Revolut

Migrating Revolut-iOS to Bazel

BazelCon 2021

Agenda

- Context: status of the codebase
- How did we do it?
- Future work

Introduction: the team

Andres Cecilia

Anton Barbasevich

Diomidis Papas

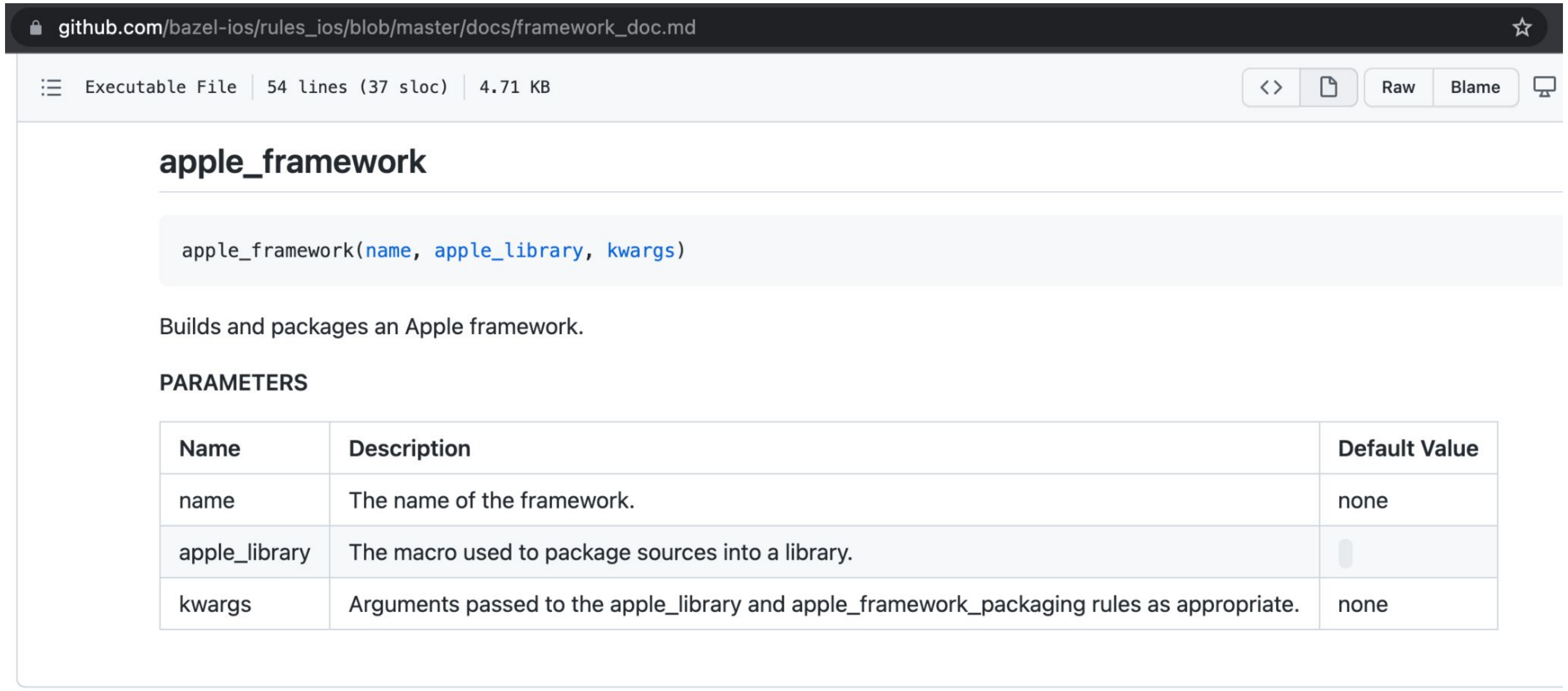
Konstantin Novikov

Context: status of the codebase

	Before Bazel (June 2020)	**Today (70% more code)**
Modules	110	188
Swift files	20K	31K
LOC	2M	3.4M
CI testing tool	Xcodebuild (unreliable builds)	Bazel (more reliable builds)
Avg CI test time	22min	11min
Modules integration	cocoapods	Bazel for CI + cocoapods for development
Module xcode projects	Manually maintained	Autogenerated

How did we do it?

- Initial spike
  - Buck or Bazel: did not support mixed objc and swift
  - Bazel + rules-ios: did support mixed objc and swift 🎉


github.com/bazel-ios/rules_ios/blob/master/docs/framework_doc.md
Executable File
54 lines (37 sloc)
4.71 KB
Raw
Blame

apple_framework

```
apple_framework(name, apple_library, kwargs)
```

Builds and packages an Apple framework.

PARAMETERS

Name	Description	Default Value
name	The name of the framework.	none
apple_library	The macro used to package sources into a library.	
kwargs	Arguments passed to the apple_library and apple_framework_packaging rules as appropriate.	none

How did we do it?

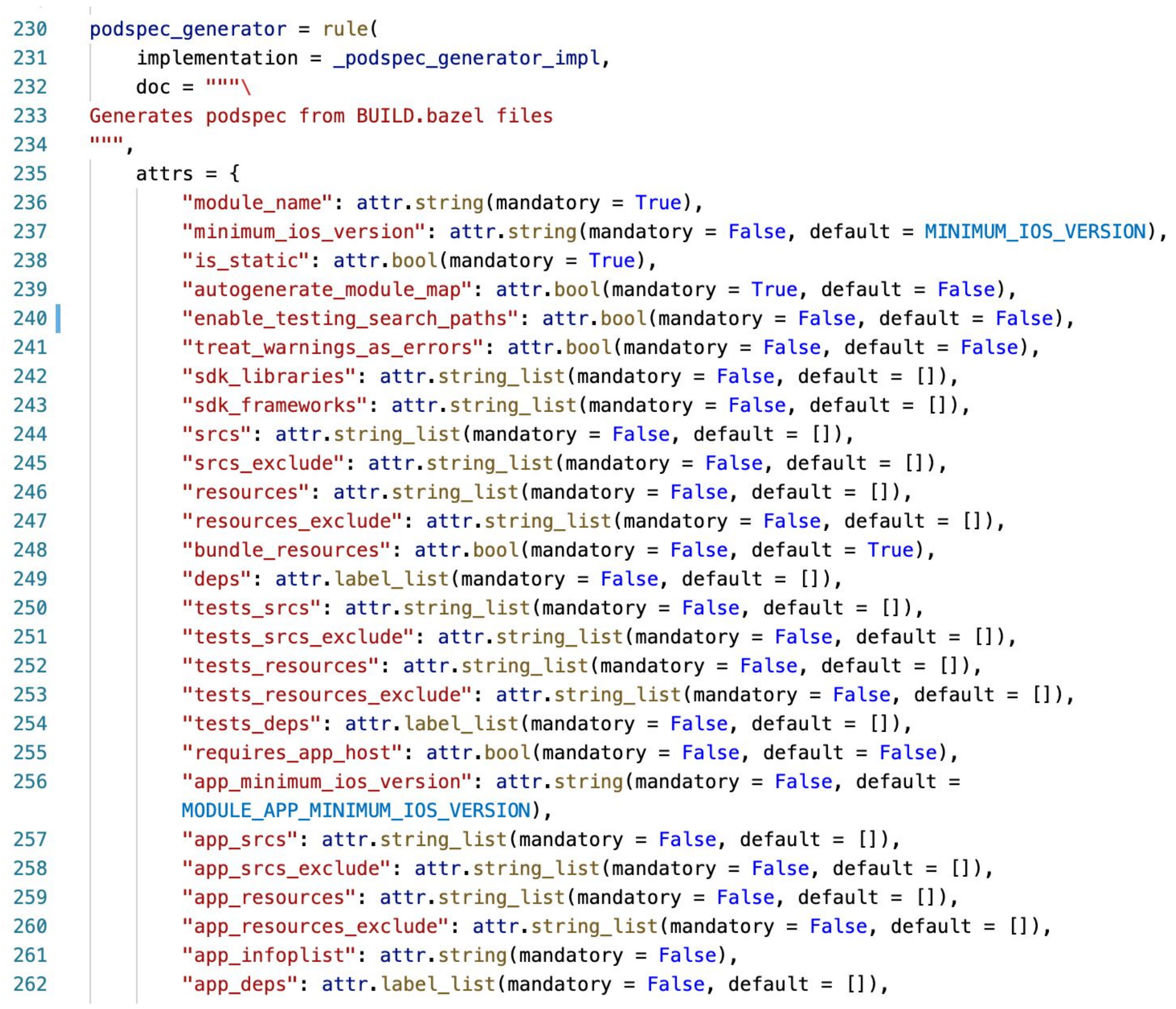
- Static/dynamic linkage support
  - Resources location changes depending on linkage
  - Required multiple code changes in order to access the bundle correctly

```swift
extension Bundle {
    public static func find<T: AnyObject>(for type: T.Type) -> Bundle {
        let currentBundle = Bundle(for: type.self)
        let bundleForStaticLinkage = currentBundle
            .url(forResource: String(describing: type), withExtension: "bundle")
            .flatMap(Bundle.init(url:))
        return bundleForStaticLinkage ?? currentBundle
    }
}
```

```swift
public extension Bundle {
    private final class Pandora {}

    static var pandora: Bundle {
        .find(for: Pandora.self)
    }
}
```

How did we do it?

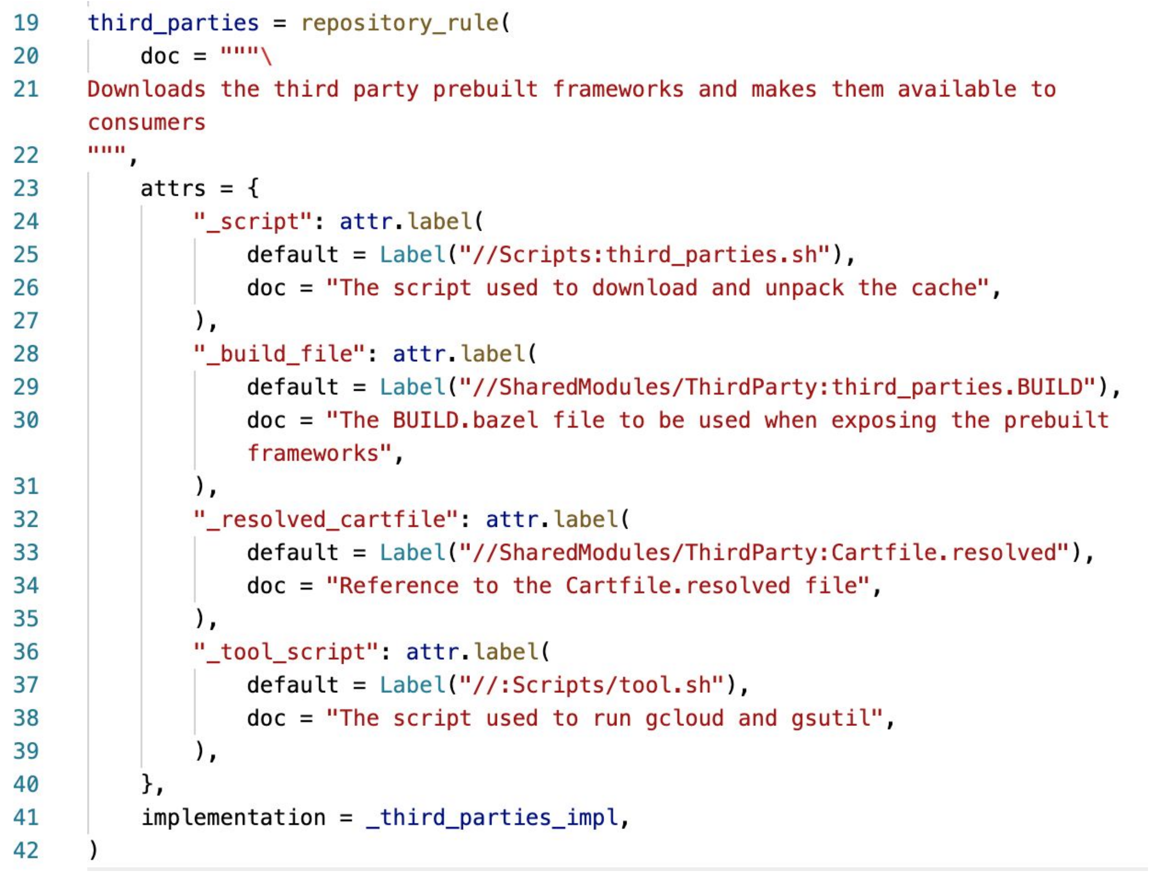
- BUILD.bazel files as the source of truth:
  - Generate podspecs using bazel: take advantage of bazel cache and parallelism
  - Allows migration without downtime, as existing cocoapods setup is unchanged
  - Currently: 30 seconds to generate 326 podspecs
  - CI check to make sure that generated podspecs are in sync with BUILD.bazel files



```
podspec_generator = rule(
    implementation = _podspec_generator_impl,
    doc = """\
Generates podspec from BUILD.bazel files
""",
    attrs = {
        "module_name": attr.string(mandatory = True),
        "minimum_ios_version": attr.string(mandatory = False, default = MINIMUM_IOS_VERSION),
        "is_static": attr.bool(mandatory = True),
        "autogenerate_module_map": attr.bool(mandatory = True, default = False),
        "enable_testing_search_paths": attr.bool(mandatory = False, default = False),
        "treat_warnings_as_errors": attr.bool(mandatory = False, default = False),
        "sdk_libraries": attr.string_list(mandatory = False, default = []),
        "sdk_frameworks": attr.string_list(mandatory = False, default = []),
        "srcs": attr.string_list(mandatory = False, default = []),
        "srcs_exclude": attr.string_list(mandatory = False, default = []),
        "resources": attr.string_list(mandatory = False, default = []),
        "resources_exclude": attr.string_list(mandatory = False, default = []),
        "bundle_resources": attr.bool(mandatory = False, default = True),
        "deps": attr.label_list(mandatory = False, default = []),
        "tests_srcs": attr.string_list(mandatory = False, default = []),
        "tests_srcs_exclude": attr.string_list(mandatory = False, default = []),
        "tests_resources": attr.string_list(mandatory = False, default = []),
        "tests_resources_exclude": attr.string_list(mandatory = False, default = []),
        "tests_deps": attr.label_list(mandatory = False, default = []),
        "requires_app_host": attr.bool(mandatory = False, default = False),
        "app_minimum_ios_version": attr.string(mandatory = False, default = MODULE_APP_MINIMUM_IOS_VERSION),
        "app_srcs": attr.string_list(mandatory = False, default = []),
        "app_srcs_exclude": attr.string_list(mandatory = False, default = []),
        "app_resources": attr.string_list(mandatory = False, default = []),
        "app_resources_exclude": attr.string_list(mandatory = False, default = []),
        "app_infoplist": attr.string(mandatory = False),
        "app_deps": attr.label_list(mandatory = False, default = []),
```

How did we do it?

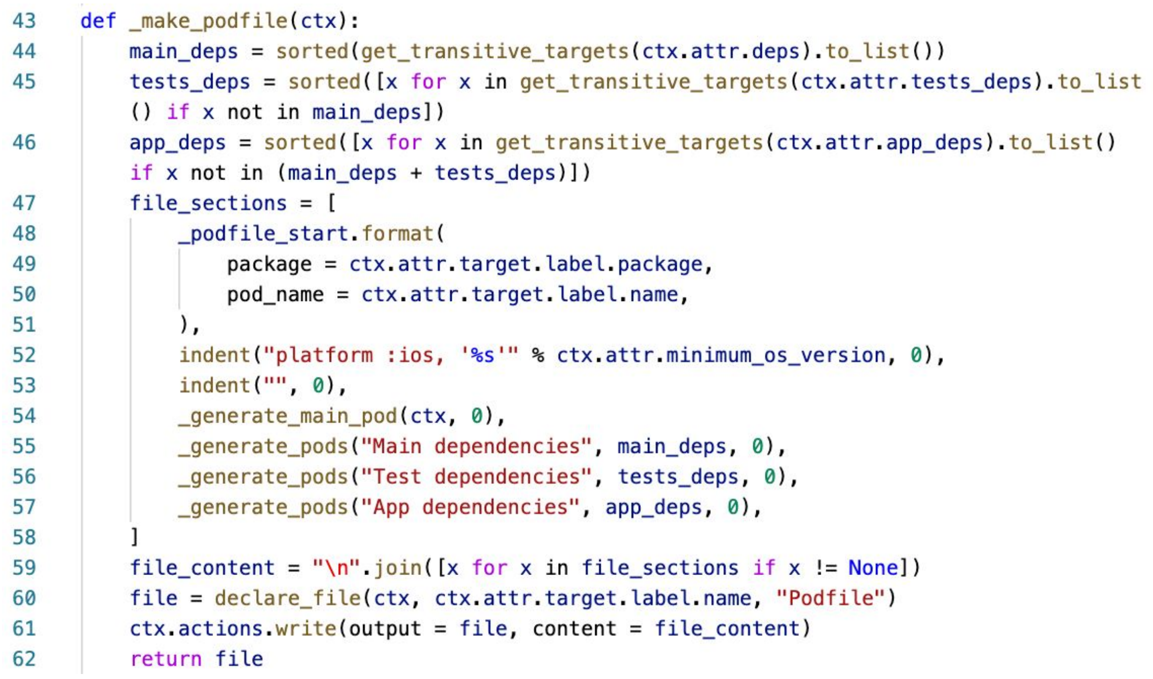
- Third parties integration:
  - Prebuild third parties using carthage, which are supported by both bazel and cocoapods
  - Support closed source binary dependencies from vendors
  - Upload binaries to google cloud
  - Download binaries using a repository rule, to take advantage of bazel cache



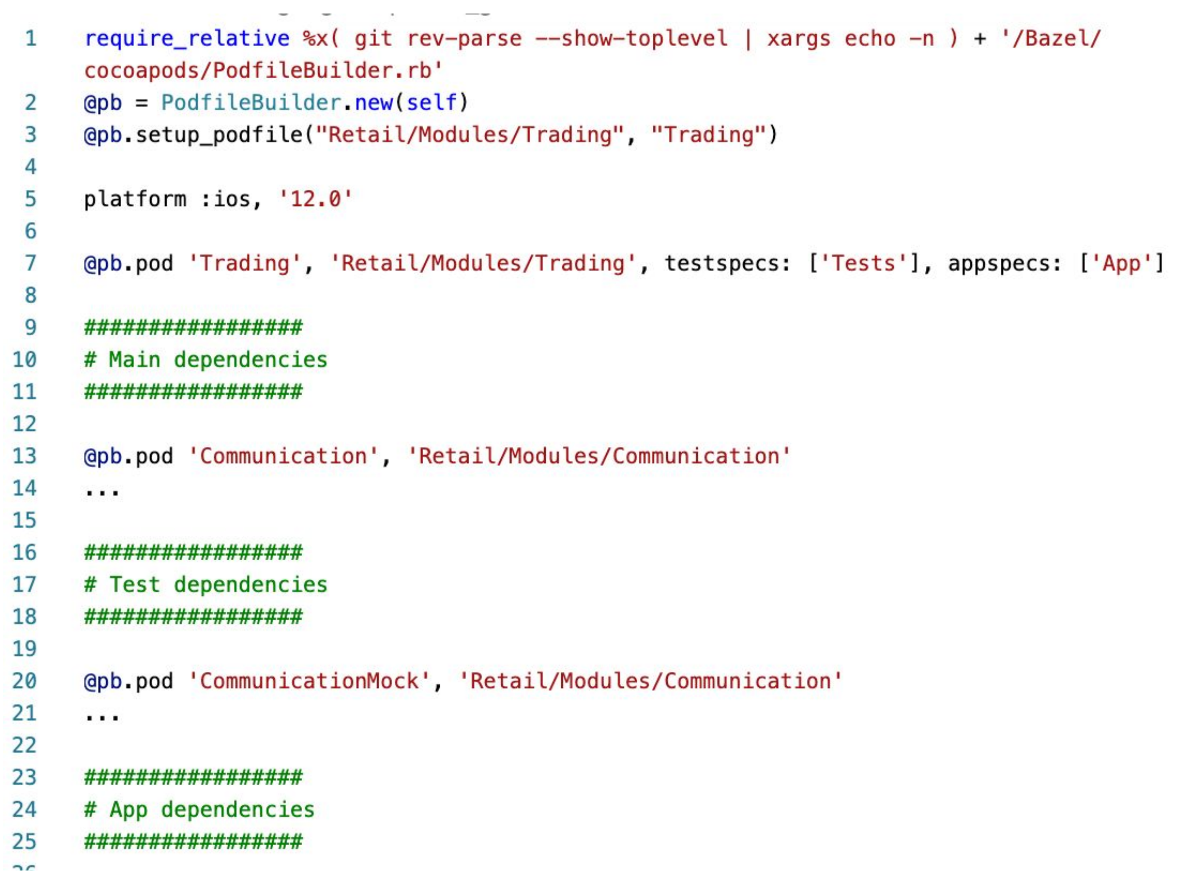
```
third_parties = repository_rule(
    doc = """\
Downloads the third party prebuilt frameworks and makes them available to consumers
""",
    attrs = {
        "_script": attr.label(
            default = Label("//Scripts:third_parties.sh"),
            doc = "The script used to download and unpack the cache",
        ),
        "_build_file": attr.label(
            default = Label("//SharedModules/ThirdParty:third_parties.BUILD"),
            doc = "The BUILD.bazel file to be used when exposing the prebuilt frameworks",
        ),
        "_resolved_cartfile": attr.label(
            default = Label("//SharedModules/ThirdParty:Cartfile.resolved"),
            doc = "Reference to the Cartfile.resolved file",
        ),
        "_tool_script": attr.label(
            default = Label("//:Scripts/tool.sh"),
            doc = "The script used to run gcloud and gsutil",
        ),
    },
    implementation = _third_parties_impl,
)
```

How did we do it?

- Generate project at module level
  - Modules have simple structure: main target, test target, sample app target
  - Generate podfile
  - Use cocoapods for xcode project generation



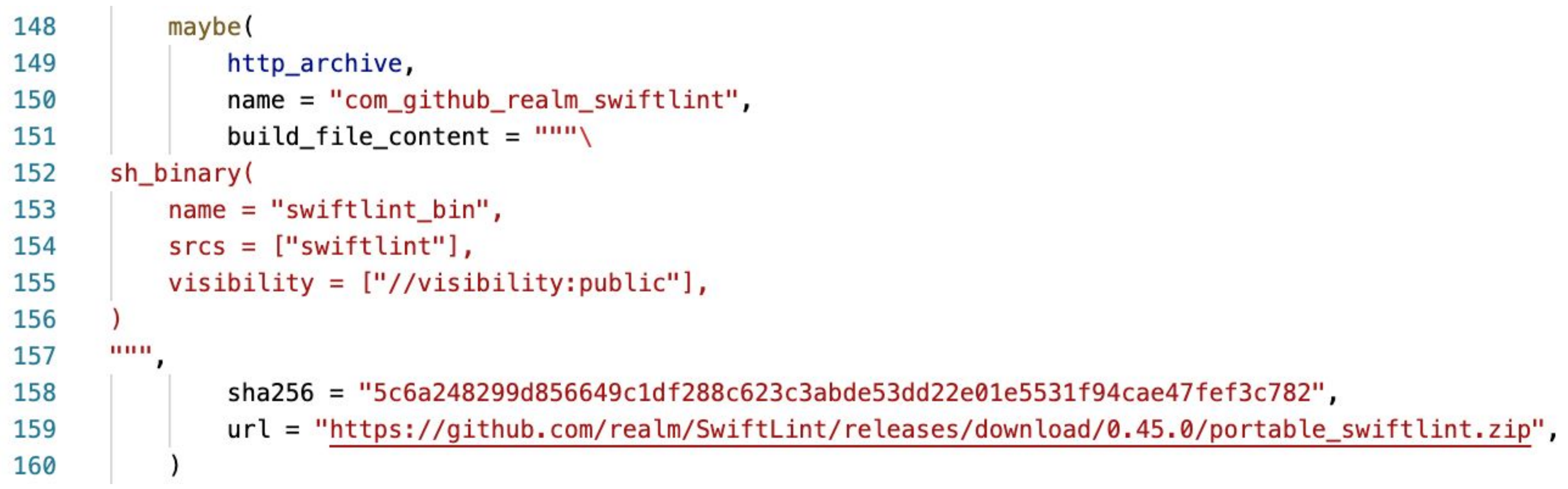
```python
def _make_podfile(ctx):
    main_deps = sorted(get_transitive_targets(ctx.attr.deps).to_list())
    tests_deps = sorted([x for x in get_transitive_targets(ctx.attr.tests_deps).to_list() if x not in main_deps])
    app_deps = sorted([x for x in get_transitive_targets(ctx.attr.app_deps).to_list() if x not in (main_deps + tests_deps)])
    file_sections = [
        _podfile_start.format(
            package = ctx.attr.target.label.package,
            pod_name = ctx.attr.target.label.name,
        ),
        indent("platform :ios, '%s'" % ctx.attr.minimum_os_version, 0),
        indent("", 0),
        _generate_main_pod(ctx, 0),
        _generate_pods("Main dependencies", main_deps, 0),
        _generate_pods("Test dependencies", tests_deps, 0),
        _generate_pods("App dependencies", app_deps, 0),
    ]
    file_content = "\n".join([x for x in file_sections if x != None])
    file = declare_file(ctx, ctx.attr.target.label.name, "Podfile")
    ctx.actions.write(output = file, content = file_content)
    return file
```



```ruby
require_relative %x( git rev-parse --show-toplevel | xargs echo -n ) + '/Bazel/cocoapods/PodfileBuilder.rb'
@pb = PodfileBuilder.new(self)
@pb.setup_podfile("Retail/Modules/Trading", "Trading")

platform :ios, '12.0'

@pb.pod 'Trading', 'Retail/Modules/Trading', testspecs: ['Tests'], appspecs: ['App']

#################
# Main dependencies
#################

@pb.pod 'Communication', 'Retail/Modules/Communication'
...

#################
# Test dependencies
#################

@pb.pod 'CommunicationMock', 'Retail/Modules/Communication'
...

#################
# App dependencies
#################
```

How did we do it?

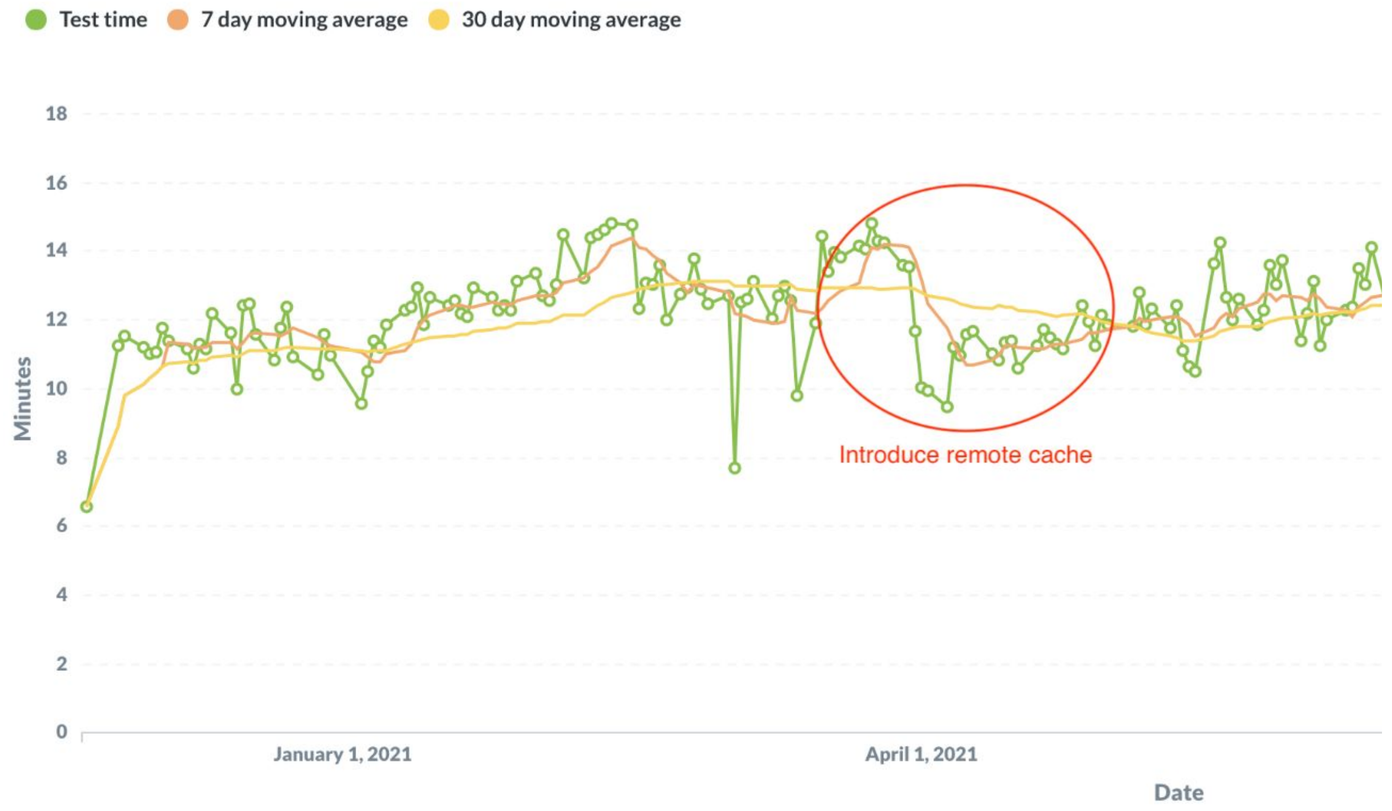
- Integration of tools with bazel
  - Unify installation and execution of the tool in one unique step
  - Avoid version mismatches (“it works in my machine”)



```
    maybe(
        http_archive,
        name = "com_github_realm_swiftlint",
        build_file_content = """\
sh_binary(
    name = "swiftlint_bin",
    srcs = ["swiftlint"],
    visibility = ["//visibility:public"],
)
""",
        sha256 = "5c6a248299d856649c1df288c623c3abde53dd22e01e5531f94cae47fef3c782",
        url = "https://github.com/realm/SwiftLint/releases/download/0.45.0/portable_swiftlint.zip",
    )
```

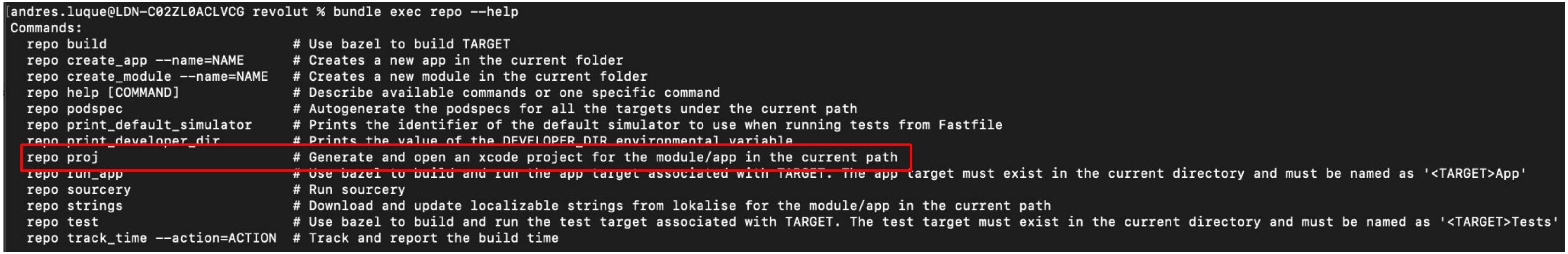
How did we do it?

- Introduction of remote cache
  - Compilation time decreased 50%
  - Total CI test time decreased from 14min to 11min



How did we do it?

- Introduction of repo CLI
  - Common stable API for developers
  - Hide implementation details
  - Able to run a command from anywhere in the repository
  - Bazel allows to write scripts in Swift: avoid chaos and increase reliability of tools/scripts



```
[andres.luque@LDN-C02ZL0ACLVCG revolut % bundle exec repo --help
Commands:
  repo build                        # Use bazel to build TARGET
  repo create_app --name=NAME       # Creates a new app in the current folder
  repo create_module --name=NAME    # Creates a new module in the current folder
  repo help [COMMAND]               # Describe available commands or one specific command
  repo podspec                      # Autogenerate the podspecs for all the targets under the current path
  repo print_default_simulator      # Prints the identifier of the default simulator to use when running tests from Fastfile
  repo print_developer_dir          # Prints the value of the DEVELOPER_DIR environmental variable
  repo proj                         # Generate and open an xcode project for the module/app in the current path
  repo run_app                      # Use bazel to build and run the app target associated with TARGET. The app target must exist in the current directory and must be named as '<TARGET>App'
  repo sourcery                     # Run sourcery
  repo strings                      # Download and update localizable strings from lokalise for the module/app in the current path
  repo test                         # Use bazel to build and run the test target associated with TARGET. The test target must exist in the current directory and must be named as '<TARGET>Tests'
  repo track_time --action=ACTION   # Track and report the build time
```

Future work

- Generate xcode projects that build with bazel under the hood
  - Blocker: incremental builds are faster with xcodebuild than bazel
- Release apps with bazel and drop cocoapods support
  - Blocker: difficulties to come up with a reliable set of bazel.rc configurations
  - No immediate need for this, as time spent on release jobs is not critical
- Improve bazel caching by flattening the dependency tree, for faster builds
- Reduce amount of times that cache gets invalidated due to changes in core UI module
  - Blocker: no clear solution yet

Revolut

Thanks