Automate your workflows with Kotlin

Fosdem - 2020

Automate your workflows with Kotlin

@martinbonnin

@mgauzins

dailymotion

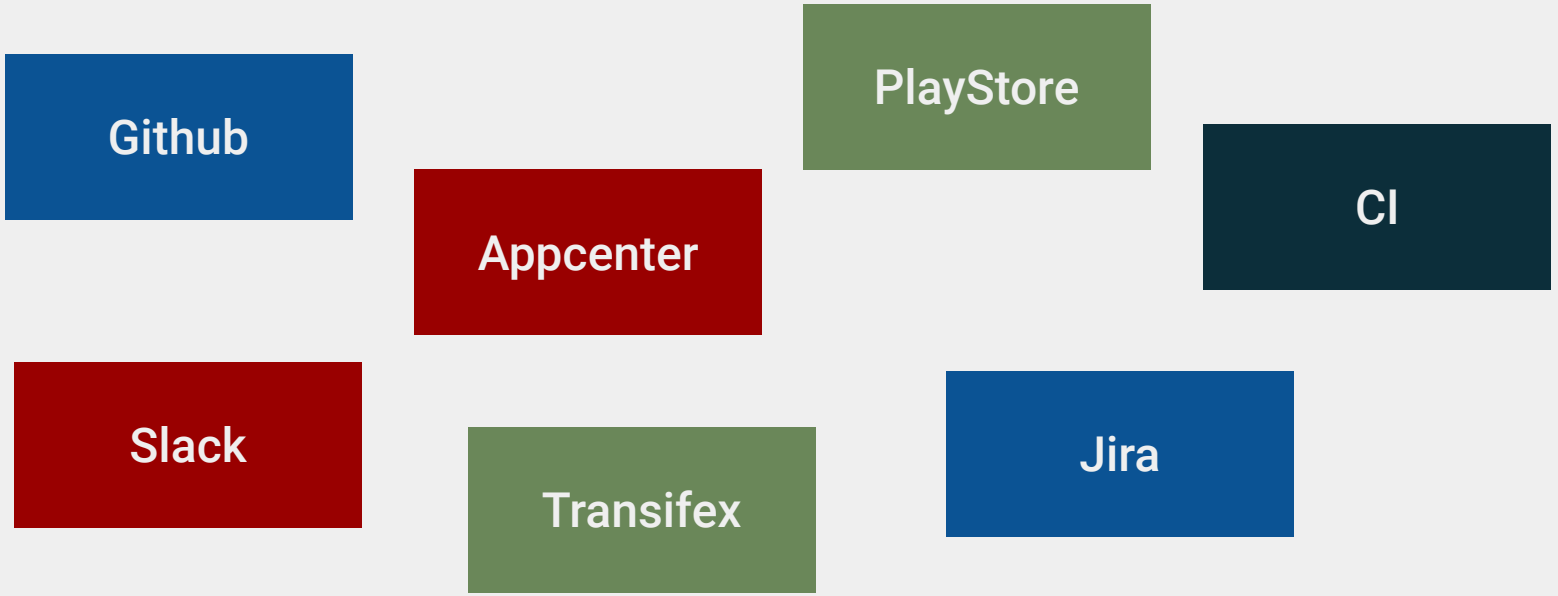
A daily work...

1. Assign a ticket
2. Create a branch
3. Code... 🤓
4. Create a pull request
5. Move ticket state
6. Merge pull request
7. Move ticket state
8. Create an alpha
    a. Increment a version
    b. Tag
    c. Push
9. Send a message to designers/product owners
10. Integrate feedbacks
11. Back to step 1

But also...

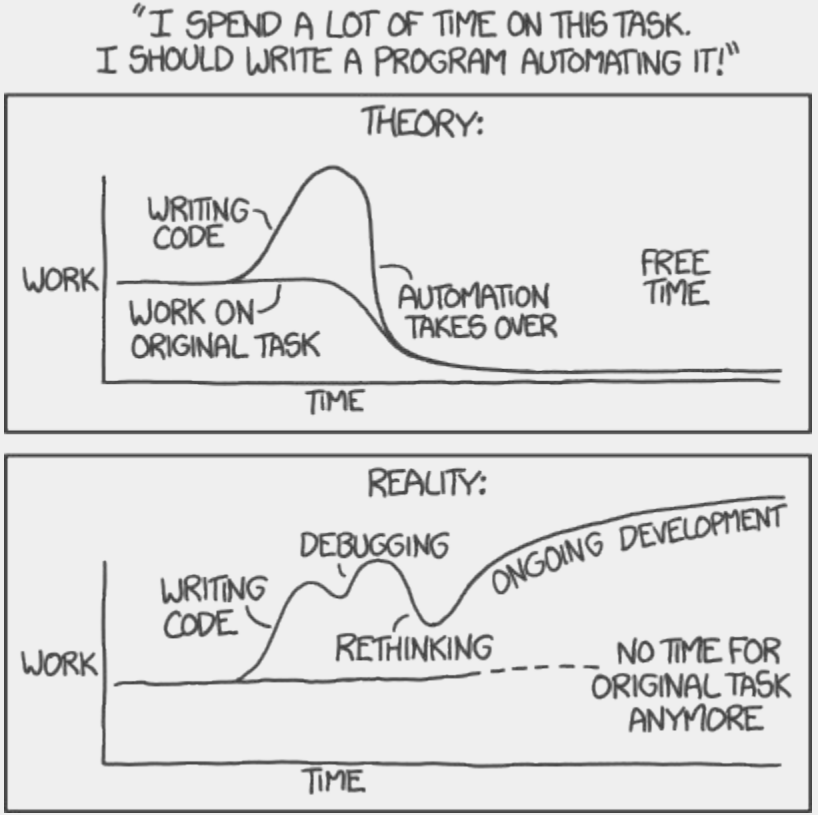
- Manage app translations
- Keep the store app up to date (images, listings, archives)
- Manage app rollout
- Notify about the updates
- Publish to alternative stores

Environment



Why automating ?

- It takes times.
- Reliability
- Reproducibility
- Documentation
- Fun
- Kotlin to the rescue



https://xkcd.com/1319/

Why Kotlin?

- The language we use every day
  - No context change (bash syndrome)
  - Every team member knows it
- Modern
- Great IDE
- Great Ecosystem

What did we replace ?

Ad-hoc scripts	General purpose tools	3rd party tools	Build system
generate_docs.sh (Bash)	Fastlane (Ruby)	Github hub (Go)	build.gradle (Groovy)
after_success.sh (Bash)		Transifex cli (Ruby)	

Tools we used

- Kotlin scripts
  - Based on Kscript
- Kotlin command line app (cli)
  - Called Kinta
  - Based on Clikt

Kscript

Scripting - motivations

- For short projects/single file projects
- No need for gradle
- Easy to setup/use

Scripting - simple example

```
// hello.kts
println("Hello ${args[0]}!")


// running the script
$ kotlinc -script hello.kts Fosdem
Hello Fosdem!
```

https://github.com/Kotlin/KEEP/blob/master/proposals/scripting-support.md

Kscript - scripting improvements

- Compiled script caching
- Shebang and interpreter usage
- Dependencies
- Standalone binaries
- IDE support
- https://github.com/holgerbrandl/kscript

Kscript - installation

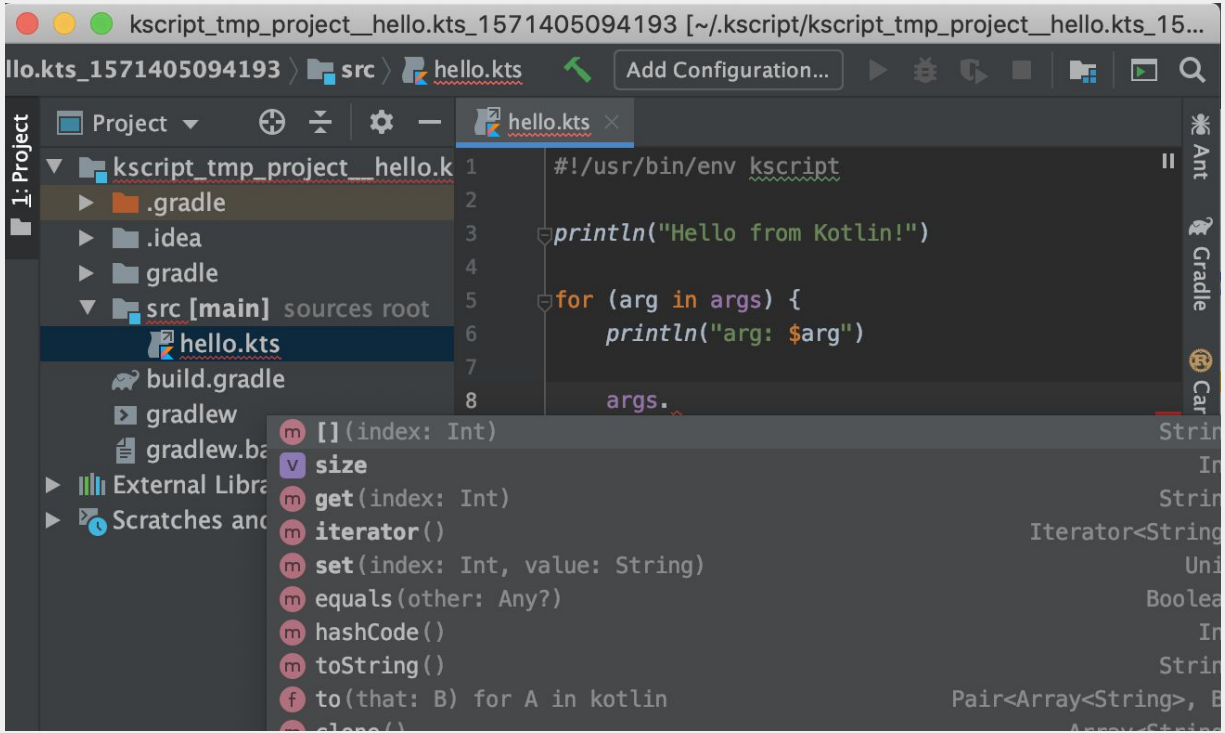
```
curl -s "https://get.sdkman.io" | bash  # install sdkman
source ~/.bash_profile                  # add sdkman to PATH

sdk install kotlin                      # install Kotlin
sdk install kscript                     # install Kscript


touch hello.kts
kscript --idea hello.kts                # start the IDE
```

Kscript

```
// weekend.kts
#!/usr/bin/env kscript
@file:DependsOn("com.squareup.okhttp3:okhttp
:4.3.1")

import okhttp3.OkHttpClient
import okhttp3.Request

val weekend =
Request.Builder().get().url("http://isitweek
endyet.com/").build().let {
   OkHttpClient().newCall(it)
}.execute().body!!.string().toLowerCase().co
ntains("yes")

if (weekend) {
   println("It is the weekend!")
} else {
   println("Not yet :-|")
}
```

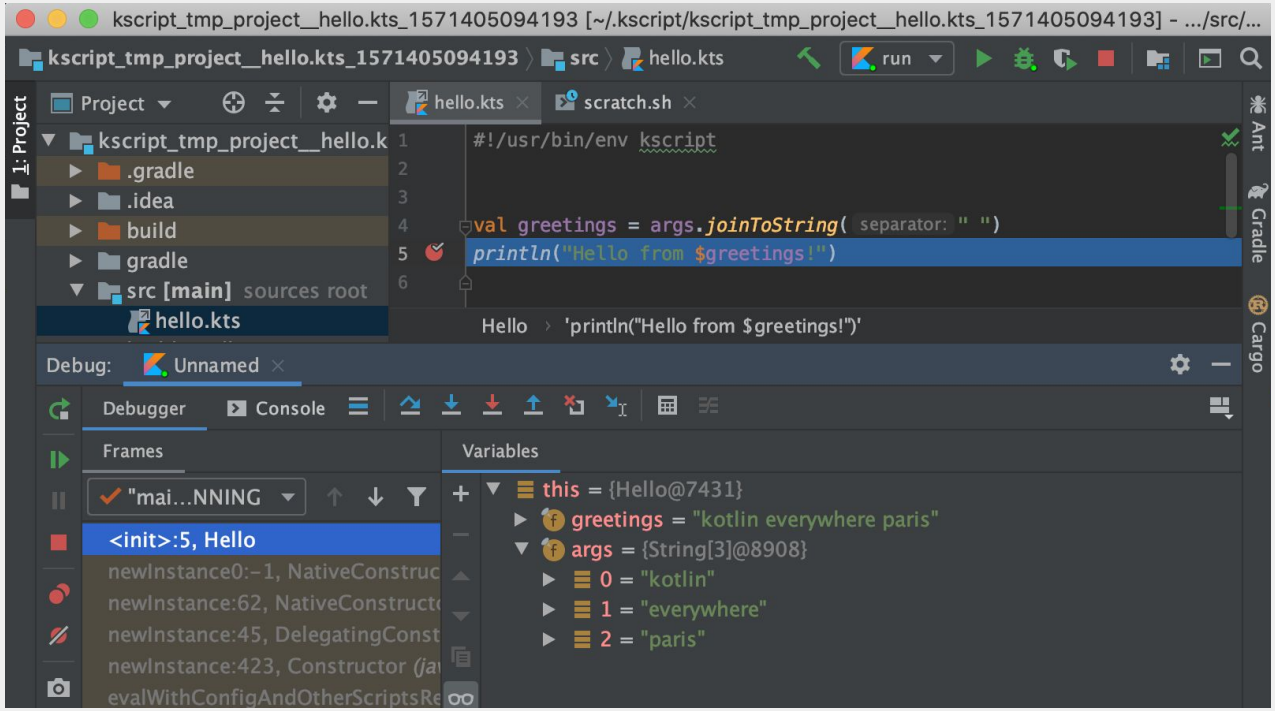
```
$ chmod +x weekend.kts

$ ./weekend.kts
It is the weekend!
```

Kscript - IDE




```
$ kscript --idea weekend.kts
```

Kscript - Debugger



Kscript - Real life examples

- Generating project website (mkdocs + github pages)
- Install scripts
- Migration from `build.gradle` to `build.gradle.kts`
- Finding duplicates in `string.xml`
- Categorizing my expenses !
- etc...

Scripting - limitations

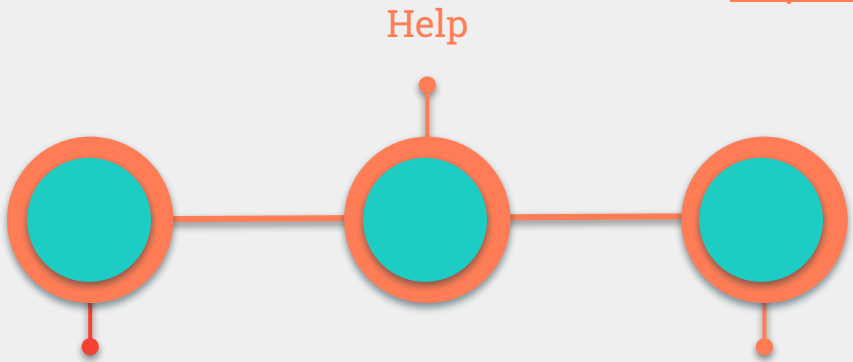
- JVM required
- JVM startup time
- Multiple files is hard to maintain
- No Gradle => no plugins, kapt, etc...

Kinta: a Kotlin Cli

CLIKT : Presentation

- ➔ Open Source library
- ➔ Command Line Interface for Kotlin

https://github.com/ajalt/clikt



Commands
customisable, nested commands

Argument/Option
composable, type safe, prompt, default

CLIKT to Kinta

- We need entry points for workflows then commands
- Provide a simple way to launch theses commands by anyone. (Command line interface)
- Reach even more platforms

Kinta CLI integration

- apply plugin: 'application'
- Create a jar
- Specify the ‘Main’ class
- Generate starting scripts

```
plugins {
    application
}

tasks.withType<Jar> {
    archiveFileName.set("kinta-cli.jar")
    manifest {
        attributes("Main-Class" to com.dailymotion.kinta.MainKt")
    }
    from(configurations.runtimeClasspath.get().map {
        if (it.isDirectory) it else zipTree(it)
    })
}

application {
    mainClassName = "com.dailymotion.kinta.MainKt"
}
```

build.gradle.kts

PublishPlayStore workflow

- What is a workflow?
- Workflow detail
  - Upload archive
  - Create a release on a specific track
  - Find a local changelog for the version
  - Upload the changelog

kinta publish beta --archiveFile=app-release.aab

PublishPlayStore workflow

```
object PublishPlayStore : CliktCommand(
    name = "publish", help = "Publish a version on the given track") {

  private val track by argument("--track", help = "The Play Store track")
  private val archiveFile by argument("--archiveFile")
  private val percent by option("--percent", help = "The user fraction receiving the update").double()

  override fun run() {

    // Beautiful code is coming...
  }
}
```

```
override fun run() {
    PlayStoreIntegration.uploadDraft(
        archiveFile = File(archiveFile),
        track = track
    )
    PlayStoreIntegration.createRelease(
        track = track,
        listVersionCodes = listOf(versionCode),
        percent = percentToApply
    )
    val changeLogs = LocalMetadataHelper.getChangelog(versionCode)

    PlayStoreIntegration.uploadWhatsNew(
        versionCode = versionCode,
        whatsNewProvider = changeLogs
    )
}
```

A Swiss knife

Builds
- nightly
- buildPR
- buildTag

Translations
- txPull
- txPush
- txPR

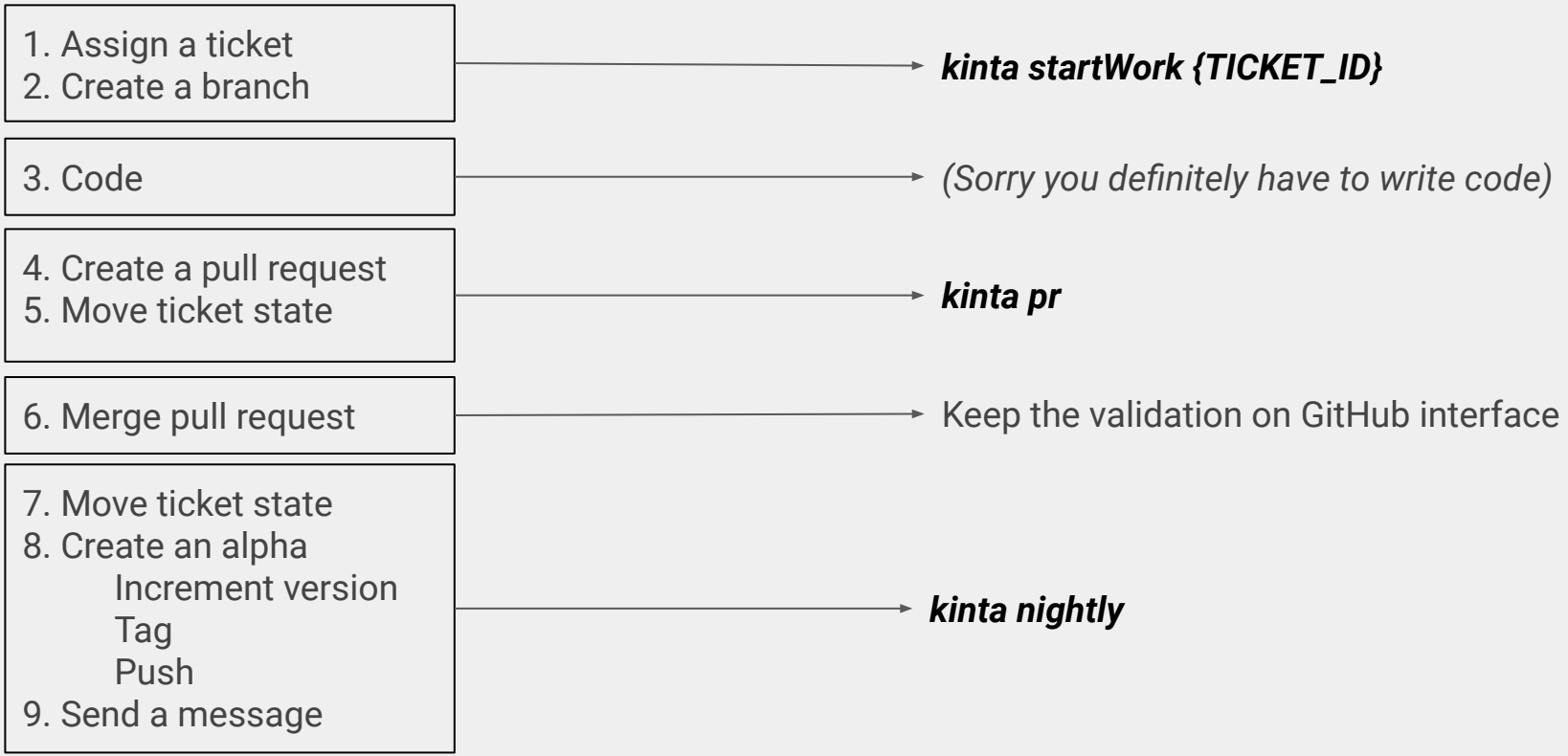
Git - tickets
- startWork
- PR

Common tools
- trigger
- branch
- hotfix
- cleanLocal
- cleanRemote

Play Store releases
- beta
- release

Play Store metadatas
- uploadWhatsNew
- uploadListing
- uploadScreenshots
- generateScreenshots

The daily work becomes simpler!



What's next

Kinta - customization

- Make the kinta tool usable outside Dailymotion
- 3rd party services have a well defined API…
- … but every organization has their own processes and workflows.
- There’s a fine line between customization and reuse

Kinta - Integration and Workflows

An Integration is:

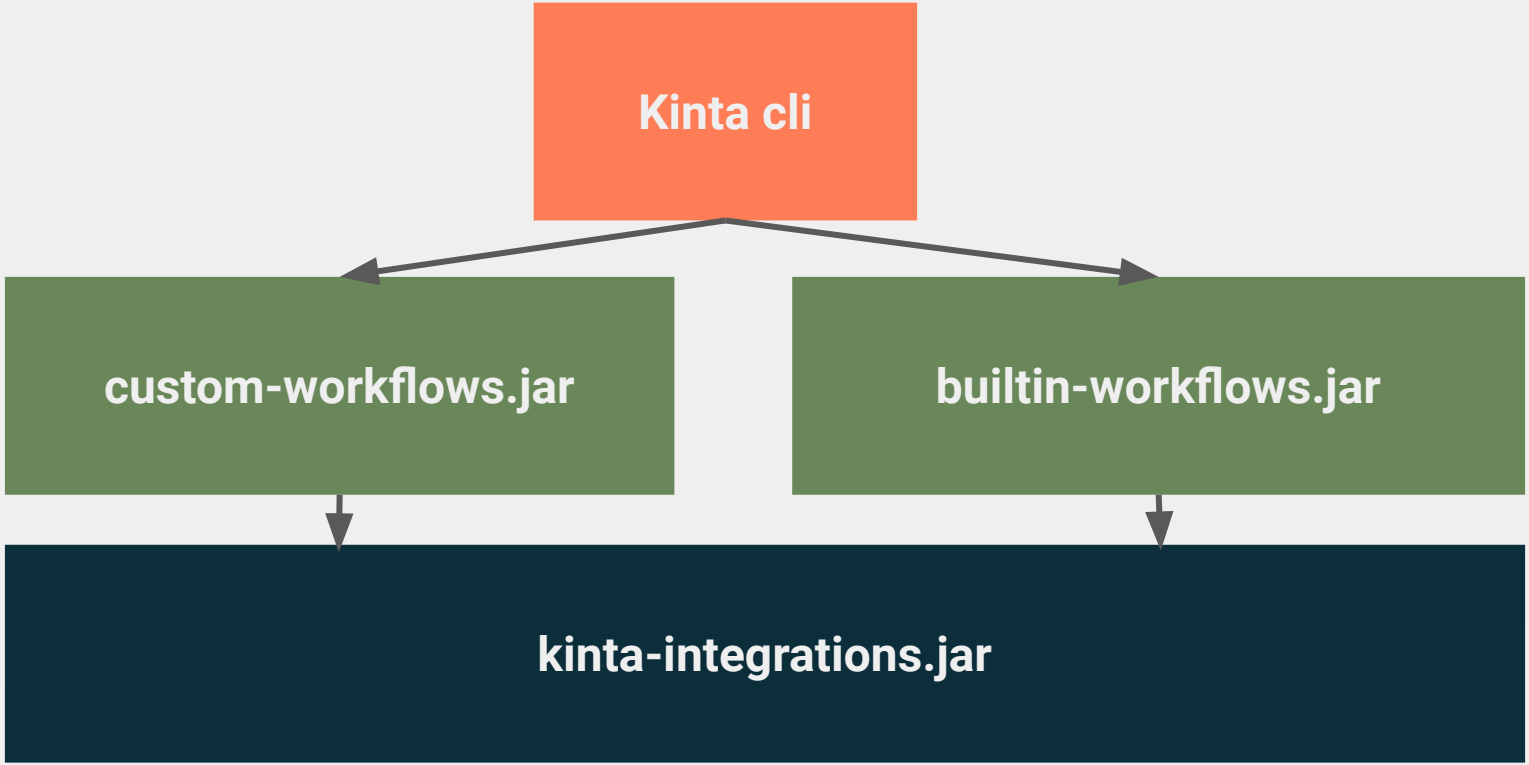
- A Kotlin class linked to a specific domain:
  - Github
  - Transifex
  - etc...
- Highly reusable
- Redistributed
- Static
  - It doesn't change often
- Composed of atomic methods
- Documented using Kdoc
- Inside the redistributed **kinta-integrations** artifact

A Workflow is:

- A Clikt command for a specific complex task:
  - Publish a release
  - Create a Translation PR
  - etc...
- Inside the host project

- Most of the times specific to the host project
- **Uses integrations** to accomplish complex tasks
- Documented using clikt

Kinta - Custom workflows



What's next

- Figuring out a way to distribute the kinta binary
- Also distribute the backend/webapp that hosts artifacts

- https://github.com/dailymotion/kinta
- Feedbacks welcome
- Disclaimer: it's still very early stage and things may break

Thanks.