

Kickstarting Kotlin Culture: The Journey from Java to Kotlin



Neil Power
[@NeilPower](#)

Speakers



Neil Power

Software Developer

[@NeilPower](#)

I was an Android Developer at Hootsuite from 2015-2017.

Recently switched to Hootsuite's Developer Products team, building out the Hootsuite Developer Platform and experimenting with Kotlin Backend Services.



Introduction

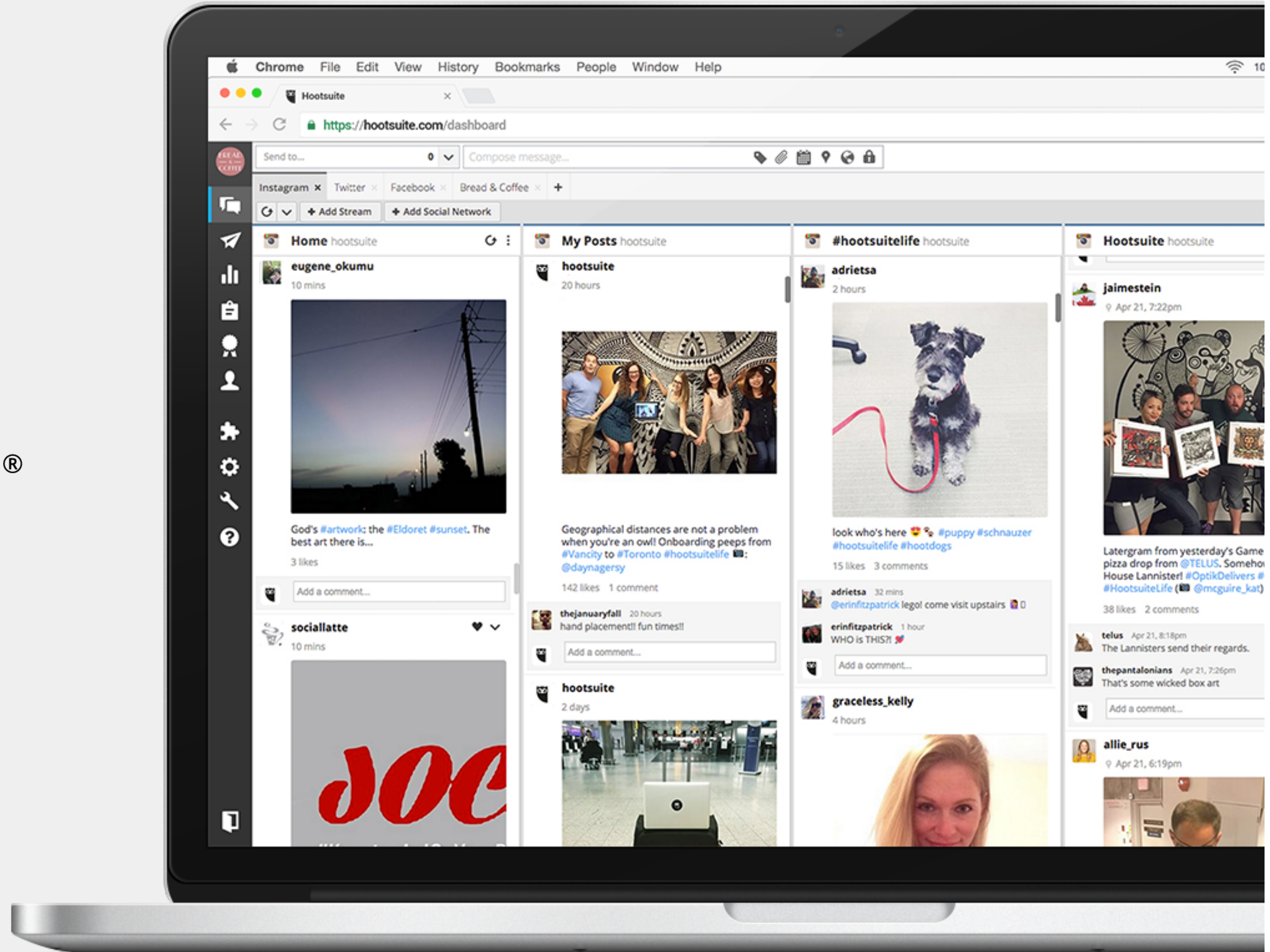
This talk is about our experience adopting Kotlin at Hootsuite. Based on the lessons we learned, we want to share our story so others can accelerate their adoption. We are growing Kotlin culture at Hootsuite and facilitating that growth in the community at large.

The core of our story is about empowering other individuals and companies to learn from our experiences and turn to the future of Kotlin in their organization.



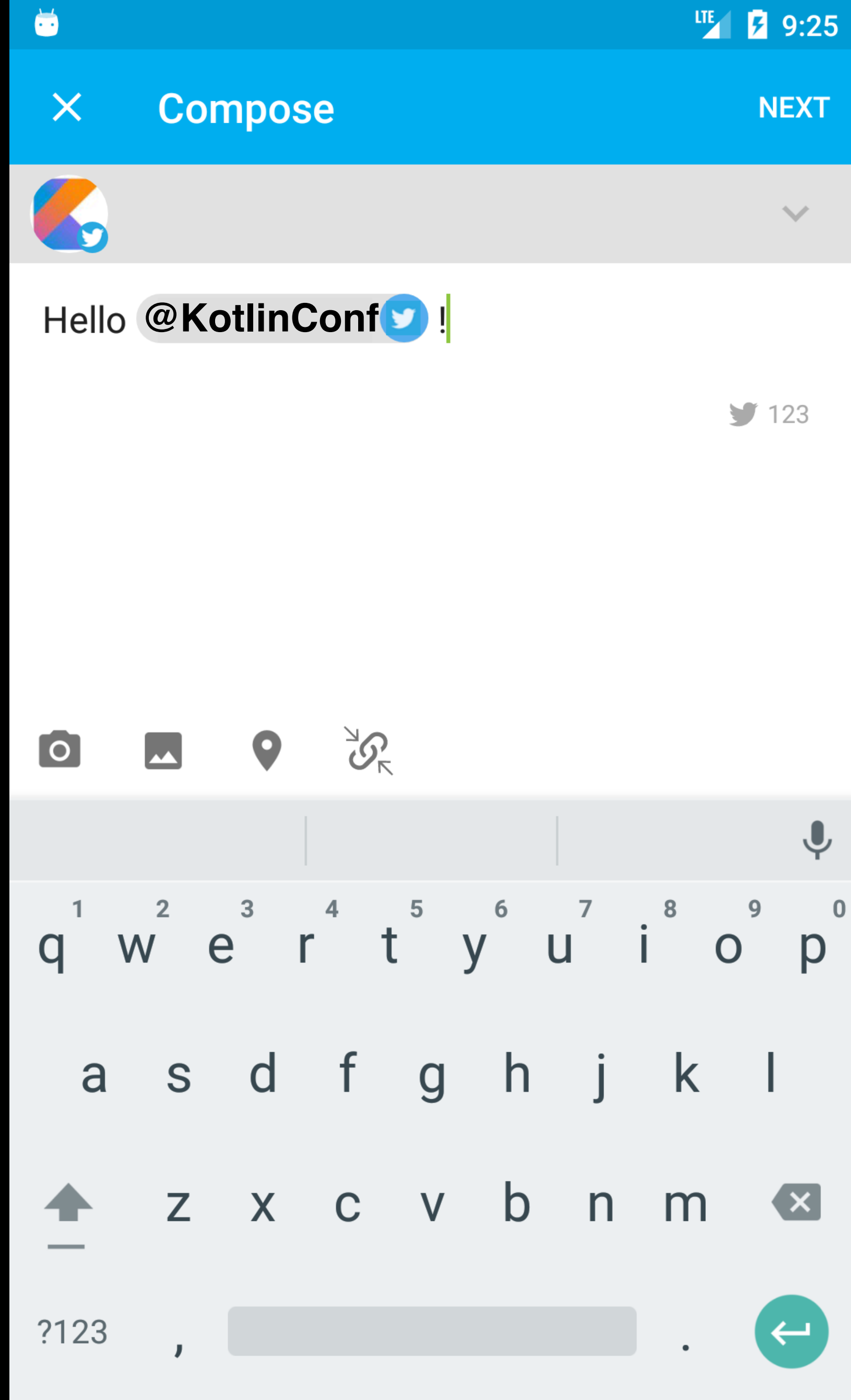
Background





Hootsuite Mobile





Why am I here?

- Experimentation
 - Hootsuite was a fairly early adopter of Kotlin in production. Great opportunity to share the lessons we learned in that process.



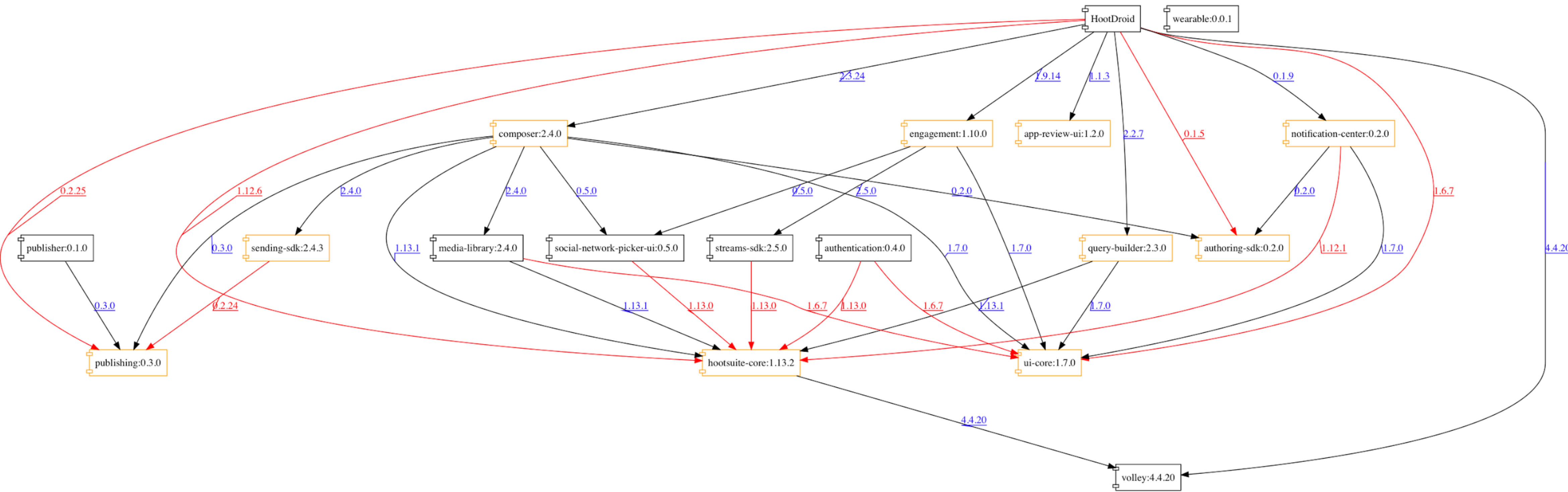
Why am I here?

- Experimentation
 - Hootsuite was a fairly early adopter of Kotlin in production. Great opportunity to share the lessons we learned in that process.
- Work out loud
 - Work passionately, know that what you're doing will be valuable to someone, and strive to share what you're working on with those around you.



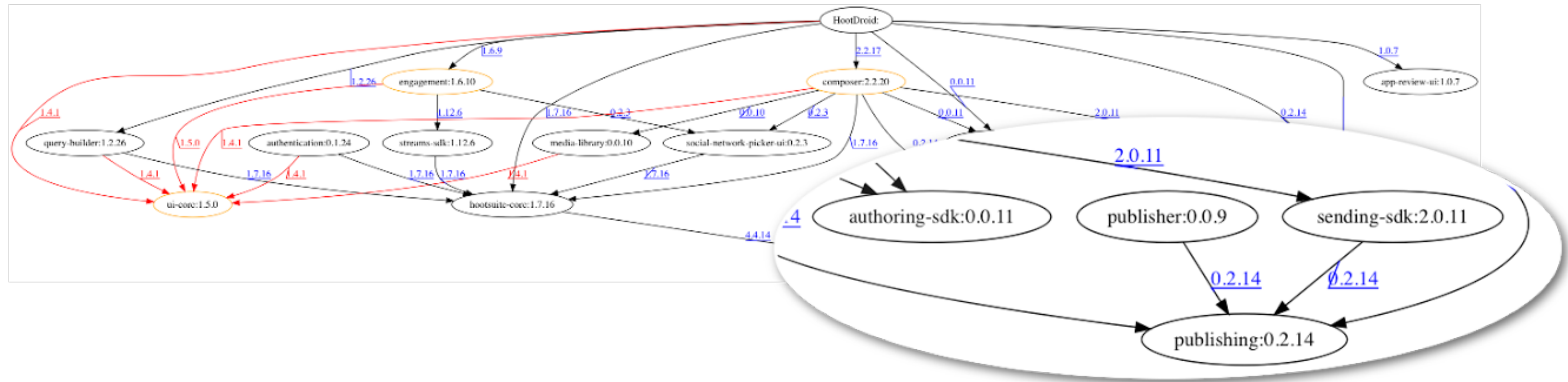
How We Work

We have a core team focussed on library driven development and being able to ‘slice’ our app into verticals. These verticals are worked on by other teams around the company.



How we Work

In this dependency graph, each node is a repository that one or more teams contributes to and maintains.



Potential Risks of Kotlin

- **Unofficial Android Language**



Potential Risks of Kotlin

- **Unofficial Android Language**
- The team learning a new language



Potential Risks of Kotlin

- **Unofficial Android Language**
- The team learning a new language
- Increased build times



Potential Risks of Kotlin

- **Unofficial Android Language**
- The team learning a new language
- Increased build times
- Stability of our CI Pipeline



Potential Risks of Kotlin

- **Unofficial Android Language**
- The team learning a new language
- Increased build times
- Stability of our CI Pipeline
- IDE Tooling support



Potential Risks of Kotlin

- **Unofficial Android Language**
- The team learning a new language
- Increased build times
- Stability of our CI Pipeline
- IDE Tooling support
- Hiring Considerations



The First Line of Kotlin



First Exposure to Kotlin

- February 2016
 - A developer who had participated in the Scala experiment at Hootsuite took notice of Kotlin and suggested trying it at Hootsuite.
 - He shared Jake Wharton's now famous "Using Project Kotlin for Android" proposal with the team.



First Exposure to Kotlin

- March 2016
 - Not everyone was yet convinced.
 - More evidence weighed and analyzed, including “Talking Kotlin with Hadi Hariri” on the Fragmented Podcast.
 - Core Android manager, Simon Tse, created a sample app to test Kotlin.



First Exposure to Kotlin

- April 2016
 - Our director, Paul Cowles, encouraged us to keep exploring
 - Simon converted the first POJO data class to Kotlin.



The First Commit

Dawn of Kotlin into the core library

[Browse files](#)

Summary:

- * Added kotlin
- * Converted `HootsuiteUser` to Kotlin
- * Tested that converted `HootsuiteUser` works when consumed in main app




Test Plan:

- * Builds
- * Unit tests rung

Reviewers: Dave, Madeleine, DenisT, wesley.alcock

Reviewed By: DenisT, wesley.alcock

Differential Revision: <http://app1.phabricator.us-east-1.hootops.com/D777>

 master  1.7.16  1.4.5



simon-tse-hs committed on Apr 6, 2016

1 parent [126f13e](#)

commit [436d115b6229cf6404241fa329142a6dd052a43b](#)

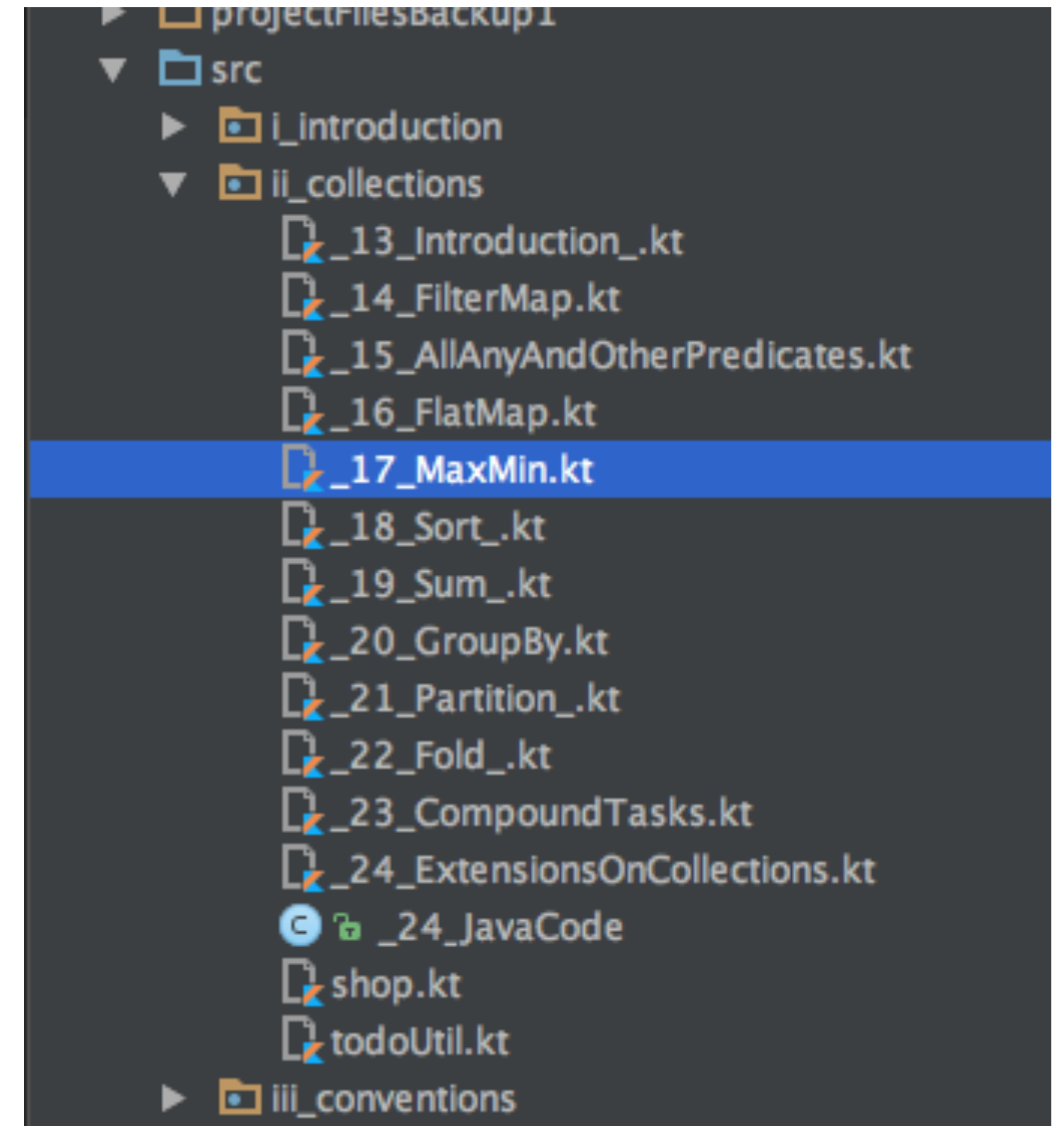


Diving In



Kotlin Koans

- Great exercises to get started.
- Each developer on the team took the Koans home.
- We shared solutions and challenges with the rest of the team.



Java with Kotlin Syntax

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    val toolbar: Toolbar = findViewById(R.id.toolbar) as Toolbar  
    setSupportActionBar(toolbar)  
    if (savedInstanceState != null) {  
        query = savedInstanceState.getParcelable(STATE_QUERY)  
    }  
}
```



Getting More Comfortable

- Reading the reference docs from <https://kotlinlang.org/>
- Experimentation with Kotlin features
- Informal knowledge sharing



Getting More Comfortable

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    setSupportActionBar(toolbar) // Synthetic Import  
    savedInstanceState?.let {  
        query = it.getParcelable(STATE_QUERY)  
    }  
}
```




Getting More Comfortable

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    → setSupportActionBar(toolbar) // Synthetic Import  
    savedInstanceState?.let {  
        query = it.getParcelable(STATE_QUERY)  
    }  
}
```



Getting More Comfortable

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    setSupportActionBar(toolbar) // Synthetic Import  
    savedInstanceState?.let {   
        query = it.getParcelable(STATE_QUERY)  
    }  
}
```



Love at First Compile

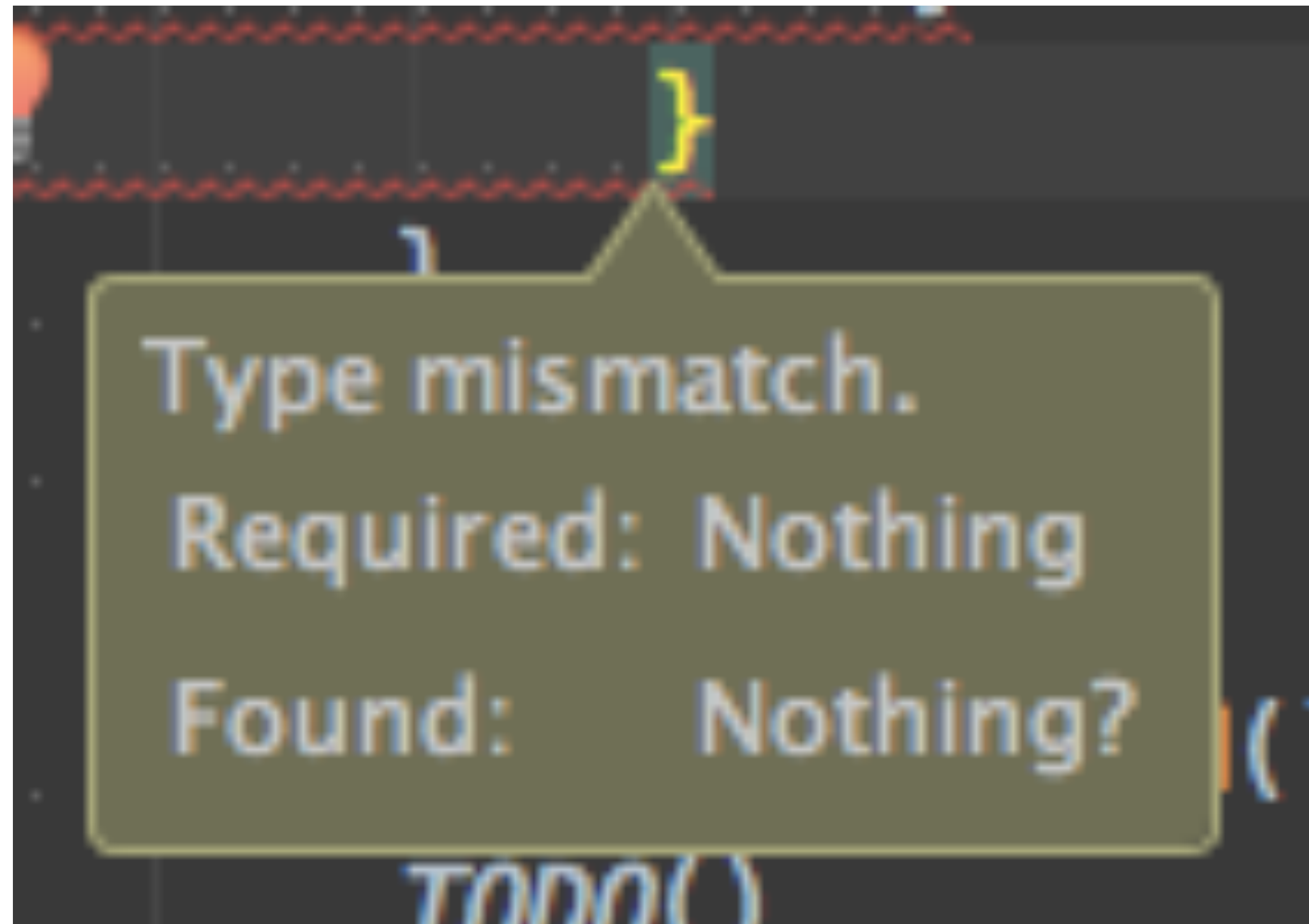


Data Classes

```
data class Query(  
    var primary: List<String>? = null,  
    var secondary: List<List<String>>? = null,  
    var exclusionary: List<String>? = null,  
    var tweetTypes: TweetType? = TweetType.ALL,  
    var excludedTypes: ExcludedType? = ExcludedType(false, false),  
    var sentiment: Sentiment? = Sentiment(false, false, false),  
    var engagement: Engagement? = null,  
    var language: String? = null,  
    var location: Location? = null)
```



Nullable Types



Rich Collections API

```
val sortedEntities = entities?.urls  
    .orEmpty()  
    .filter { url -> !isQuotedURL(url, quotedStatus) }  
    .mapNotNull { url -> Tag(referenceId = url.url) }  
    .sortedBy(Tag::offset)
```



First Class Functions

```
internal fun <T> likePost(  
    postComplete: PostComplete,  
    socialProfileId: Long,  
    apiLikePost: () -> Observable<T>): Observable<Int>
```



Idiomatic Kotlin



Zippping Infinite Sequences

```
(messageTags
    .orEmpty()
    .asSequence()
    .zip(generateSequence { TagLocation.MESSAGE } ) +

storyTags
    .orEmpty()
    .asSequence()
    .zip(generateSequence { TagLocation.STORY })))
.mapNotNull { (textTag, tagLocation) ->
    Tag(referenceId = textTag.id ?: return@mapNotNull null)
}.toList()
```



Reified Inline Functions

```
inline fun <reified T> Intent.getNotificationList(key: String):  
    List<Notification> where T : Notification, T : Parcelable =  
    this  
        .getParcelableArrayListExtra<T>(key)  
        .orEmpty().map { it as Notification }
```



The First Kotlin Libraries



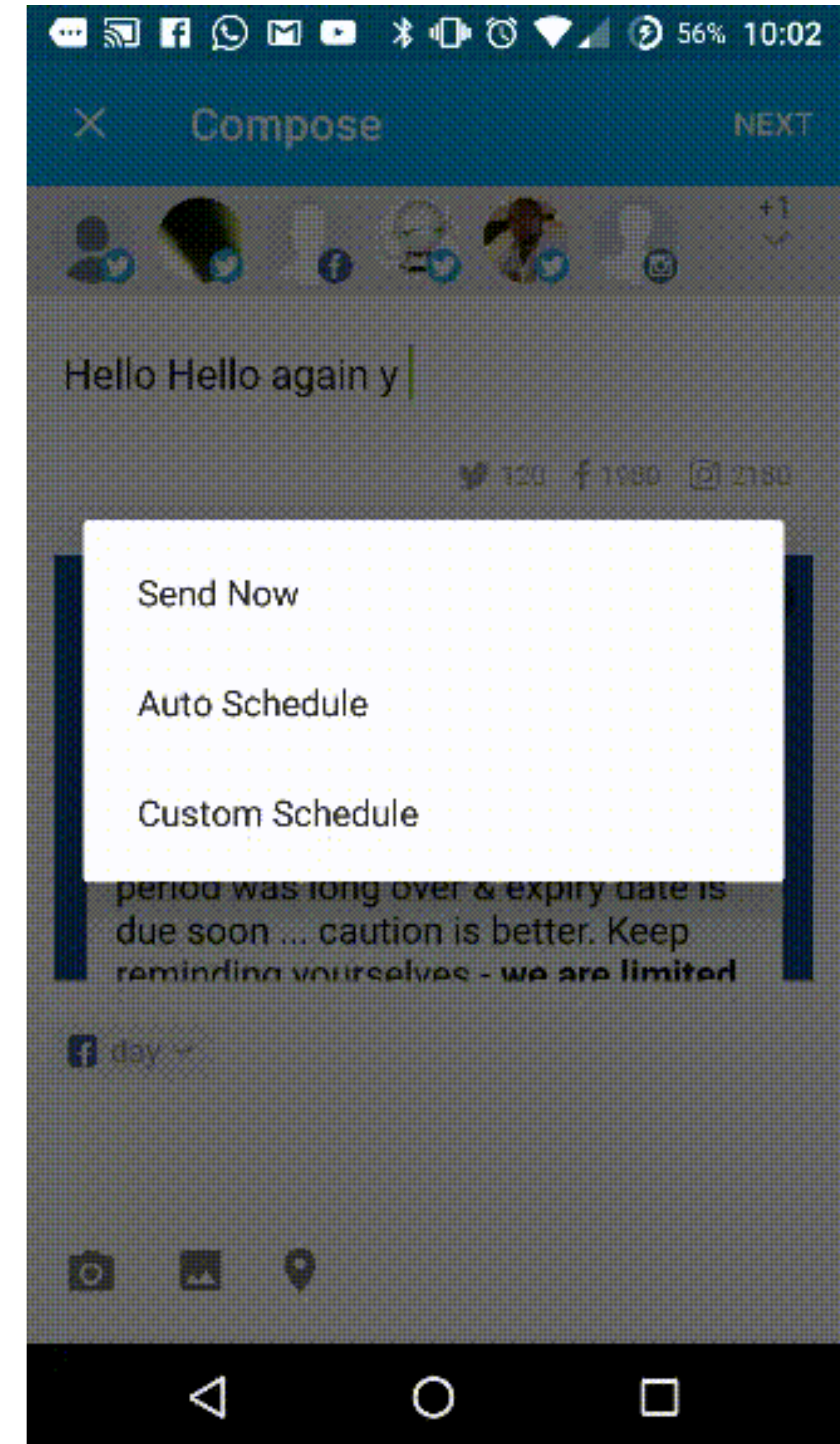
First Kotlin Feature

- We decided to write a new feature in Kotlin.
- Should have enough technical complexity to test Kotlin.
- Should be decoupled from the main code base.



Compose Feedback

- First Library >90% Kotlin
- Interesting Technical Components
 - Job Queue
 - Persistence
 - API calls
 - Custom Views

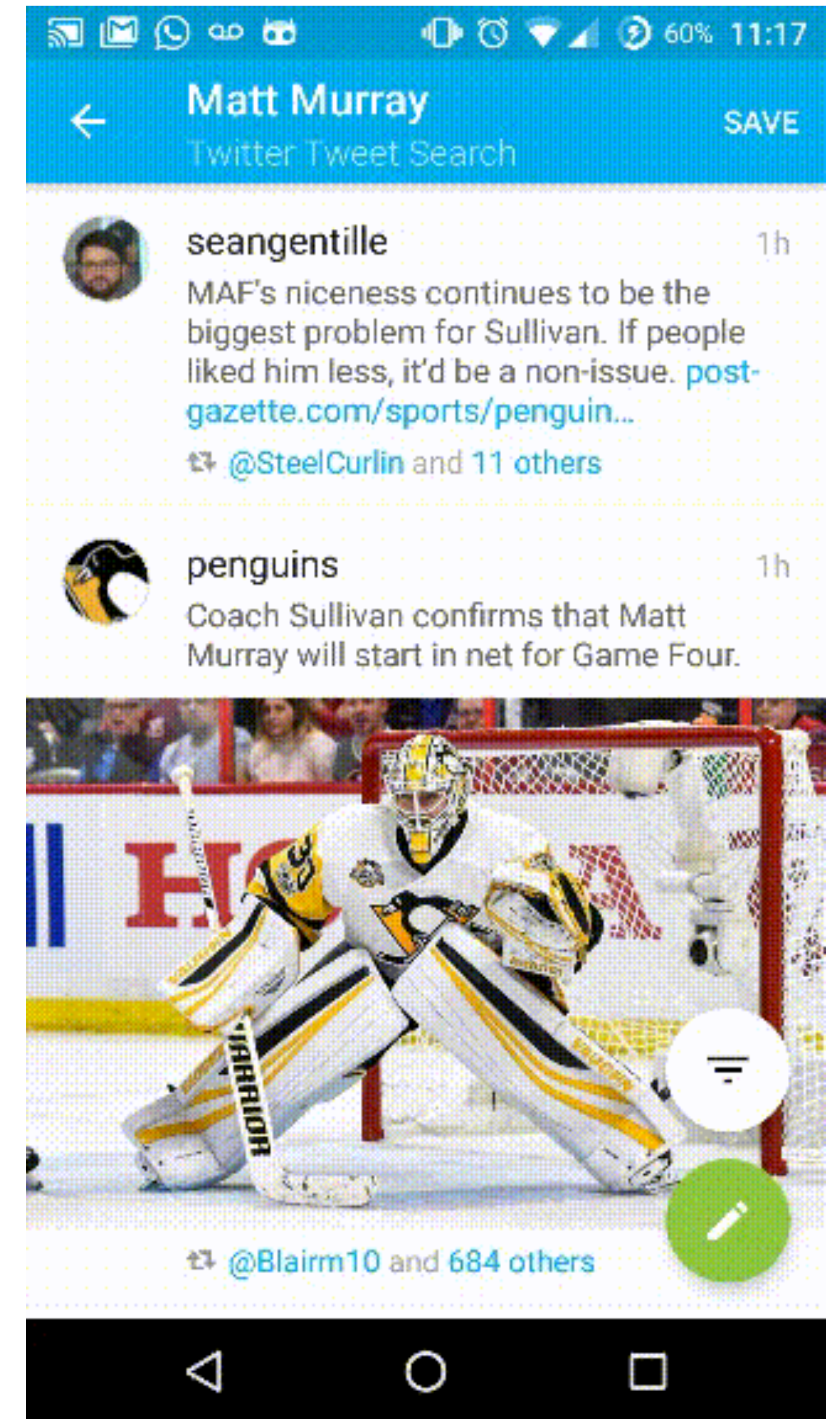


First 100% Kotlin Library



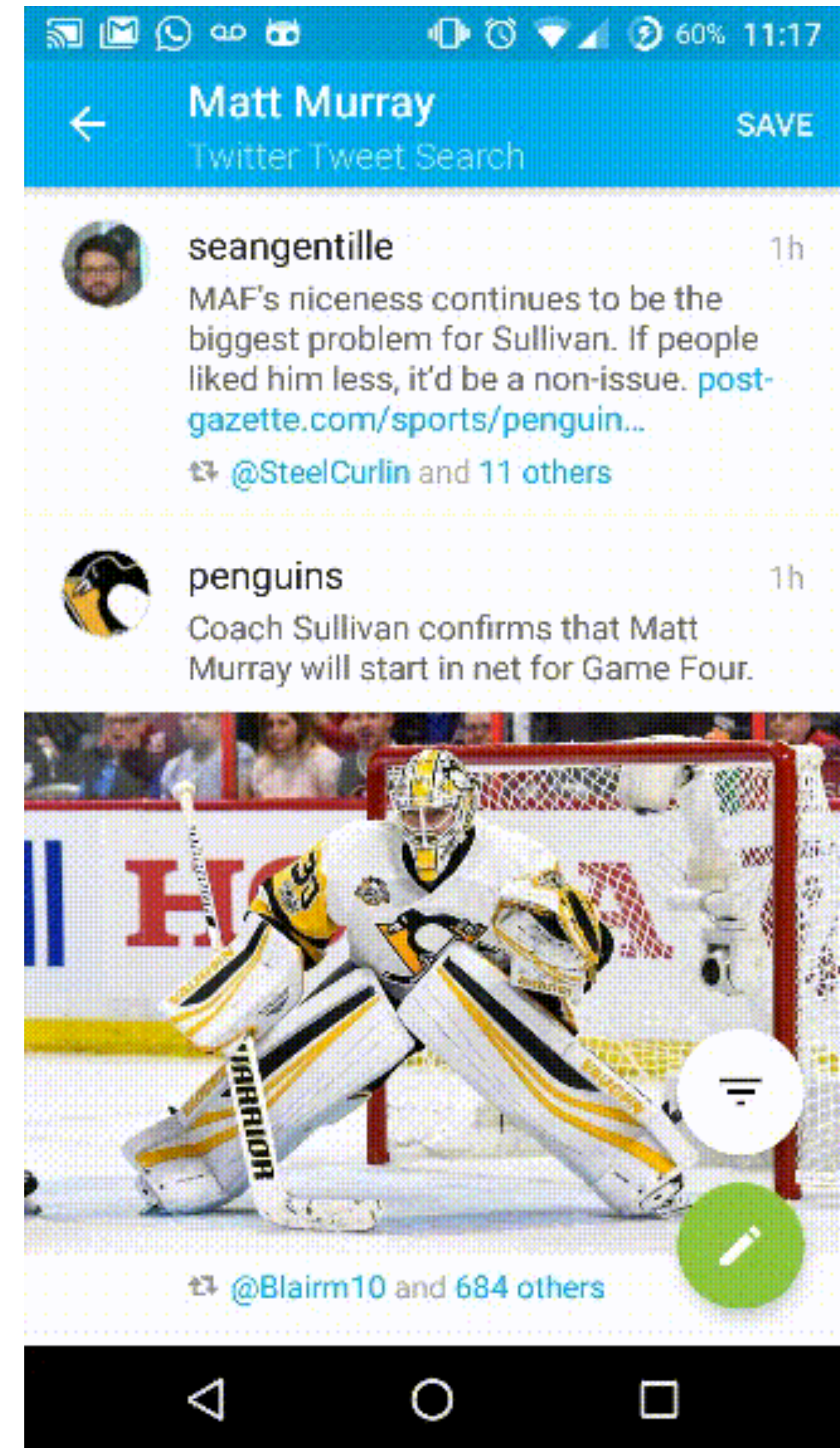
Query Builder

- Twitter Search Query Builder
- Started >90% Kotlin, first to 100%
- Challenges
 - Issues with Dagger + Kapt



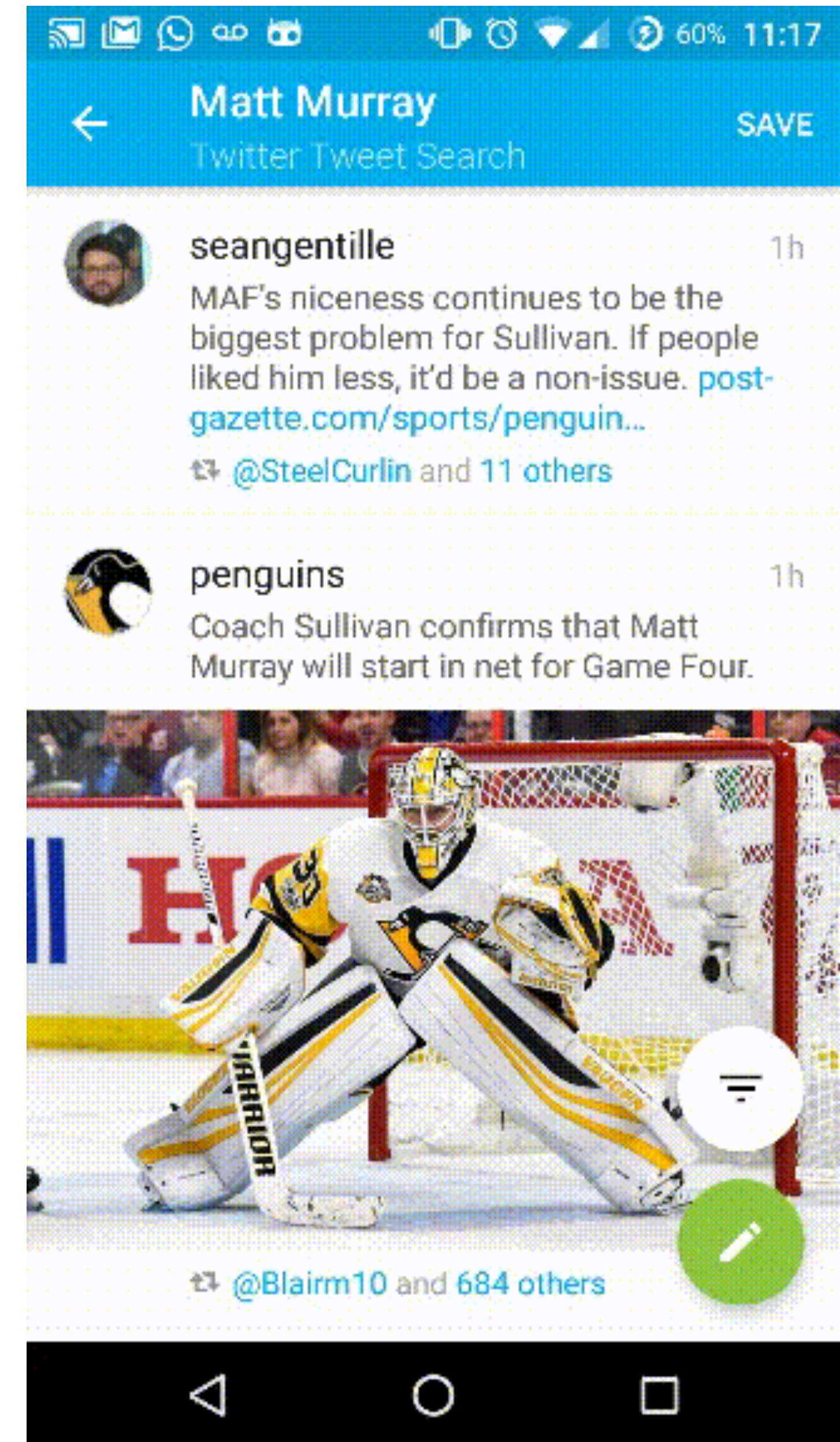
Query Builder

- Twitter Search Query Builder
- Started >90% Kotlin, first to 100%
- Challenges
 - Issues with Dagger + Kapt
 - Integration with ANTLR



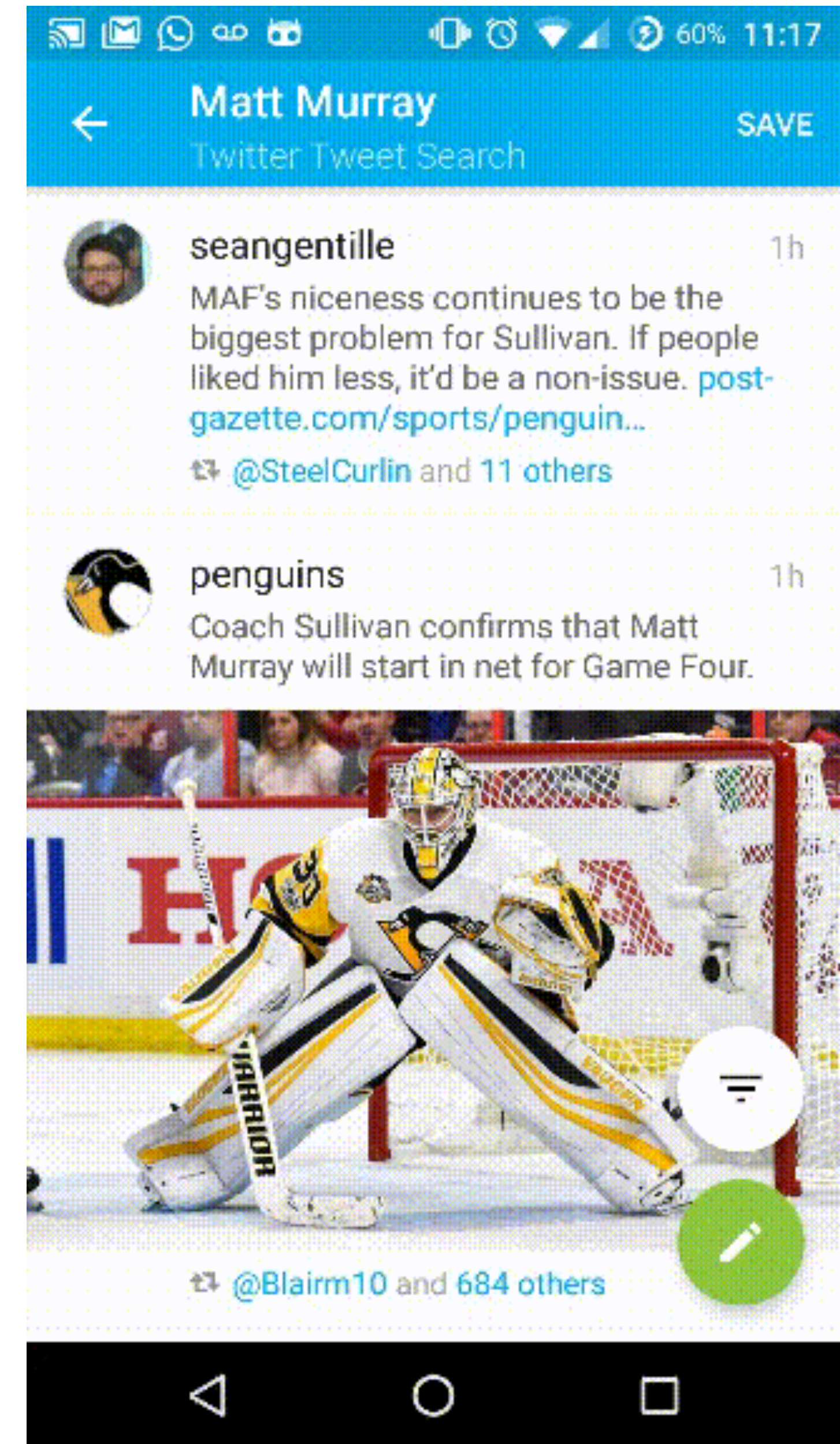
Query Builder

- Twitter Search Query Builder
- Started >90% Kotlin, first to 100%
- Challenges
 - Issues with Dagger + Kapt
 - Integration with ANTLR
 - Communication with AWS Lambda



Query Builder

- Isolated Experiment, away from the main app.
- Lower risk for customers
- Extracting into a library was great for mitigating our uncertainty with Kotlin.
 - Faster Build Cycle
 - Testing in Isolation



Kotlin Love from the Team

- We work in sprints with retrospectives.
 - What is working?
 - What's not working?



Kotlin Love from the Team

Good

1. Using advanced **Kotlin** features, **Kotlin** guild
3. Good work on publisher service
4. **Kotlin** working well and good support via slack

1. No Major Build Issues

2. **Kotlin** Use -> + Learning, Build

- jira, rebasing, **kotlin** great

- Storio **Kotlin** support ready

9. **Kotlin**

- Crazy **Kotlin**

- **Kotlin** Love

Growing Kotlin Culture



Evangelizing Kotlin

- Kotlin Koans were a great start.
- Kotlin started on the core team, feature teams still on Java.
- How could we continue to grow Kotlin at Hootsuite?
 - Start to develop Kotlin standards.
 - Elevate Kotlin knowledge on the team.
 - Share lessons learned from the community.



What are Guilds?

“Collective action is a wonderful thing. [People] getting together to have a conversation about the things they care about, with the aim to do something positive for themselves and their community.

How can this collective action happen inside your organization? At Hootsuite, the answer is Guilds.” - Noel Pullen



Evangelizing Kotlin

- The Solution
 - Kotlin Guild



Evangelizing Kotlin

- The Solution
 - Kotlin Guild
 - Kotlin Book Club



Evangelizing Kotlin

- The Solution
 - Kotlin Guild
 - Kotlin Book Club
 - Vancouver Kotlin Meetup Group




Kotlin Guild





Kotlin

Kotlin Guild

 Open group

Joined ▼

 Share

 Notifications



Discussion

Members

Files

Events

Videos

More ▼

Search this group



Members 16 Admins 1 Blocked 0

Default ▼

+ Add

Import

 Find a Member



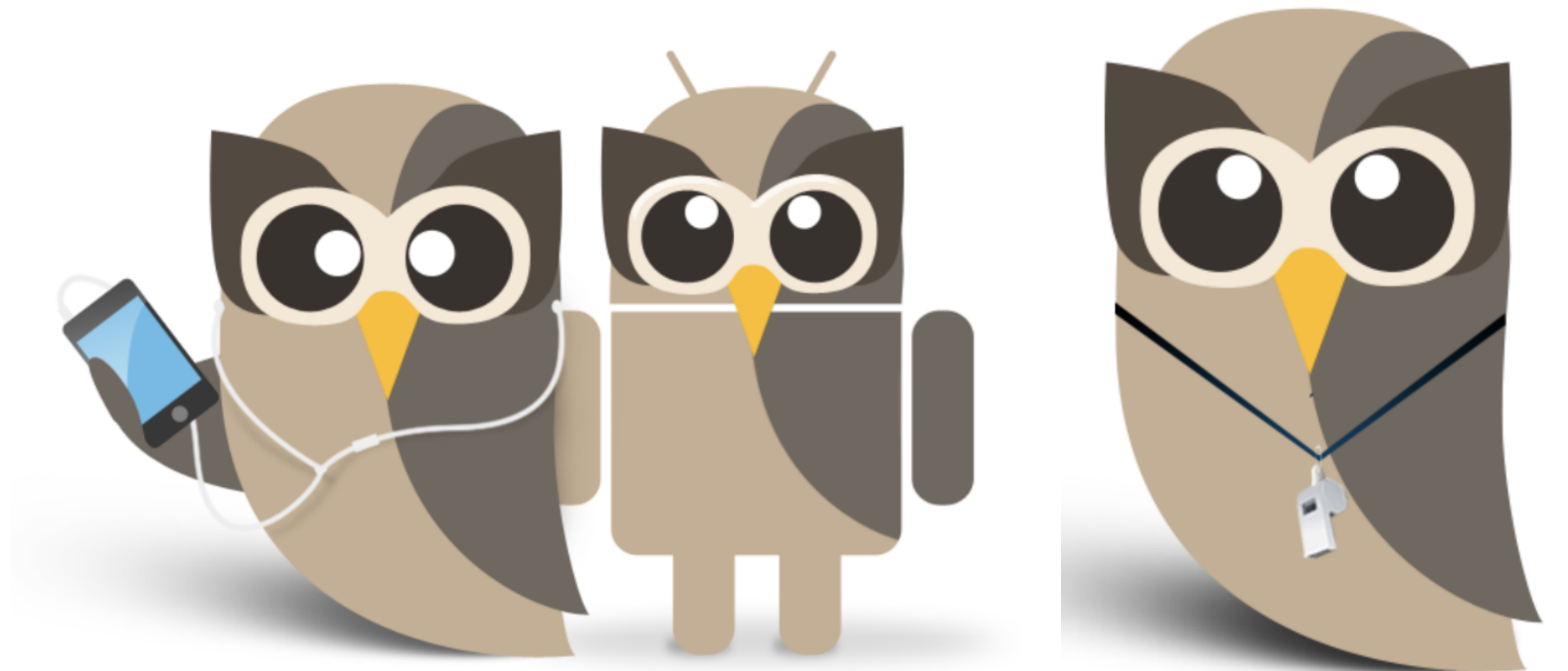
Kotlin Guild

- Purpose
 - Sharing Best Practices
 - Discussing Interesting Articles/Videos
 - Creating a Unified Kotlin Style at Hootsuite



Kotlin Guild

- Format
 - Bi-Weekly Meetings
 - Opt-In
 - Shared Agenda
 - Company Wide



Kotlin Guild

2016-08-11: Session 3

- Neil : Kotlin Koans Review
- Noah: Delegation in Query Builder
- Denis: Kotlin language proposals



KotlinConf Announced

Session 20: 2017-03-23

- Kotlin Conf Brainstorm - <https://blog.jetbrains.com/kotlin/2017/03/23/kotlin-conf-2017/> - Neil Power : Kotlin Conf has been announced, let's brainstorm sessions that we can submit to the conference!
- Kotlin 1.1 Event Retro - Neil Power



KotlinConf Announced

Session 20: 2017-03-23

- Kotlin Conf Brainstorm - <https://blog.jetbrains.com/kotlin/2017/03/23/kotlin-conf-announced/> - Neil Power : Kotlin Conf has been announced, let's brainstorm sessions that we can submit to the conference!
- Kotlin 1.1 Event Retro - Neil Power



Sequences?



Something New

- Encountered Sequences through the guild.
 - A first for the whole team.



Sequences

```
listOf(1, 2)  
  .asSequence()  
  .map { print("map "); it * it }  
  .filter { print("filter "); it % 2 == 0 }  
  .toList()
```



Unlazy Expectations

```
listOf(1, 2)  
  .asSequence()  
  .map { print("map "); it * it }  
  .filter { print("filter "); it % 2 == 0 }  
  .toList()
```

Expected: map map filter filter



Lazy Sequence Operations

```
listOf(1, 2)  
  .asSequence()  
  .map { print("map "); it * it }  
  .filter { print("filter "); it % 2 == 0 }  
  .toList()
```

Expected: map map filter filter

Actual: map filter map filter



Structured Learning



Kotlin Book Club

- What?
 - A more structured time to learn Kotlin.
 - Using reference material, an hour long meeting every 2 weeks with personal study between meetings.



Kotlin Book Club

- Why?
 - Increase Developer Engagement with Kotlin



Kotlin Book Club

- Why?
 - Increase Developer Engagement with Kotlin
 - Accelerate Learning by learning together



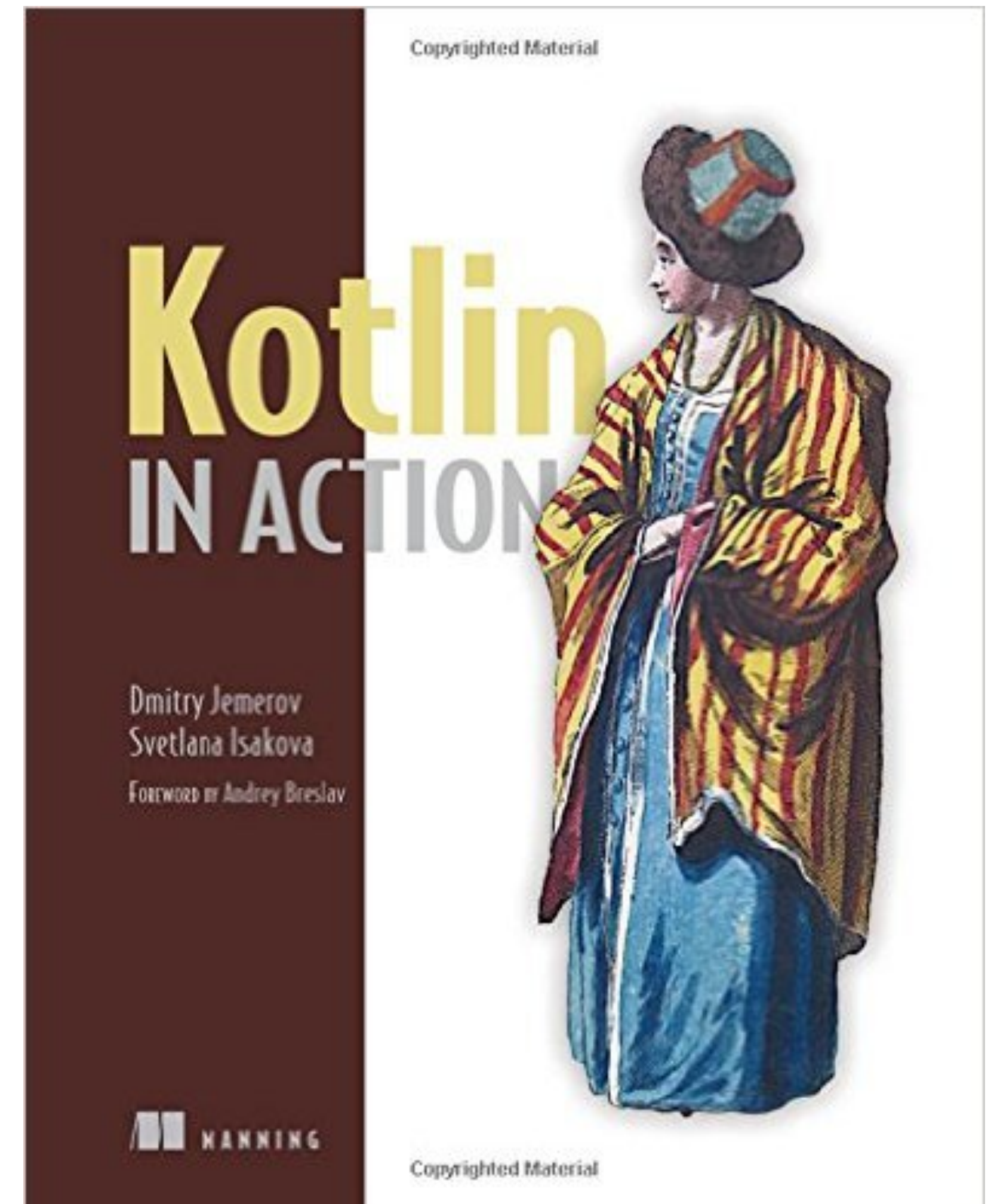
Kotlin Book Club

- Why?
 - Increase Developer Engagement with Kotlin
 - Accelerate Learning by learning together
 - Find the next Sequences



Kotlin Book Club

- Kotlin in Action
 - One chapter a session.
 - Round Table Discussion.
 - Exercises based on Book Material.



Kotlin in Action

- Pragmatic
 - “Kotlin is a practical language designed to solve real-world problems.”



Kotlin in Action

- Pragmatic
 - “Kotlin is a practical language designed to solve real-world problems.”
- Concise
 - “The syntax clearly expresses the intent of the code you read.”



Kotlin in Action

- Pragmatic
 - “Kotlin is a practical language designed to solve real-world problems.”
- Concise
 - “The syntax clearly expresses the intent of the code you read.”
- Safe
 - “Language design prevents certain types of errors in a program.”



Kotlin in Action

- Pragmatic
 - “Kotlin is a practical language designed to solve real-world problems.”
- Concise
 - “The syntax clearly expresses the intent of the code you read.”
- Safe
 - “Language design prevents certain types of errors in a program.”
- Interoperable
 - “Regardless of the kind of APIs, you can work with them from Kotlin.”



Nothing is Everything and Everything is Anything.

- Kotlin Book Club



Building Standards

- When should we use an extension function?
 - Not to simply replace passing a variable to a function.



Building Standards

- When should we use an extension function?
 - Not to simply replace passing a variable to a function.
- When should we be creating or overloading operators?
 - Infix operators are interesting but not always appropriate.



Building Standards

- When should we use an extension function?
 - Not to simply replace passing a variable to a function.
- When should we be creating or overloading operators?
 - Infix operators are interesting but not always appropriate.
- When should we be using sequences?
 - The overhead means that using lists is frequently better.



Building Standards

- When should we use an extension function?
 - Not to simply replace passing a variable to a function.
- When should we be creating or overloading operators?
 - Infix operators are interesting but not always appropriate.
- When should we be using sequences?
 - The overhead means that using lists is frequently better.
- How should we use let/with/run/apply/also?



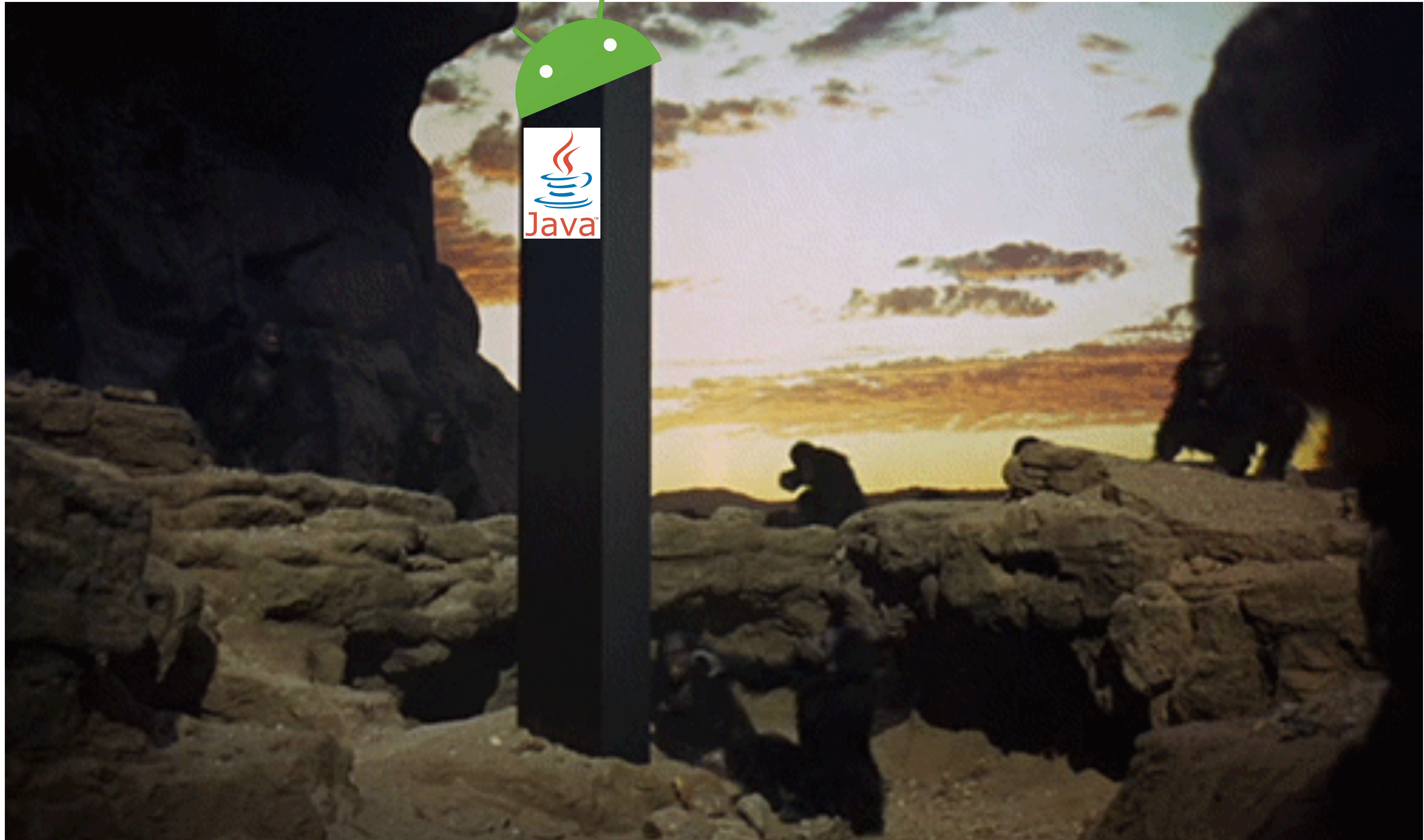
Building Standards

		receiver	
		this	it
returns	this	apply	also
	result	run	let



Library Driven Development





Decomposition



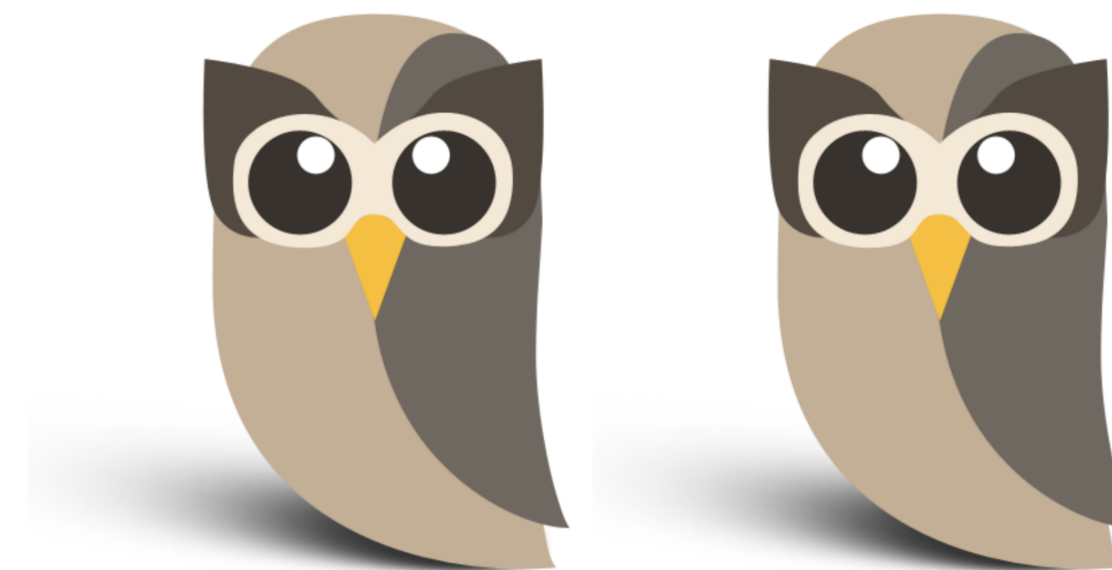
Decomposition into Libraries

- Late 2015
 - We started to think about extracting functionality into Libraries
 - Each library is its own repository



Decomposition into Libraries

- 2016
 - New features starting to be in their own libraries
 - Tools extracted into libraries
 - The start of Kotlin



Decomposition into Libraries

- 2017
 - Many libraries (mostly Kotlin)
 - Ownership distributed across teams



Decomposition into Libraries

- What?
 - New features in their own libraries.
 - Decompose HootDroid into reusable library components.

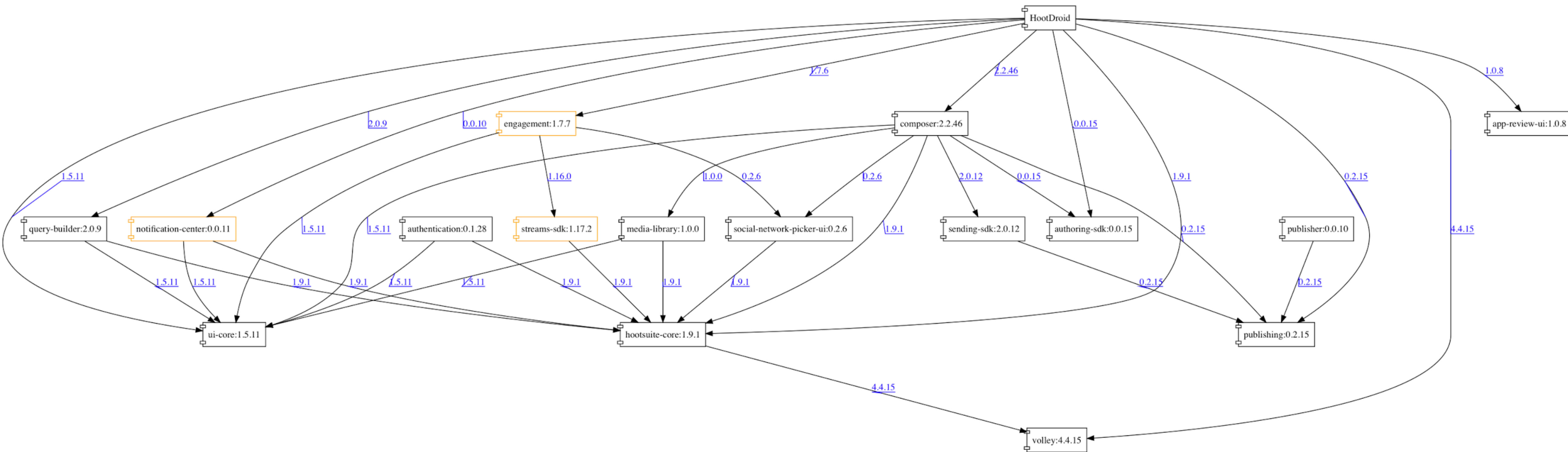


Decomposition into Libraries

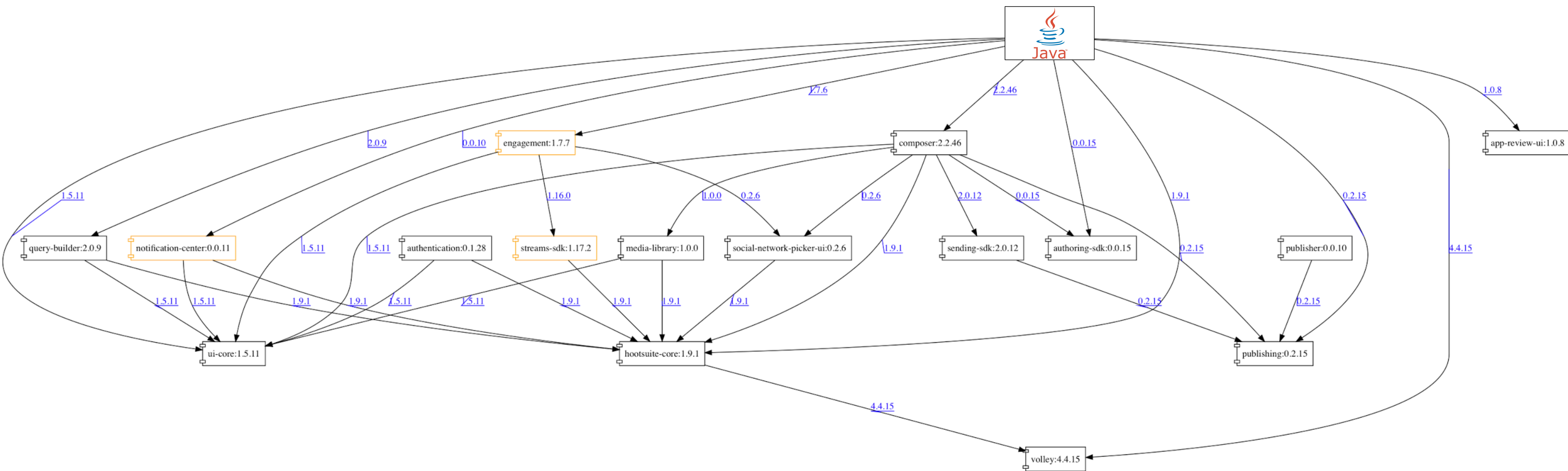
- Why?
 - Better Architecture.
 - Separating libraries forces you to consider interfaces between features and main app.
 - Introduce Kotlin in a steady, but controlled way.



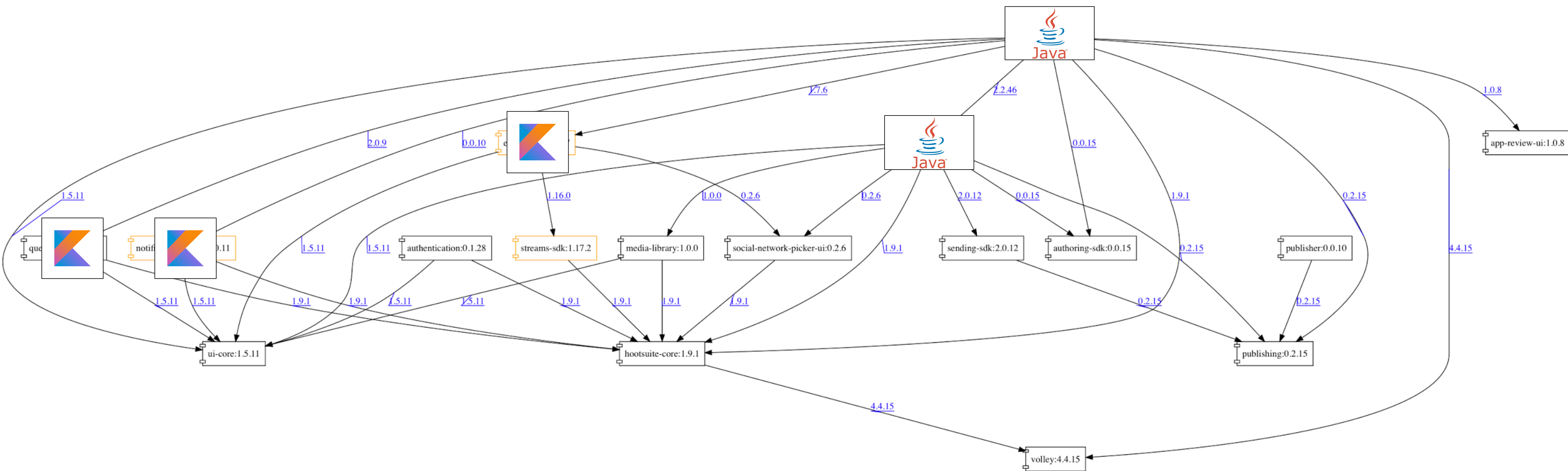
Library Dependency Graph



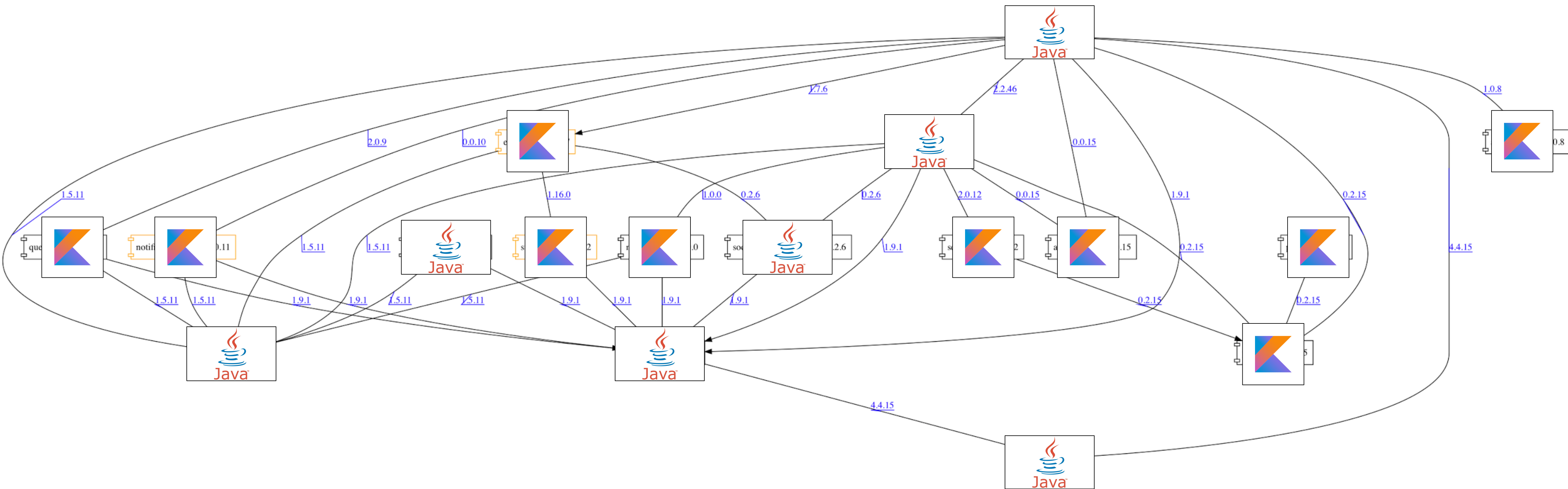
Library Dependency Graph



Library Dependency Graph



Library Dependency Graph



Decomposition into Libraries

- Benefits
 - Easy to make new libraries Kotlin only.
 - Easy to delegate ownership to different teams.
 - Faster builds when using precompiled modules.



Decomposition into Libraries

- Drawbacks
 - Increased overhead for project configuration.
 - Need to keep libraries versioned and in sync.



Library Driven Development Tools



Tools

- Library Driven Development creates some development overhead.
 - Who wants to manually bump versions of internal libraries?
 - At Hootsuite, we use Atlas and Peon to mitigate this overhead.



Atlas

- Atlas is a command line Ruby application which determines project dependencies.
- Originally written by myself and Ben Hergert, we use it to create dependency graphs for iOS, Android, and Scala projects.
- Atlas can visualize the graphs to help developers understand their projects better.

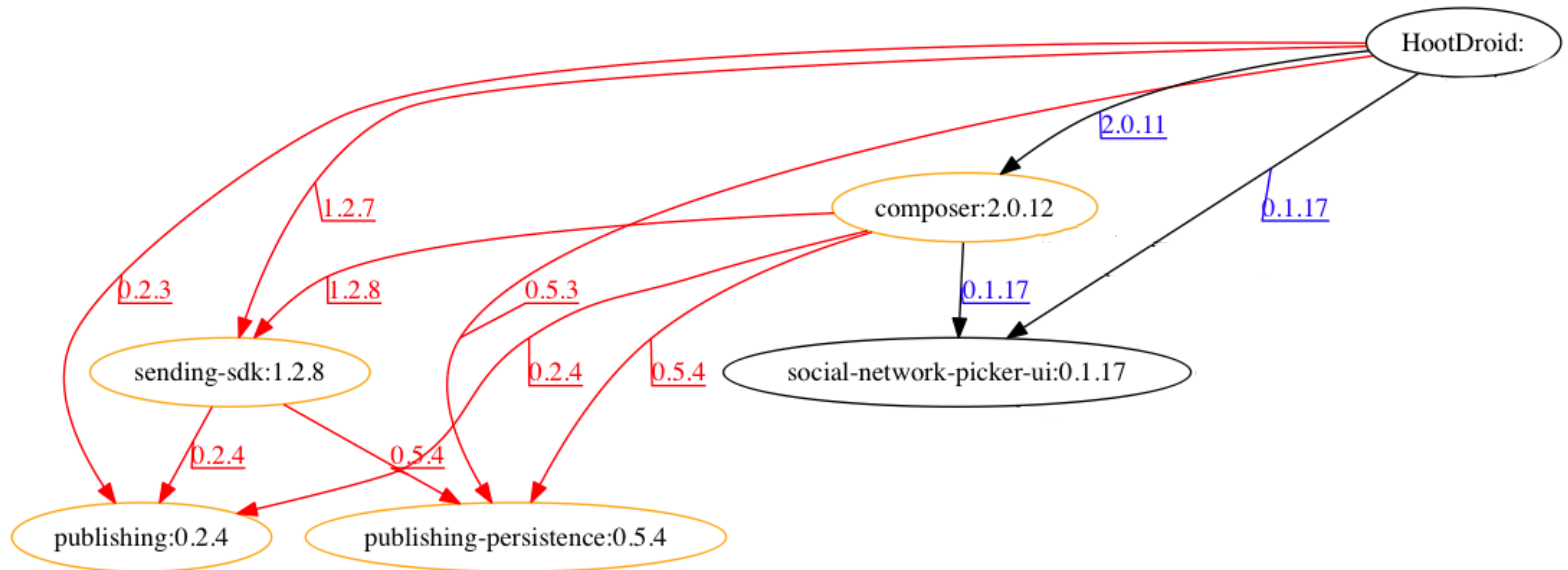


- Below is our current graph



Atlas

- Visually, Atlas will show out of date dependencies as well as mismatched dependencies.



Atlas

- What do we use Atlas for?
 - Graph building and sorting for Peon.
 - Onboarding new Android Developers.
 - Atlas graphs shown on status boards in the office.
 - Diagnose bugs in development.



Peon

- Peon is another command line Ruby application that was written by myself and Simon Tse.
- It takes as input, the graph data structure that Atlas outputs as well as dependency updates you wish to apply.
- Peon then topologically sorts the graph and creates ordered pull requests with the given changes.



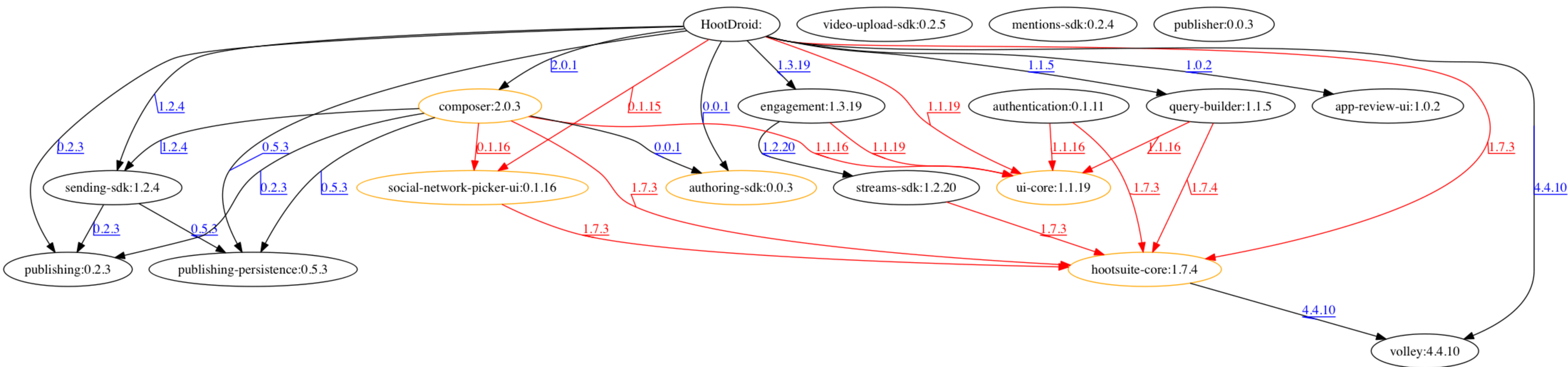


peon APP 1:06 PM ☆

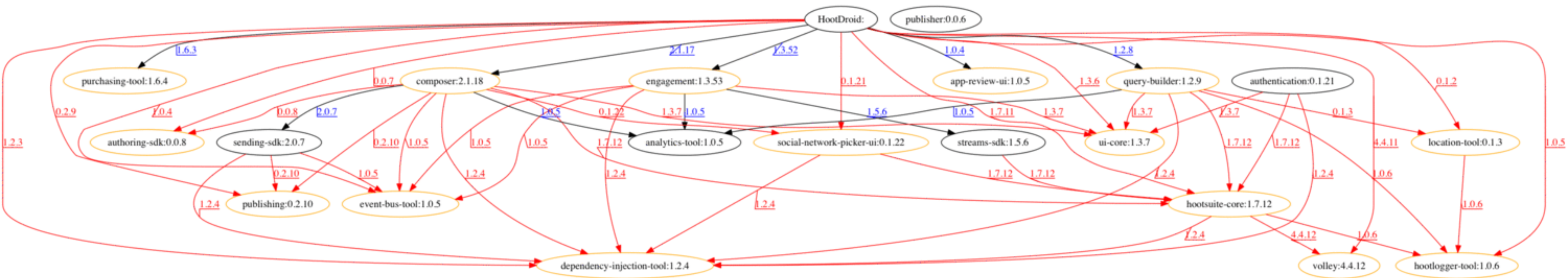
Job's done: Kotlin 1.1.0

- 1 - com.hootsuite.android:event-bus-tool:1.0.3 - <https://github.hootops.com/Android/event-bus-tool/pull/10>
- 2 - com.hootsuite.android:dependency-injection-tool:1.2.3 - <https://github.hootops.com/Android/dependency-injection-tool/pull/19>
- 3 - com.hootsuite.android:legacy:volley:4.4.11 - <https://github.hootops.com/Android/Volley/pull/13>
- 4 - com.hootsuite.android:hootlogger-tool:1.0.5 - <https://github.hootops.com/Android/hootlogger-tool/pull/13>
- 5 - com.hootsuite.android:hootsuite-core:1.7.9 - <https://github.hootops.com/Android/hootsuite-core/pull/59>
- 6 - com.hootsuite.android:streams-sdk:1.5.3 - <https://github.hootops.com/Android/streams-sdk/pull/141>
- 7 - com.hootsuite.android:ui-core:1.3.5 - <https://github.hootops.com/Android/ui-core/pull/96>
- 8 - com.hootsuite.android:analytics-tool:1.0.3 - <https://github.hootops.com/Android/analytics-tool/pull/10>
- 9 - com.hootsuite.android:engagement:1.3.50 - <https://github.hootops.com/Android/engagement/pull/198>
- 10 - com.hootsuite.android:social-network-picker-ui:0.1.19 - <https://github.hootops.com/Android/social-network-picker-ui/pull/49>
- 11 - com.hootsuite.android:publishing:0.2.7 - <https://github.hootops.com/Android/publishing/pull/18>
- 12 - com.hootsuite.android:sending-sdk:2.0.4 - <https://github.hootops.com/Android/sending-sdk/pull/56>
- 13 - com.hootsuite.android:authoring-sdk:0.0.6 - <https://github.hootops.com/Android/authoring-sdk/pull/8>
- 14 - com.hootsuite.android:composer:2.1.11 - <https://github.hootops.com/Android/compose/pull/222>
- 15 - com.hootsuite.android:purchasing-tool:1.6.3 - <https://github.hootops.com/Android/purchasing-tool/pull/16>
- 16 - com.hootsuite.android:app-review-ui:1.0.3 - <https://github.hootops.com/Android/app-review-ui/pull/9>
- 17 - com.hootsuite.android:location-tool:0.1.1 - <https://github.hootops.com/Android/location-tool/pull/5>
- 18 - com.hootsuite.android:query-builder:1.2.5 - <https://github.hootops.com/Android/query-builder/pull/204>
- 19 - Root Level Change - <https://github.hootops.com/Android/HootDroid/pull/1239>
- 20 - com.hootsuite.android:authentication:0.1.18 - <https://github.hootops.com/Android/authentication/pull/30>
- 21 - com.hootsuite.android:publisher:0.0.4 - <https://github.hootops.com/Android/publisher/pull/4>

Minor Version Changes



Major Graph Changes

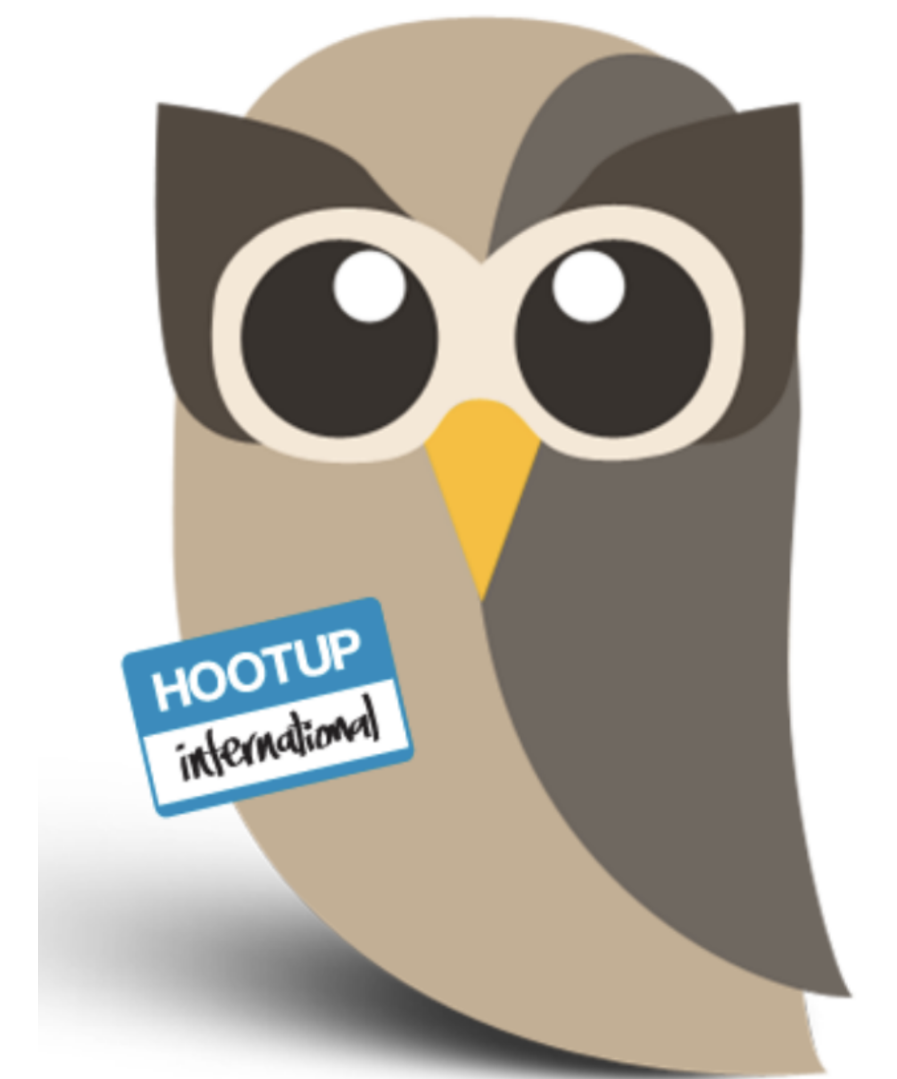


Community Contribution




Helping to Spread Kotlin

- We wanted to grow the Kotlin Culture beyond Hootsuite
- It started with the Kotlin 1.1 Event
 - Only Official Event in Canada
 - Founded the Vancouver Kotlin Meetup Group for the Event.
 - Engage with JetBrains.



It Begins



Kotlin 1.1 Event

Thank you for submitting your event!

We will contact you as soon as possible but not later than in 72 hours.
For more details, do not hesitate to contact Alina at alina@jetbrains.com or on Twitter @meilalina.

Never submit passwords through Google Forms.

This form was created inside of Dev.by. [Report Abuse](#) - [Terms of Service](#) - [Additional Terms](#)



Getting Ready for the 1.1 Event



Kotlin 1.1

- Kotlin 1.1 Event
 - Talks by Hootsuite Kotlin Developers.
 - Engage with Kotlin Community.
 - First Meetup in Vancouver.



Kotlin Meetup Group

- Kotlin Meetup Group
 - Help grow Kotlin in Vancouver.
- Kotlin Night #1
 - Sponsored by JetBrains
 - Talks by Hootsuite and external Kotlin developers.
- Some people here today!
 - Thanks JetBrains for the tickets!



Kotlin Meetup Group

- Kotlin Beginners Night
 - Often requested by people new to Kotlin.
 - Introductory talks.
 - Workshops on language features.



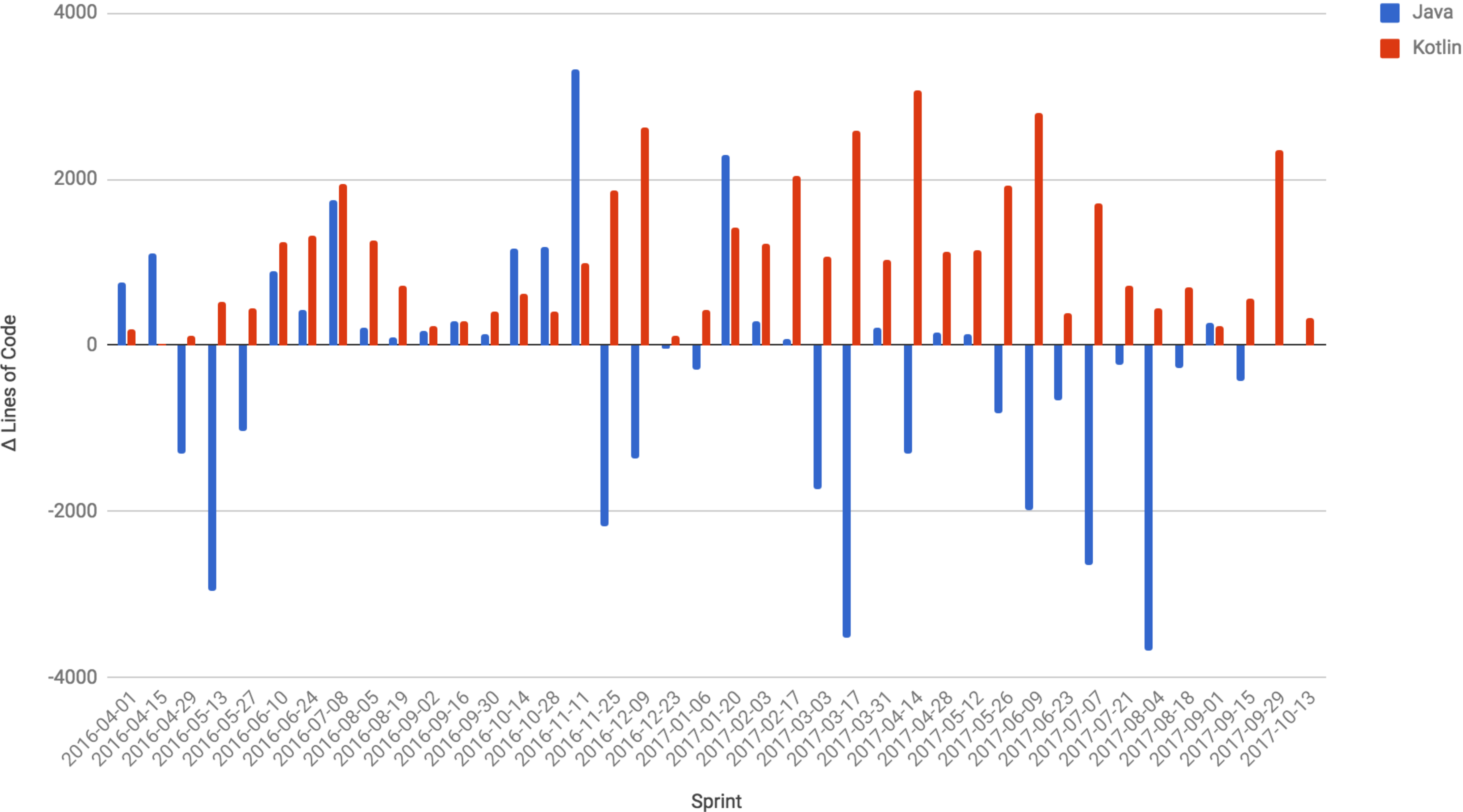
History of Kotlin and Java at Hootsuite



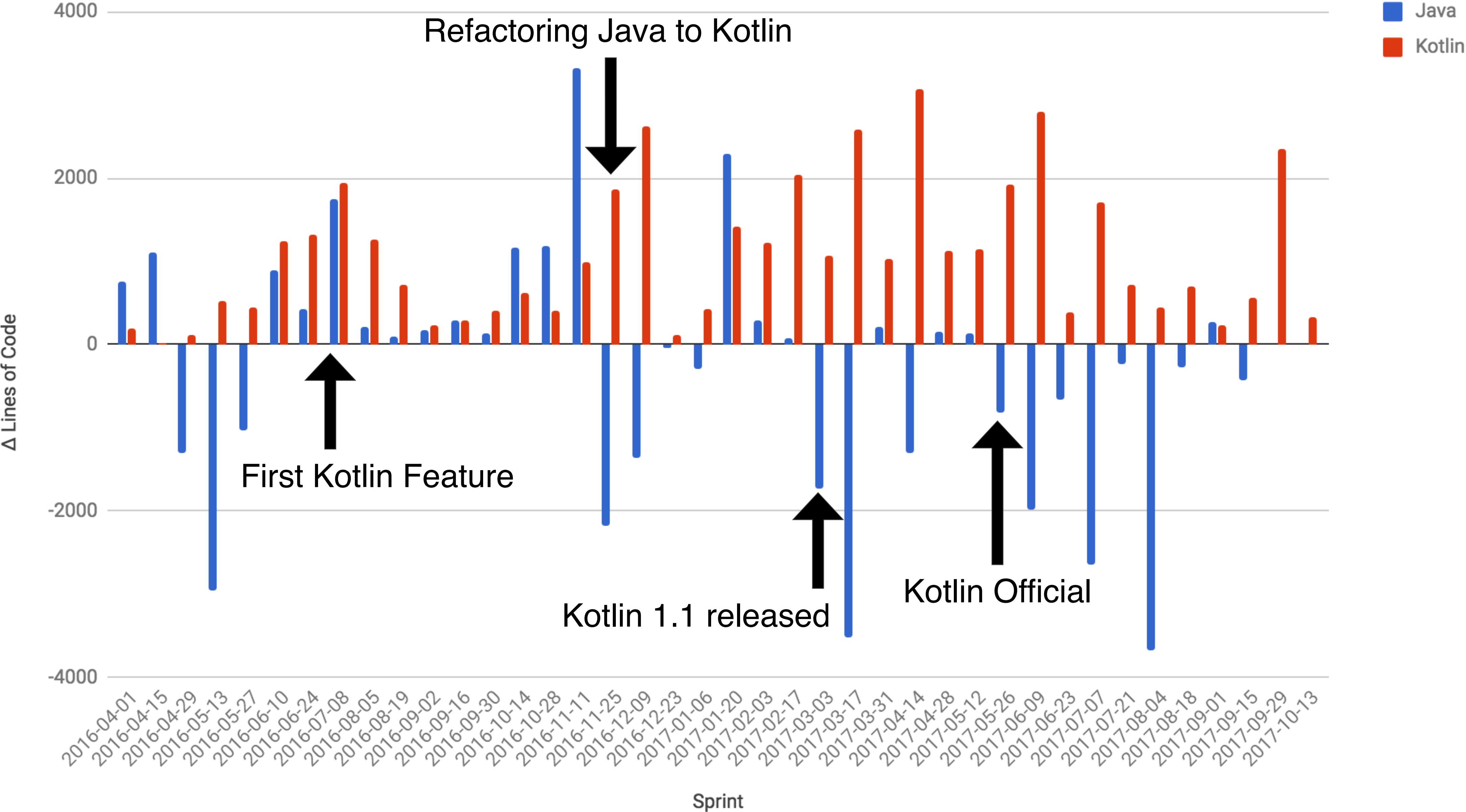
Per Sprint Contributions



Change in Lines of Code by Language / Sprint



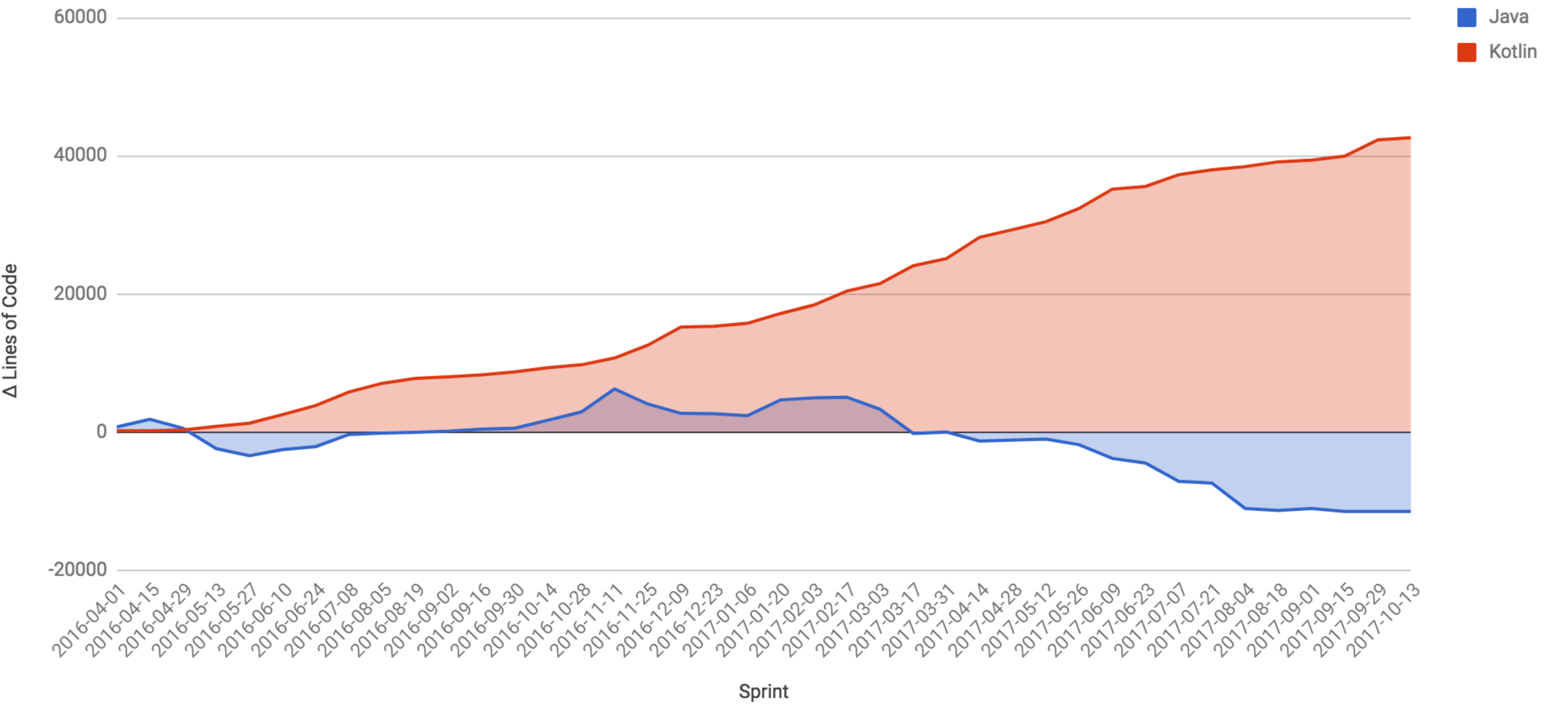
Change in Lines of Code by Language / Sprint



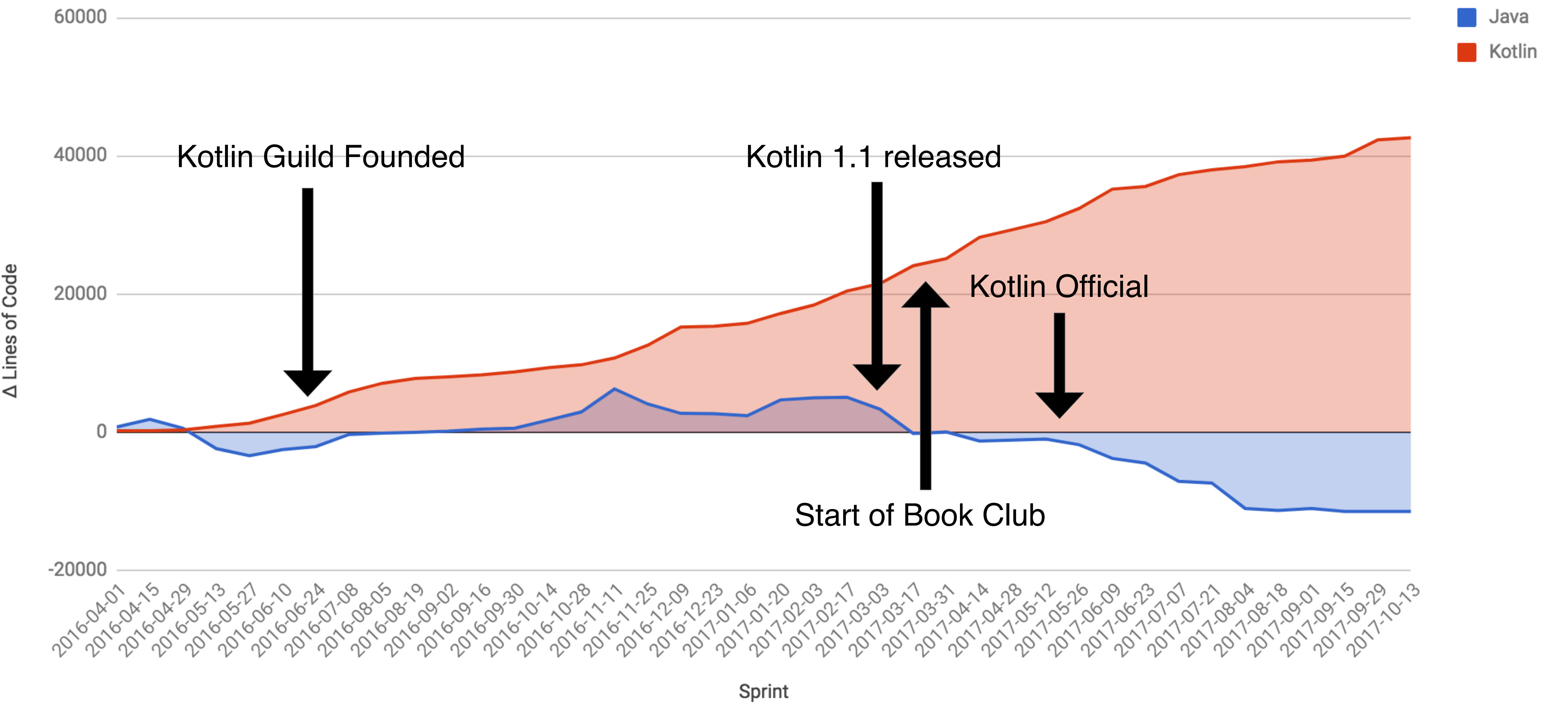
Cumulative Lines of Code



Cumulative Change in Lines of Code by Language



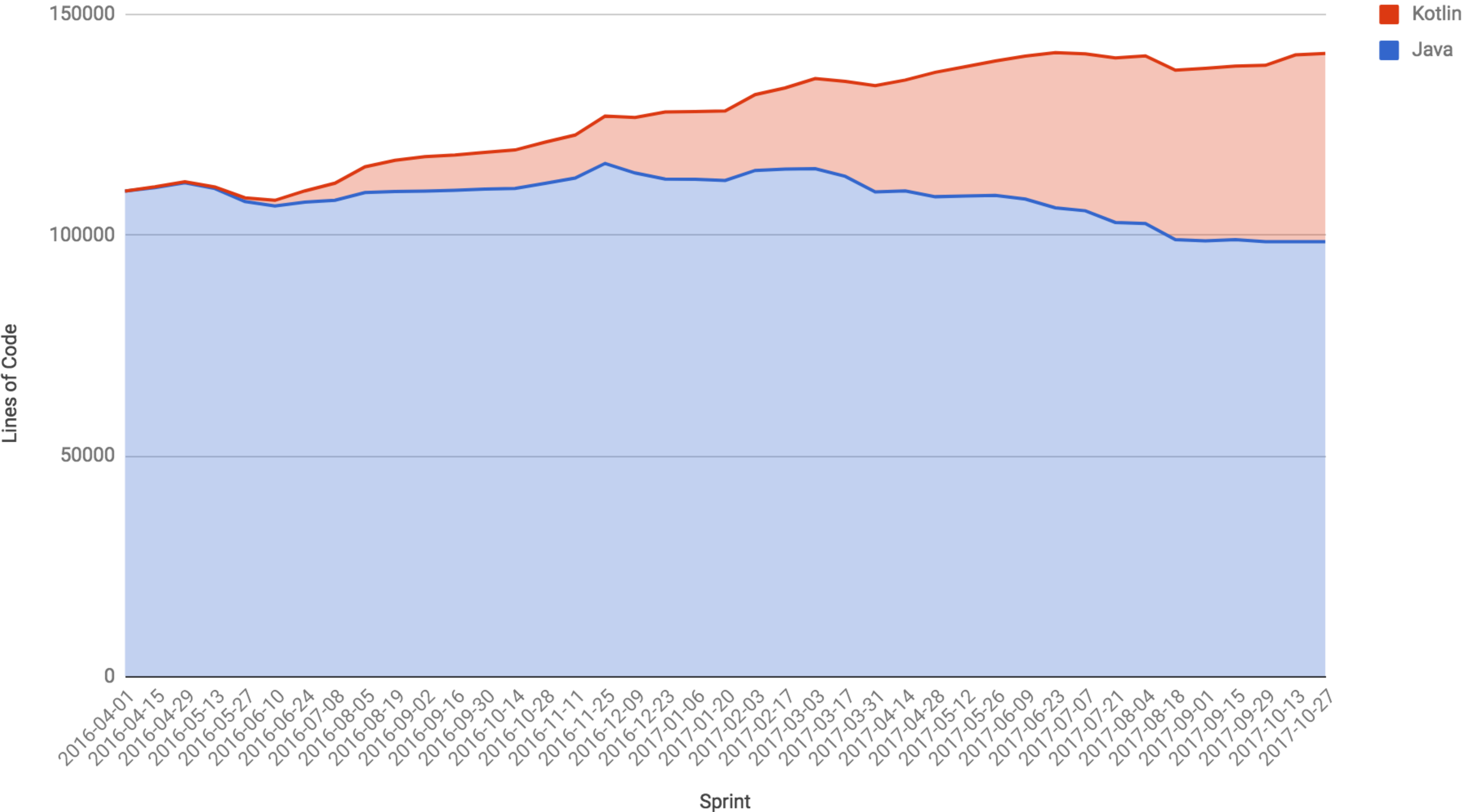
Cumulative Change in Lines of Code by Language



Overall Lines of Code



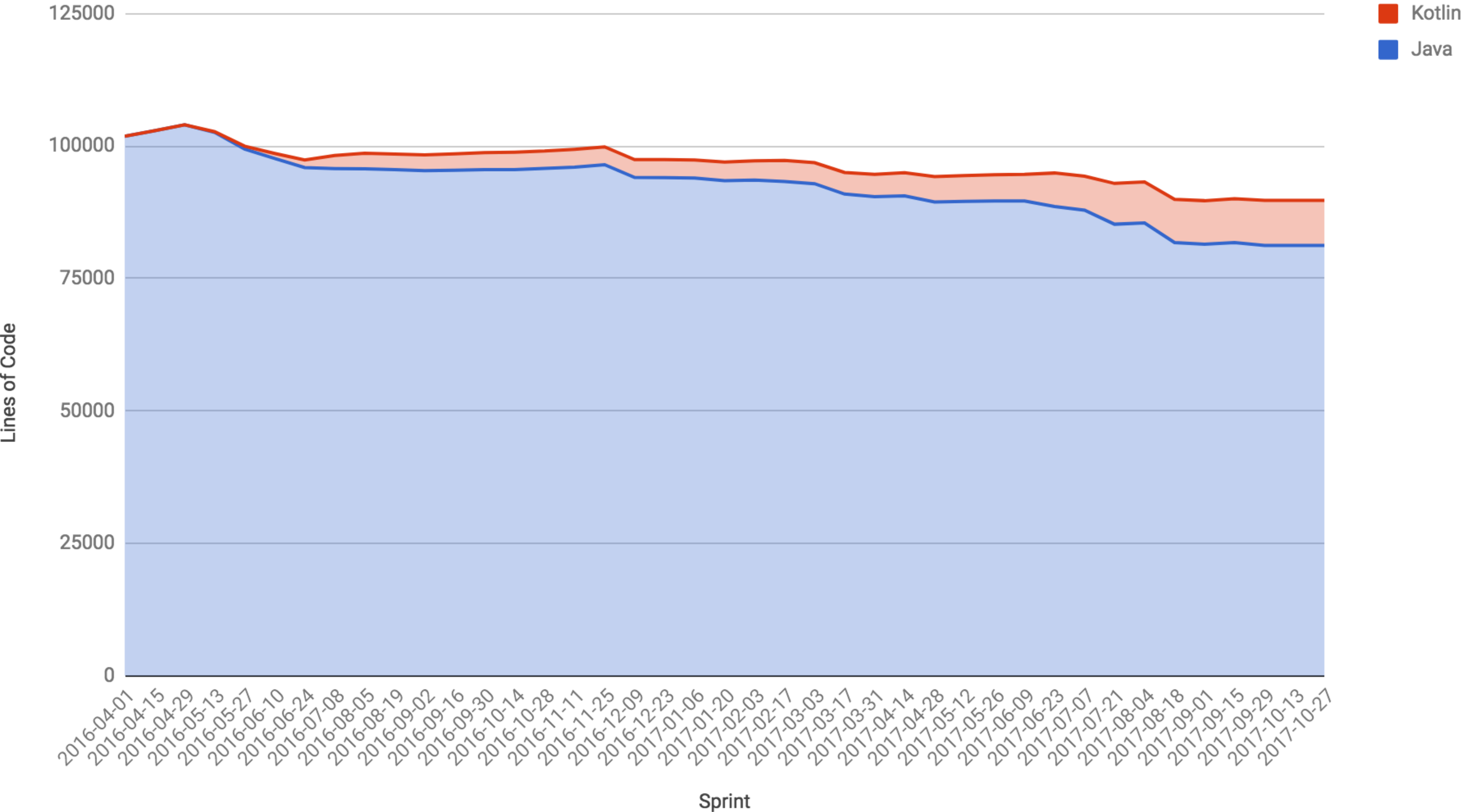
Hootsuite Android Code Base by Language



Main App



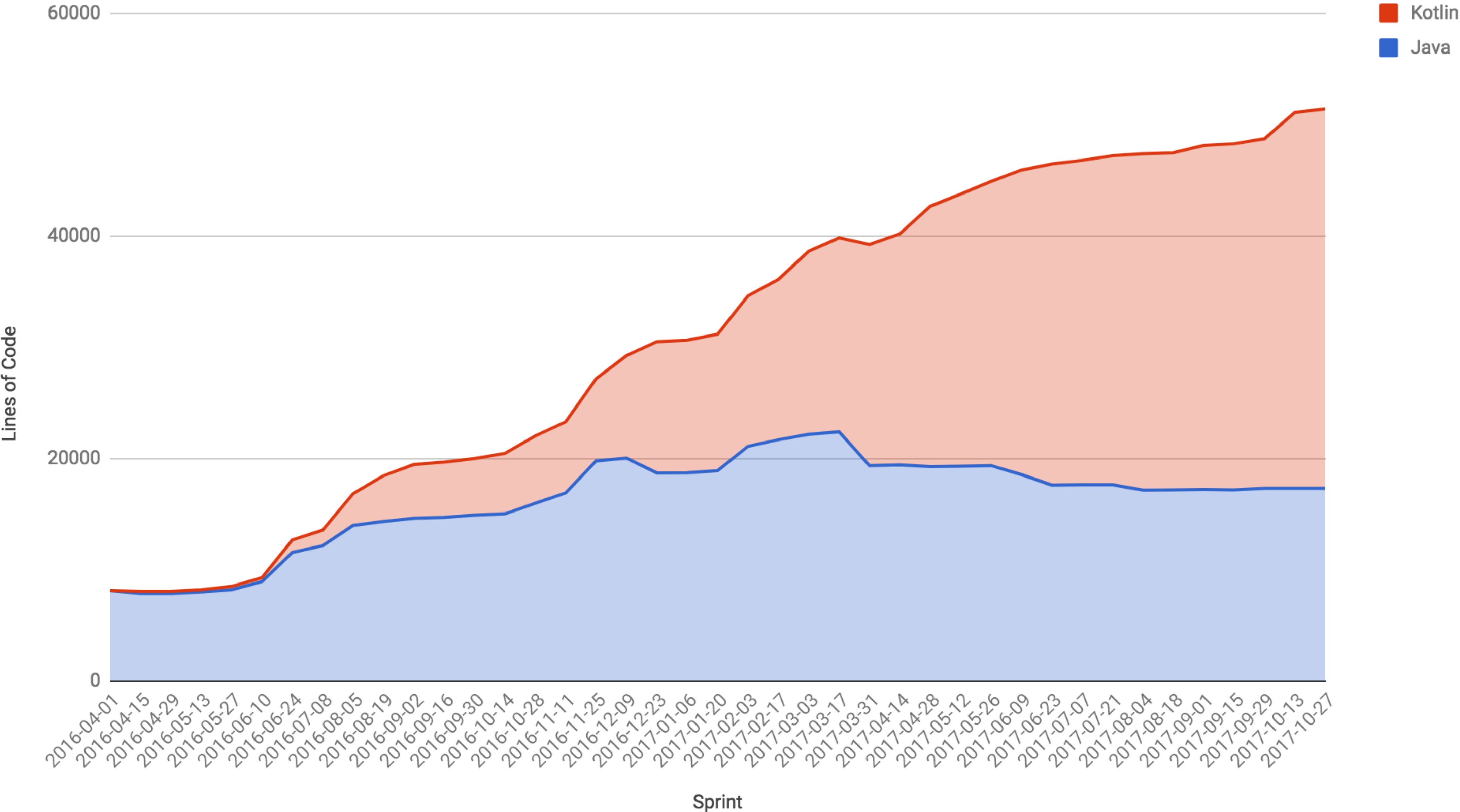
Main App Composition by Language



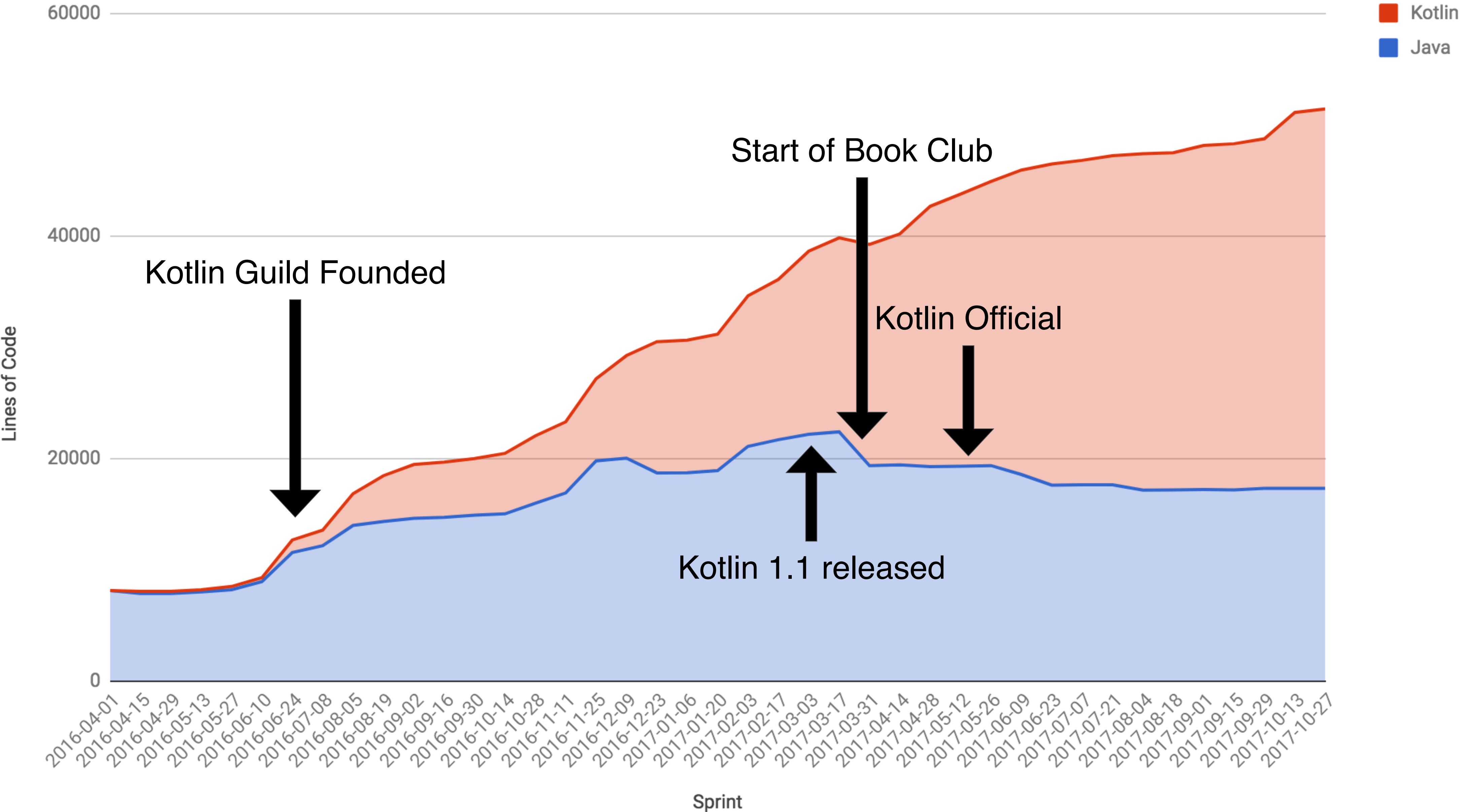
Libraries



Composition of Libraries by Language



Composition of Libraries by Language



2016 Milestones



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6	0	0	0



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6	0	0	0
2016-04-15	6	6	0	0	1



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6	0	0	0
2016-04-15	6	6	0	0	1
2016-05-27	9	9	0	0	2



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6	0	0	0
2016-04-15	6	6	0	0	1
2016-05-27	9	9	0	0	2
2016-06-10	11	11	0	0	5



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6	0	0	0
2016-04-15	6	6	0	0	1
2016-05-27	9	9	0	0	2
2016-06-10	11	11	0	0	5
2016-06-24	12	11	1	1	5



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6	0	0	0
2016-04-15	6	6	0	0	1
2016-05-27	9	9	0	0	2
2016-06-10	11	11	0	0	5
2016-06-24	12	11	1	1	5
2016-07-22	13	10	2	3	6



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6	0	0	0
2016-04-15	6	6	0	0	1
2016-05-27	9	9	0	0	2
2016-06-10	11	11	0	0	5
2016-06-24	12	11	1	1	5
2016-07-22	13	10	2	3	6
2016-08-05	14	11	1	3	7



For all sad words of tongue and pen, The saddest are these, 'I had to convert Kotlin back to Java'.

- The Hootsuite Android Team



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6		0	0
2016-04-15	6	6		0	1
2016-05-27	9	9		0	2
2016-06-10	11	11		0	5
2016-06-24	12	11	1	1	5
2016-07-22	13	10	2	3	6
2016-08-05	14	11	1	3	7



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6		0	0
2016-04-15	6	6		0	1
2016-05-27	9	9		0	2
2016-06-10	11	11		0	5
2016-06-24	12	11	1	1	5
2016-07-22	13	10	2	3	6
2016-08-05	14	11	1	3	7
2016-09-02	15	10	2	5	8



Change in Libraries over Time

removing apt (#53)

[Browse files](#)

Summary
Removing apt in favor of kapt

Test Plan
Build and Run app

Reviewer

@simon-tse-hs

JIRA Ticket

<https://jira.hootsuite.com/browse/AND->

 master (#53)  2.0.0 ... 1.0.16



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6		0	0
2016-04-15	6	6		0	1
2016-05-27	9	9		0	2
2016-06-10	11	11		0	5
2016-06-24	12	11	1	1	5
2016-07-22	13	10	2	3	6
2016-08-05	14	11	1	3	7
2016-09-02	15	10	2	5	8
2016-10-04	15	10	3	5	8



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-04-01	6	6		0	0
2016-04-15	6	6		0	1
2016-05-27	9	9		0	2
2016-06-10	11	11		0	5
2016-06-24	12	11	1	1	5
2016-07-22	13	10	2	3	6
2016-08-05	14	11	1	3	7
2016-09-02	15	10	2	5	8
2016-10-04	15	10	3	5	8
2016-10-28	16	10	3	6	10

2017



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-10-28	16	10	3	6	10



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-10-28	16	10	3	6	10
2017-03-03	18	12	4	6	12



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-10-28	16	10	3	6	10
2017-03-03	18	12	4	6	12
2017-04-28	20	10	5	10	15



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-10-28	16	10	3	6	10
2017-03-03	18	12	4	6	12
2017-04-28	20	10	5	10	15
2017-06-23	20	6	9	14	18



Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-10-28	16	10	3	6	10
2017-03-03	18	12	4	6	12
2017-04-28	20	10	5	10	15
2017-06-23	20	6	9	14	18
2017-07-21	20	5	10	15	18

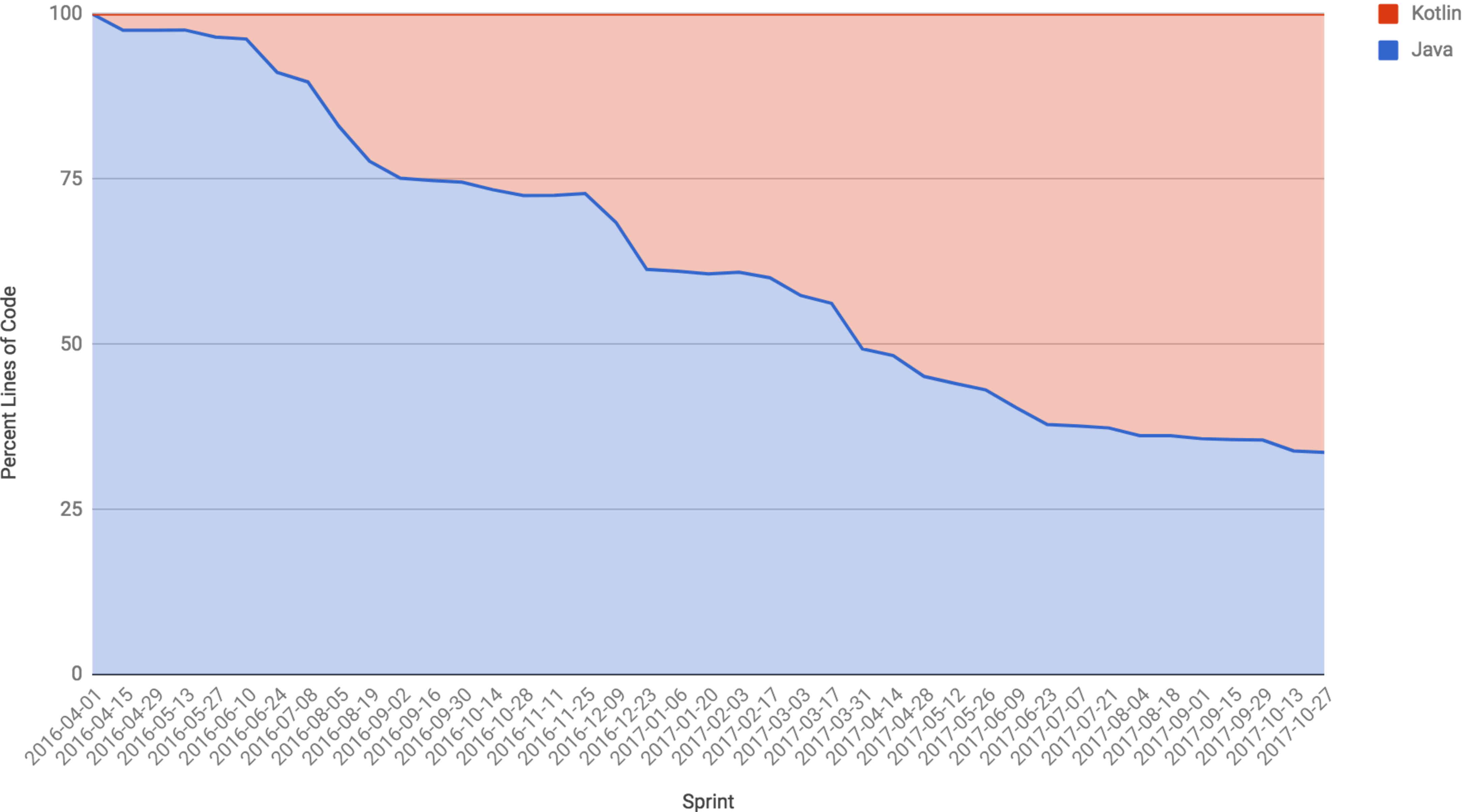


Change in Libraries over Time

Sprint	Libraries	Majority Java	All Kotlin	Majority Kotlin	Partially Kotlin
2016-10-28	16	10	3	6	10
2017-03-03	18	12	4	6	12
2017-04-28	20	10	5	10	15
2017-06-23	20	6	9	14	18
2017-07-21	20	5	10	15	18
2017-10-27	21	5	13	16	18



Android Library Code by Language (Normalized)



Kotlin Official



Hopeful Developers

- Kotlin had been getting a lot of press all year.
- The team at Hootsuite were very hopeful for an announcement about Kotlin.
- We had almost given up hope but...

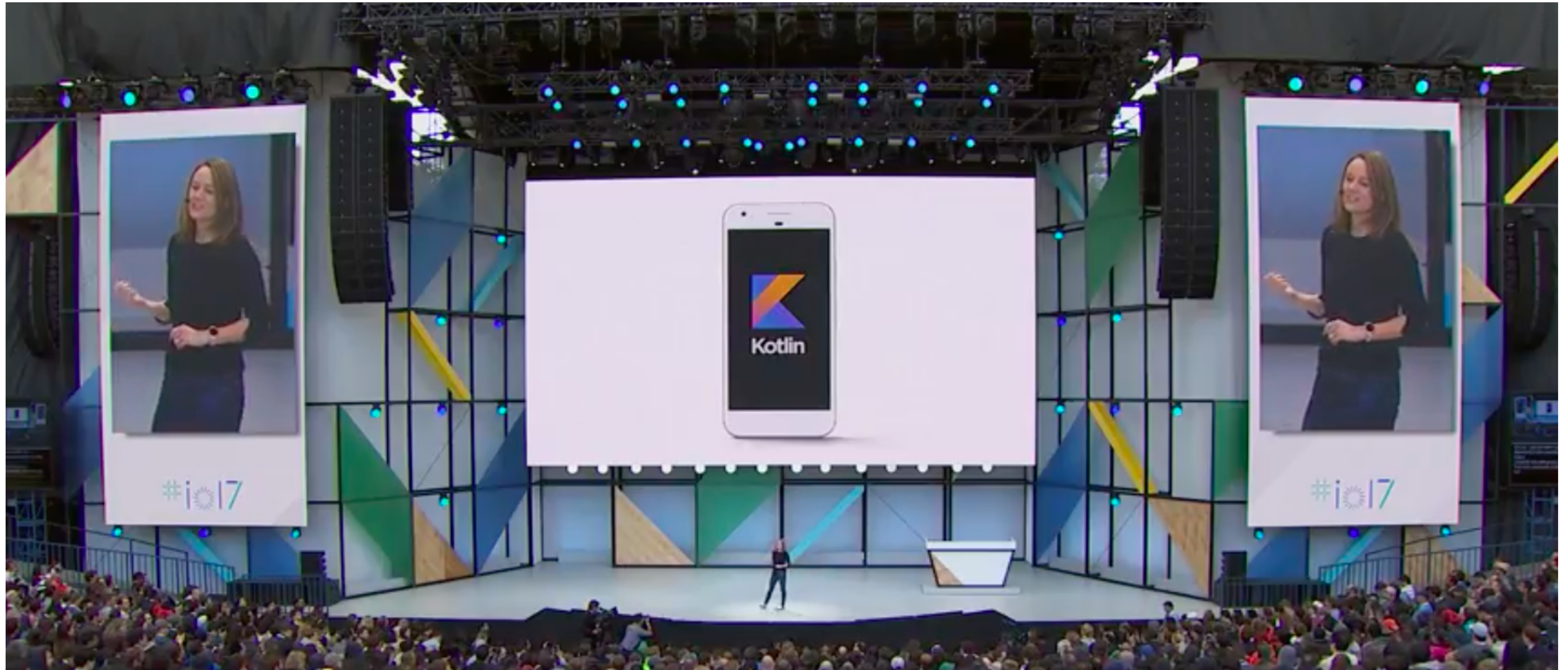


“We have never added a new programming language to Android”

Stephanie Saad Cuthbertson —
director of product management for Android



Production Ready



Risks Review



Mitigated Risks of Kotlin

- **Unofficial Android Language**
 - Now Official!



Mitigated Risks of Kotlin

- Increased build times.
 - Use of precompiled library modules.



Mitigated Risks of Kotlin

- Stability in our CI Pipeline.
 - Some early issues, now resolved.



Mitigated Risks of Kotlin

- IDE Tooling support
 - Android Studio and Kotlin are both JetBrains.



#HootLove



Mitigated Risks of Kotlin

- Hiring Considerations.
 - People have seen our blog and want to work with Kotlin.



Conclusion



Conclusion

- Kotlin has been a boon to productivity



Conclusion

- Kotlin has been a boon to productivity



Conclusion

- Increased Developer Happiness



Conclusion

- Better Code Shipped



Conclusion

- Interoperability key to Adoption
- Incremental addition was the strategy for adoption.



Conclusion

- Library Driven Development
 - Decomposition of our app into Kotlin libraries.



Conclusion

- Grown and Spread Kotlin Culture
 - Kotlin Guild
 - Kotlin Book Club
 - Meetup Group



Thank you!

Neil Power

Software Developer

@NeilPower



Owly Plush Questions

```
println(3.let { 12 } + 6.run { 9 } +  
9.apply { 6 } + 12.also { 3 })
```

```
println(null  
  .toString()  
  .filter { it != 'l' }  
  .map { if (it == 'n') 'm' else it }  
  .joinToString(separator = ""))
```



Thank you!



Neil Power
[@NeilPower](#)

#kotlinconf17

