Refactoring Critical Paths with Kotlin Scientist

John Li

@tbfJohn

in/JohnLeeroy

Overview

- What is critical code?
- The challenges of updating critical code
- GitHub and Scientist
- Validating correctness, performance, and error handling
- Practice integrating Scientist in the Codelab
- Course Summary

What is Critical Code?

Critical Code is ...

- Any piece of software that is required to operate your business
- High business significance
- Moderate or high complexity

Updating Critical Code can be challenging

Updating code can is risky

- Introduce a crash
- Introduce a bug
- Degrade performance

Updating code can is time-consuming

- Update version number
- Build a release app
- Sign the app
- Upload the app
- Start the staged-rollout
- App is rolled out to 100% of users

GitHub and Scientist

Branching by Abstraction

```kotlin
// old implementation
class PermissionRepository {
  fun isPullable(repository: Repository, user: User): Boolean {
     //...
    }
}
// new abstraction
interface PermissionRepository {
  fun isPullable(repository: Repository, user: User): Boolean
}
```

Branching by Abstraction

```
// old implementation
class PermissionRepositoryV1: PermissionRepository  {
  override fun isPullable(repository: Repository, user: User): Boolean
 {
     //old implementation
    }
}
```

Branching by Abstraction

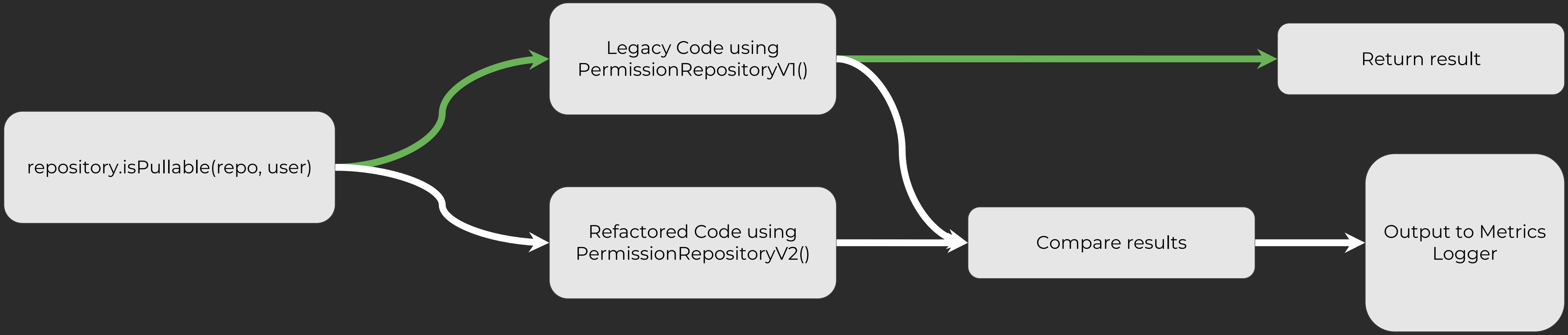
```
// old implementation
class PermissionRepositoryV1: PermissionRepository  {
  override fun isPullable(repository: Repository, user: User): Boolean
 {
      //old implementation
    }
}
// new abstraction
class PermissionRepositoryV2: PermissionRepository  {
  override fun isPullable(repository: Repository, user: User): Boolean
 {
      //new implementation
    }
}
```

Branching by Abstraction

```
var permissionRepo: PermissionRepository  =
  if(featureFlagRepo.isUsingNewUserPermissionLogic) {
    PermissionRepositoryV1()
  } else {
    PermissionRepositoryV2()
  }
```

Limitations of Branch by Abstraction

- correctness isn’t measured
- performance isn’t measured
- error-handling can result in different side-effects


repository.isPullable(repo, user)
Legacy Code using PermissionRepositoryV1()
Refactored Code using PermissionRepositoryV2()
Return result
Compare results
Output to Metrics Logger

Kotlin Scientist

Up Next

droidcon academy

Email Validator Example

- We have a regex pattern that we sourced from StackedOverflow to validate emails
- We want to migrate to android.util.Patterns implementation

Setting up Scientist

```
val scientist = scientist<Boolean, Unit> {
   publisher { result -> Log.d(“experiment”, result) }
}
```

droidcon academy

Conducting an Experiment

```
val experiment = experiment<Boolean, Unit> {
   name { experimentName }
   control {
       regexEmailValidator.isValidEmail(email)
   }
   candidate {
       textUtilEmailValidator.isValidEmail(email)
   }
}
val isValidEmail = scientist conduct experiment
```

Codelab Intro

Up Next

Scientist Extensions

- Use scientists in your unit tests
- Use Scientist with Firebase Performance Monitoring or other APM
- Log experiment results to a remote data store
  - Observe the experiment over time
  - Fix edge case scenarios in production with reported errors

Course Summary

Up Next

Course Summary

- Critical code is code that …
  - have high domain significance
  - medium or high complexity
- Refactoring critical code can be risky
- Refactoring critical code can be slow
- Scientist can help you measure
  - correctness
  - performance
  - error handling