Revolut

UNLOCKING HYPER-GROWTH: CHALLENGES (AND SOLUTIONS) FOR MASSIVE CODEBASES

PLSwift 2022

2022 Revolut

Agenda

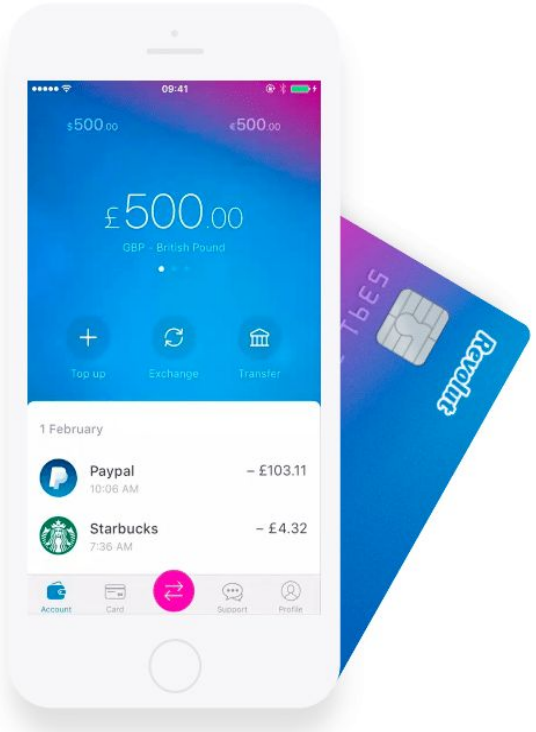
- Revolut: a growth story
- 1 - Increase of local build times
- 2 - Increase of CI build times
- 3 - Tooling gets out of control
- Module layout
- Conclusion

Revolut: a growth story

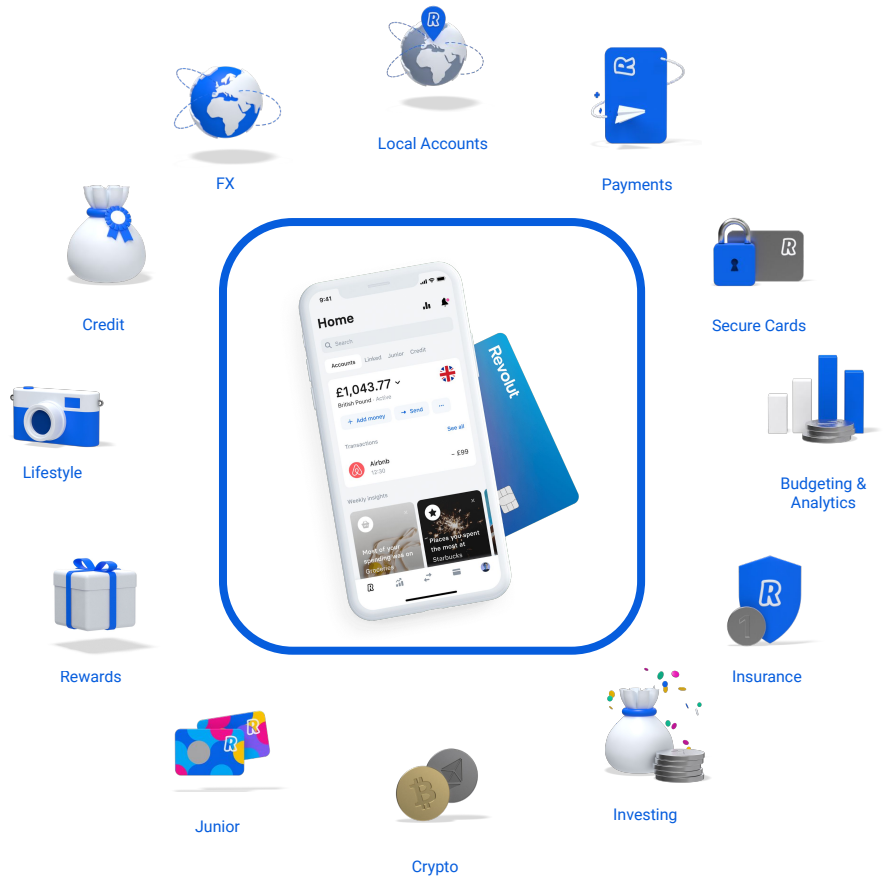
2022 Revolut

What is Revolut?

2015



2022



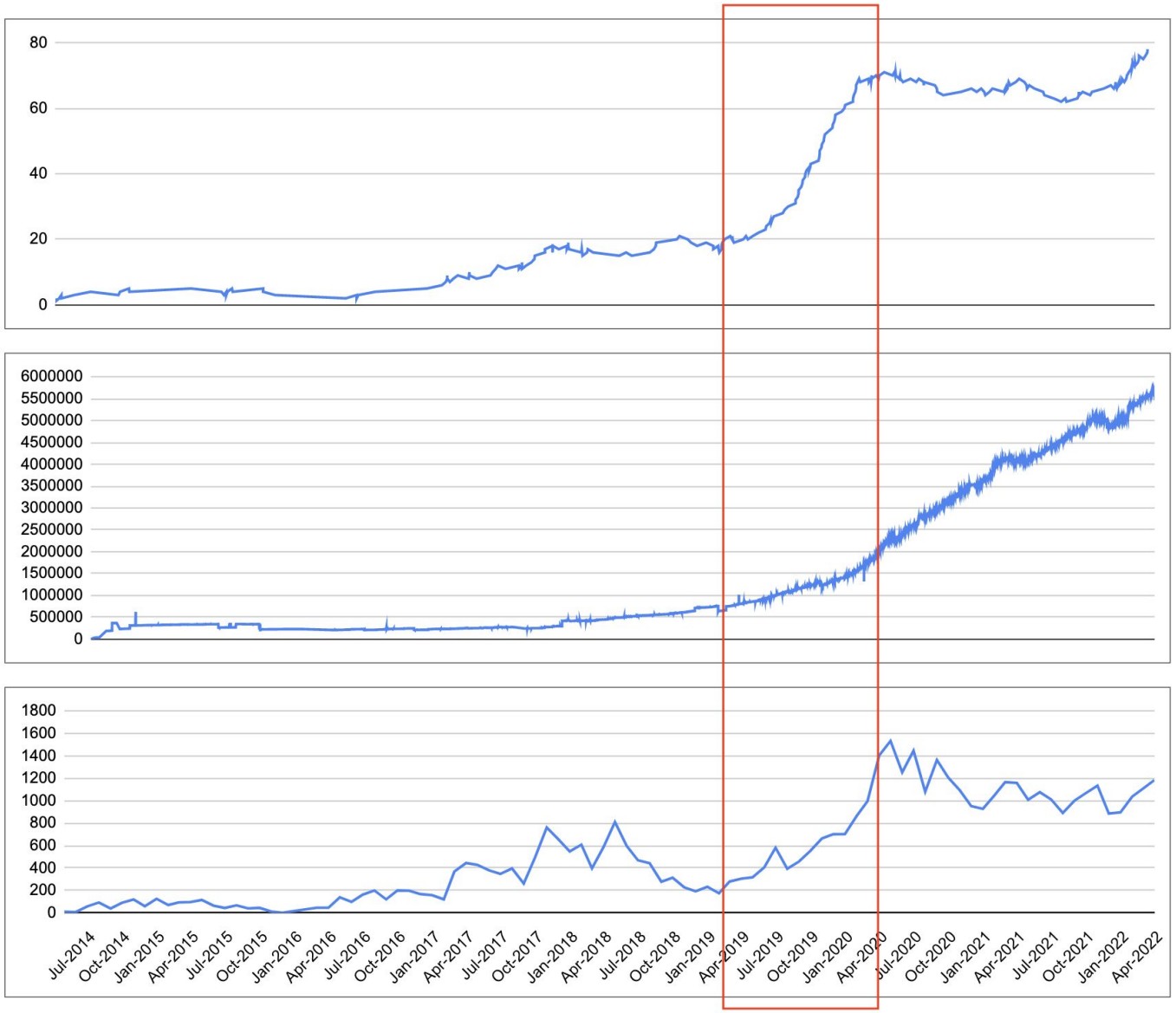
18M users
36 countries
90 iOS developers

Repository statistics

Amount of developers

Amount of lines of code

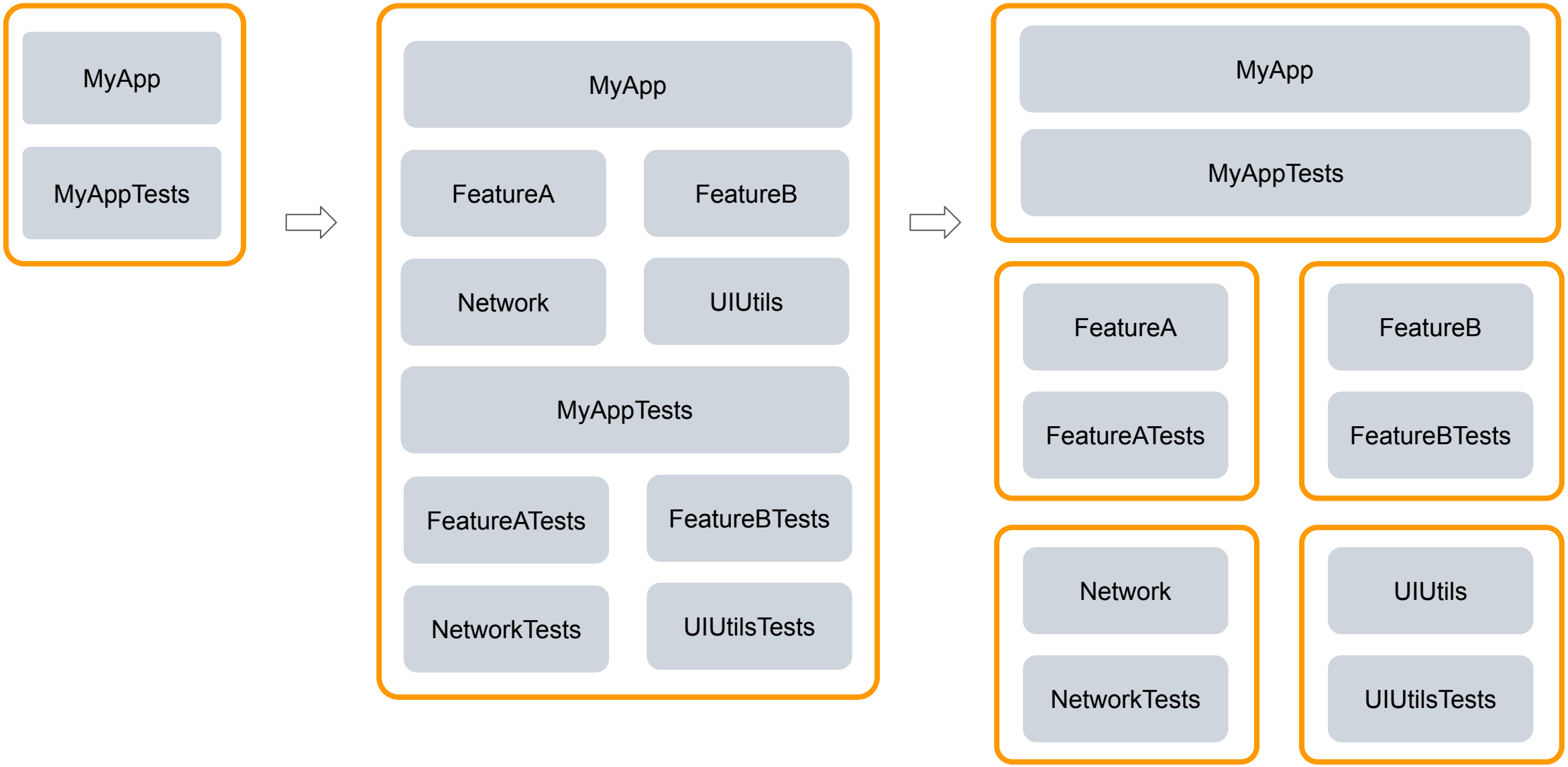
Amount of commits



1 - Increase of local build times

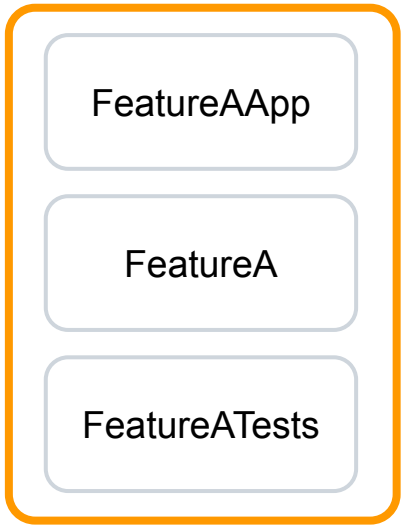
2022 Revolut

Modularization

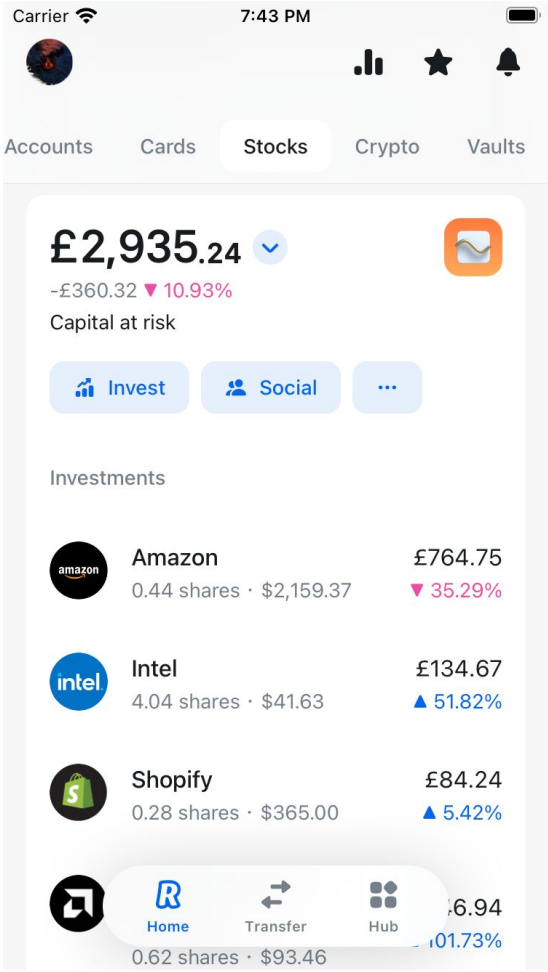


2022 Revolut

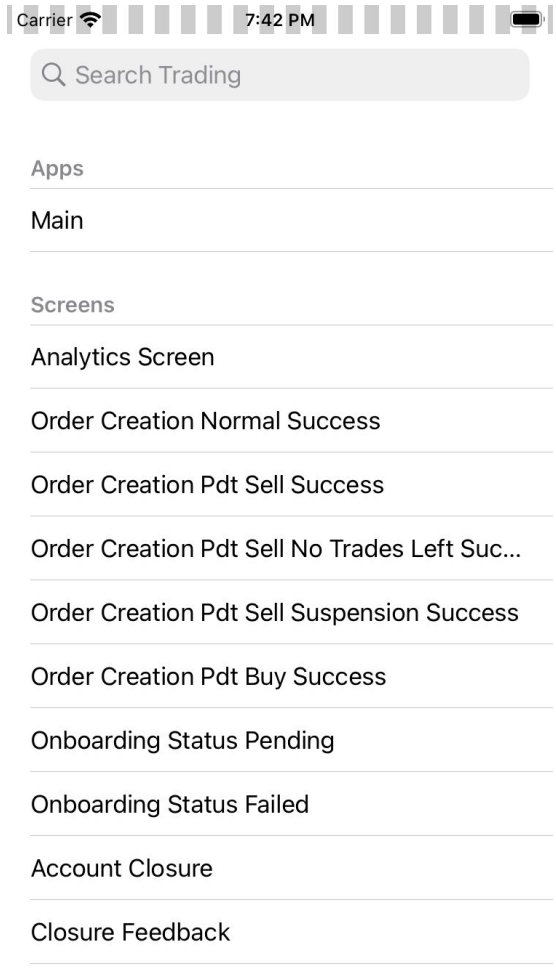
The importance of sample apps

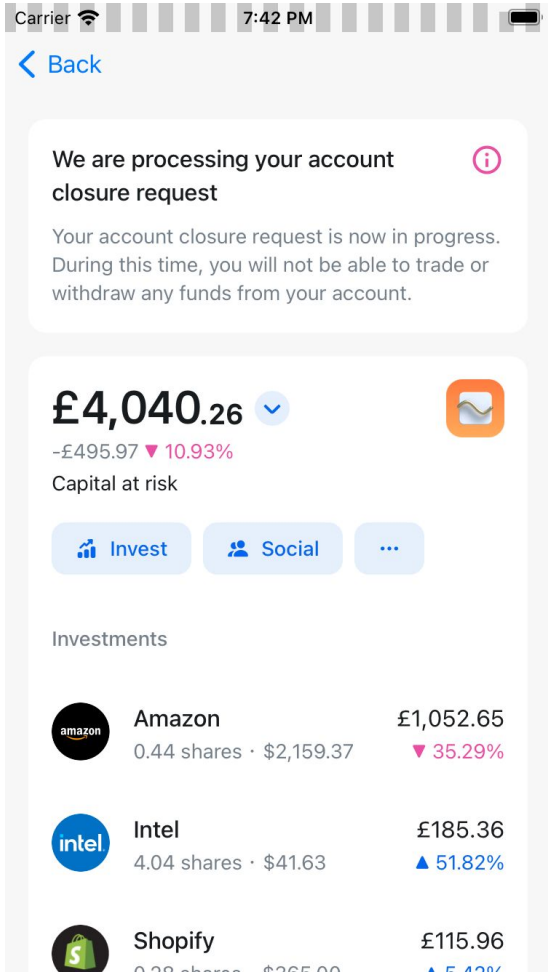


Revolut retail app



Trading sample app





Monorepo VS multirepo

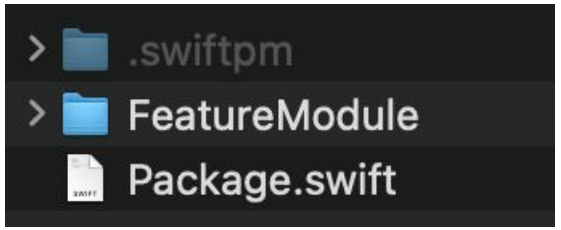
Multirepo

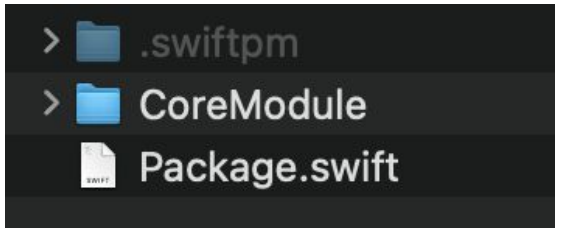
- Well known in the open-source community
- Clear ownership of the code
- Requires versioning
- Difficult integration
- Difficult to implement tooling and controls
- Difficult CI support
- Unsustainable at large scale

Monorepo

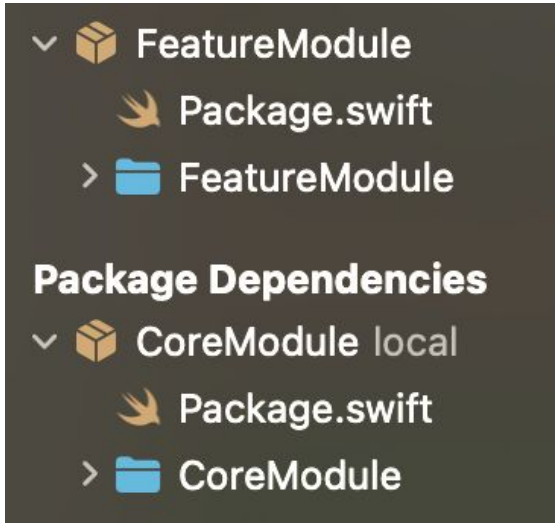
- Easy integration (no versioning)
- Easy CI setup
- Easy implementation of tooling and controls
- Ideal for build systems focussed on large scale
- Not well known
- Requires ownership definition

Integration of local modules: SPM



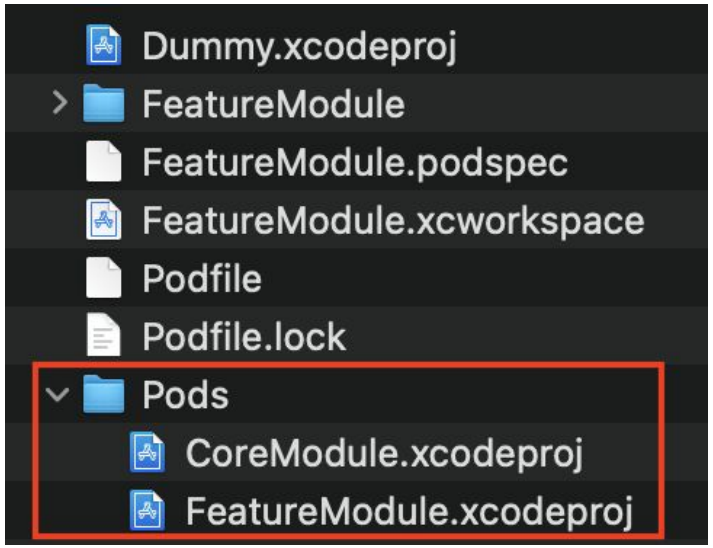


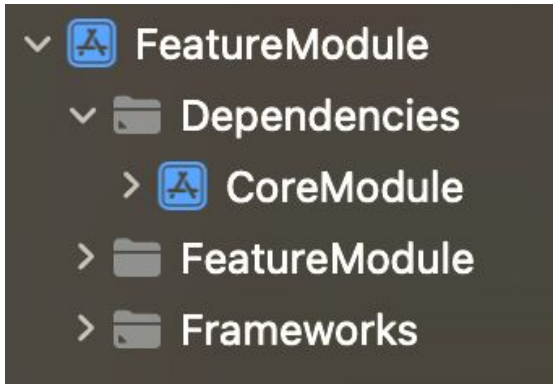

```swift
// swift-tools-version:5.5

import PackageDescription

let package = Package(
    name: "FeatureModule",
    products: [
        .library(
            name: "FeatureModule",
            targets: ["FeatureModule"]
        ),
    ],
    dependencies: [
        .package(name: "CoreModule", path: "../CoreModule")
    ],
    targets: [
        .target(
            name: "FeatureModule",
            dependencies: [
                .product(name: "CoreModule", package: "CoreModule")
            ],
            path: "FeatureModule"
        ),
    ]
)
```



Integration of local modules: CocoaPods

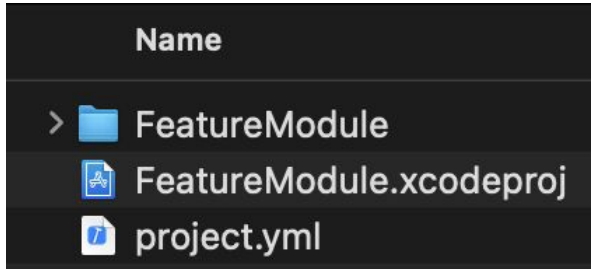
```ruby
Pod::Spec.new do |spec|
    spec.name = 'FeatureModule'
    spec.version = '1.0.0'
    spec.summary = 'FeatureModule'
    spec.description = 'FeatureModule'
    spec.homepage = 'https://revolut.com'
    spec.source = { :git =>'https://revolut.com' }
    spec.license = { :type => 'Proprietary' }
    spec.author = { 'Revolut' => 'something@revolut.com' }
    spec.swift_version = '5.0'

    spec.ios.deployment_target = '13.0'

    spec.source_files = [
        'FeatureModule/**/*.swift',
    ]
    spec.dependency 'CoreModule'
end
```

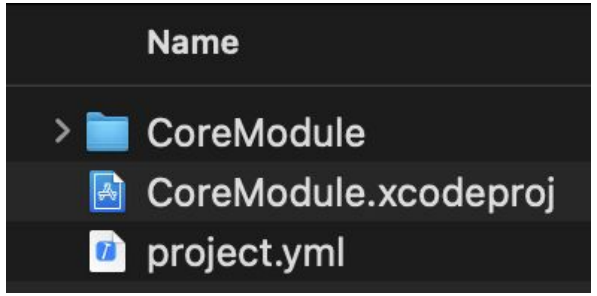




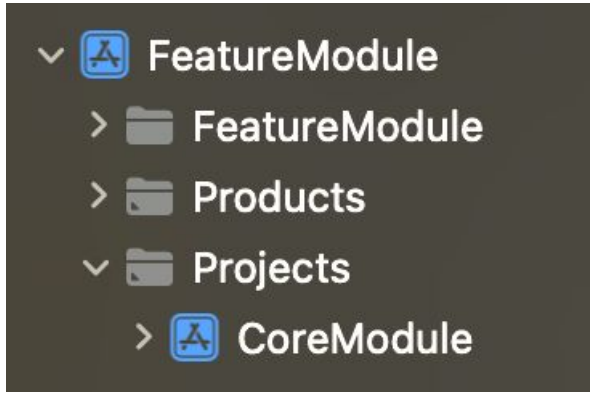

```ruby
platform :ios, '13.0'

install! 'cocoapods', :generate_multiple_pod_projects => true

project 'Dummy.xcodeproj'
workspace 'FeatureModule.xcworkspace'

pod 'FeatureModule', :path => '.'
pod 'CoreModule', :path => '../CoreModule'
```

Integration of local modules: Xcodegen






```yaml
name: FeatureModule

projectReferences:
  CoreModule:
    path: ../CoreModule/CoreModule.xcodeproj

targets:
  FeatureModule:
    platform: iOS
    type: framework
    sources: FeatureModule
    dependencies:
      - target: CoreModule/CoreModule
```



2 - Increase of CI build times

2022 Revolut

CI hosting options

Third party hosted virtual machines

- Slow

GitHub

Azure Pipelines

Third party hosted barebone machines

- Fastest
- Reproducible environment
- Expensive

aws

MacStadium

Self hosted barebone machines

- Fastest
- Full control
- Most affordable
- Non reproducible environment
- Setup complexity
- Maintenance

CI hosting in Revolut

- 12 M1 machines, 8 intel machines (~60 PRs per day, 90 developers)
- Setup performed with a script (600 loc)
- Most problematic during maintenance: OS updates
- Importance of NOT installing tools globally
  - Ruby tools: use bundler and keep installation local
  - Other tools: binaries stored locally to the repo

apple 3/20	
LDN-Bermondsey-M1	idle
LDN-Big-Ben	idle
LDN-Buckingham	idle
LDN-Camden-M1	idle
LDN-Canada-Water-M1	idle
LDN-Canary-Wharf-M1	idle
LDN-Chelsea	idle
LDN-Chinatown-M1	idle
LDN-Cutty-Sark	idle
LDN-Greenwich-M1	idle
LDN-Hackney-Wick-M1	
LDN-Harrods	idle
LDN-Kew-Gardens	idle
LDN-London-Eye-M1	idle
LDN-Paddington	idle
LDN-Poplar-M1	idle
LDN-Shoreditch-M1	idle
LDN-Stratford-M1	
LDN-The-Shard	idle
LDN-Westminster-M1	

.bundle > config

```
---
BUNDLE_PATH: "vendor/bundle"
```

Reducing CI build times: prebuilding third parties

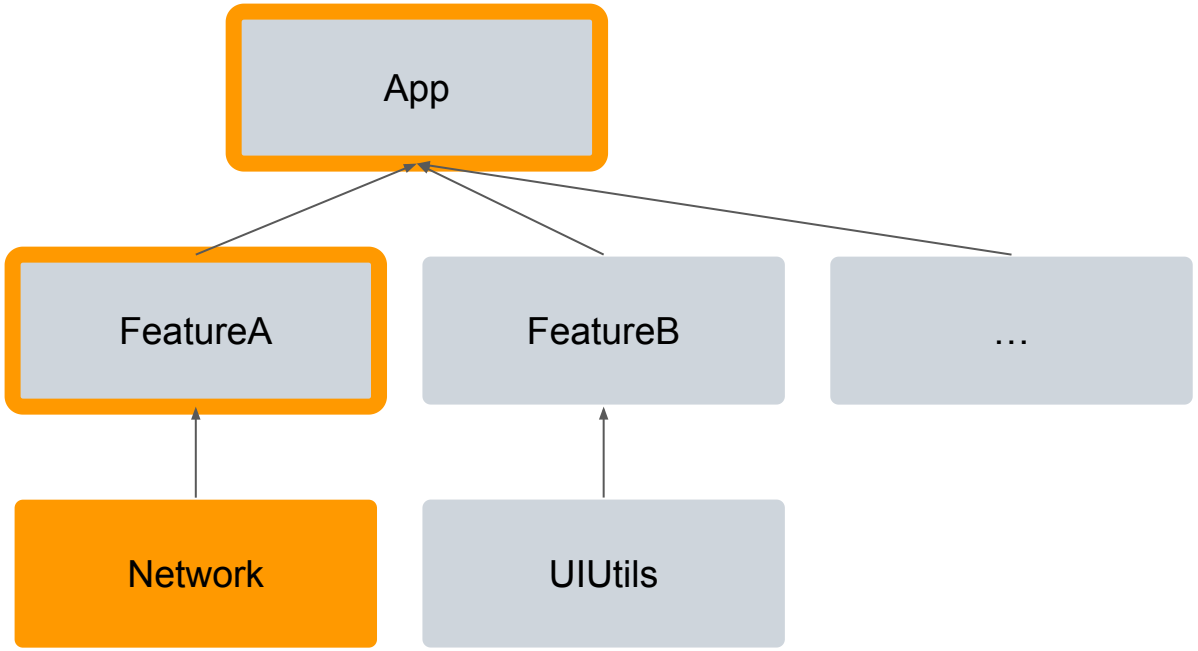
- Create a standard cocoapods project
- Add all dependencies to the Podfile
- Create a shared scheme with all pods
- Build shared scheme with Carthage

```ruby
def make_shared_scheme(installer)
    scheme = Xcodeproj::XCScheme.new()

    installer.pods_project.targets.each do |target|
        if !target.name.start_with?("Pods")
            scheme.add_build_target(target)
        end
    end

    scheme_name = File.basename(installer.aggregate_targets.first.user_project.path, '.*')
    scheme.save_as(installer.pods_project.path, scheme_name)
    puts "Created shared scheme #{scheme_name}"
end
```

```
carthage build --use-xcframeworks --no-use-binaries --platform iOS --no-skip-current
```

- Hosting: submodules or cloud storage
- Integration: your tool of choice

Reducing CI build times: importance of caching



- Leverage local cache
- Leverage remote cache
  - Make build deterministic
- Cache test results
  - Detect flaky tests

Reducing CI build times: bazel

	Before Bazel (June 2020)	Today (125% more code)
Modules	110	260
Swift files	20K	38K
LOC	2M	4.15M
CI testing tool	Xcodebuild (unreliable builds)	Bazel (more reliable builds)
Avg CI test time	22min	11min

3 - Tooling gets out of control

2022 Revolut

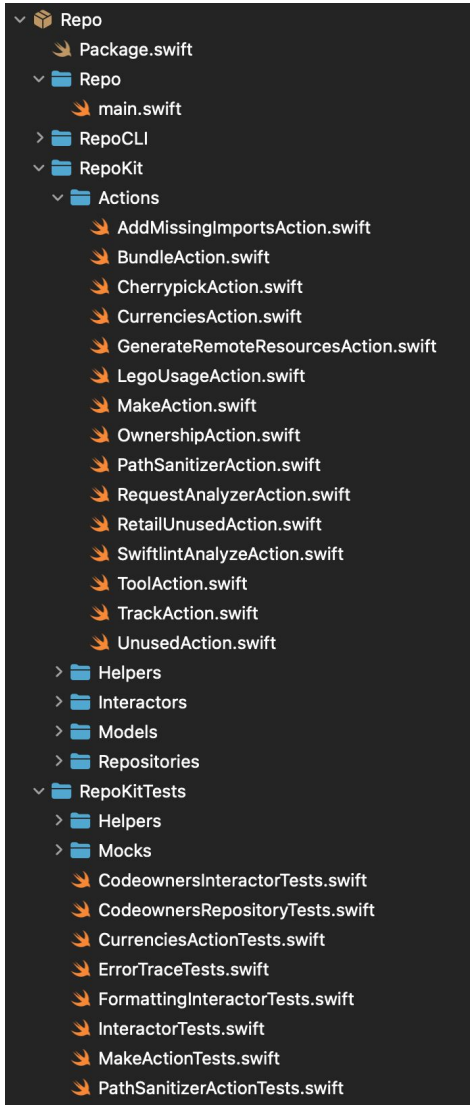
Create a stable API for recurring tasks

- Able to run commands from any location
- Stable and documented API
- Hide internal implementation

```
class RevolutRepo < Formula
  desc "The entrypoint for the revolut repository CLI"
  version '1.0.0'

  url "file:///dev/null"

  def install
    (bin + "repo").write(
      %{
#!/bin/zsh

set -euo pipefail

bundle exec repo $@
      }
    )
  end
end
```

```
desc(
  "proj",
  "Generate and open an xcode project for the module/app in the current path"
)
CLI.clean_option
def proj()
  if options[:clean]
    makefile("proj_clean")
  end

  makefile("proj")
end
```

Write the scripts in Swift

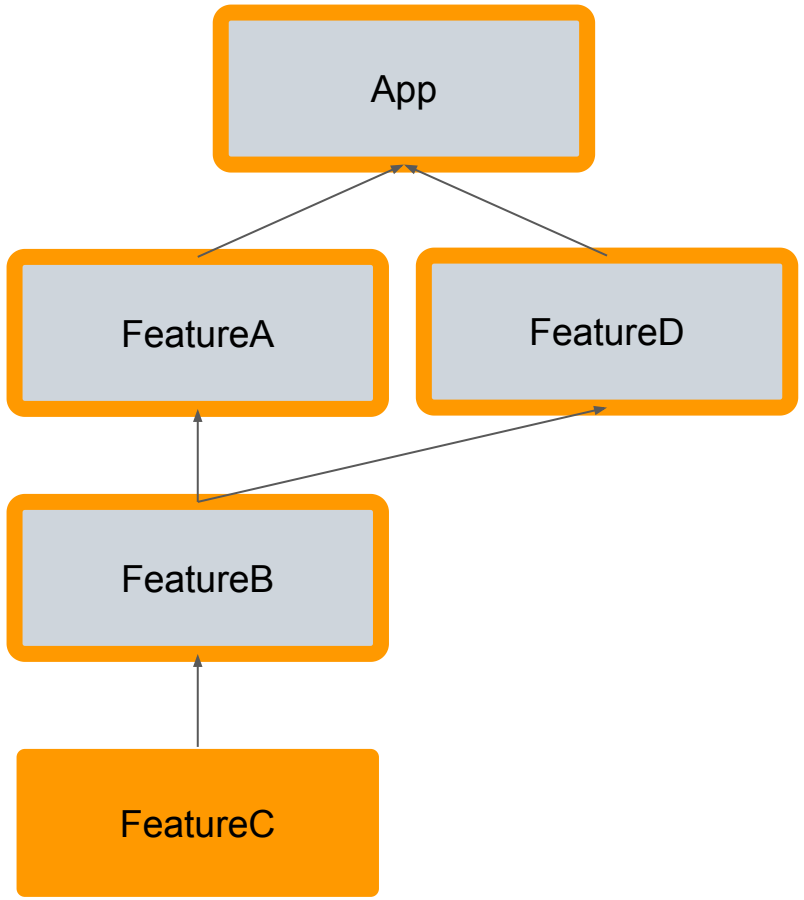


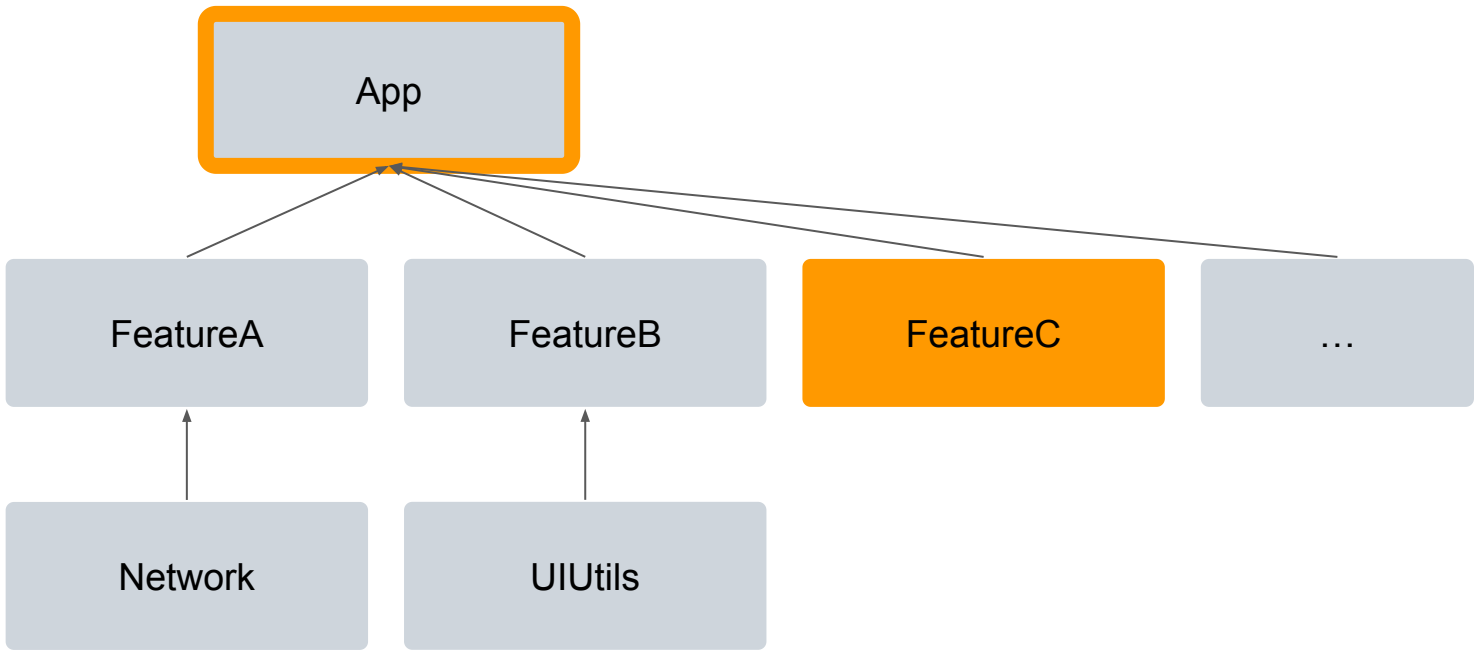
- One unique language and location for all scripts
- Known language and development platform
- Type safety
- Testability
- Unlock implementation of more complex tooling

Module layout

2022 Revolut

The importance of an horizontal dependency tree





Conclusions

2022 Revolut

Takeaways

- Reduce local build times: modularize
- Reduce CI build times: M1 barebone machines + bazel
- Support the infrastructure: use Swift as the main scripting language
- Support further growth: horizontal dependency tree + introduce cache for build and test

Revolut

Questions