Mobile Developer Productivity at Uber Scale

Ty Smith
Android Platform Tech Lead

Uber

@tsmith

Agenda

01 Mobile at Uber Overview
02 Development Stack and Workflow
03 Measuring Devx
04 Making Builds Faster
05 IDE and Devtools

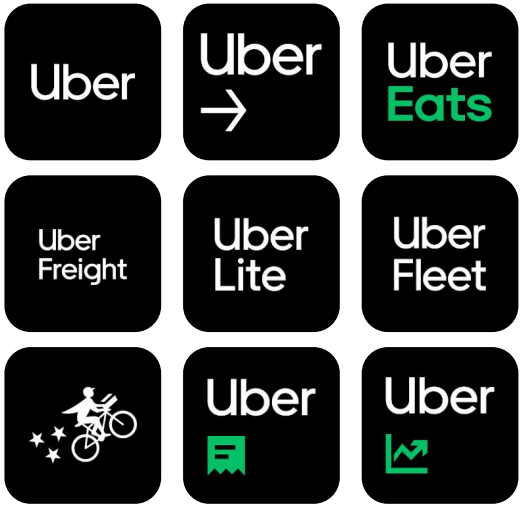
Mobile Scale

800+
Mobile Engineers

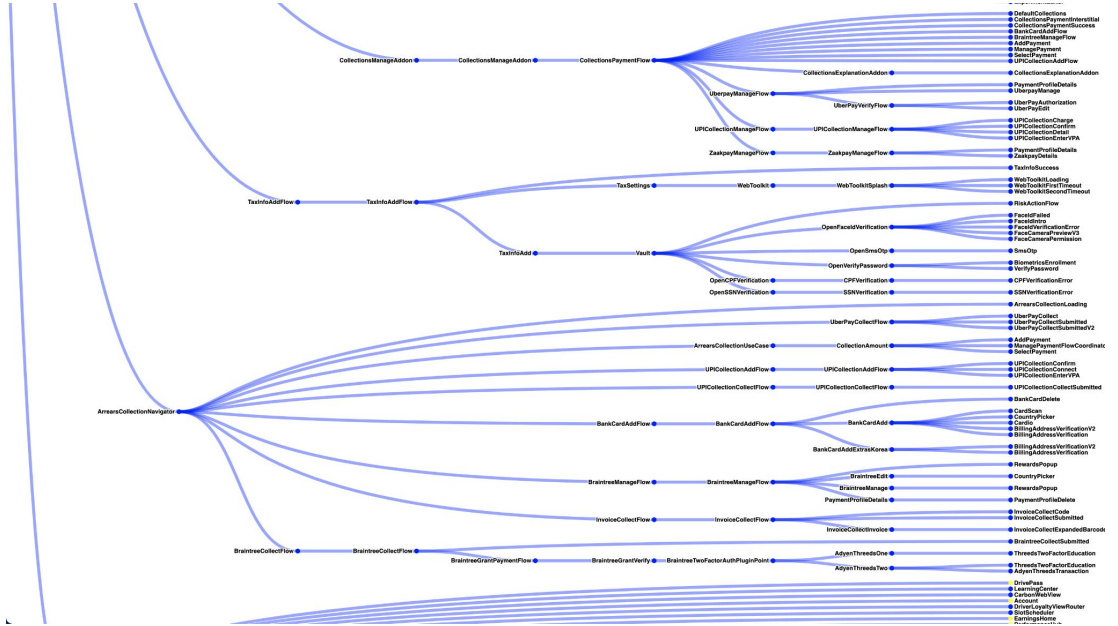
Tens of Thousands
Build Modules

Tens of Millions
Mobile LOC

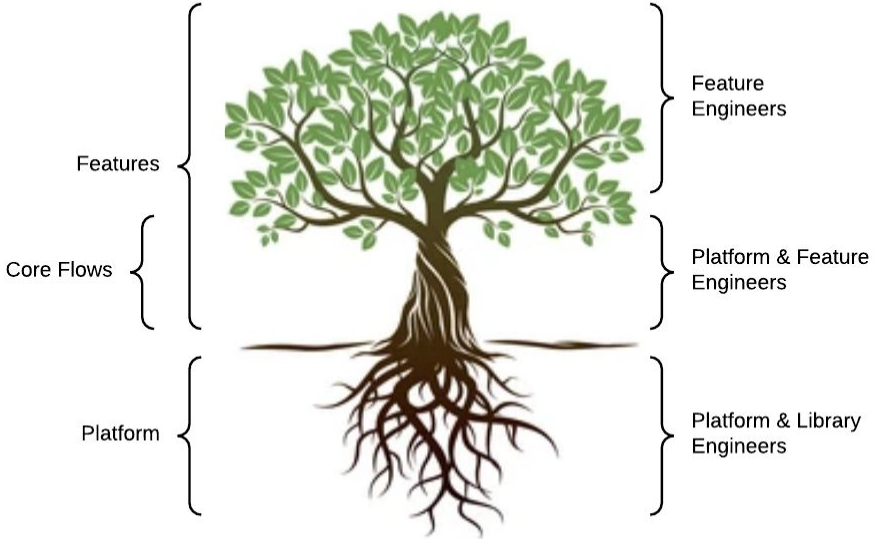
Thousands
RIBs - Our Mobile Architecture



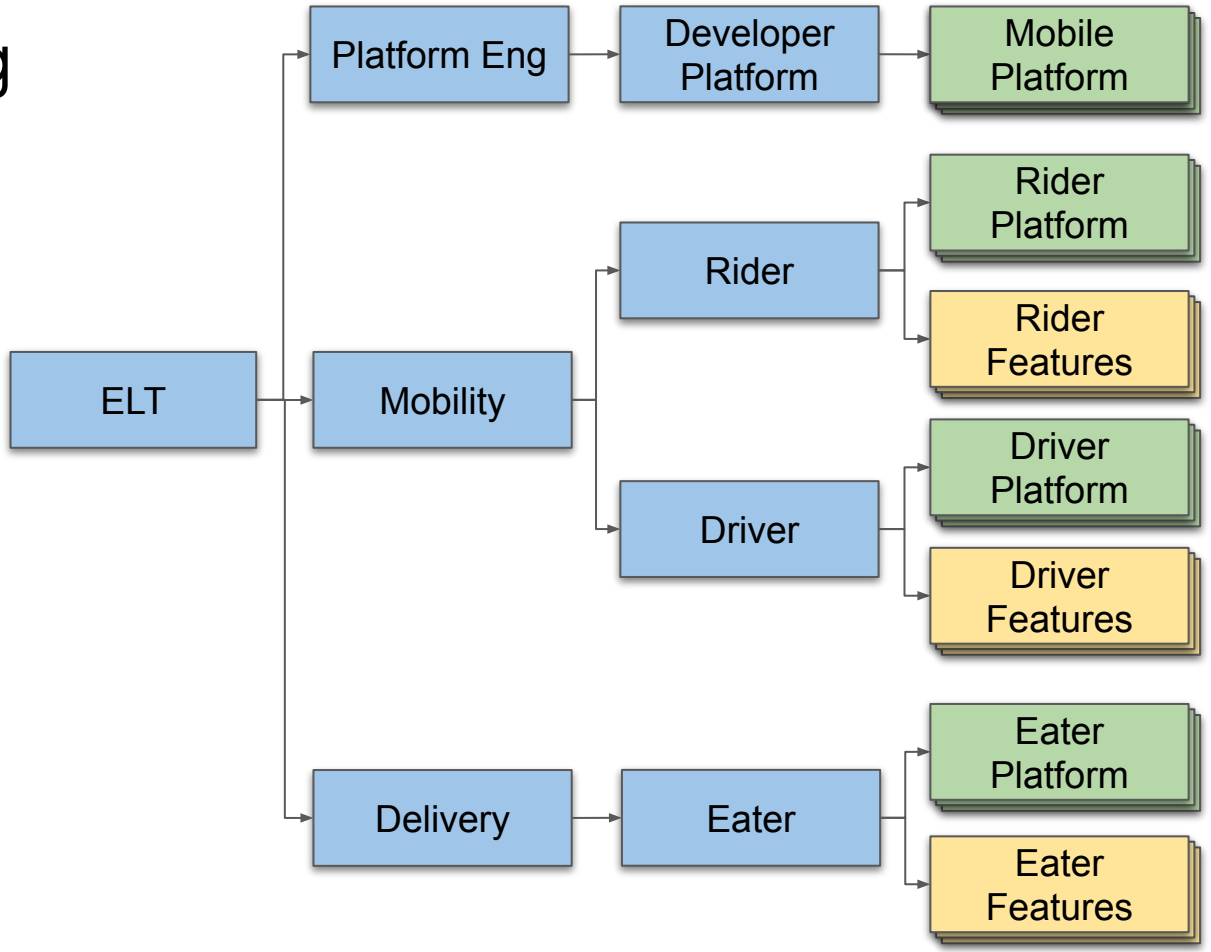
And 100's of internal apps



Mobile Engineering Teams at Uber

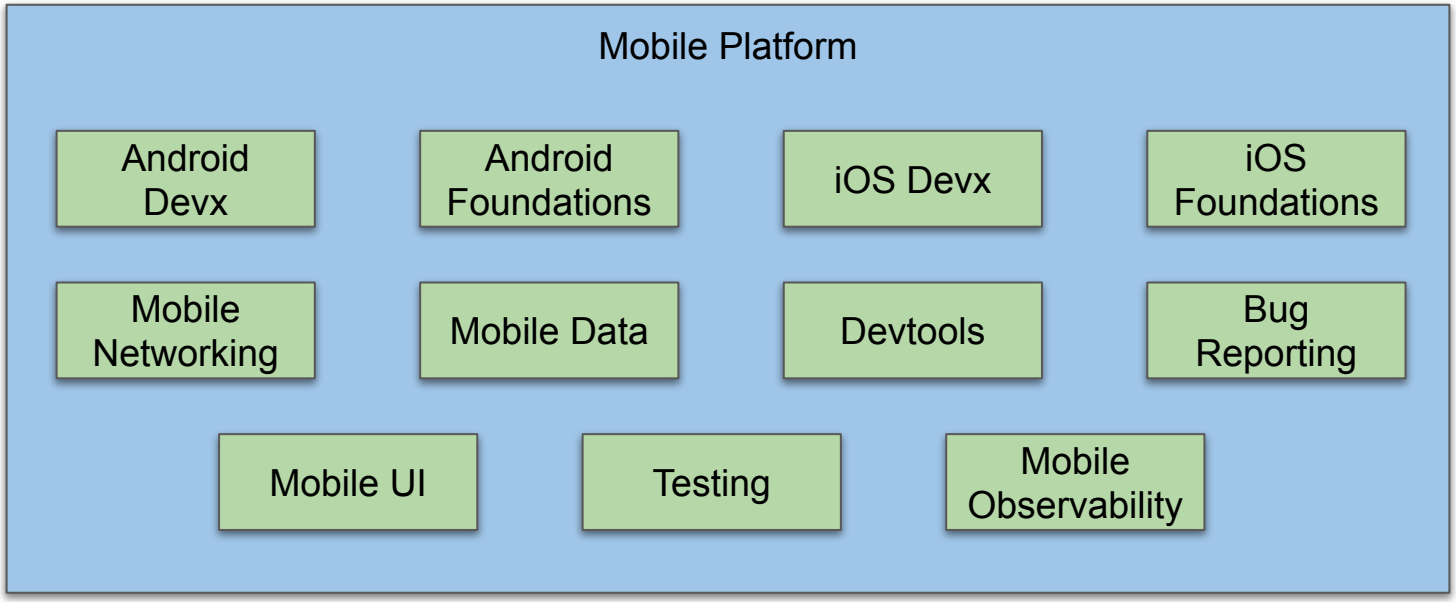


Mobile Org



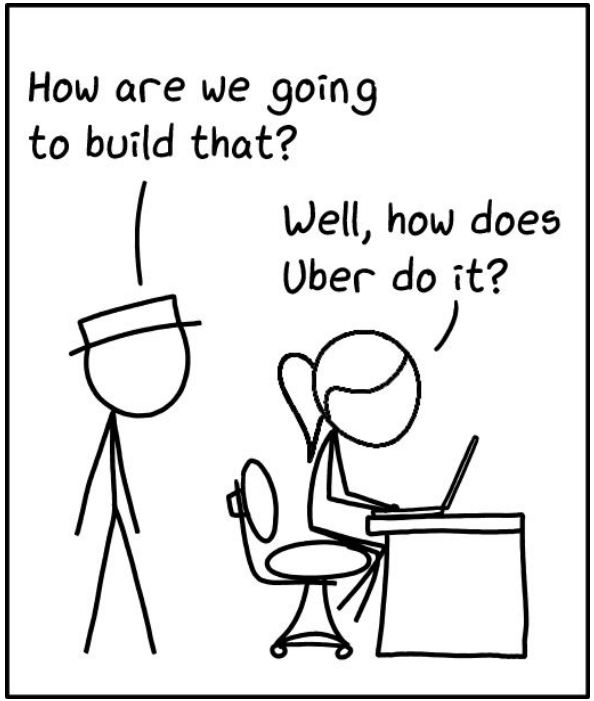
*Org structure simplified

Mobile Platform Teams



Mobile Platform Vision

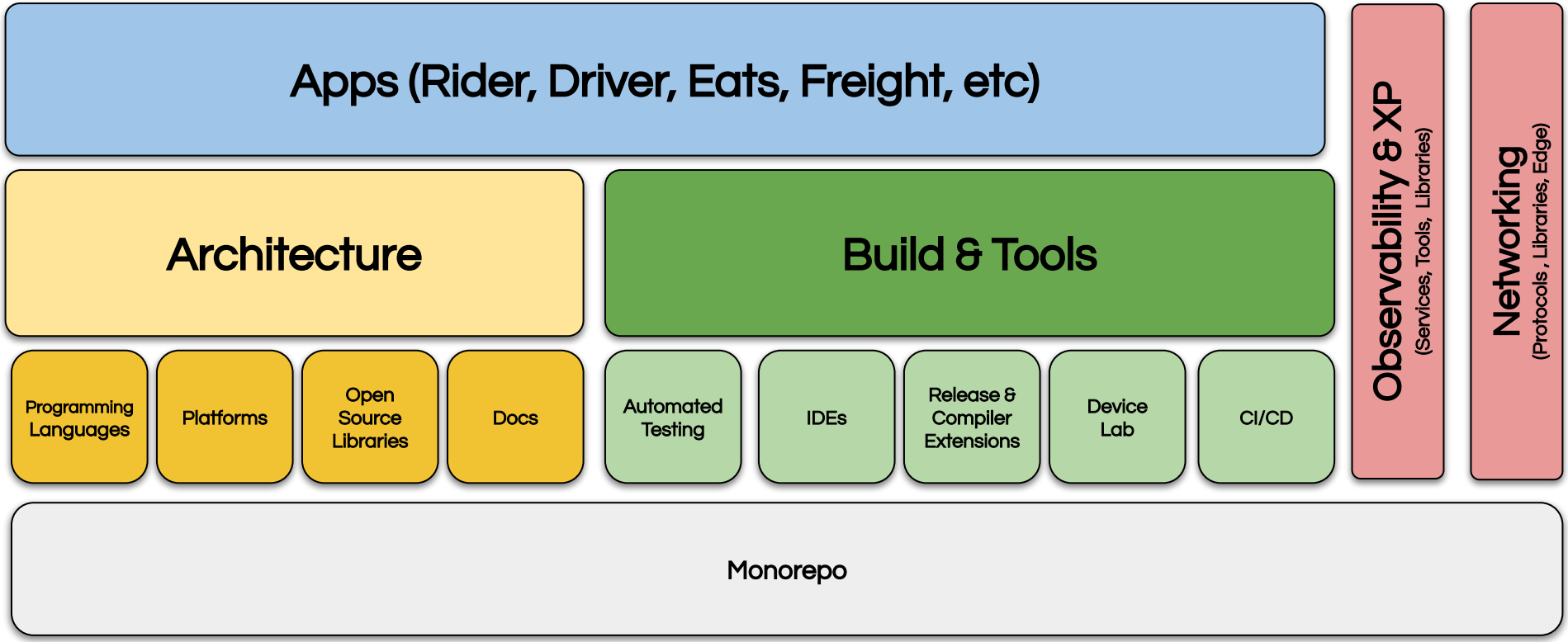
Be the industry leader
for how developers **build,**
deploy, and manage high-quality
software productively
and at scale.



Development Stack and Workflow

Uber

Mobile Tech Stack



Android Tech Stack

Architecture

- RIBS - Converged Architecture
- Kotlin, Compose, Coroutines
- Motif - Code generated DI

Frameworks

- Citrus - Parameterized XP and Plugins
- Simple-store - Async KVS Storage

UI

- Stylist/Artist - View & Style generators
- Base Mobile - Compose Based UI Components
- Base SDUI - Server Driven UI Rendering

Data and Observability

- Unified Reporter - Analytics pipeline
- Jenga - GRPC Model generation pipeline
- Healthline - Crash & Observability
- Blackswan - Crash Recovery

Build

- Buck/Bazel - Hermetic Build system*
- Kaptish - Fast Kotlin Annotation Processing*
- Buildkite - Sharded containerized CI/mergequeue

IDE and Tools

- Intellij Plugins*
- Flipper + Custom Plugins*
- Devpods - Cloud IDEs*
- QuickUI - UI Hotloading*
- LDA - Local Developer Analytics*
- Mobile Studio - Plugin based debug drawer

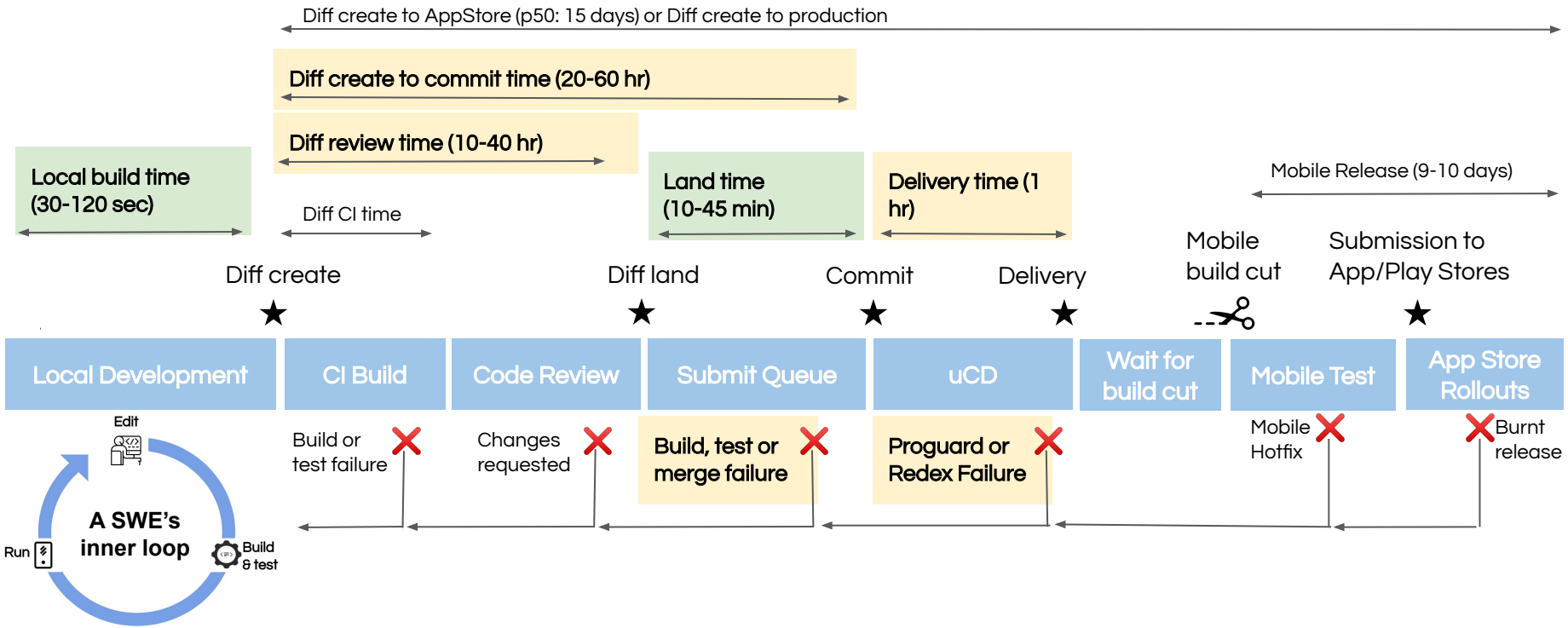
Reliability

- Detekt/Errorprone/Lint/Shellcheck - Static Analysis
- Ktfmt/GJF - Code Formatter
- Piranha - automated XP code cleanup
- Nanoscope - Hyper performant profiler

Education

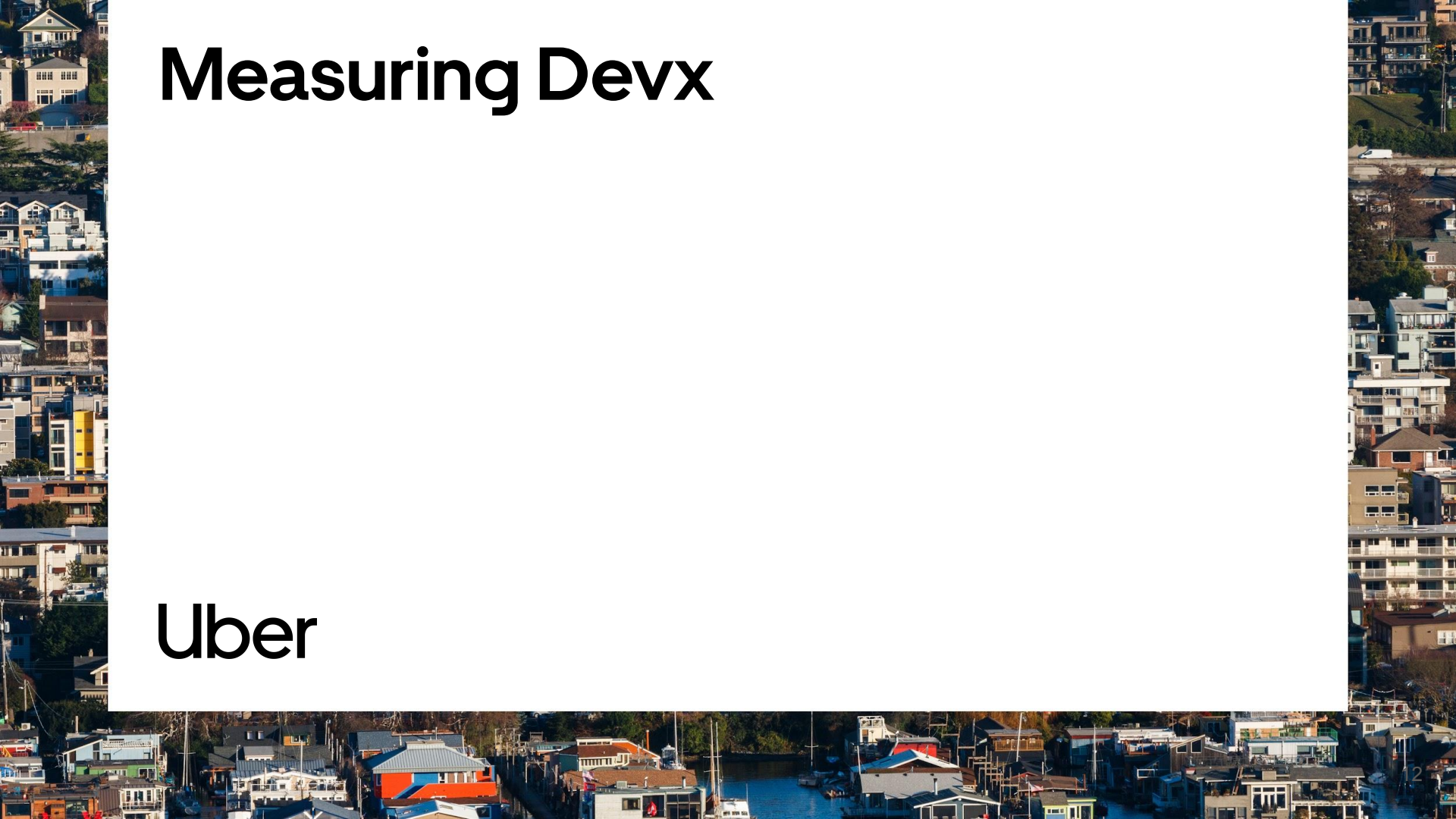
- Centralized documentation, Codelabs, developer onboarding, and support

Mobile Code Development



Measuring Devx

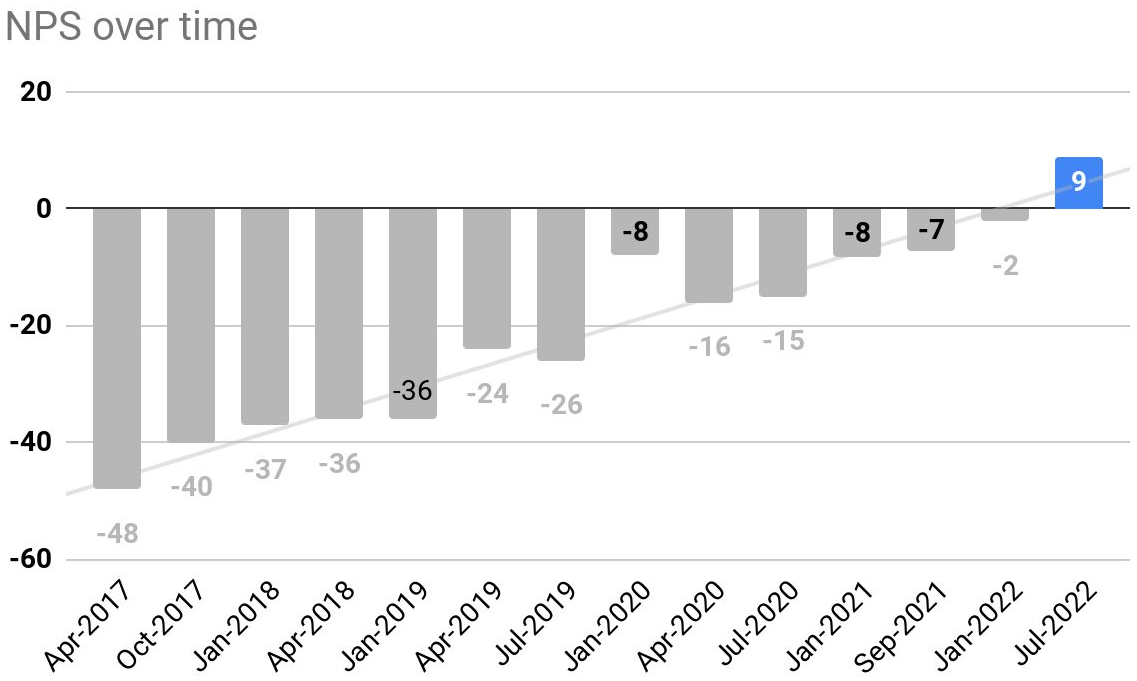
Uber



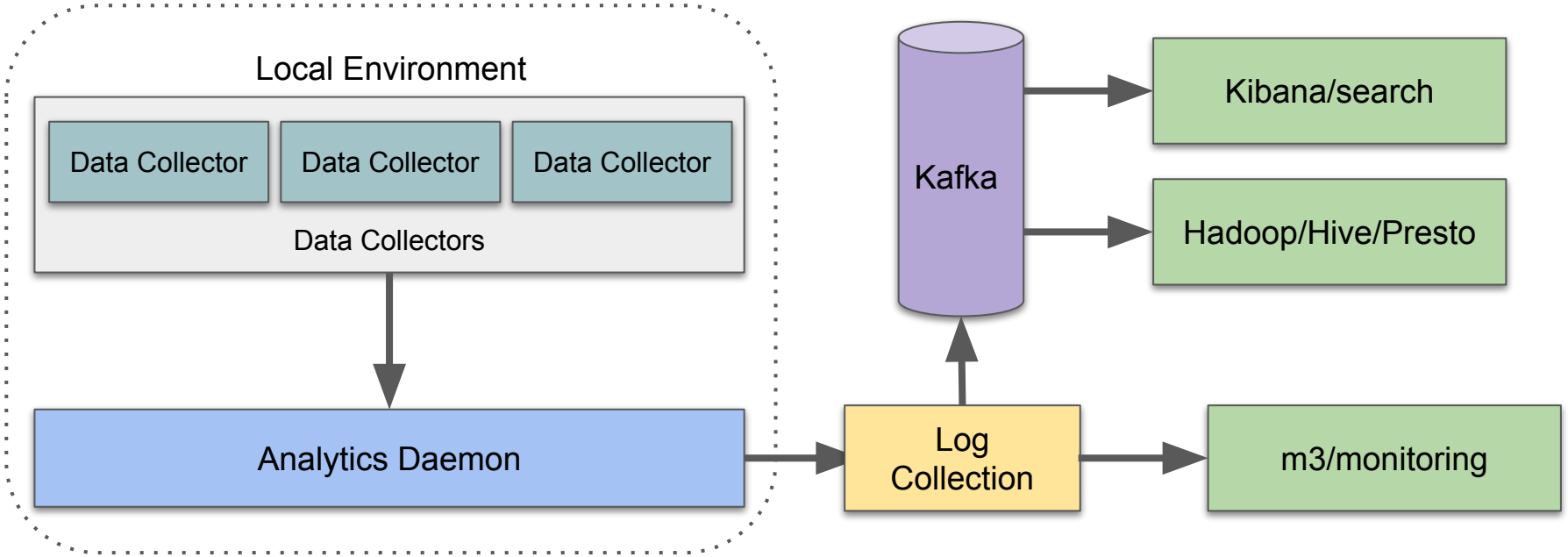
What We Measure

- Net Promoter Score (NPS)
- Developer Throughput
- Build Time
- Failure Rate
- Git time
- Release Funnel
- IDE Performance
- CI/CD Time
- Tooling uptime
- App Performance
- App Reliability

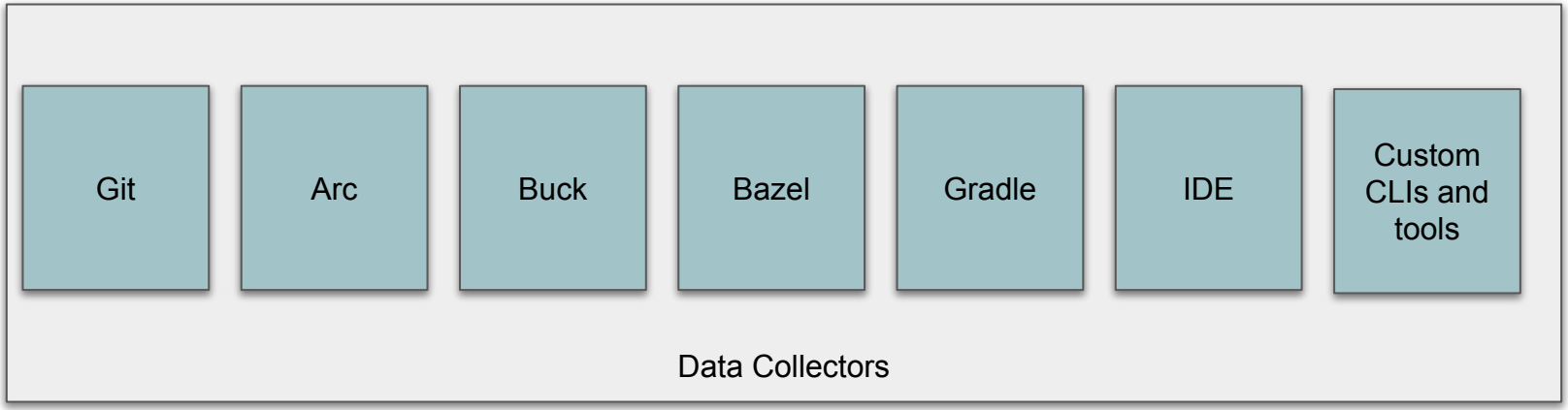
Developer Satisfaction (NPS)



Local Developer Analytics (LDA)



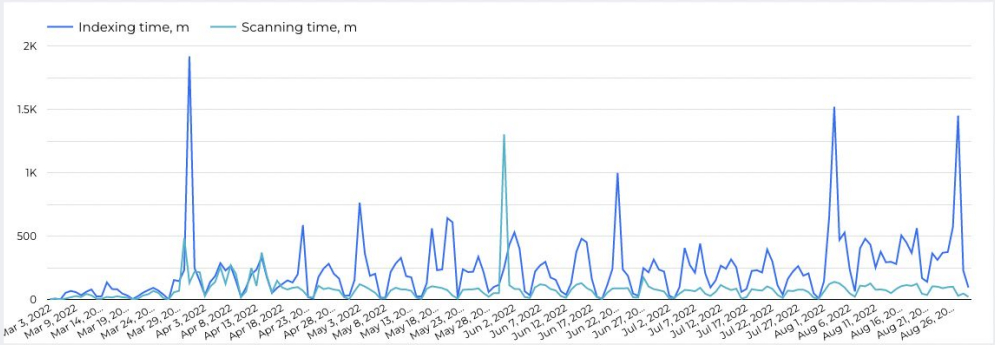
Local Developer Analytics

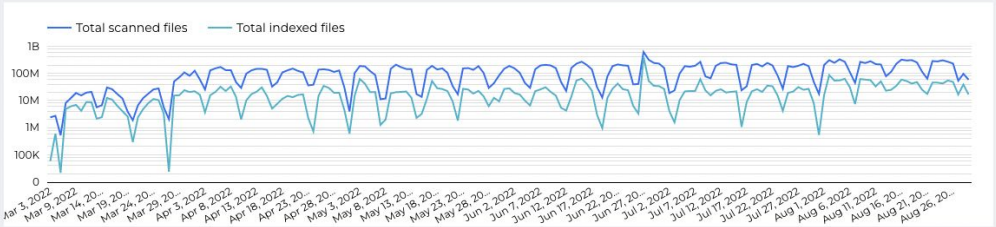


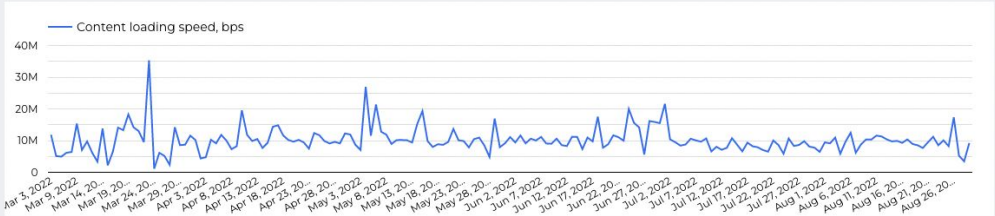
Repo: MOANDROIDJ (1)

Source

Select date range

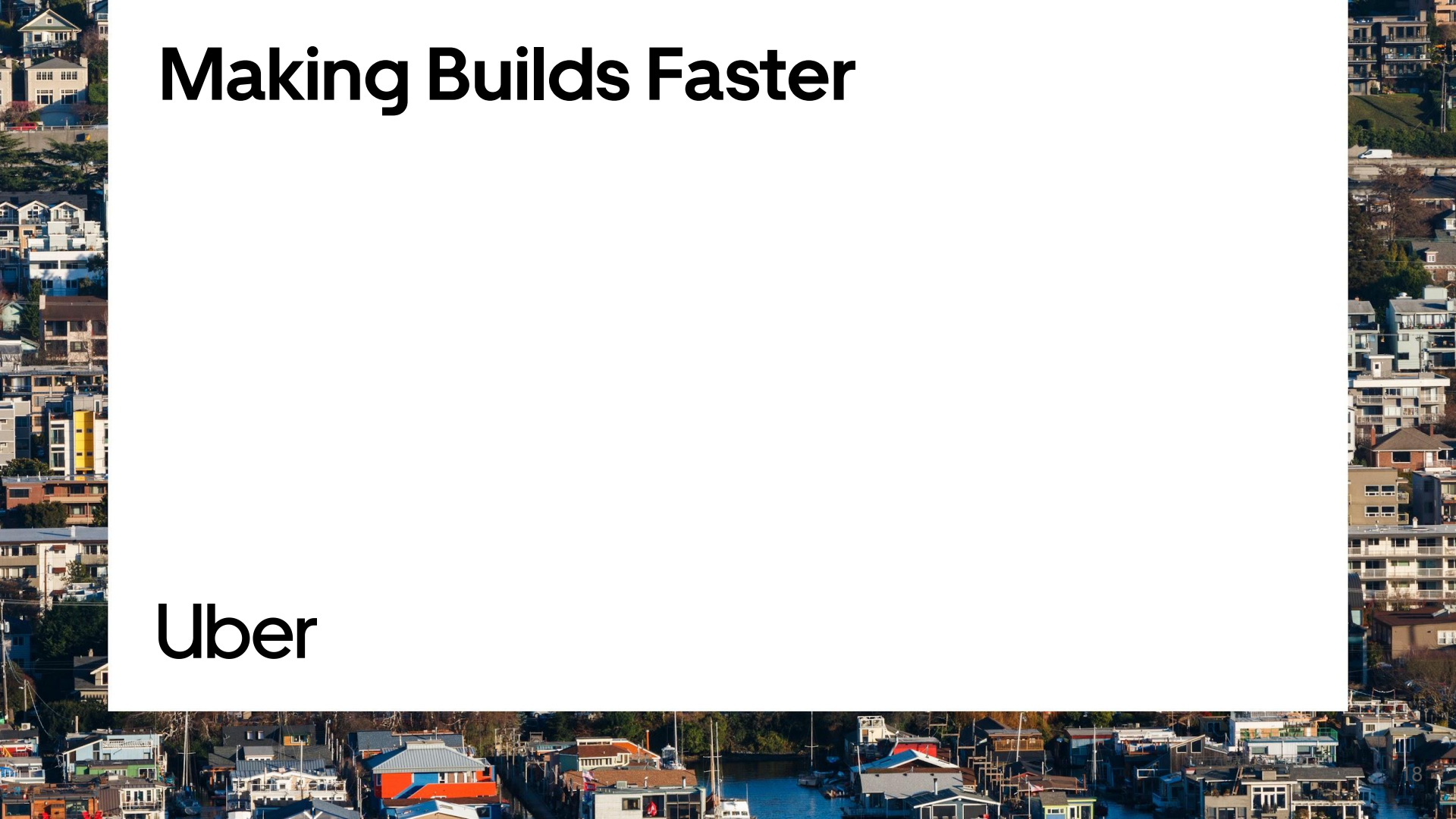






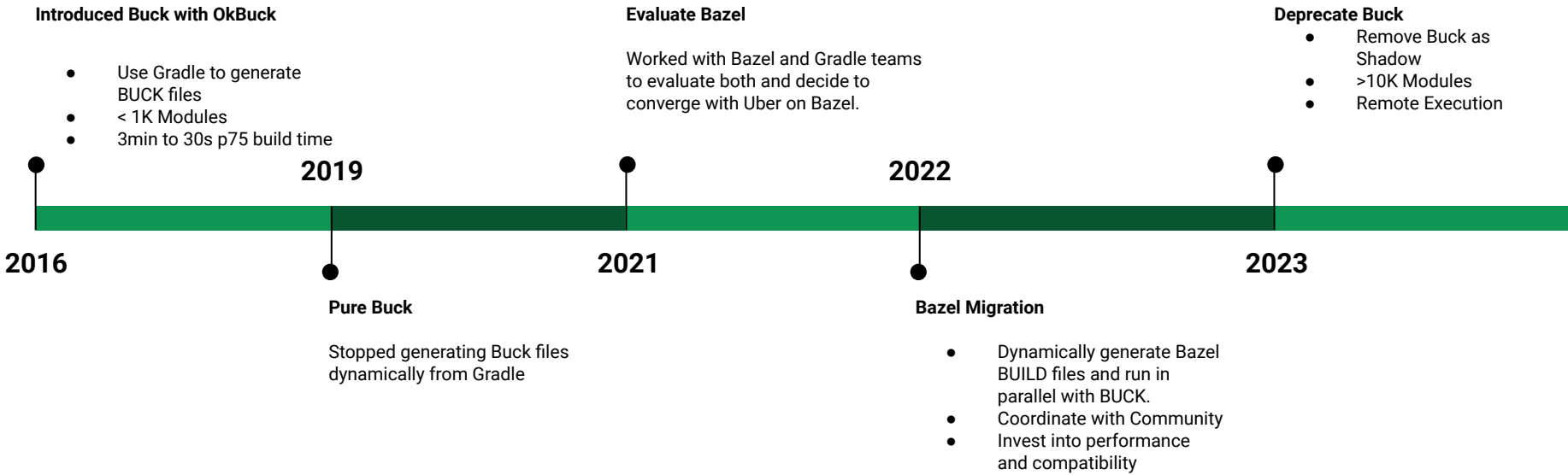
Data Last Updated: 8/29/2022 4:05:34 PM

Making Builds Faster



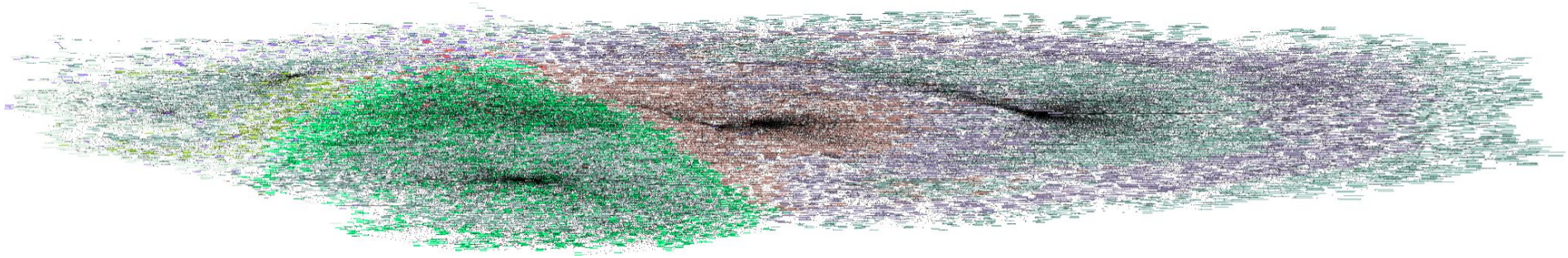
Uber

Uber's Android Build Systems



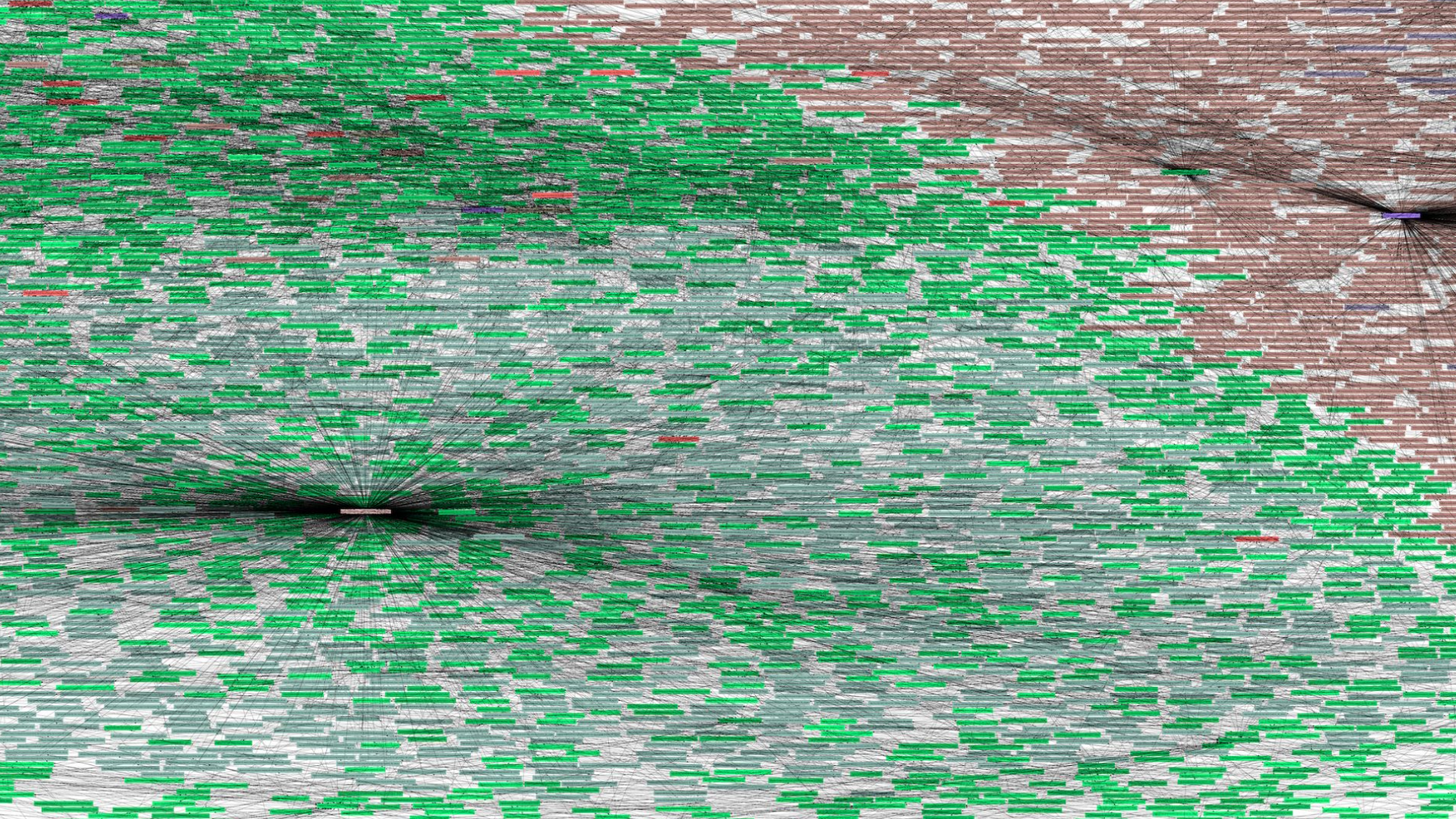
Mobile vs Backend Builds

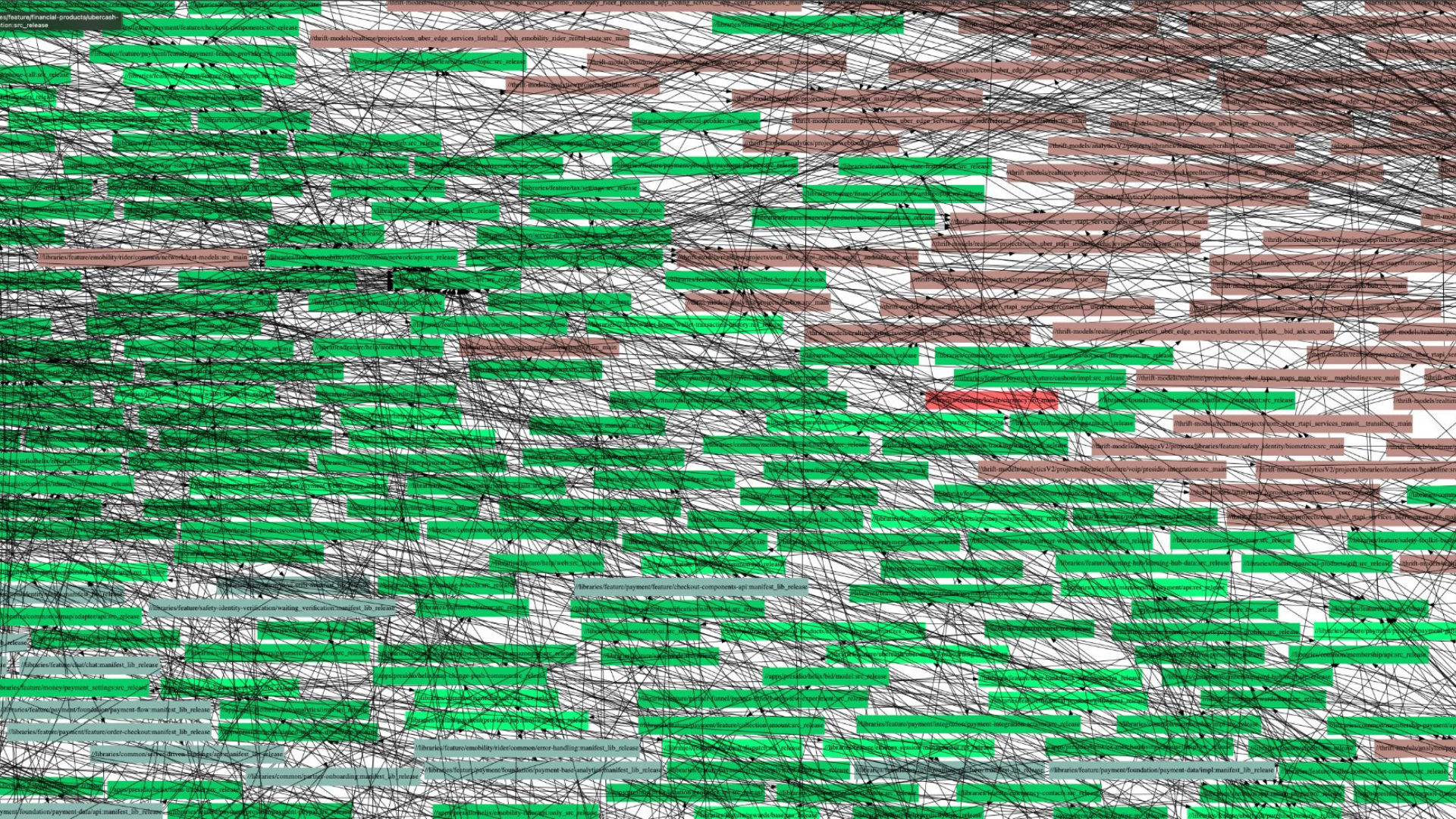
Rider App Dependency Graph

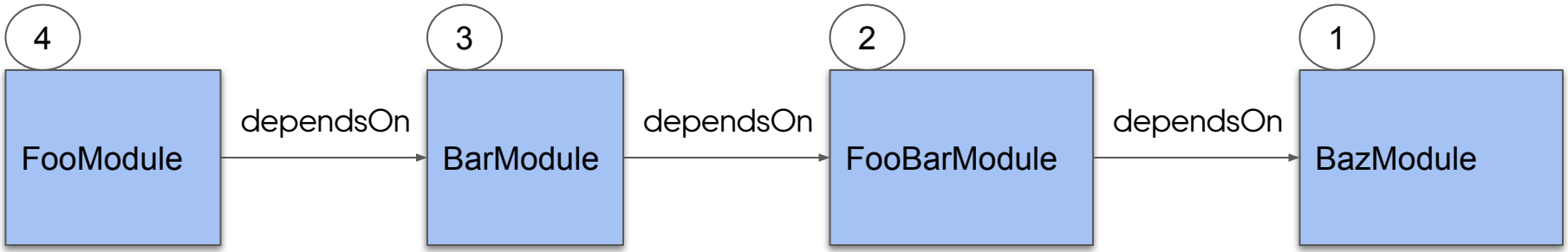


Over 12,000 module dependencies

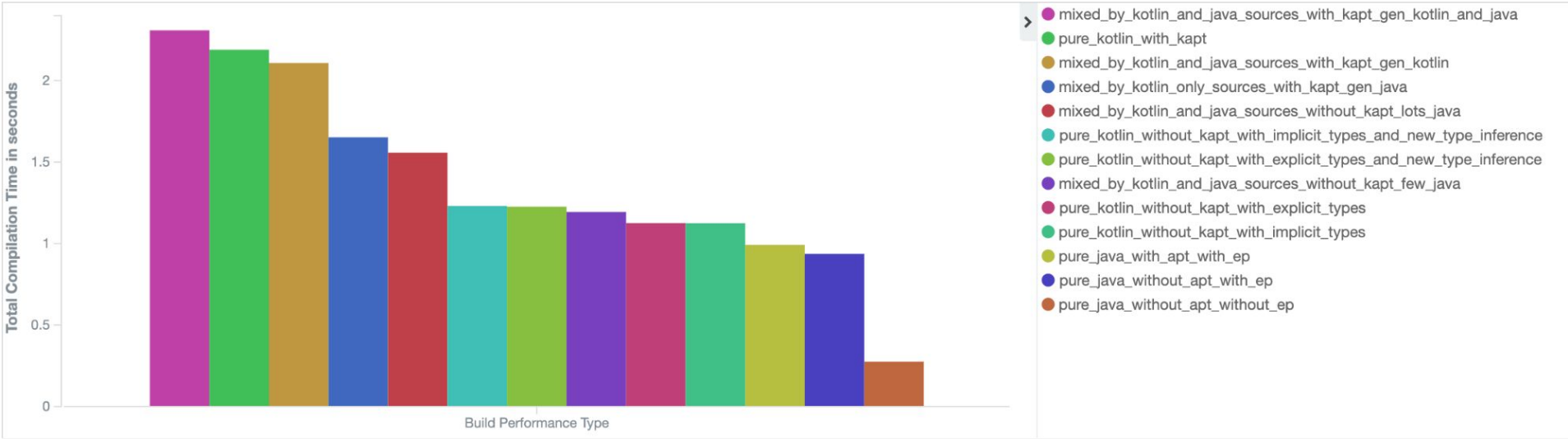
*Reduced via transitive reduction




//libraries/feature/financial-products/ubercash-ui:src_release
//libraries/feature/payment/feature/checkout-components:src_release
//thrift-models/realtime/projects/com_uber_edge_services_fireball__push_emobility_rider_rental_stage:src_main
//libraries/feature/payment/feature/payment-feature-provider:src_release
//libraries/feature/payment/feature/cashout/impl:src_release
//libraries/feature/social_profiles:src_release
//libraries/feature/safety-state-transport:src_release
//libraries/feature/financial-products/payment-offers:src_release
//libraries/feature/emobility/rider/common/network/gen-models:src_main
//libraries/feature/emobility/rider/common/network/api:src_release
//libraries/feature/wallet/home/wallet-home:src_release
//libraries/feature/wallet/home/wallet-transaction-history:src_release
//libraries/feature/help/web:src_release
//libraries/feature/safety-identity-verification/waiting_verification:manifest_lib_release
//libraries/feature/payment/feature/checkout-components-api:manifest_lib_release
//libraries/feature/chat:manifest_lib_release
//libraries/feature/payment/foundation/payment-flow:manifest_lib_release
//libraries/feature/payment/feature/order-checkout:manifest_lib_release
//libraries/feature/payment/feature/collection/amount:src_release
//libraries/feature/emobility/rider/common/error-handling:manifest_lib_release
//libraries/common/partner-onboarding:manifest_lib_release
//libraries/feature/payment/foundation/payment-base/analytics:manifest_lib_release
//libraries/feature/payment/foundation/payment-data/impl:manifest_lib_release
//libraries/feature/payment/foundation/payment-data/api:manifest_lib_release


4
FooModule
dependsOn
3
BarModule
dependsOn
2
FooBarModule
dependsOn
1
BazModule

Kotlin Builds 🐢



Kotlin is up to 2x slower than Java (with Errorprone)

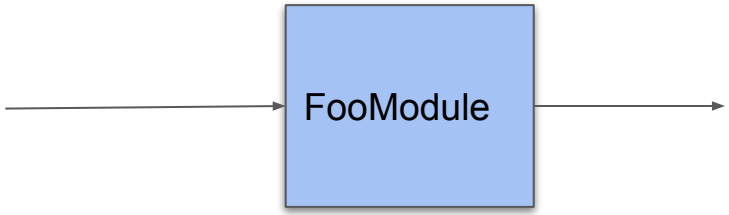
Faster Compilation

- Ins and Outs
- Cache Keys
- ABI Jars

Faster Compilation

Build Target Inputs

- Toolchain
- Dependencies
- Configuration

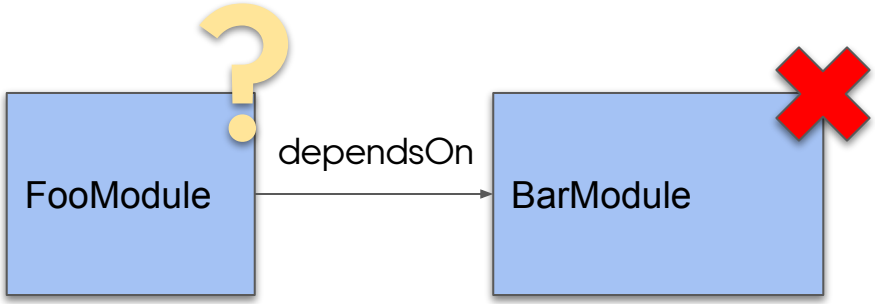


Build Target Outputs

- JARs/AARs

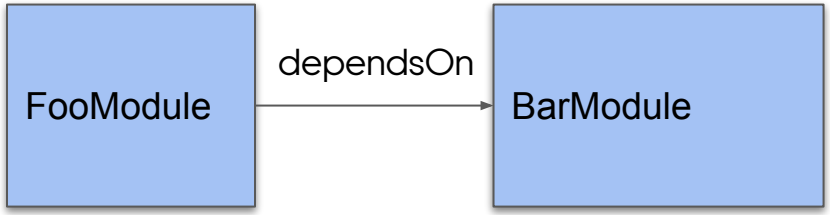
Cache Key: 9e92c406b5efc492a714d29c5e26516d

ABI Jars



ABI Jars

compile_against_abis = false



Bar Module
src/foo.kt

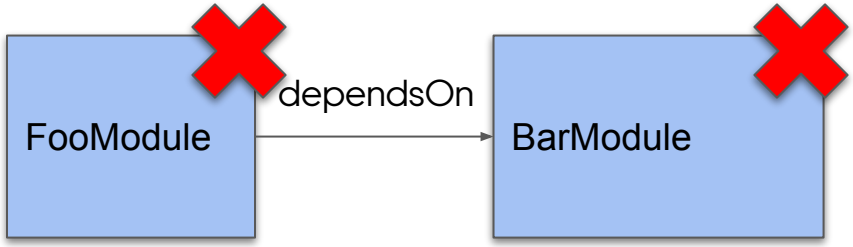
fun bar() = “Foo”

Bar.jar
Foo.class
Cache-key: 9e92c406b5efc492a714d29c5e26516d

```
public final class FooKt {
   @NotNull
   public static final String bar() {
      return "Foo";
   }
}
```

ABI Jars

compile_against_abis = false



Bar Module
src/foo.kt

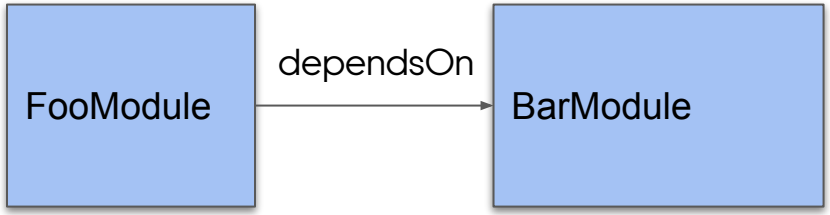
fun bar() = “FooBar”

Bar.jar
Foo.class
Cache-key: a28d01554c0f541c04ec3a2fe28396ed

```
public final class FooKt {
   @NotNull
   public static final String bar() {
      return "FooBar";
   }
}
```

ABI Jars

compile_against_abis = true



Bar Module
src/foo.kt

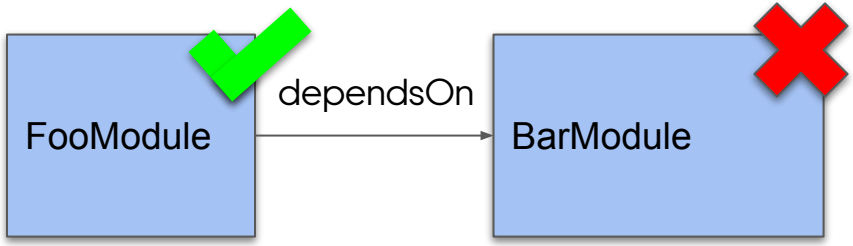
fun bar() = “Foo”

Bar.abi.jar
Foo.class
Cache-key: d3b7138441694bea985e32ef219697b6

```
public final class FooKt {
   @NotNull
   public static final String bar() {  }
}
```

ABI Jars

compile_against_abis = true



Bar Module
src/foo.kt

fun bar() = “FooBar”

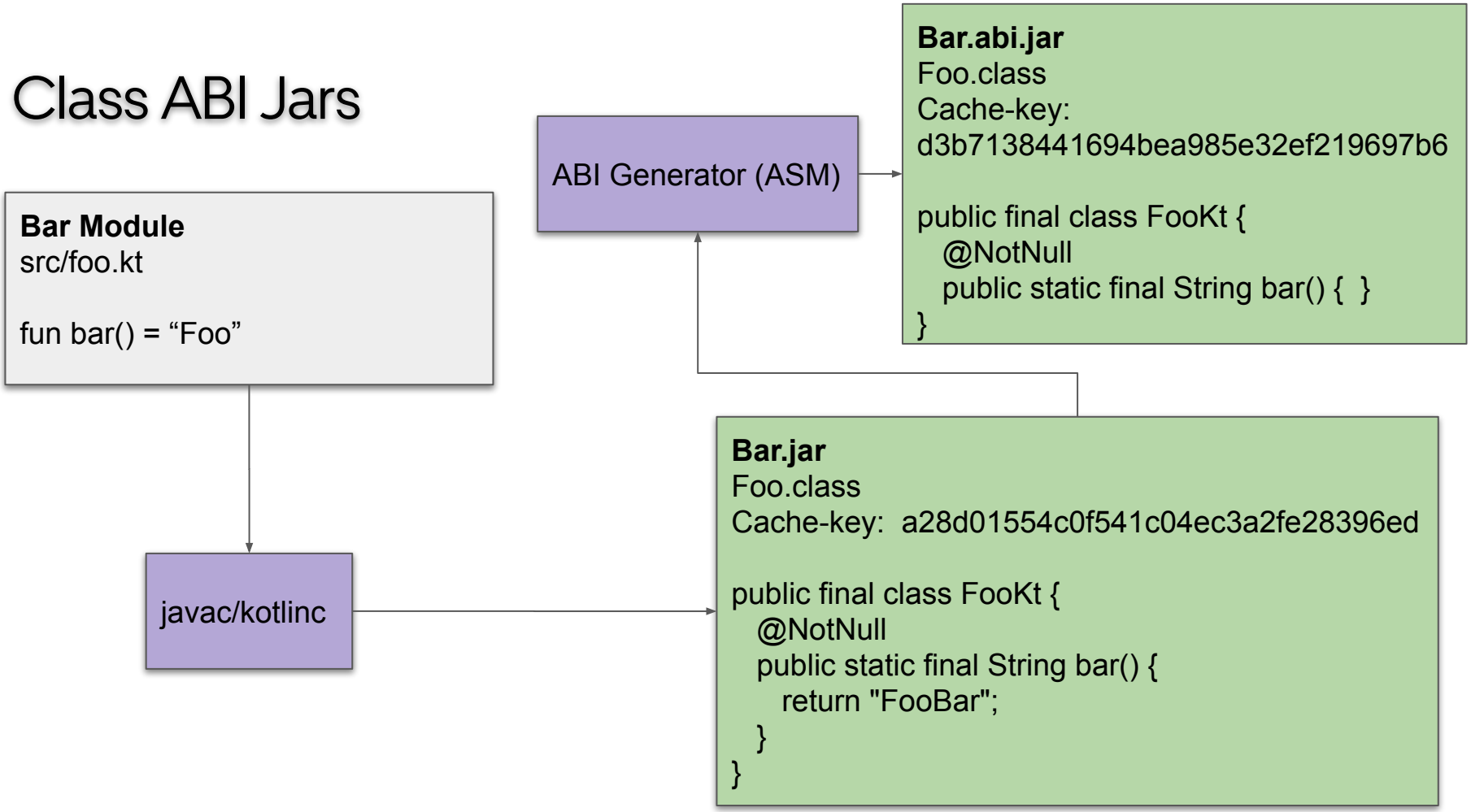
Bar.abi.jar
Foo.class
Cache-key: d3b7138441694bea985e32ef219697b6

```
public final class FooKt {
   @NotNull
   public static final String bar() {  }
}
```

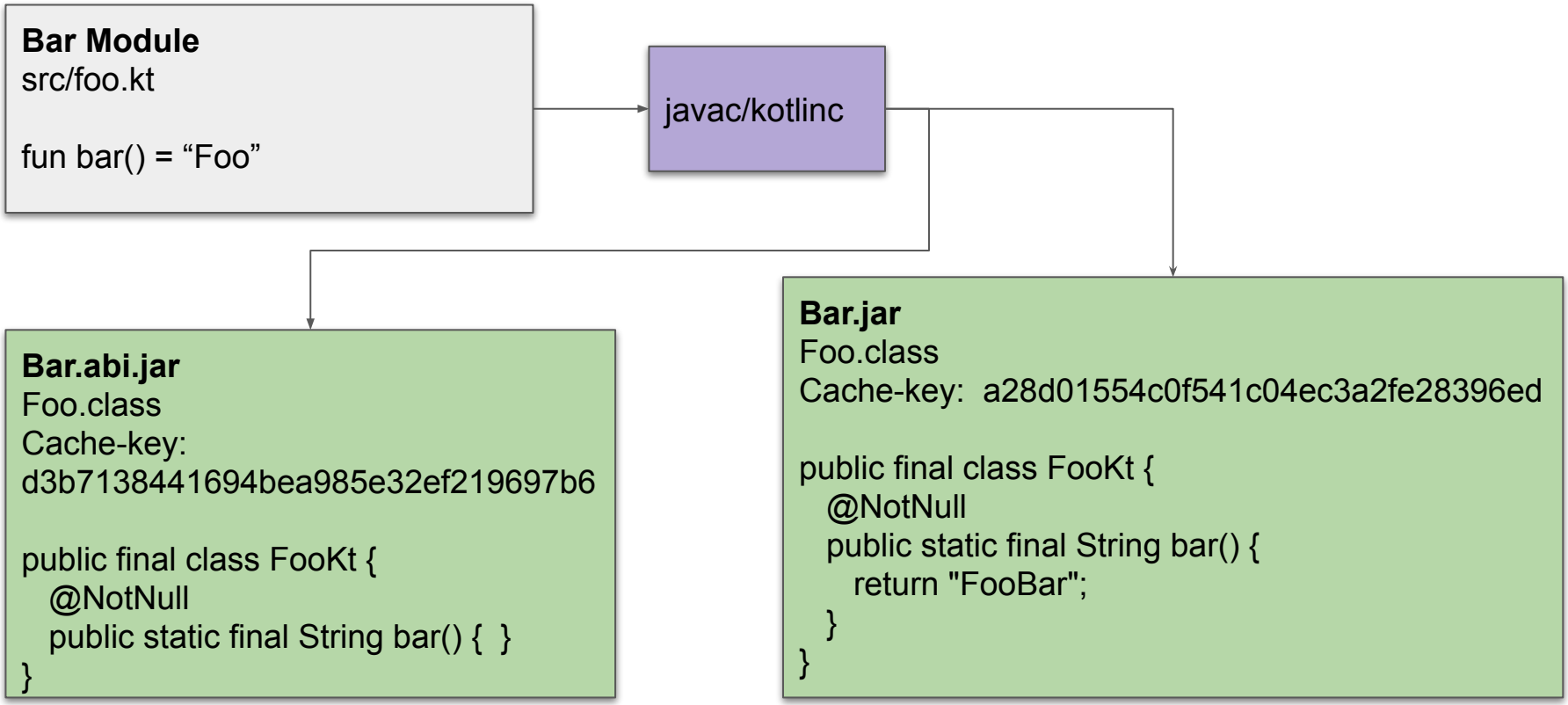
Faster Compilation

- Ins and Outs
- Cache Keys
- ABI Jars
  - Class ABI jars
  - Src ABI Jars

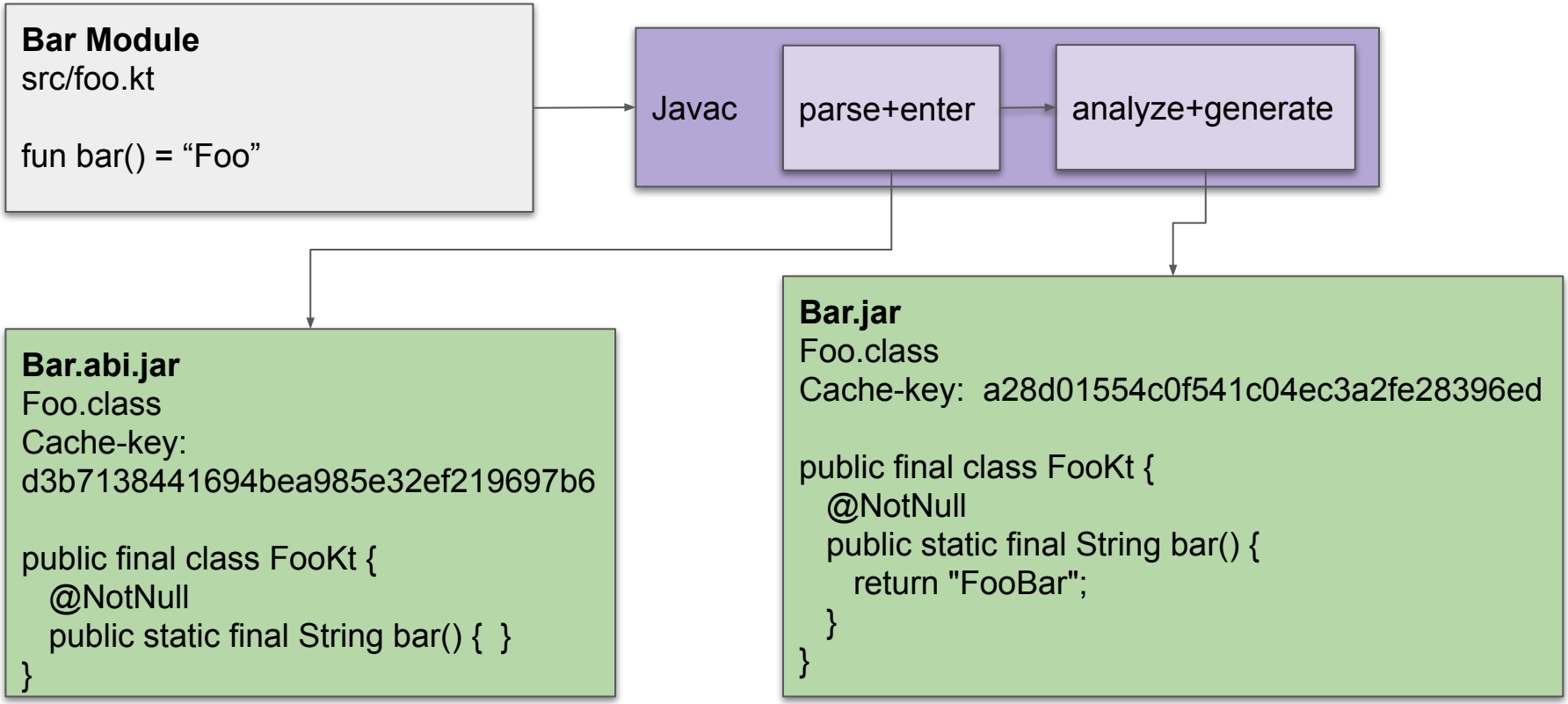
Class ABI Jars



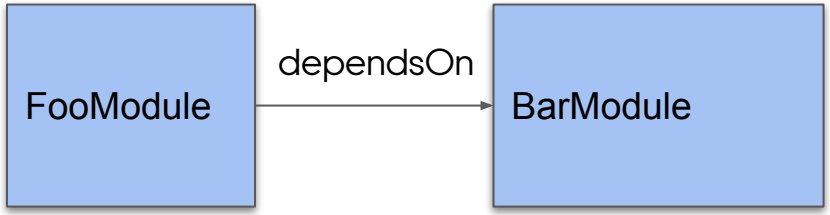
Source ABI Jars

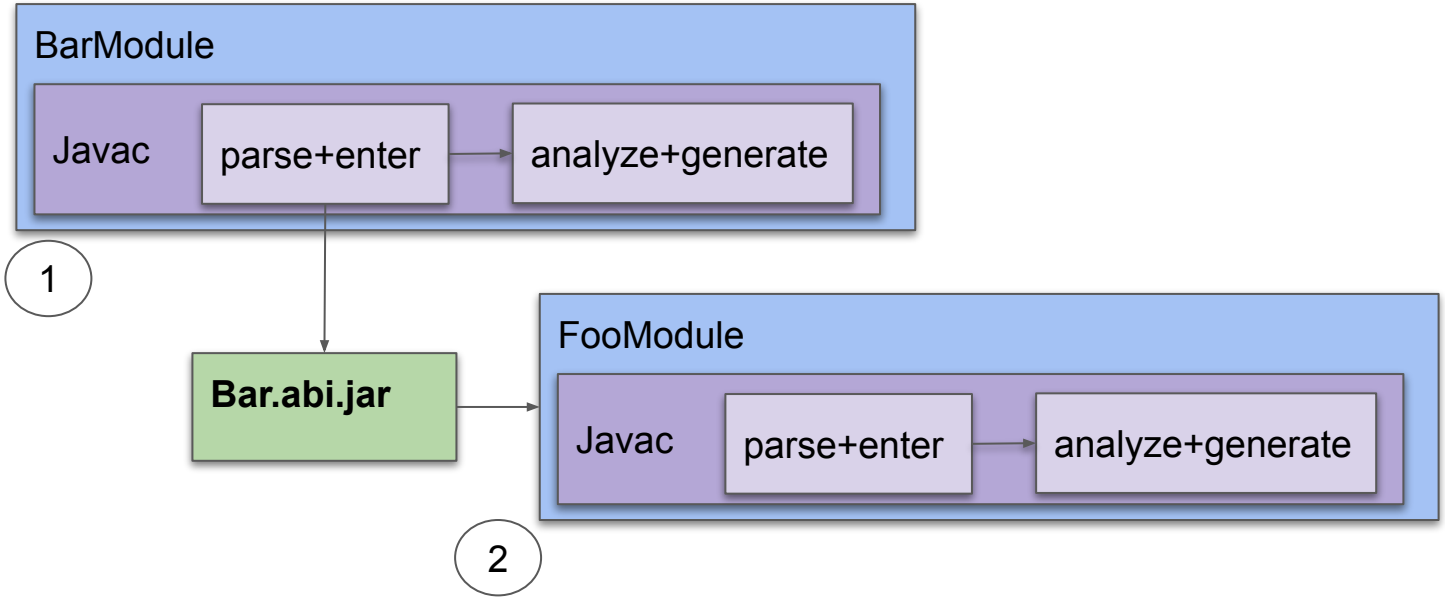


Source ABI Jars (Java)

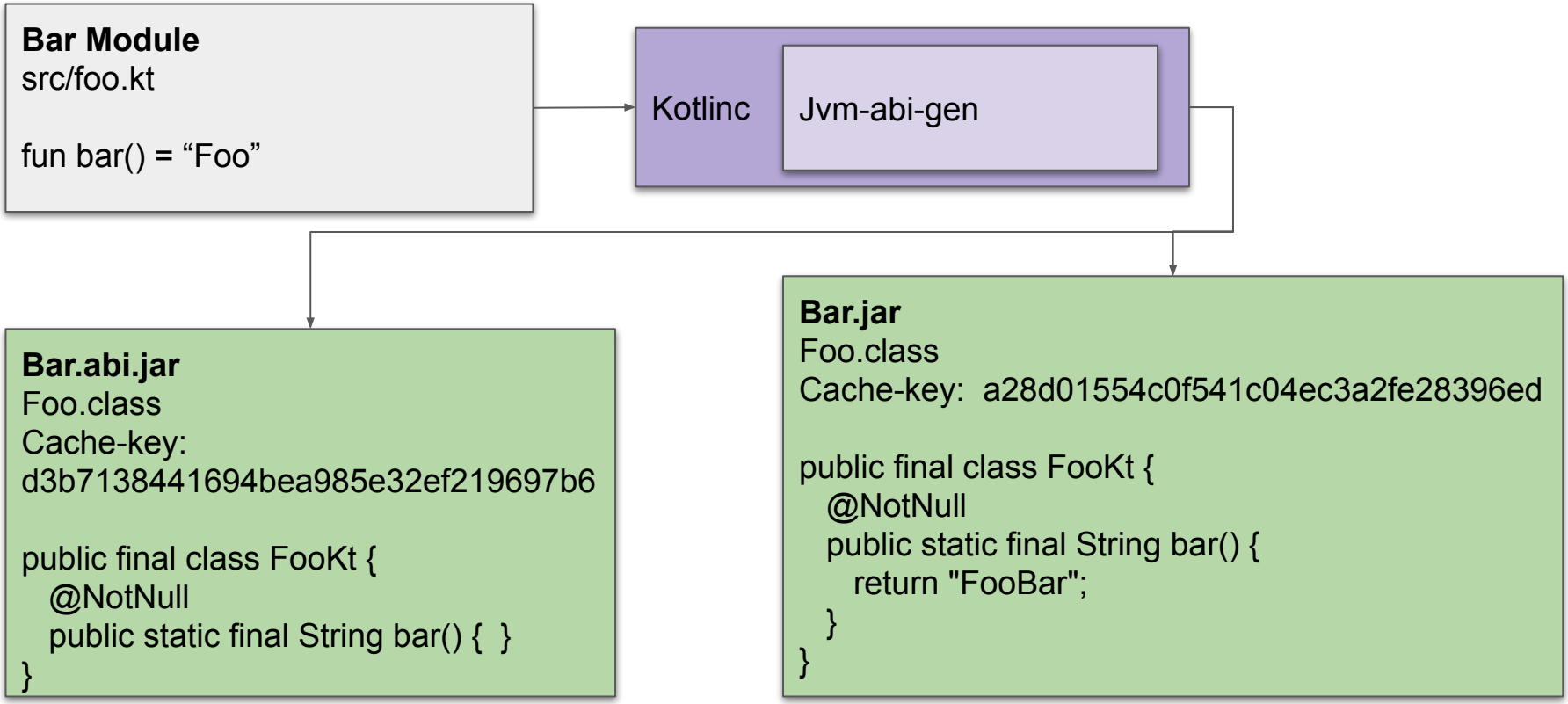


Rule Pipelining (Java)

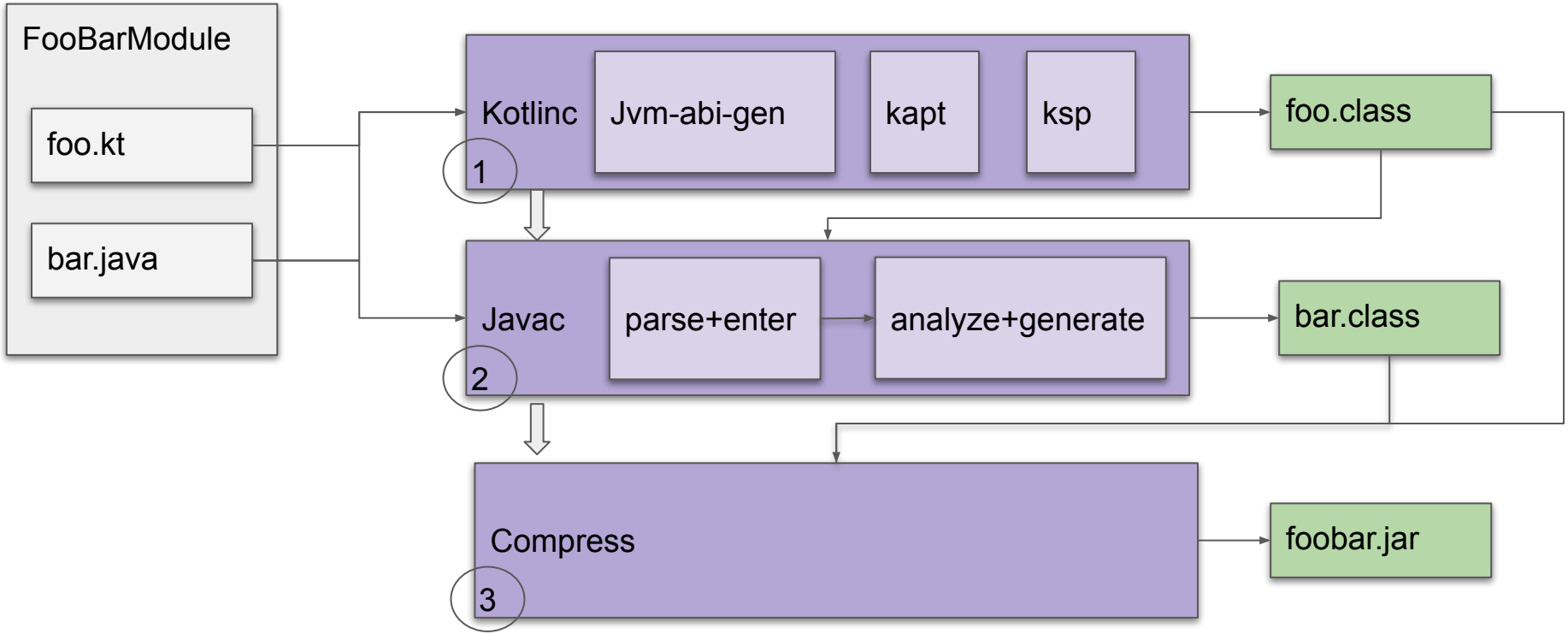




Source ABI Jars (Kotlin)

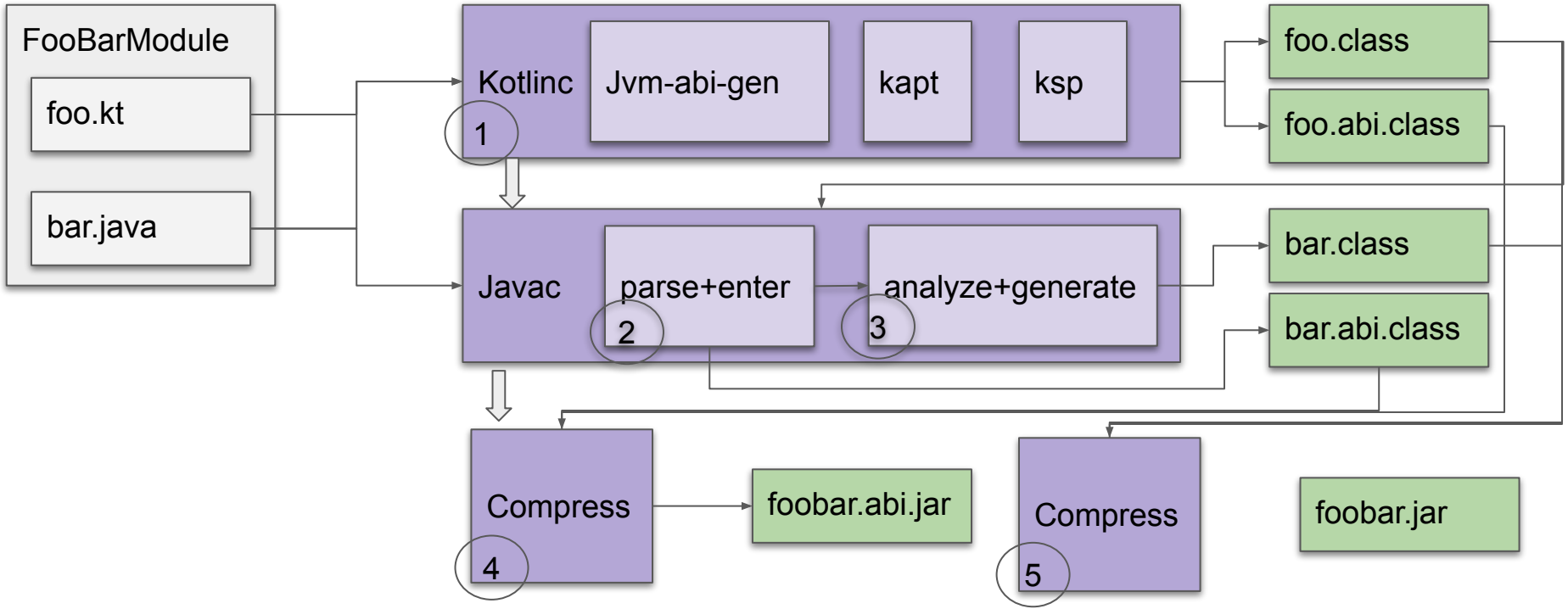


Mixed Java and Kotlin Sources



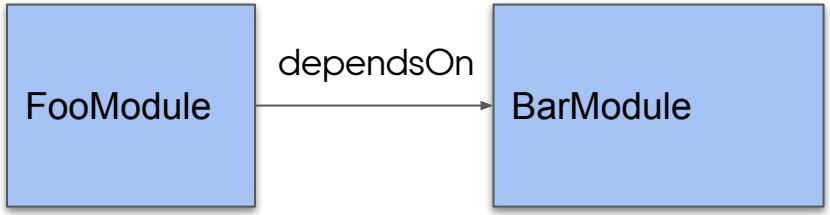
Kotlin Rule Pipelining

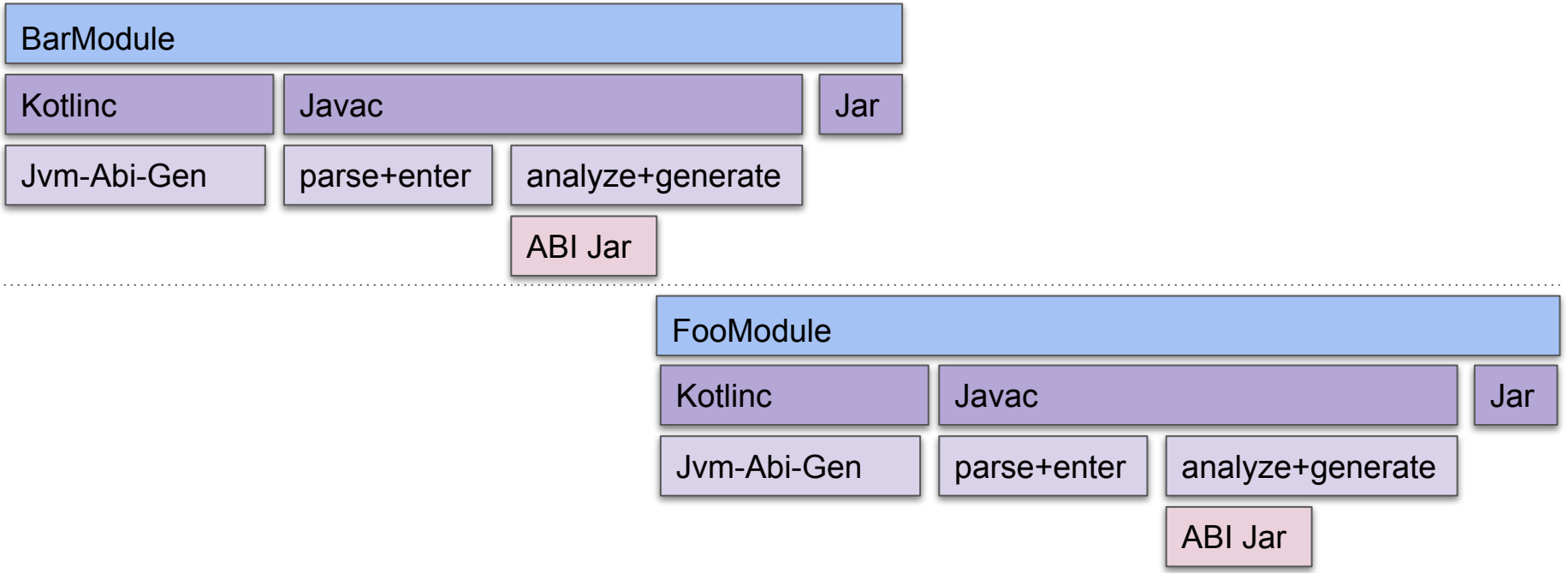
Making Mixed Java/Kt Sources Faster



Kotlin Rule Pipelining

Making Mixed Java/Kt Sources Faster

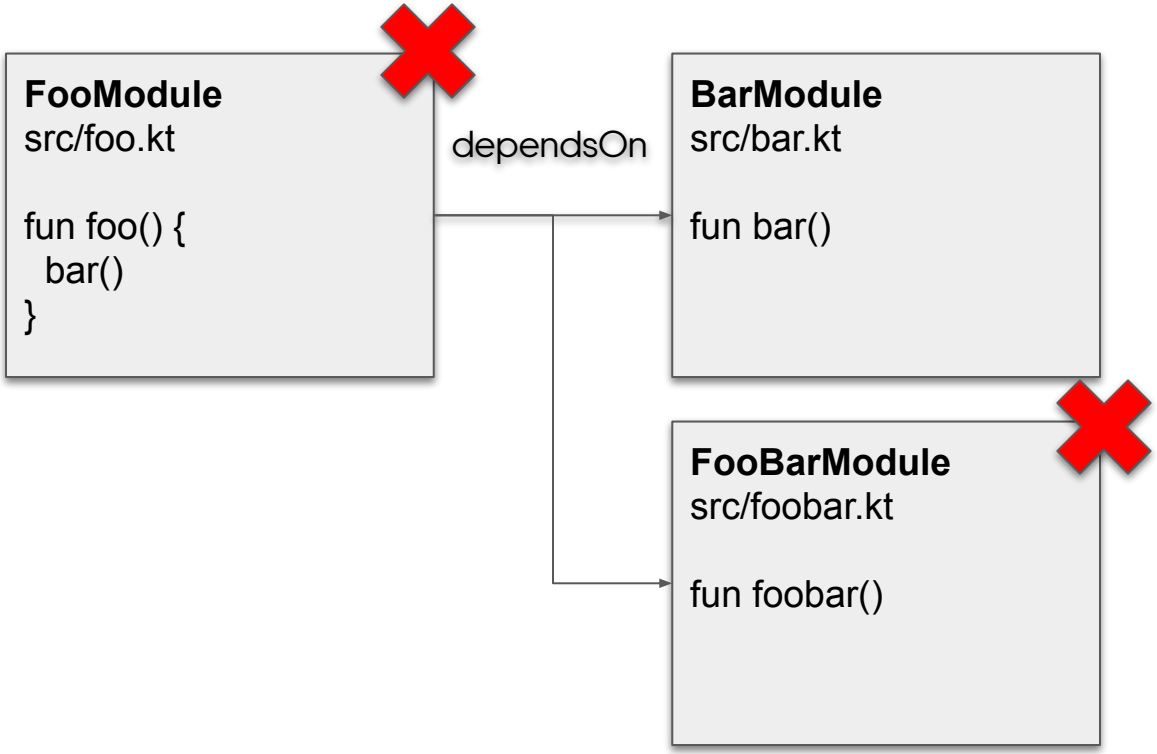




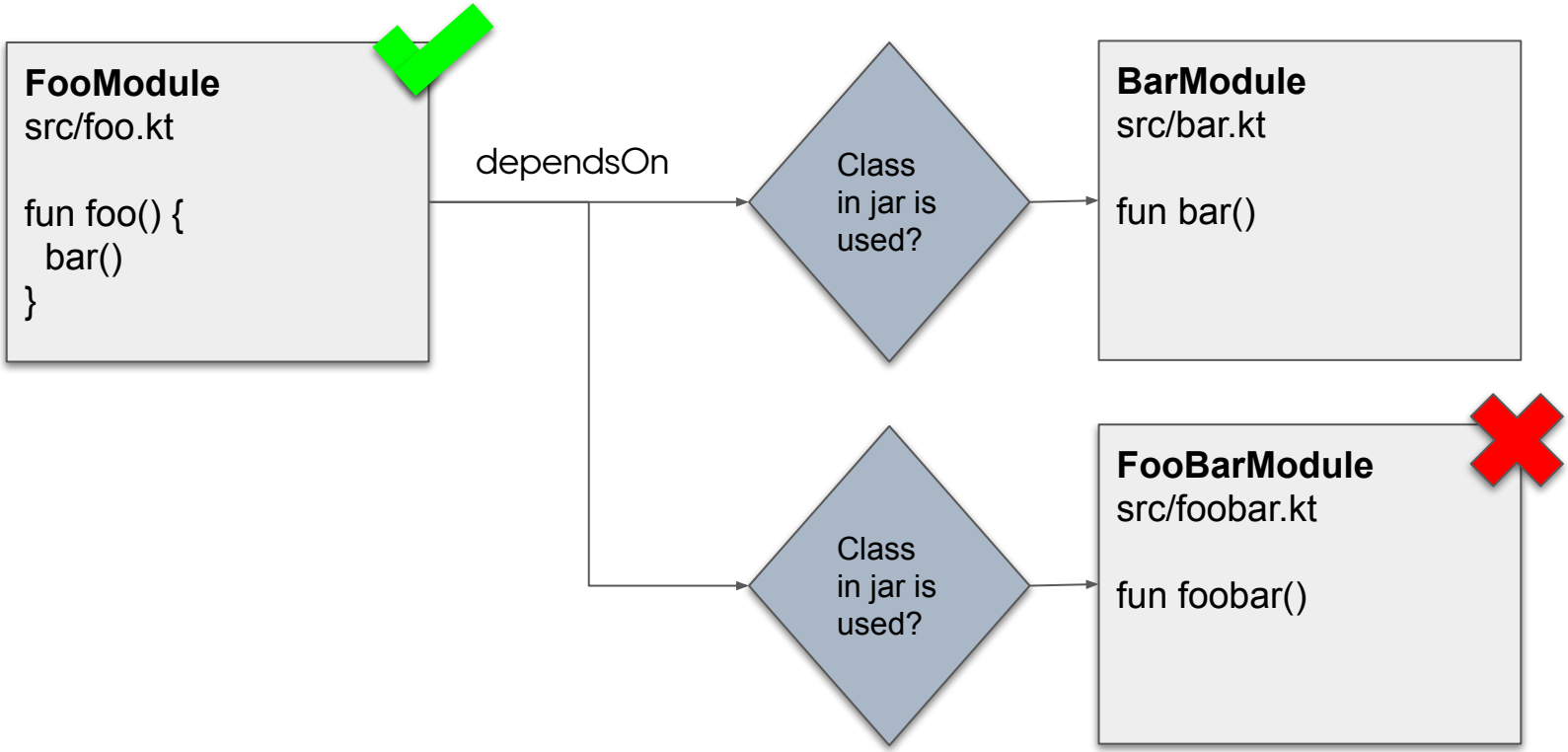
Faster Compilation

- Ins and Outs
- Cache Keys
- ABI Jars
  - Class ABI jars
  - Src ABI Jars
- Per class compiler avoidance

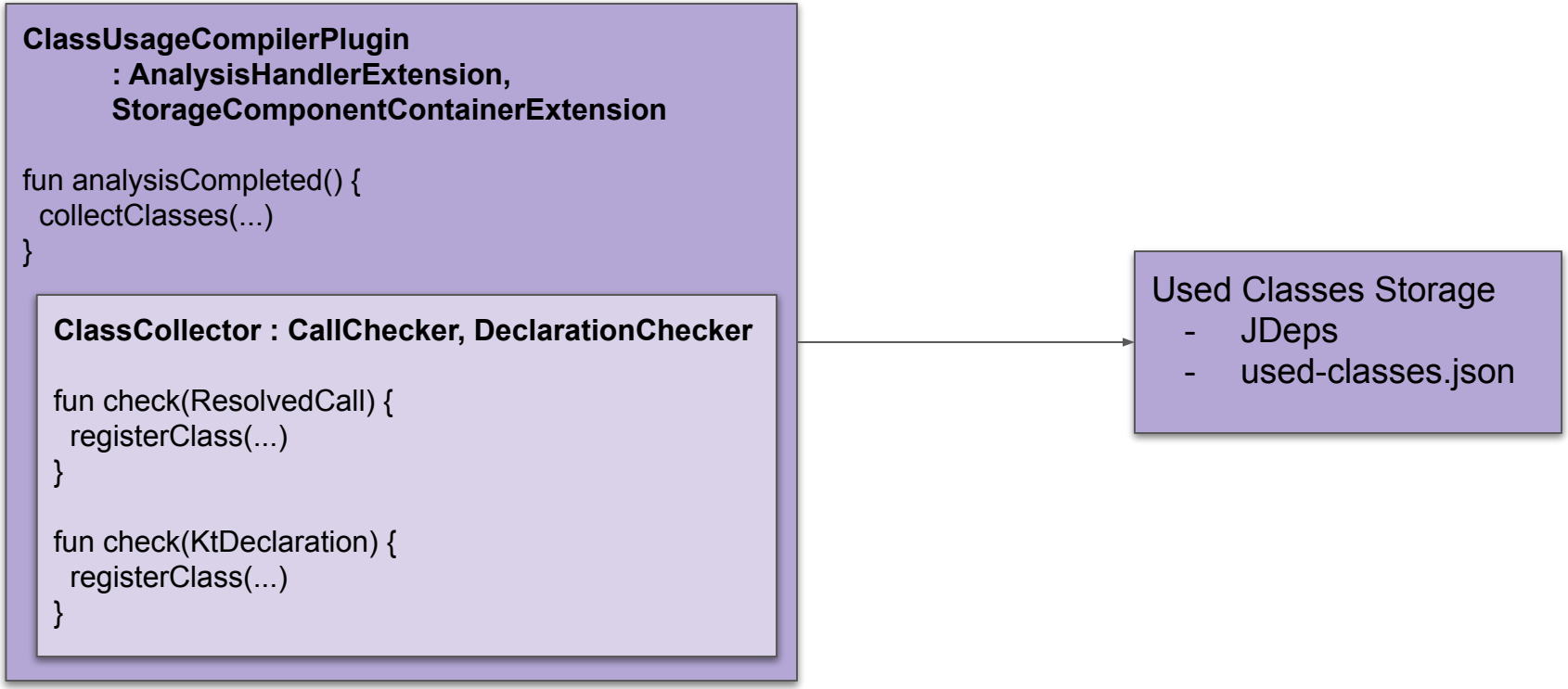
Per Class Compiler Avoidance



Per Class Compiler Avoidance



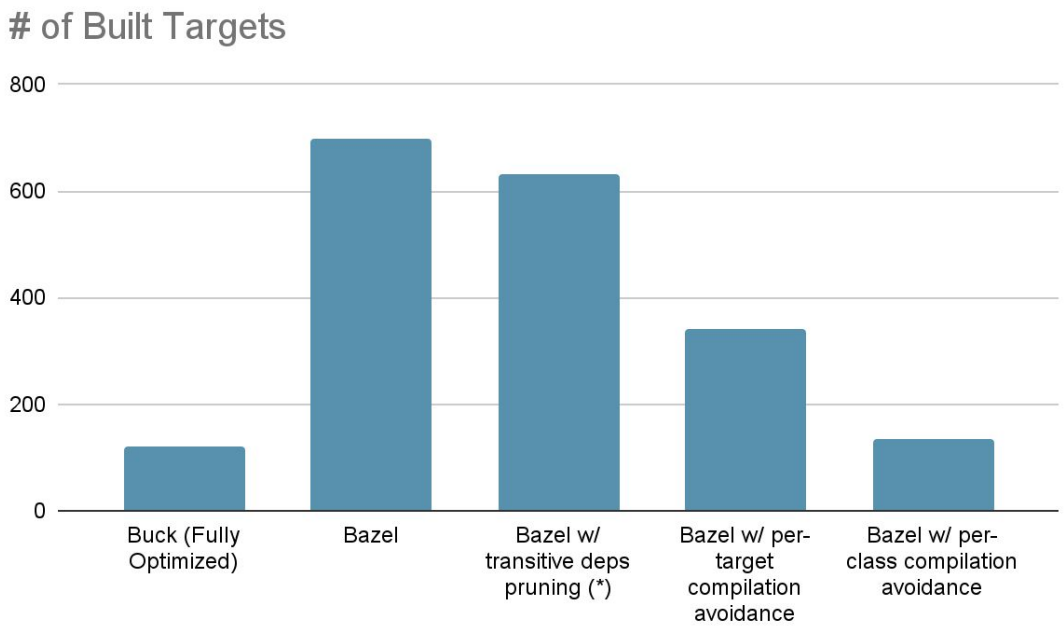
Kotlin Class Usage Tracking Compiler Plugin



Faster Compilation

- Ins and Outs
- Cache Keys
- ABI Jars
  - Class ABI jars
  - Src ABI Jars
- Per class compiler avoidance
- Annotation Processing

Compiler Avoidance Data

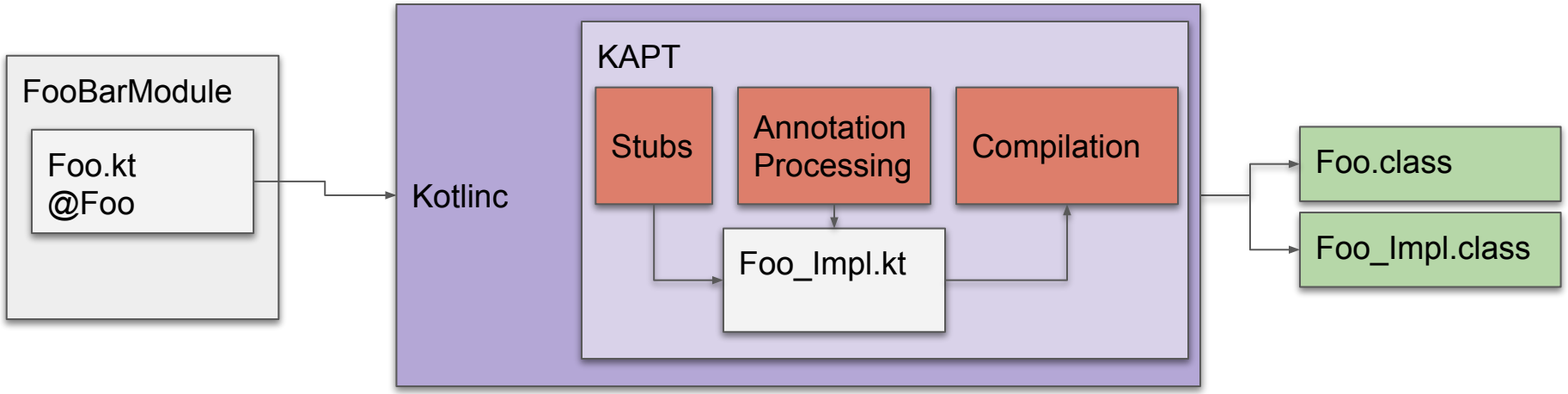


github.com/bazelbuild/bazel/pull/16457

Annotation Processing

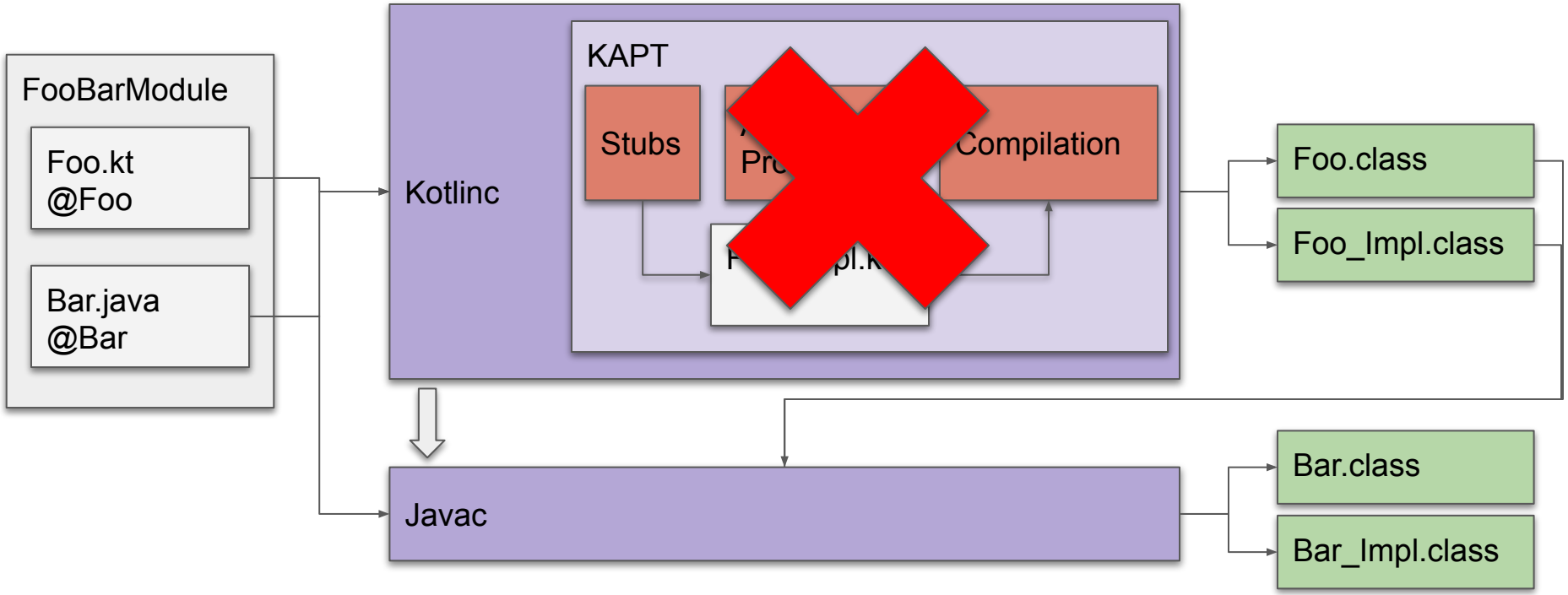
- Kotlin Annotation Processing Tool (KAPT)
  - Slow
- Java Annotation Processing (AP)
  - Fast
  - Doesn’t support Kotlin
- Kotlin Symbol Processing (KSP)
  - Fast
  - Doesn’t have wide support

Kotlin Annotation Processing

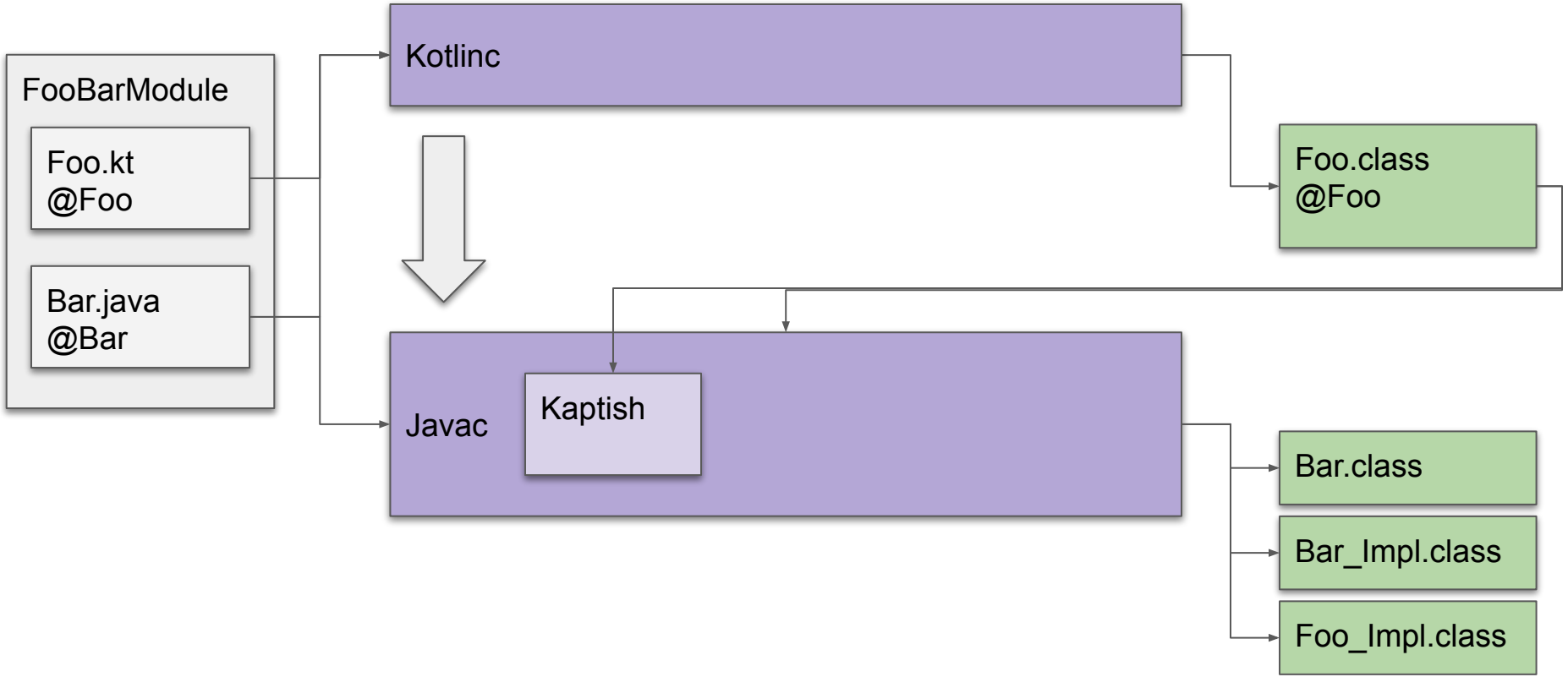


KAPT slows build time by more than 150% compared to Kotlin without AP or Javac AP

Kotlin Annotation Processing



Kotlin Annotation Processing



Kaptish

```
class Kaptish : Plugin {
  override fun getName() = "Kaptish"

  override fun init(task: JavacTask, vararg args: String?) {
      val classes = getClasses(task.context)
      Arguments.instance(task.context).classNames.addAll(classes)
  }
```

“The javac command can also process annotations in Java source files and classes.” - Javac docs

Kaptish Downsides

- Cannot reference generated code directly in Kotlin.
- Must have class retained annotations
- Cannot generate Kotlin code

Faster Compilation

- Ins and Outs
- Cache Keys
- ABI Jars
  - Class ABI jars
  - Src ABI Jars
- Per class compiler avoidance
- Annotation Processing
- Future:
  - KSP
  - K2 Compiler
  - Graph Flattening

IDEs and Devtools

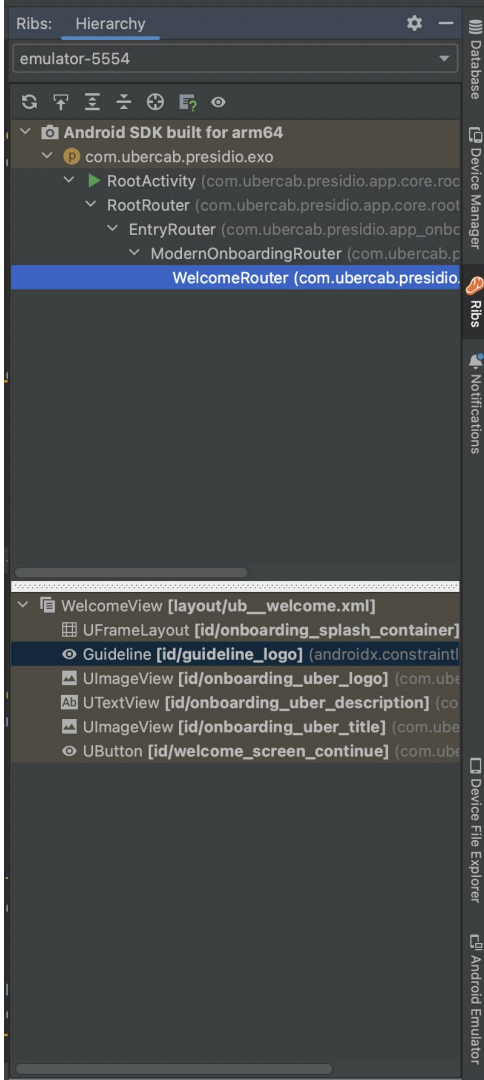
Uber

Local development

- M1 Max hardware running Mac OS
- Intellij 2022
- Uber IDE Plugins
- Automatic IDE settings (Vmoptions, default settings, code style)
- Integrate LDA
- Managed IT and standard applications via Chef
- Developer setup Android environment ❌

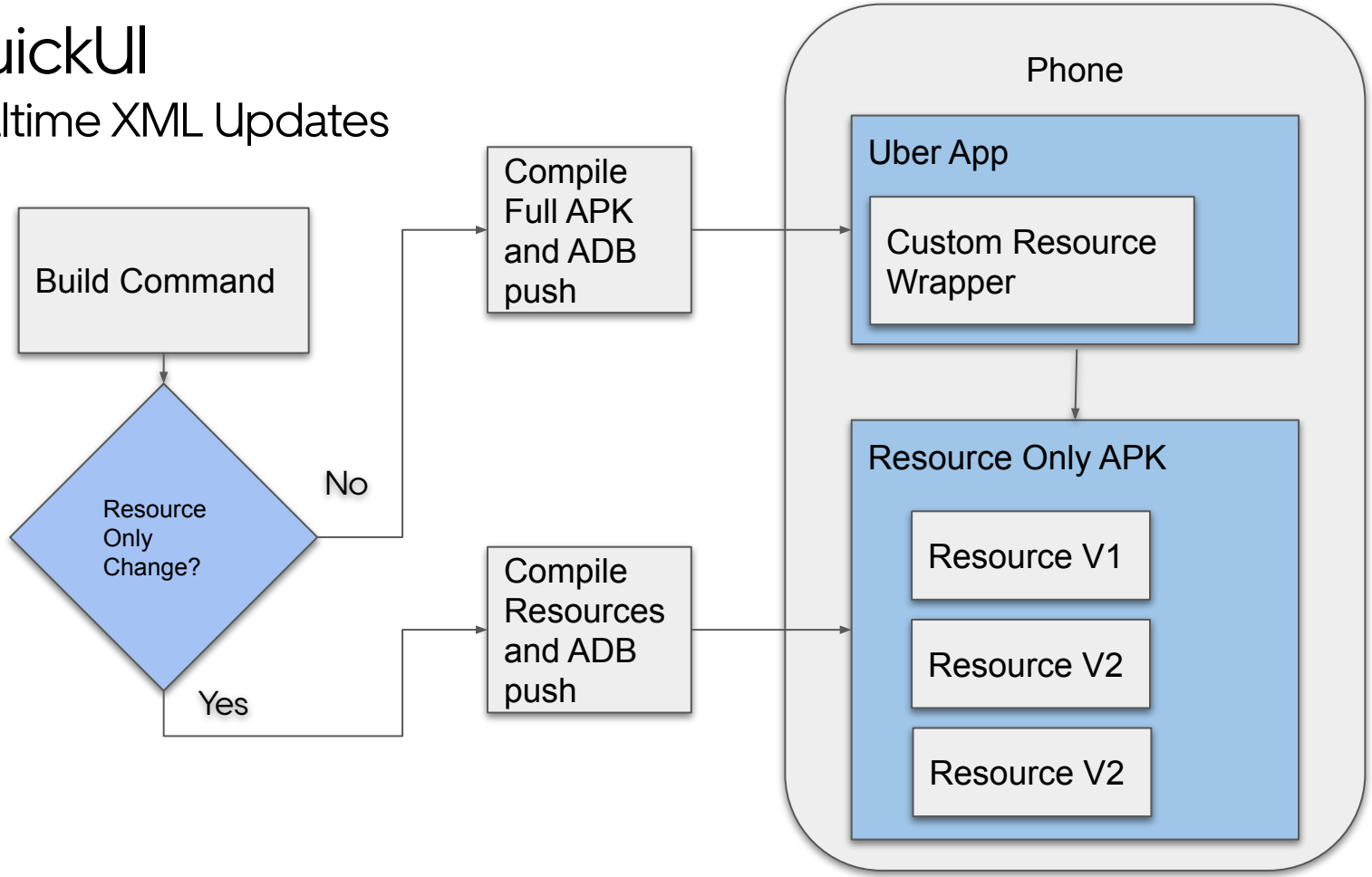
Custom Intellij Plugins

- Local Developer Analytics
- Scaffold new Module
- Scaffold new RIB with Compose Views
- Scaffold new Sandbox app
- Motif Dependency explorer
- Realtime RIB explorer
- Index new apps/modules
- Manage third party plugins
- Analytics line markers
- Live Templates
- Realtime UI updates

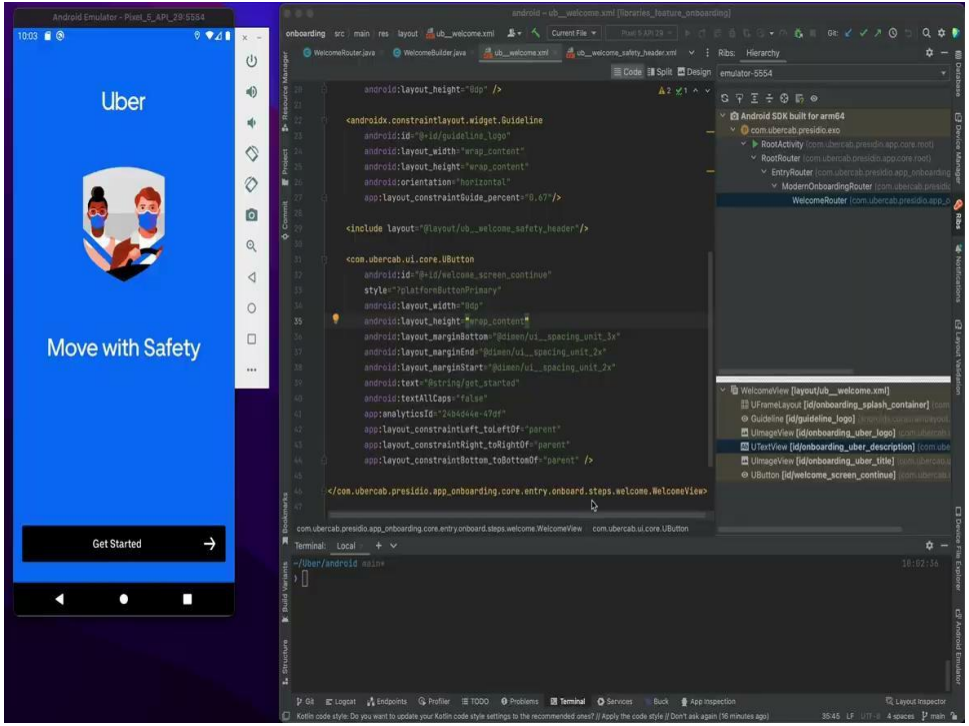


QuickUI

Realtime XML Updates



QuickUI Demo



Devpod (Cloud IDEs)

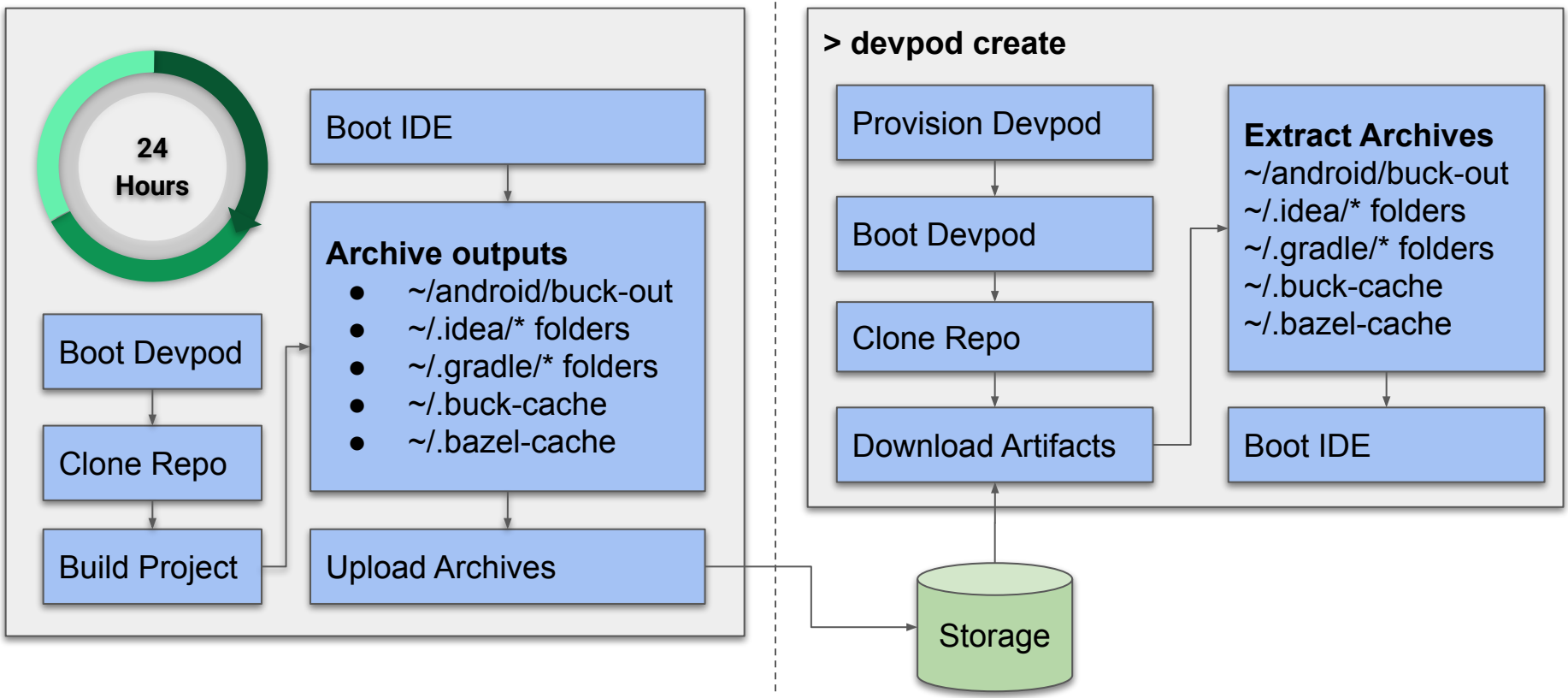
- Beefy Dev Servers
- Global distributed
- Containerized dev environment
- Preloaded Caches
- IDE as a daemon

- User customization
- Jetbrains Gateway and VSCode
- Supports multiple devpods
- Instant context Switching

“I used to barely being able to make large scale migrations on my macbook w/o steam coming out of it, and now im changing 20K LOC in 2K files w/ all modules in Intellij indexed on a cloud machine over airplane wifi. Crazy times!”

Devpods + Android Emulators & Devices

Snapshots


24
Hours
Boot IDE
Archive outputs
~/android/buck-out
~/.idea/* folders
~/.gradle/* folders
~/.buck-cache
~/.bazel-cache
Boot Devpod
Clone Repo
Build Project
Upload Archives
> devpod create
Provision Devpod
Boot Devpod
Clone Repo
Download Artifacts
Extract Archives
~/android/buck-out
~/.idea/* folders
~/.gradle/* folders
~/.buck-cache
~/.bazel-cache
Boot IDE
Storage

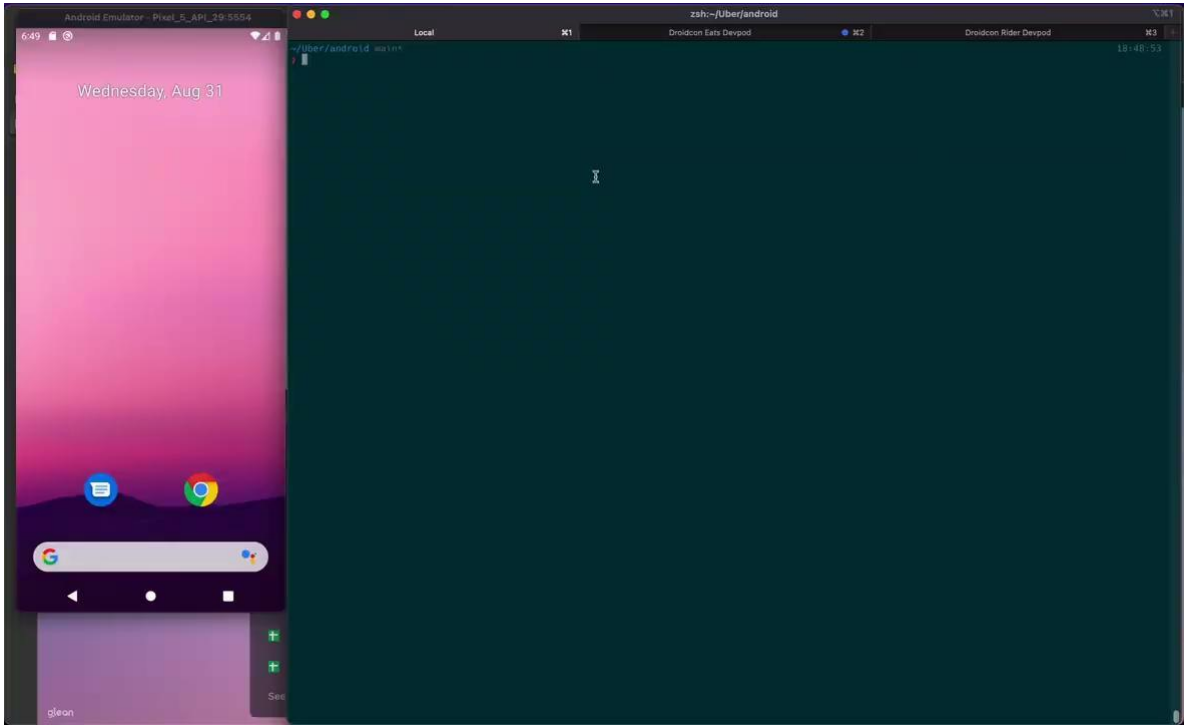
Jetbrains Cached Index

Indexing Project target: a project in Android monorepo

Regular Indexing	Jetbrains Shared Index Plugin	Cached Index
38 minutes	**32 minutes**	**seconds**

* provided time corresponds to Intellij IDEA “Updating Index” stage

Devpod Demo


Android Emulator - Pixel_5_API_29:5554
6:49
Wednesday, Aug 31
zsh:~/Uber/android
Local
Droidcon Eats Devpod
Droidcon Rider Devpod

Recap

01 Mobile at Uber Overview
02 Development Stack and Workflow
03 Measuring Devx
04 Making Builds Faster
05 IDE and Devtools

Resources

- **RIBs** - github.com/uber/ribs
- **Motif** - github.com/uber/motif
- **Nanoscope** - github.com/uber/nanoscope
- **Nullaway** - github.com/uber/nullaway
- **Piranha** - github.com/uber/piranha
- **Gradle/Buck/Bazel evaluation** - github.com/uber-common/android-build-eval
- **Napt -** github.com/sergei-lapin/napt
- **Flipper** - github.com/facebook/flipper
- **Detekt** - github.com/detekt/detekt
- **KTFMT** - github.com/facebookincubator/ktfmt
- **Devpod like product** - coder.com, Gitpod.io, & Github Code Spaces

Mobile Developer Productivity at Uber Scale

Ty Smith
Android Platform Tech Lead

Uber

@tsmith