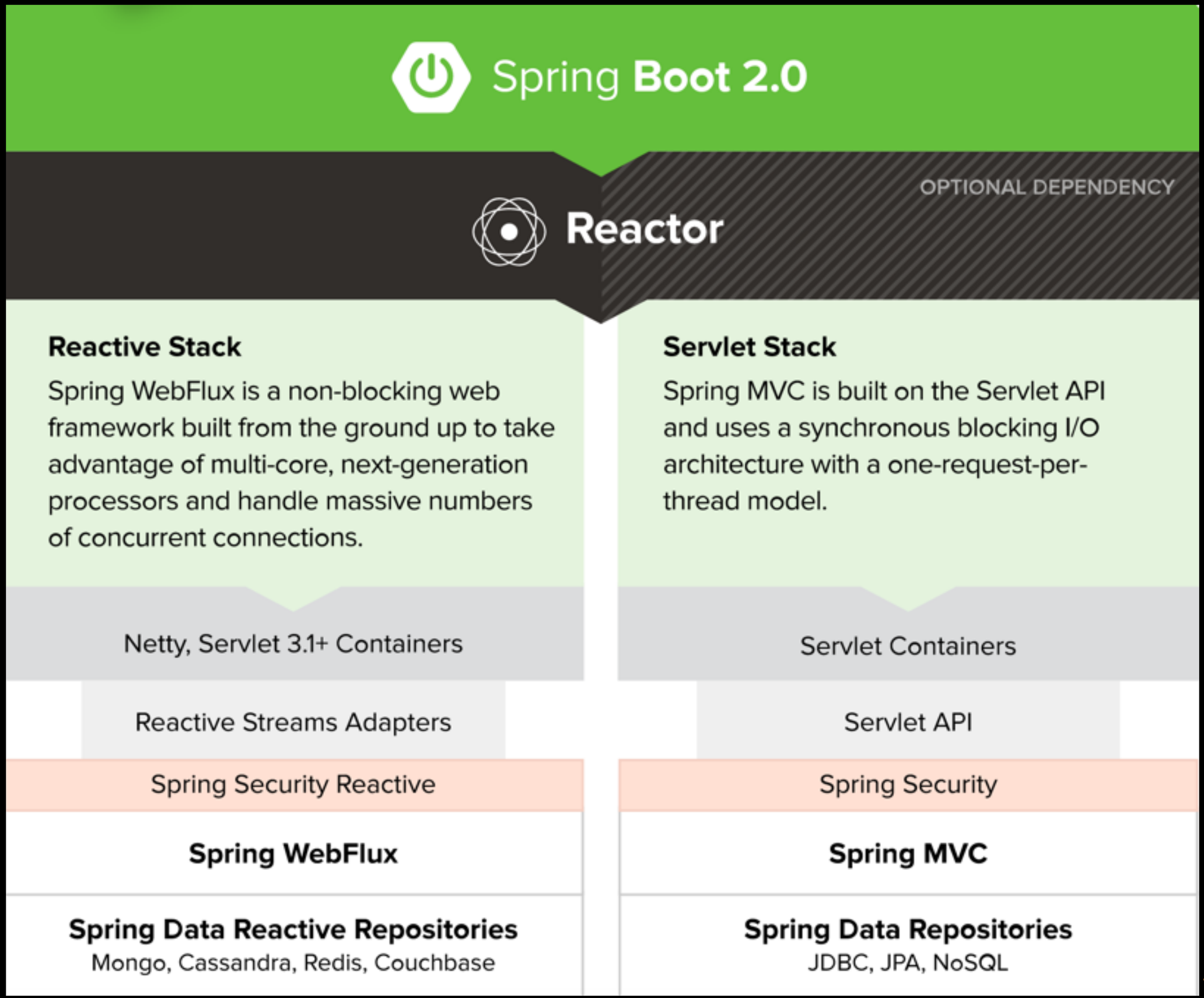
Kotlin Server Side: Spring

Spring Boot 2
Spring Framework 5
Spring WebFlux + Reactive (RxJava)
Kotlin 1.1 +

Kotlin Server Side: Spring



Kotlin Server Side: Spring



Kotlin Server Side: Spring

- Spring WebFlux (start.spring.io)
  Reactive (RxJava)
  https://docs.spring.io/spring-framework/docs/5.0.x/kdoc-api/spring-framework/
- Opinionated (but can still choose)
- Supports Discovery
- Reactive
  - Responsive (need to code for it)
  - Function Driven
- Polyglot (JVM languages)

Kotlin Server Side: Spring

- Clustering Services (Eureka, Zuul, Consul)
- Supports Servlet 3.1 (non-blocking I/O)
- Kotlin Friendly Null Safety Spring
- Special Annotations “Open” Kotlin classes
- No special CGLIB or Aspects
- Jackson Kotlin Module
- Bean Registration through Lambda
- Reative respones
  - Mono (single result)
  - Flux (stream result)

Kotlin Server Side: Spring

start.spring.io

Apps | Smart Bookmarks | Google | DuckDuckGo | DuckDuckGo | DuckDuckGo | DuckDuckGo | DuckDuckGo | Other Bookmarks

SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Kotlin and Spring Boot 2.0.0 M5

Project Metadata

Artifact coordinates

Group

com.lse.example.kotlin.boot

Artifact

03-spring

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Web × JPA × H2 × DevTools × Actuator ×

Generate Project ⌘ + ⏎

Don't know what to look for? Want more options? Switch to the full version.

LSE

Kotlin Server Side: Spring

start.spring.io

Apps | Smart Bookmarks | Google | DuckDuckGo | DuckDuckGo | DuckDuckGo | DuckDuckGo | DuckDuckGo | Other Bookma

SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Kotlin and Spring Boot 2.0.0 M5

Project Metadata

Artifact coordinates

Group

com.lse.example.kotlin.boot

Artifact

03-spring-fnx

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Reactive Web × Actuator × JPA × H2 × DevTools ×

Generate Project ⌘ + ⏎

Don't know what to look for? Want more options? Switch to the full version.

Kotlin Server Side: Spring

DEMO

Kotlin Server Side: Summary

- Spring version leverages existing success
- Kotlin + Spring simplifies development
  - Spring Web Service Approach
  - Spring Reactive (WebFlux) Approach
- Ktor future looks good for native services
- Vertx (originally Node.x) is similar but different to Spring

Kotlin Server Side: Resources

- Kotlin Lang
  http://kotlinlang.org
- Try Kotlin
  https://try.kotlinlang.org
- Awesome Kotlin
  https://kotlin.link
- Async Programming w/ Kotlin
  https://www.youtube.com/watch?v=eVxi3BGE5Rc
- Kotlin Native (23m update+build)
  https://github.com/JetBrains/kotlin-native.git
- Native Servless Kotlin
  https://juan-medina.com/2017/07/30/kotlin-native-serverless
- Kotlin Superpower
  https://superkotlin.com

Kotlin Server Side: Resources

- Kotlin Coroutines
  https://github.com/Kotlin/kotlinx.coroutines
- Vert.x
  http://vertx.io/docs/vertx-core/kotlin
- Spring Framework 5 with Kotlin

  https://spring.io/blog/2017/01/04/introducing-kotlin-support-in-spring-framework-5-0
  https://spring.io/blog/2017/08/01/spring-framework-5-kotlin-apis-the-functional-way
- Kotlin w/ Spring Boot 2
  https://www.youtube.com/watch?v=5zGznI6gtOo
- Apache Bench (yum install httpd-tools)
  http://httpd.apache.org/docs/2.0/programs/ab.html

Questions ?

- Slides: https://github.com/lseinc/intro-kotlin-microservices.git

David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com
@DavidDLucas

LSE

LSE