

# Разработка CLI инструментов на Swift

Игорь Шамрин

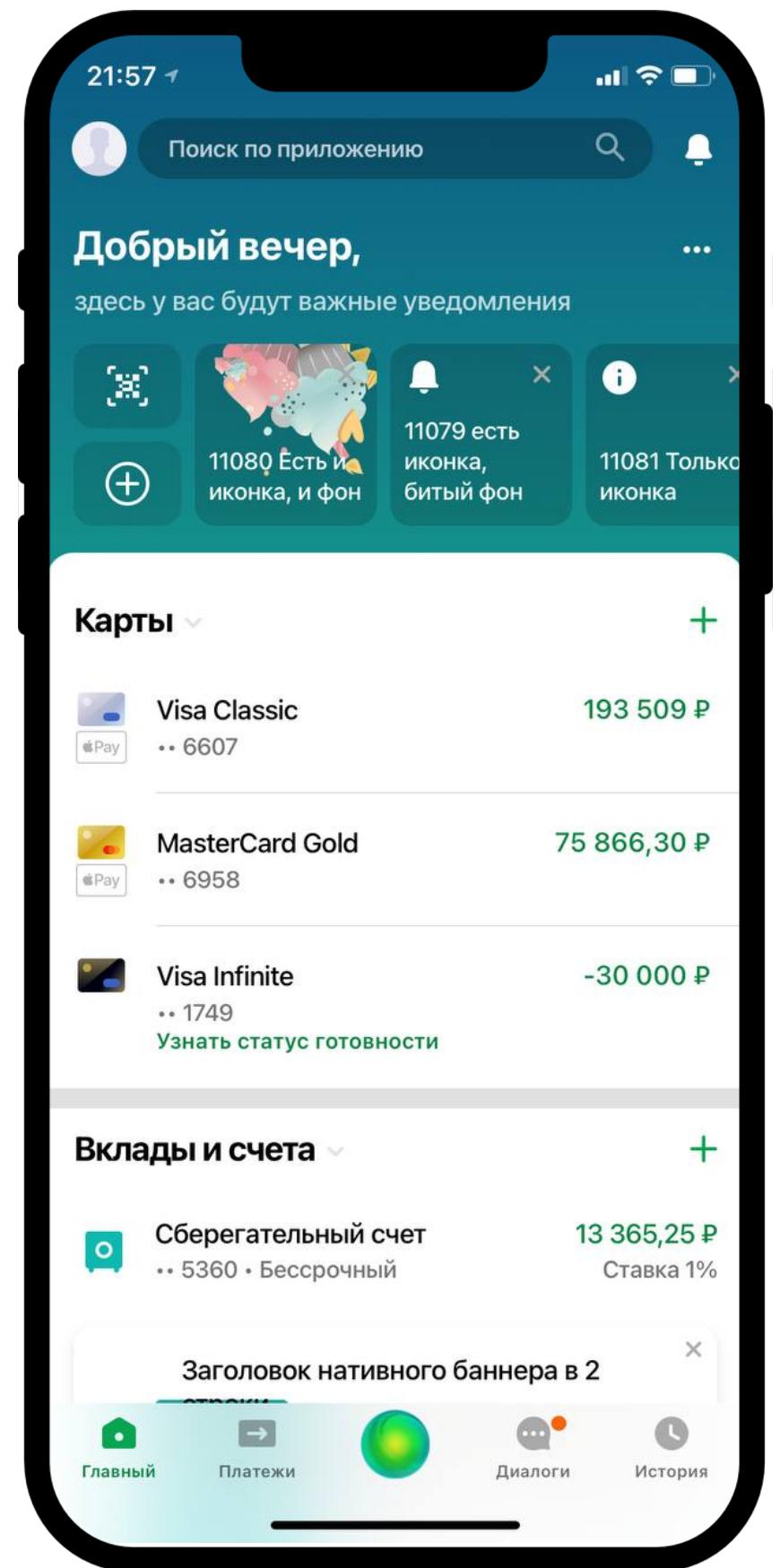
Яндекс Вертикали – Авто.ру

19 // 10 // 24

# Мой опыт

# Мой опыт

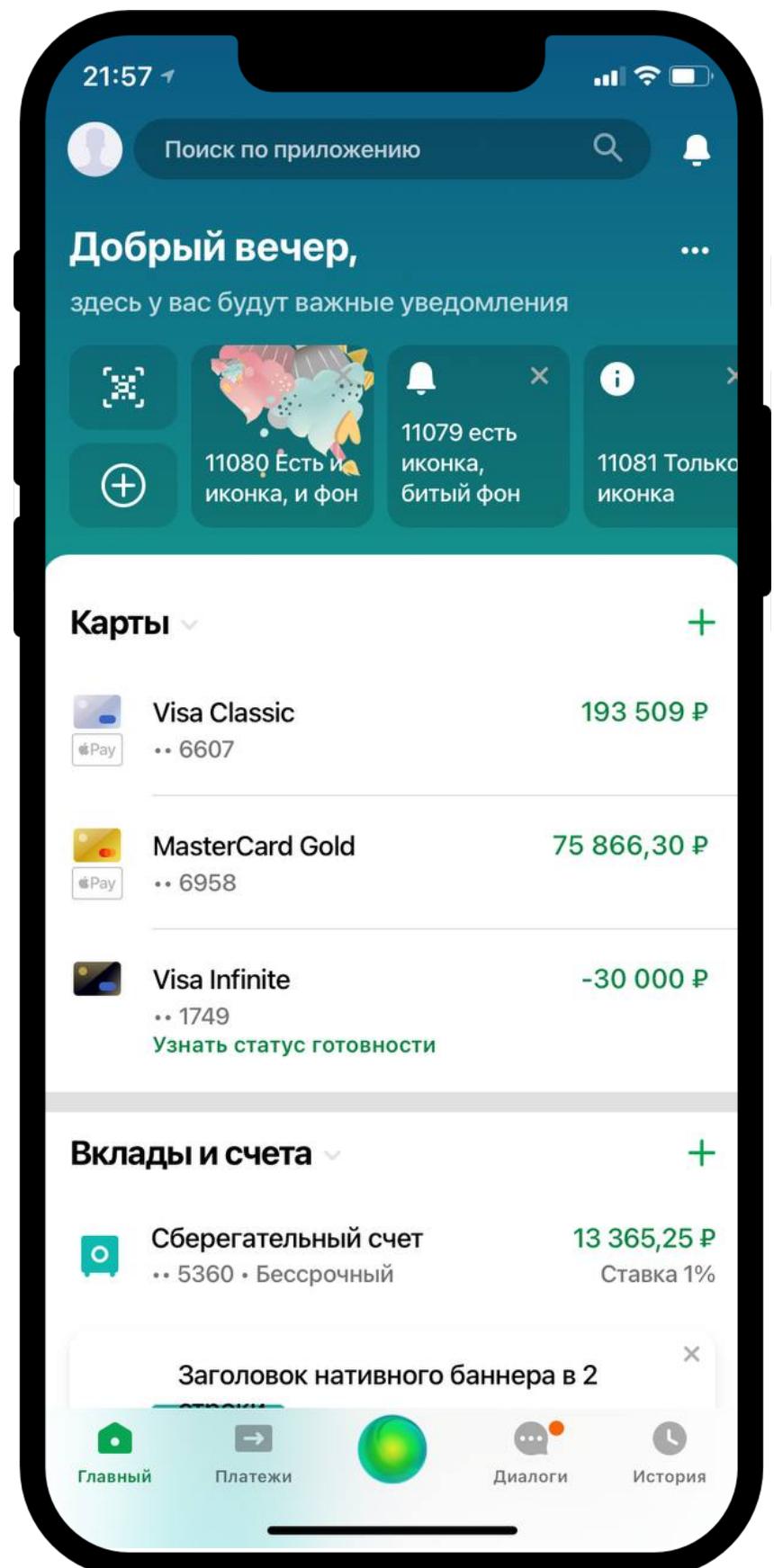
## СБЕР



Сбербанк Онлайн

# Мой опыт

СБЕР



Сбербанк Онлайн

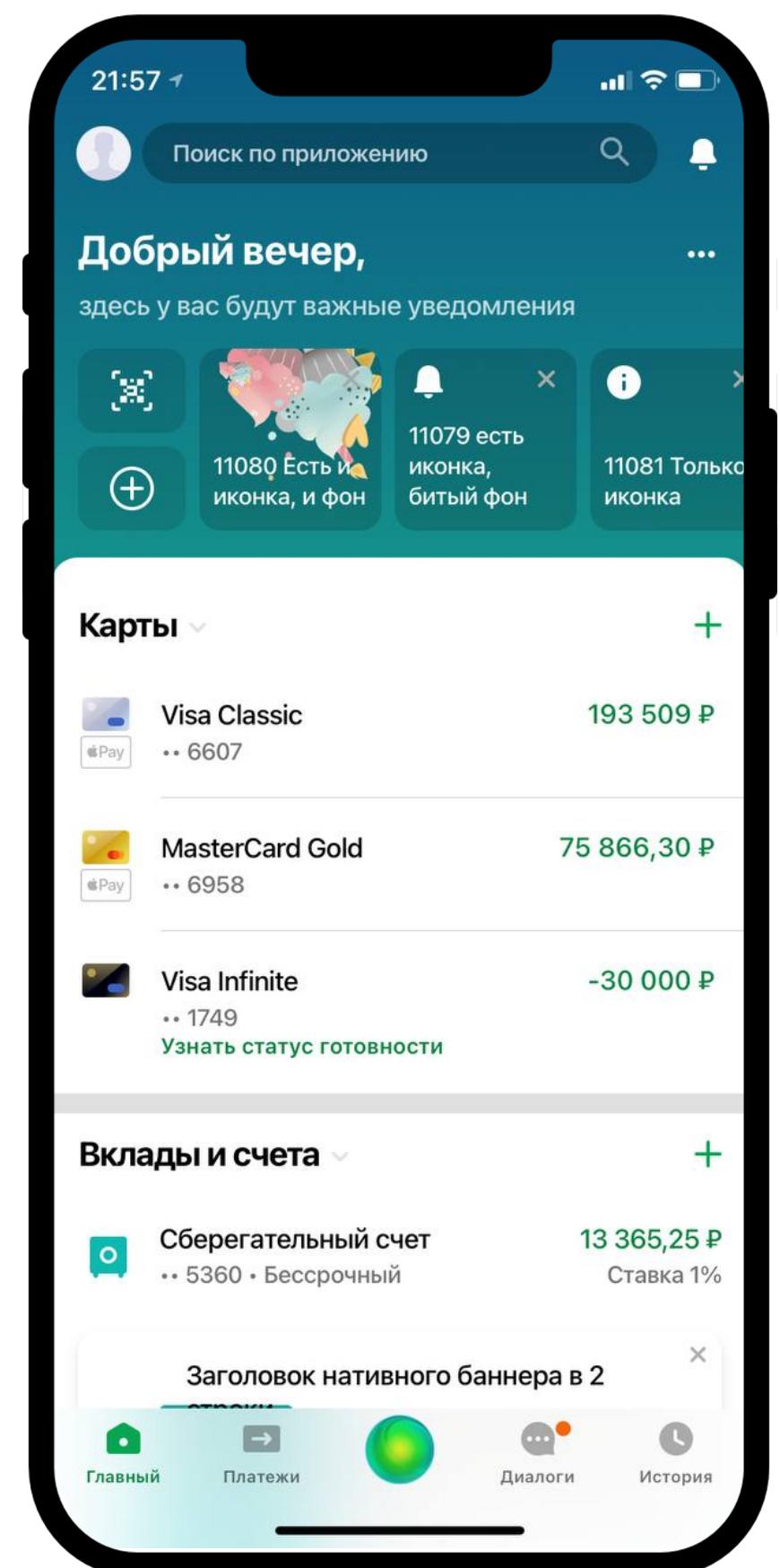
Яндекс



Авто.ру

# Мой опыт

## СБЕР



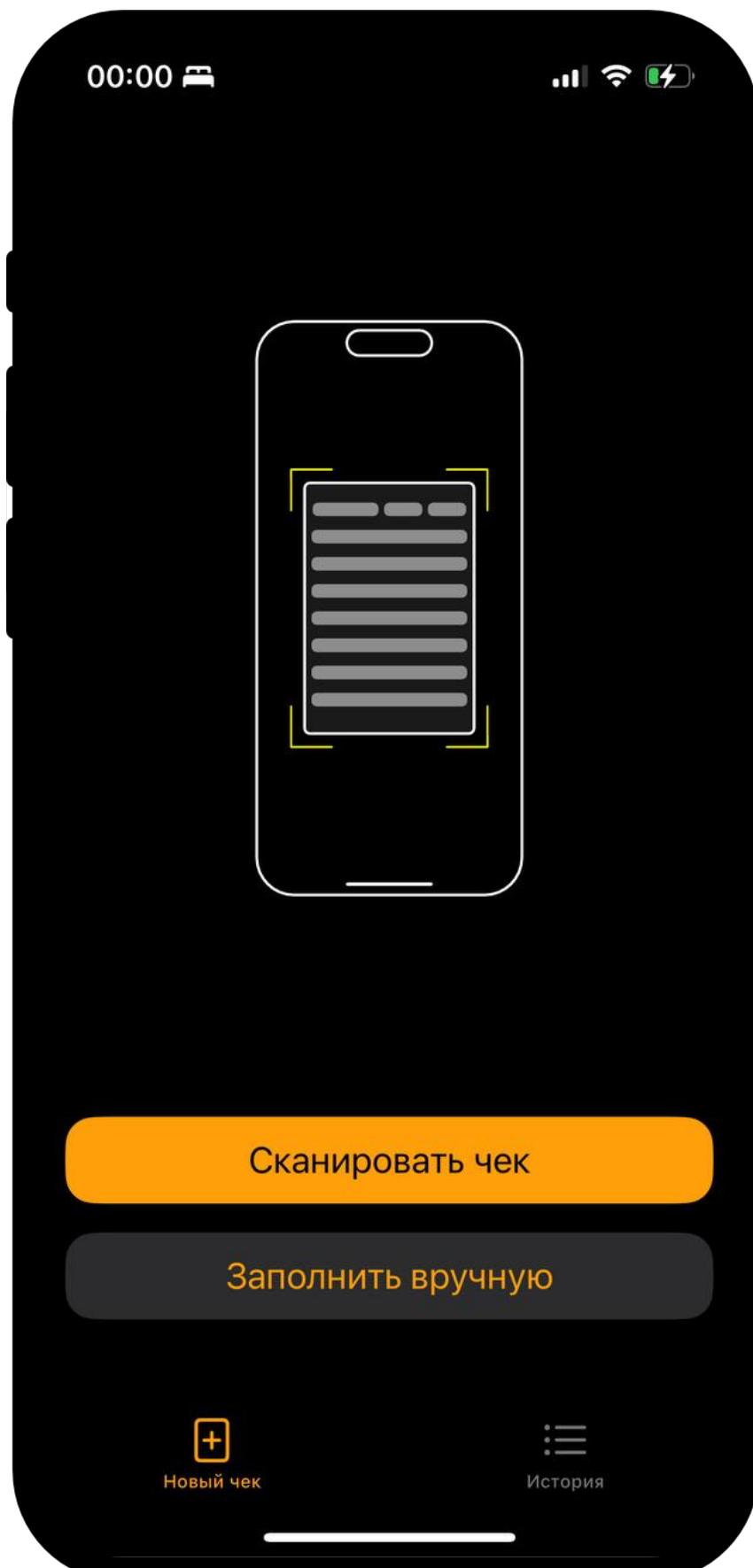
Сбербанк Онлайн

## Яндекс



auto.ru

## Пет проекты



Монетки

# Содержание доклада

- |    |  |    |                         |
|----|--|----|-------------------------|
| 01 | Зачем мобильному разработчику писать CLI     | 05 | Работа с логами         |
| 02 | Как сделать консольный интерфейс?            | 06 | Нюансы сборки и запуска |
| 03 | API для взаимодействия со структурой проекта | 07 | BuildTools в Авто.ру    |
| 04 | Анализ кода проекта                          |    |                         |

# CLI - Command line interface

# Пример CLI

```
git commit -m "Hello Mobius!" --verbose
```

# Пример CLI



# Зачем CLI мобильному разработчику?

# Зачем CLI мобильному разработчику? 🤔

01

Запуск  
кода на CI

# Зачем CLI мобильному разработчику? 🤔

01

Запуск  
кода на СІ

02

Инструменты  
для разработки

01

# Запуск кода на СІ



# CI: Build AdHoc

## Steps

- tools issue-info --auth-token \*my token\*
- tools pr-info --auth-token \*my token\*
- tools xcode-archive --configuration Debug --colored-logs --path \*путь до проекта\*
- ...
- tools lint --config \*путь до конфига\*
- tools import-check --xcodebuild-log-path .ci\_build/xcodebuild.log --path \*путь до проекта\*
- tools warnings-check --warnings-logs-file-path .ci\_build/warnings.log --path \*путь до проекта\*

# CI: Build AdHoc

## Steps

- tools issue-info --auth-token \*my token\*
- tools pr-info --auth-token \*my token\*
- tools xcode-archive --configuration Debug --colored-logs --path \*путь до проекта\*
- ...
- tools lint --config \*путь до конфига\*
- tools import-check --xcodebuild-log-path .ci\_build/xcodebuild.log --path \*путь до проекта\*
- tools warnings-check --warnings-logs-file-path .ci\_build/warnings.log --path \*путь до проекта\*

# CI: Build AdHoc

## Steps

- `tools init --path ... --auth-token ...`
- `tools issue-info --auth-token *my token*`
- `tools pr-info --auth-token *my token*`
- `tools xcode-archive --configuration Debug --colored-logs --path *путь до проекта*`
- ...
- `tools lint --config *путь до конфига*`
- `tools import-check --xcodebuild-log-path .ci_build/xcodebuild.log --path *путь до проекта*`
- `tools warnings-check --warnings-logs-file-path .ci_build/warnings.log --path *путь до проекта*`

# CI: Build AdHoc

## Steps

- **tools init --path ... --auth-token ...**
- tools issue-info
- tools pr-info
- tools xcode-archive --configuration Debug --colored-logs
- ...
- tools lint --config \*путь до конфига\*
- tools import-check --xcodebuild-log-path .ci\_build/xcodebuild.log
- tools warnings-check --warnings-logs-file-path .ci\_build/warnings.log

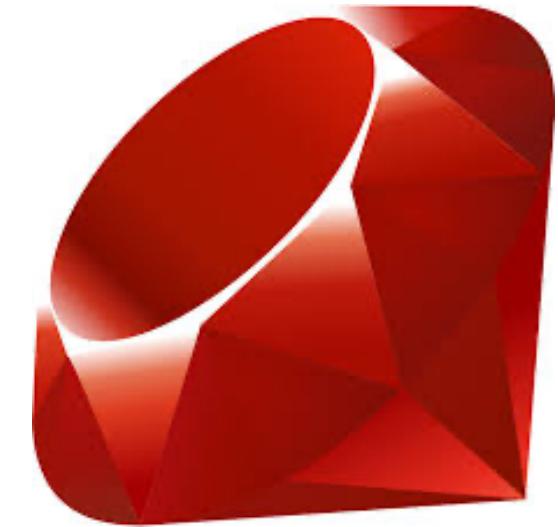
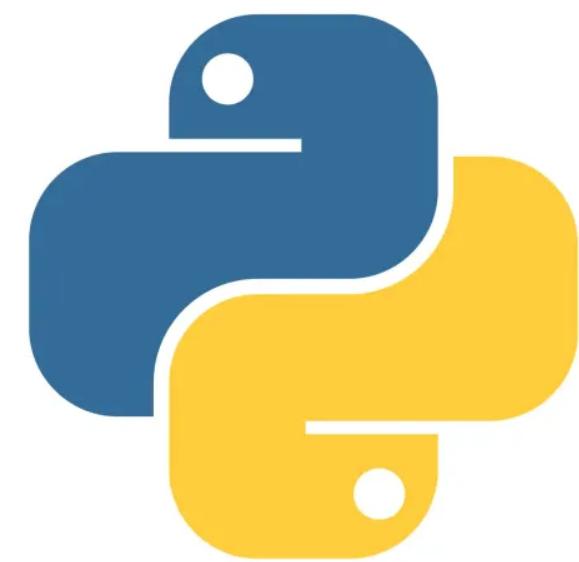
# Какой язык программирования выбрать?

# Какой язык программирования выбрать?

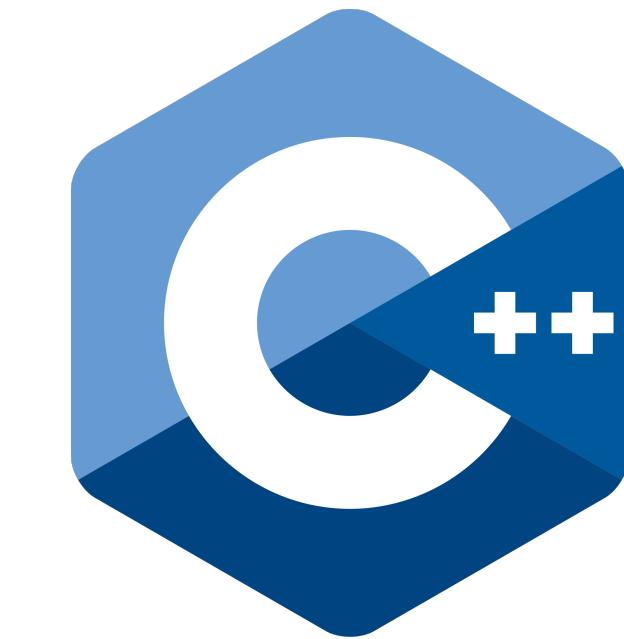
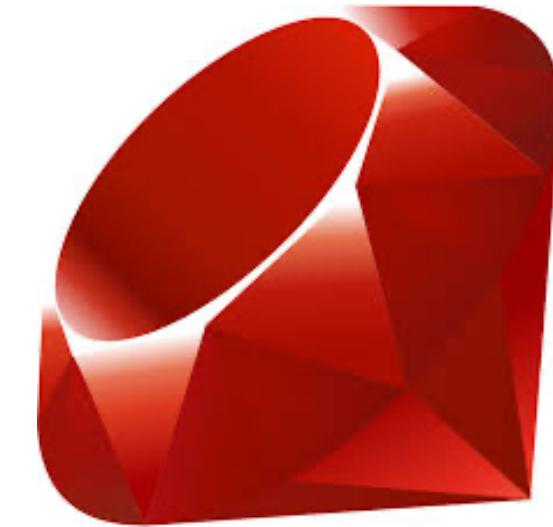
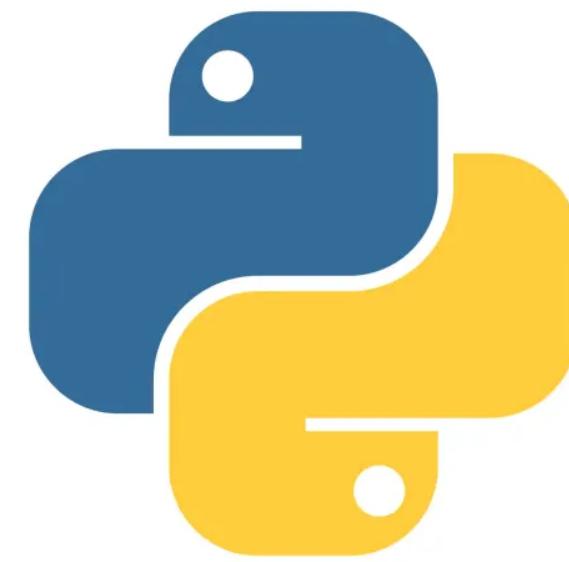
Любой язык с готовым парсером  
аргументов

# Какой язык программирования выбрать?

# Какой язык программирования выбрать?



# Какой язык программирования выбрать?

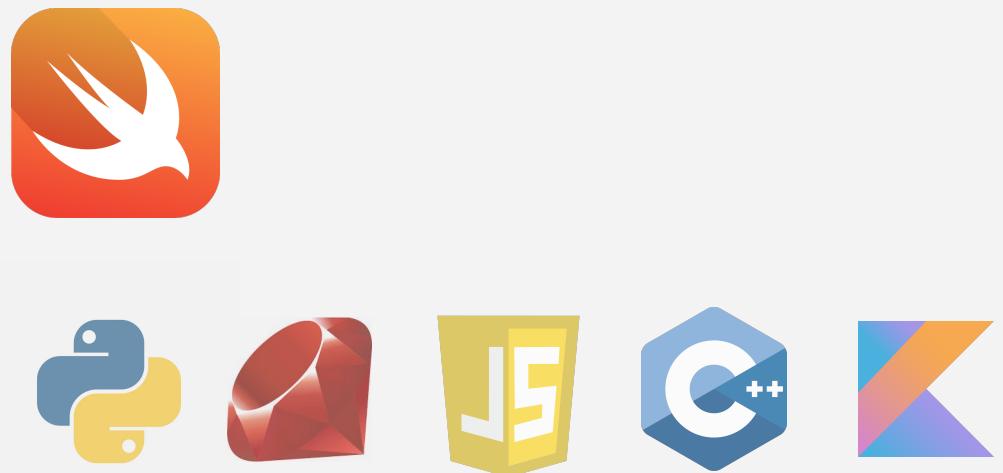


# Что ещё хотим от языка для С1?

# Что ещё хотим от языка для СІ?

Знакомый всем  
разработчикам  
язык

# Что ещё хотим от языка для СI?



Знакомый всем  
разработчикам  
язык

# Что ещё хотим от языка для СI?



Знакомый всем  
разработчикам  
язык

API для  
взаимодействия  
с проектом

# Что ещё хотим от языка для СI?



Знакомый всем  
разработчикам  
язык



API для  
взаимодействия  
с проектом

# Что ещё хотим от языка для СI?



Знакомый всем  
разработчикам  
язык



API для  
взаимодействия  
с проектом

Библиотеки для  
сборки и анализа  
кода

# Что ещё хотим от языка для СI?



Знакомый всем  
разработчикам  
язык



API для  
взаимодействия  
с проектом



Библиотеки для  
сборки и анализа  
кода

# Что ещё хотим от языка для СI?



Знакомый всем  
разработчикам  
язык



API для  
взаимодействия  
с проектом



Библиотеки для  
сборки и анализа  
кода

Готовые  
инструменты  
для логирования

# Что ещё хотим от языка для С1?



Знакомый всем  
разработчикам  
язык



API для  
взаимодействия  
с проектом



Библиотеки для  
сборки и анализа  
кода



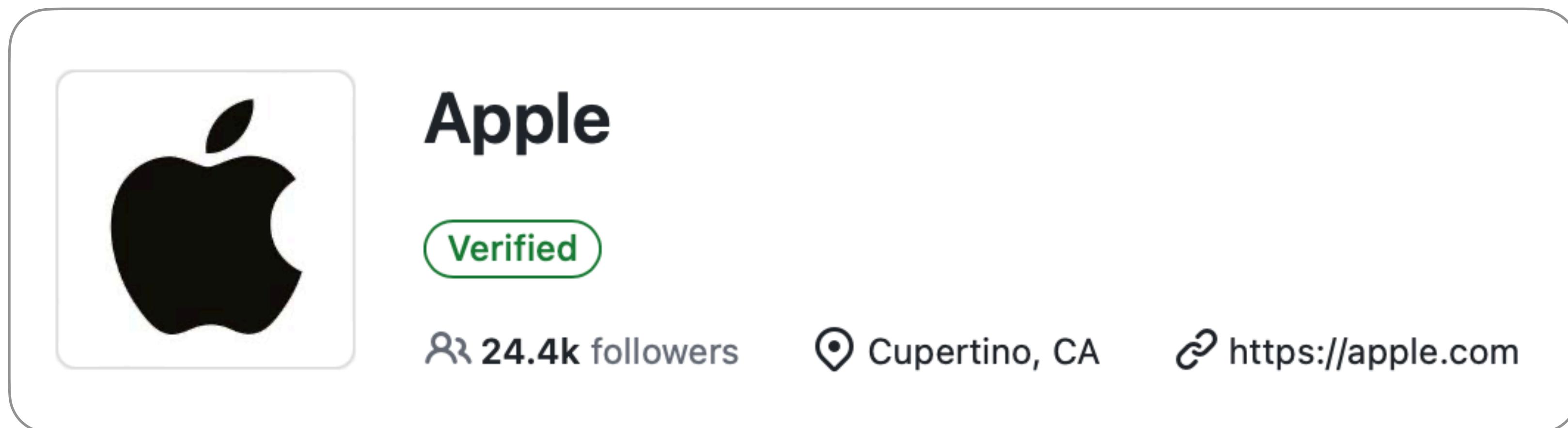
Готовые  
инструменты  
для логирования

02

# Swift Argument Parser



# Swift Argument Parser



A GitHub repository card for "swift-argument-parser". At the top left, the repository name "swift-argument-parser" is shown in blue, with a "Public" badge next to it. A green wavy progress bar is on the right. Below the name is a description: "Straightforward, type-safe argument parsing for Swift". At the bottom, there are several metrics: a red circle for Swift, a star icon for 3,274 stars, a license icon for Apache-2.0, a code icon for 309 issues, a circular icon for 76 open issues, a pull request icon for 11 pull requests, and a text indicating it was "Updated 3 days ago".

## MyProgam hello --name “Mobius” --newLine

```
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )
    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
        if newLine {
            print("Hello \(name)".replacingOccurrences(of: " ", with: "\n"))
        } else {
            print("Hello \(name)")
        }
    }
}
```

## MyProgam hello --name “Mobius” --newLine

```
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )
    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
        if newLine {
            print("Hello \(name)".replacingOccurrences(of: " ", with: "\n"))
        } else {
            print("Hello \(name)")
        }
    }
}
```

## MyProgam hello --name “Mobius” --newLine

```
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )

    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
        if newLine {
            print("Hello \(name)".replacingOccurrences(of: " ", with: "\n"))
        } else {
            print("Hello \(name)")
        }
    }
}
```

## MyProgam hello --name “Mobius” --newLine

```
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )

    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
        if newLine {
            print("Hello \(name)".replacingOccurrences(of: " ", with: "\n"))
        } else {
            print("Hello \(name)")
        }
    }
}
```

# MyProgram hello --name “Mobius” --newLine

```
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )

    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
        if newLine {
            print("Hello \(name)".replacingOccurrences(of: " ", with: "\n"))
        } else {
            print("Hello \(name)")
        }
    }
}
```



## Terminal: ./MyProgram --help

USAGE: my-program <subcommand>

OPTIONS:

-h, --help Show help information.

SUBCOMMANDS:

hello Напечатать hello



## Terminal: ./MyProgram hello --help

OVERVIEW: Напечатать hello

USAGE: my-program hello --name <name> [--new-line]

OPTIONS:

--name <name>	Имя для вывода
--new-line	Печатать каждое слово с новой строки
-h, --help	Show help information.

# Пользователи



Swift Package  
Manager



SwiftLint



SwiftFormat



Tuist



Periphery

03

# API для взаимодействия со структурой проекта



# API для взаимодействия со структурой проекта

Найти все таргеты  
тестов в SPM проекте

# SwiftPM-auto

>  swift\_cli

swift\_cli > Package > No Selection

```
1 // swift-tools-version: 5.10
2
3 import PackageDescription
4
5 let package = Package(
6   name: "MyCLI",
7   dependencies: [
8     .package(url: "https://github.com/apple/swift-argument-parser", from: "1.0.0"),
9     .package(url: "https://github.com/apple/swift-package-manager.git", branch: "main")
10   ],
11   targets: [
12     .executableTarget(
13       name: "MyCLI",
14       dependencies: [
15         .product(name: "ArgumentParser", package: "swift-argument-parser"),
16         .product(name: "SwiftPM-auto", package: "swift-package-manager")
17       ],
18       path: "Sources"),
19   ]
20 )
21
```

# Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

# Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

# Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

# Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

# Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

04

# Анализ кода проекта



# Статический анализ кода

# Статический анализ кода

Проверить, что в Feature  
пакетах не создаются  
экземпляры класса Date

# Статический анализ кода

Проверить, что в Feature пакетах не создаются экземпляры класса Date

Исключение:  
SwiftUI Preview

# SwiftSyntax



The Swift Programming Language

Verified

2.5k followers Worldwide https://swift.org @SwiftLang



swift-syntax Public

A set of Swift libraries for parsing, inspecting, generating, and transforming Swift source code.

Swift 3,130 Apache-2.0 393 65 (7 issues need help) 42 Updated 42 minutes ago



Swift-syntax – это набор библиотек, которые работают с древовидным представлением исходного кода Swift с точностью до исходного кода, называемым SwiftSyntax tree

# AST Explorer

# AST Explorer

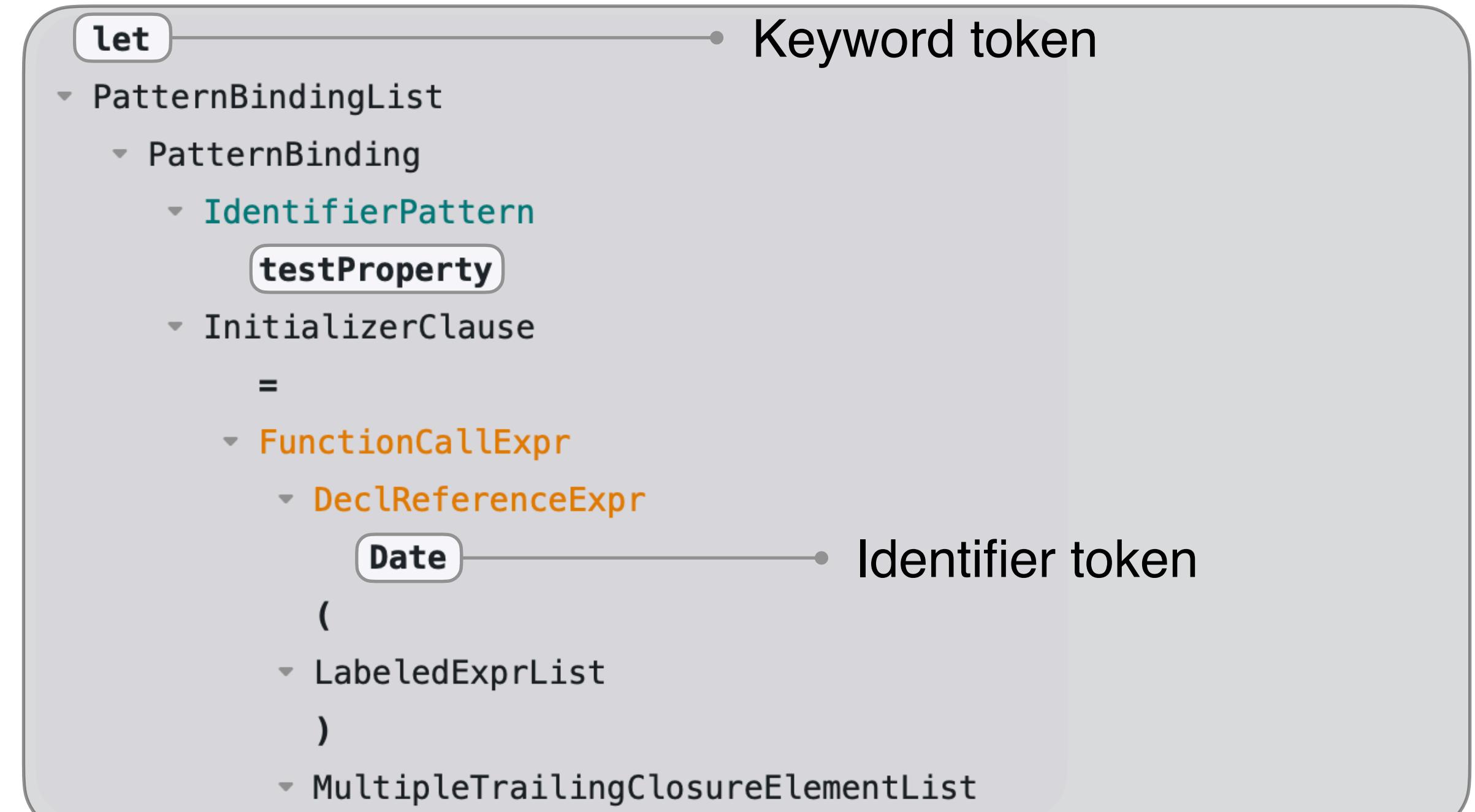
```
struct Test {  
    let testProperty = Date()  
}
```

```
let  
  ▾ PatternBindingList  
    ▾ PatternBinding  
      ▾ IdentifierPattern  
        testProperty  
      ▾ InitializerClause  
        =  
        ▾ FunctionCallExpr  
          ▾ DeclReferenceExpr  
            Date  
            (  
              ▾ LabeledExprList  
              )  
            ▾ MultipleTrailingClosureElementList
```

# AST Explorer

```
struct Test {  
    let testProperty = Date()  
}
```

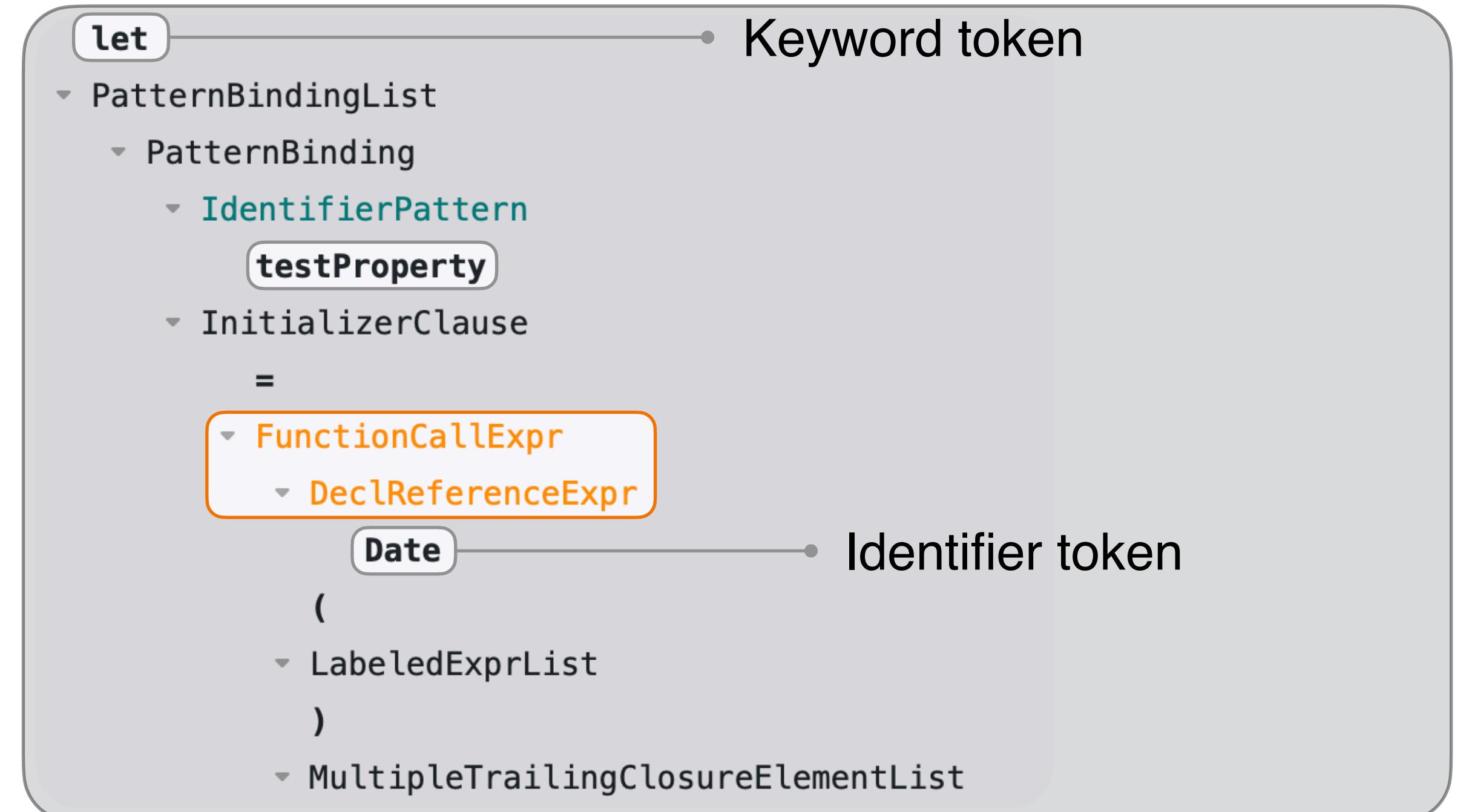
## TokenSyntax



# AST Explorer

```
struct Test {  
    let testProperty = Date()  
}
```

TokenSyntax ExprSyntax



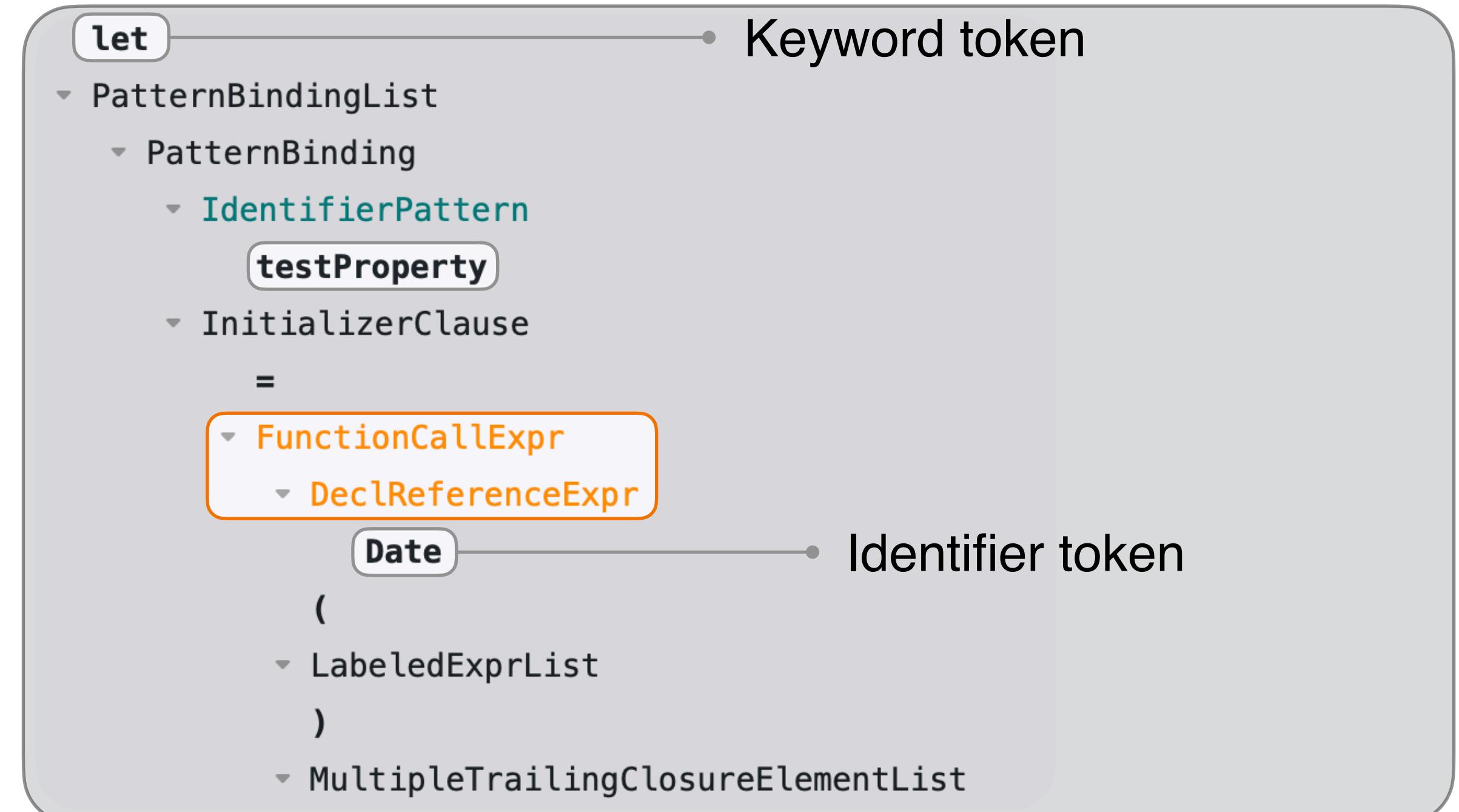
# AST Explorer

```
import SwiftSyntax

static func isDateInit(token: TokenSyntax) -> Bool {
    if token.tokenKind == .identifier("Date"),
        let declReferenceExpr = token.parent?.as(DeclReferenceExprSyntax.self),
        let functionCallExpr = declReferenceExpr.parent?.as(FunctionCallExprSyntax.self),
        functionCallExpr.arguments.isEmpty { return true }

    return false
}
```

TokenSyntax ExprSyntax



# AST Explorer

```
import SwiftSyntax

static func isDateInit(token: TokenSyntax) -> Bool {
    if token.tokenKind == .identifier("Date"),
        let declReferenceExpr = token.parent?.as(DeclReferenceExprSyntax.self),
        let functionCallExpr = declReferenceExpr.parent?.as(FunctionCallExprSyntax.self),
        functionCallExpr.arguments.isEmpty { return true }

    return false
}
```

# Поиск Date init: preview check

```
struct MyView_Previews: PreviewProvider {  
    static var previews: some View {  
        MyView(date: Date())  
    }  
}
```

# Поиск Date init: preview check

```
struct MyView_Previews: PreviewProvider {  
    static var previews: some View {  
        MyView(date: Date())  
    }  
}
```



# Поиск Date init: preview check

```
#Preview {  
    MyView(date: Date())  
}
```



# Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
                ...
                // Previews
                if let structDecl = parentToken.as(StructDeclSyntax.self),
                   let inheritedTypeList = structDecl.inheritanceClause?.inheritedTypes.as(InheritedTypeListSyntax.self),
                   inheritedTypeList.contains(where: { inheritedTypeSyntax in
                       let identifierTypeName = inheritedTypeSyntax.type.as(IdentifierTypeSyntax.self)?.name.text
                       return identifierTypeName == "PreviewProvider"
                   })
                {
                    continue sourceTokensLoop
                }
            }
            let converter = SourceLocationConverter(fileName: filePath, tree: tree)
            resultLocations.append(converter.location(for: token.position))
        }
    }

    return resultLocations
}
```

# Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
                ...
                // Previews
                if let structDecl = parentToken.as(StructDeclSyntax.self),
                   let inheritedTypeList = structDecl.inheritanceClause?.inheritedTypes.as(InheritedTypeListSyntax.self),
                   inheritedTypeList.contains(where: { inheritedTypeSyntax in
                       let identifierTypeName = inheritedTypeSyntax.type.as(IdentifierTypeSyntax.self)?.name.text
                       return identifierTypeName == "PreviewProvider"
                   })
                {
                    continue sourceTokensLoop
                }
            }
        }
        let converter = SourceLocationConverter(fileName: filePath, tree: tree)
        resultLocations.append(converter.location(for: token.position))
    }
}

return resultLocations
}
```

# Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
                ...
                // Previews
                if let structDecl = parentToken.as(StructDeclSyntax.self),
                   let inheritedTypeList = structDecl.inheritanceClause?.inheritedTypes.as(InheritedTypeListSyntax.self),
                   inheritedTypeList.contains(where: { inheritedTypeSyntax in
                       let identifierTypeName = inheritedTypeSyntax.type.as(IdentifierTypeSyntax.self)?.name.text
                       return identifierTypeName == "PreviewProvider"
                   })
                {
                    continue sourceTokensLoop
                }
            }
            let converter = SourceLocationConverter(fileName: filePath, tree: tree)
            resultLocations.append(converter.location(for: token.position))
        }
    }

    return resultLocations
}
```

# Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
                ...
                // Previews macro
                if let macroDecl = parentToken.as(MacroExpansionExprSyntax.self), macroDecl.name.text == "Preview" {
                    continue sourceTokensLoop
                }

                ...
                ...
                ...
                ...
            }
        }

        let converter = SourceLocationConverter(fileName: filePath, tree: tree)
        resultLocations.append(converter.location(for: token.position))
    }

    return resultLocations
}
```

# Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
                ...
                // Previews macro
                if let macroDecl = parentToken.as(MacroExpansionExprSyntax.self), macroDecl.name.text == "Preview" {
                    continue sourceTokensLoop
                }

                ...
                ...
                ...
                ...
            }
        }

        let converter = SourceLocationConverter(fileName: filePath, tree: tree)
        resultLocations.append(converter.location(for: token.position))
    }

    return resultLocations
}
```

# Date lint command

```
23:28:39 error: Использование Date вместо UtcClock: Features/ReportDetails/Sources/ReportDetails/ReportDetailsView.swift:46:16
23:28:39 error: Lint failed with errors:
23:28:39 error: The operation couldn't be completed. (LintCheck.DateLintCheckCommand.DateLintError error 0.)
Error: LintCheckError()
```

04

# Проверка новых warning-ов на CI



# Парсим логи с Xbeatify

```
s/arm64 -Xcc -I/Users/igorshamrin/arc/arcadia/classifieds/mobile-autoru-client-ios/.ci_build/derived_data
/Build/Intermediates.noindex/ArchiveIntermediates/QAssist/IntermediateBuildFilesPath/QAssist.build/Debug-
iphoneos/QAssist.build/DerivedSources -Xcc -DDEBUG\=1 -module-name QAssist -frontend-parseable-output -di
sable-clang-spi -target-sdk-version 17.4 -target-sdk-name iphoneos17.4 -external-plugin-path /Application
s/Xcode-15.3.0-Release.Candidate.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhone
OS17.4.sdk/usr/lib/swift/host/plugins\#/Applications/Xcode-15.3.0-Release.Candidate.app/Contents/Develop
er/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS17.4.sdk/usr/bin/swift-plugin-server -external-plugi
n-path /Applications/Xcode-15.3.0-Release.Candidate.app/Contents/Developer/Platforms/iPhoneOS.platform/De
veloper/SDKs/iPhoneOS17.4.sdk/usr/local/lib/swift/host/plugins\#/Applications/Xcode-15.3.0-Release.Candid
ate.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS17.4.sdk/usr/bin/swift-plug
in-server -external-plugin-path /Applications/Xcode-15.3.0-Release.Candidate.app/Contents/Developer/Platf
orms/iPhoneOS.platform/Developer/usr/lib/swift/host/plugins\#/Applications/Xcode-15.3.0-Release.Candidat
e.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/swift-plugin-server -external-plugi
n-path /Applications/Xcode-15.3.0-Release.Candidate.app/Contents/Developer/Platforms/iPhoneOS.platform/De
veloper/usr/local/lib/swift/host/plugins\#/Applications/Xcode-15.3.0-Release.Candidate.app/Contents/Devel
oper/Platforms/iPhoneOS.platform/Developer/usr/bin/swift-plugin-server -plugin-path /Applications/Xcode-1
5.3.0-Release.Candidate.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/lib/swift/host/plu
gins -plugin-path /Applications/Xcode-15.3.0-Release.Candidate.app/Contents/Developer/Toolchains/XcodeDef
ault.xctoolchain/usr/local/lib/swift/host/plugins -o /Users/igorshamrin/arc/arcadia/classifieds/mobile-au
toru-client-ios/.ci_build/derived_data/Build/Intermediates.noindex/ArchiveIntermediates/QAssist/Intermedi
ateBuildFilesPath/QAssist.build/Debug-iphoneos/QAssist.build/Objects-normal/arm64/FileStorage.o -o /Users
/igorshamrin/arc/arcadia/classifieds/mobile-autoru-client-ios/.ci_build/derived_data/Build/Intermediates.
noindex/ArchiveIntermediates/QAssist/IntermediateBuildFilesPath/QAssist.build/Debug-iphoneos/QAssist.buil
d/Objects-normal/arm64/AppDelegate.o -index-unit-output-path /QAssist.build/Debug-iphoneos/QAssist.build/
Objects-normal/arm64/FileStorage.o -index-unit-output-path /QAssist.build/Debug-iphoneos/QAssist.build/Ob
jects-normal/arm64/AppDelegate.o
/Users/igorshamrin/arc/arcadia/classifieds/mobile-autoru-client-ios/utils/mobile-autoru-client-settings-i
os/QAssist/AppDelegate.swift:13:47: error: argument passed to call that takes no arguments
    let vc = AppSettingsViewController(a: 10)
               ~~~~^~~

SwiftDriverJobDiscovery normal arm64 Compiling FieldsRow.swift, LabelRow.swift, MultipleSelectorRow.swift
, PickerInlineRow.swift, PickerInputRow.swift, PickerRow.swift (in target 'Eureka' from project 'Eureka')

SwiftCompile normal arm64 Compiling\ AppSettingsViewController.swift /Users/igorshamrin/arc/arcadia/class
ifieds/mobile-autoru-client-ios/utils/mobile-autoru-client-settings-ios/QAssist/Screens/AppSettingsViewCo
ntroller.swift (in target 'QAssist' from project 'QAssist')
    cd /Users/igorshamrin/arc/arcadia/classifieds/mobile-autoru-client-ios/utils/mobile-autoru-client-set
tings-ios
    builtin-swiftTaskExecution -- /Applications/Xcode-15.3.0-Release.Candidate.app/Contents/Developer/Too
lchains/XcodeDefault.xctoolchain/usr/bin/swift-frontend -frontend -c /Users/igorshamrin/arc/arcadia/class
ifieds/mobile-autoru-client-ios/utils/mobile-autoru-client-settings-ios/QAssist/Data/FileStorage.swift /U
ser/igorshamrin/arc/arcadia/classifieds/mobile-autoru-client-ios/utils/mobile-autoru-client-settings-ios
/QAssist/AppDelegate.swift /Users/igorshamrin/arc/arcadia/classifieds/mobile-autoru-client-ios/utils/mobi
```

Стандартный xcarchive (~ 335МБ)

```
[Eureka] Compiling RuleURL.swift
[Eureka] Compiling PopoverSelectorRow.swift
[Eureka] Compiling PushRow.swift
[Eureka] Compiling SegmentedRow.swift
[Eureka] Compiling ListCheckRow.swift
[Eureka] Compiling SliderRow.swift
[Eureka] Compiling StepperRow.swift
[Eureka] Compiling DecimalFormatter.swift
[Eureka] Compiling FieldRow.swift
[Eureka] Compiling GenericMultipleSelectorRow.swift
[Eureka] Compiling OptionsRow.swift
[Eureka] Compiling Protocols.swift
[Eureka] Compiling SelectorRow.swift
[Eureka] Compiling MultipleSelectorViewController.swift
[QAssist] Compiling Alert.swift
[QAssist] Compiling Preset.swift
[QAssist] Compiling AppSettings.swift
[QAssist] Compiling ErrorUtils.swift
△ /Users/igorshamrin/arc/arcadia/classifieds/mobile-autoru-client-ios/utils/mobile-autoru-client-settin
gs-ios/QAssist/Utils/ErrorUtils.swift:6:13: switch covers known cases, but 'DecodingError' may have ad
ditional unknown values, possibly added in future versions
    switch decodingError {
        ^
[QAssist] Compiling UserPresets.swift
[QAssist] Compiling FileStorage.swift
[QAssist] Compiling AppDelegate.swift
✗ /Users/igorshamrin/arc/arcadia/classifieds/mobile-autoru-client-ios/utils/mobile-autoru-client-settin
gs-ios/QAssist/AppDelegate.swift:13:47: argument passed to call that takes no arguments
    let vc = AppSettingsViewController(a: 10)
               ~~~~^~~
[QAssist] Compiling Version.swift
[QAssist] Compiling GeneratedAssetSymbols.swift
[QAssist] Compiling UIBarButtonItem.swift
[QAssist] Compiling AppSettingsViewController.swift
[QAssist] Compiling AppConfig.swift
** ARCHIVE FAILED **
```

Отформатированный xcarchive (< 1МБ)

# Парсим логи с Xcbeautify

```
import XcbeautifyLib

func onWriteLine(_ line: String, fromErrorStream: Bool) {
    ...

    if let formattedLine = formatNextLine() {
        outputHandler.write(parser.outputType, formattedLine)

        if parser.outputType == .warning {
            writeWarning(formattedLine) // <-- Записываем warning в warnings.log
        }
    }
}
```

# Парсим логи с Xcbeautify

```
import XcbeautifyLib

func onWriteLine(_ line: String, fromErrorStream: Bool) {
    ...

    if let formattedLine = formatNextLine() {
        outputHandler.write(parser.outputType, formattedLine)

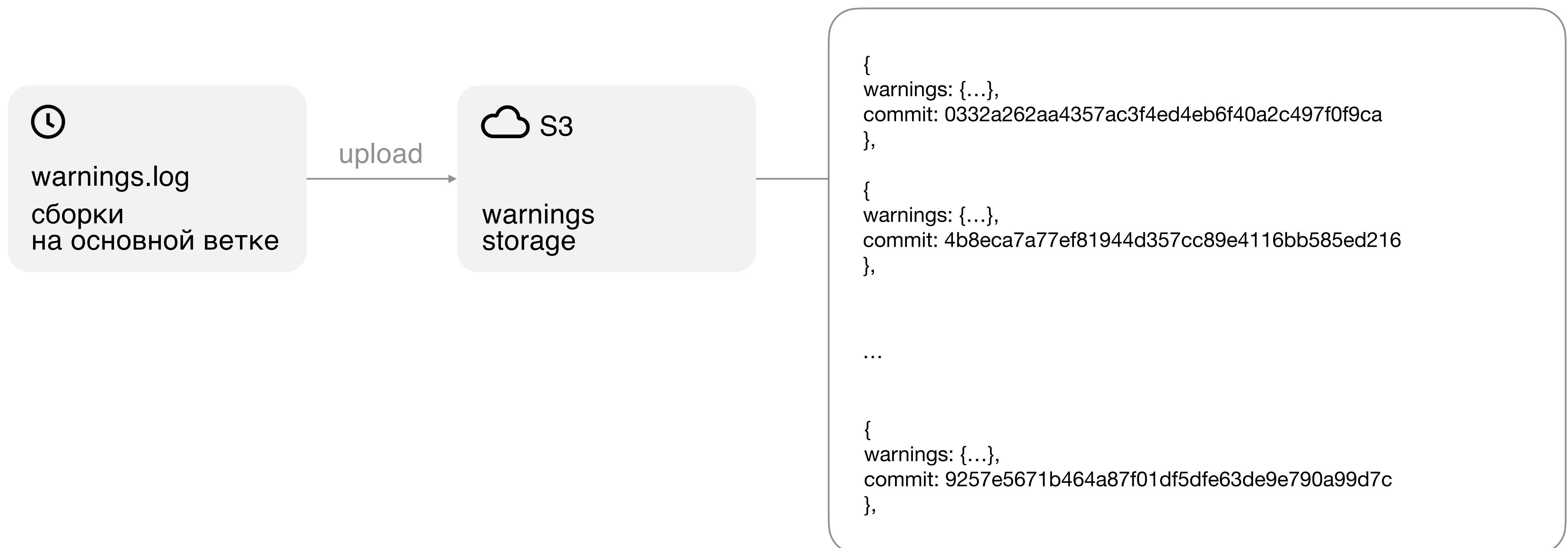
        if parser.outputType == .warning {
            writeWarning(formattedLine) // <-- Записываем warning в warnings.log
        }
    }
}
```

# warnings.log

```
>
{
    /Contracts/Sources/ComplaintContracts/ComplaintService.swift:10:14: : [ 1 item ]
        0 : main actor-isolated property 'service' can not be mutated from a non-isolated context; this is an error in Swift 6
            self.service = service
            ^
    ]
    /Contracts/Sources/SplitContracts/SplitService.swift:38:9: : [ 1 item ]
        0 : main actor-isolated property 'objectWillChange' can not be mutated from a non-isolated context; this is an error in Swift 6
            objectWillChange = service.erasedObjectWillChange()
            ^
    ]
▶ /Services/PhoneConfirmationServices/Sources/PhoneConfirmationFlowImpl.swift:20:14: : [ 1 item ]
▶ /packages/user_profile/main/UserProfileInteractor.swift:558:13: : [ 1 item ]
▶ /Shared/Views/Sources/Views/SwipeableView.swift:71:17: : [ 1 item ]
▶ /Contracts/Sources/PublicProfileContracts/PublicProfileInfoService.swift:12:9: : [ 1 item ]
▶ /Features/UGC/Sources/UGCUserProfile/UGCUserProfileViewModel.swift:107:14: : [ 1 item ]
▶ /Features/NotificationBell/Sources/NotificationBell/NotificationBellItemFactory.swift:27:14: : [ 1 item ]
```

# Проверка новых warning-ов на CI

Обновляем состояние по таймеру



# Проверка новых warning-ов на CI

Обновляем состояние по таймеру



# Другие проверки



SwiftFormat

Lint & Format



Periphery, SwiftPM

Unused code  
check



SourceKitten

Import check



SwiftPM

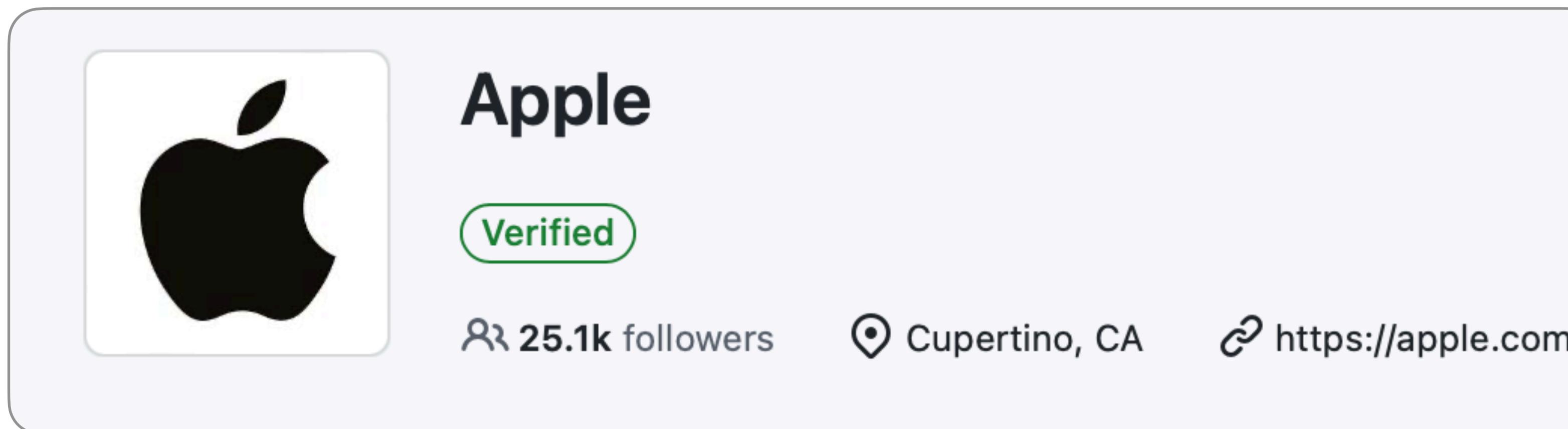
Dependency check

05

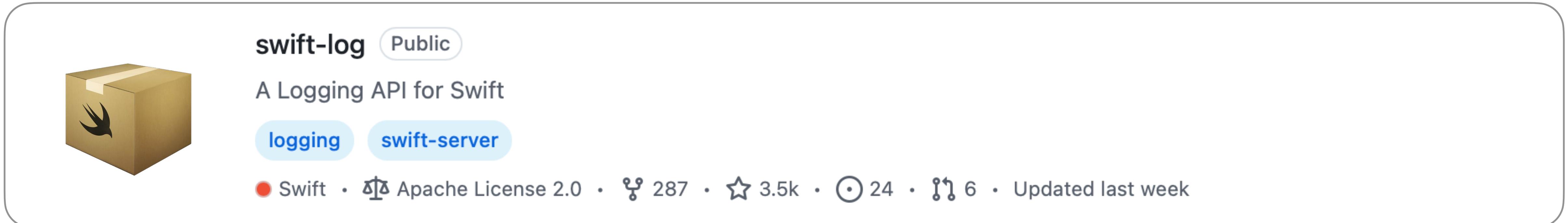
Собираем логи красиво



# Swift-log



A screenshot of a Twitter profile card for the account "Apple". The card features the Apple logo in a white square. To the right of the logo, the word "Apple" is written in a large, bold, black sans-serif font. Below "Apple" is a green oval button containing the word "Verified" in white. Further down, there are three pieces of information: a user icon followed by "25.1k followers", a location pin icon followed by "Cupertino, CA", and a link icon followed by the URL "https://apple.com". The entire card has a thin gray border.



A screenshot of a GitHub repository card for "swift-log". On the left is a small icon of a brown cardboard box with a black bird logo on it. To the right of the icon, the repository name "swift-log" is displayed in bold black text, with a "Public" badge next to it. Below the name is a brief description: "A Logging API for Swift". Underneath the description are two blue circular tags: "logging" and "swift-server". At the bottom of the card, there is a row of metrics: a red circle followed by "Swift", a blue square followed by "Apache License 2.0", a yellow triangle followed by "287", a purple star followed by "3.5k", a gray circle followed by "24", a green hexagon followed by "6", and the text "Updated last week". The entire card has a thin gray border.

# Swift-log

```
let logger = Logger(label: "MyCommand")

logger.info("response: \$(string)")

logger.error("xcodebuild run failed, exit code = \$(status)")

logger.warning("issueKey is nil")
```

# Swift-log

```
extension LoggingSystem {
    static func setUp() {
        bootstrap { label in
            var handlers: [any LogHandler] = [LogHandlerImpl(label: label)]
            do {
                if let logsDirectory = ProcessInfo.processInfo.environment["LOGS_DIRECTORY"] {
                    try handlers.append(FileLogger(directoryPath: logsDirectory, fileName: label))
                }
            } catch { ... }

            return MultiplexLogHandler(handlers)
        }
    }
}
```

# Html страничка с логами

## Build and PR checks

### Common logs (stdout + stderr) log

► **XcodeBuild.log**

▼ **ImportCheckCommand.log**

Start checking packages/stock\_card/main/StockCardViewController.swift

► **WarningsCheckCommand.log**

### Errors logs (stderr) log

▼ **WarningsCheckCommand\_errors.log**

Появились новые warning-и

Список warning-ов, отличающихся от trunk:

```
[!] /packages/stock_card/main/StockCardViewController.swift:170:16: variable 'b' was never used; consider replacing with '_' or removing it
var a, b: Int
^

[!] /packages/stock_card/main/StockCardViewController.swift:170:13: variable 'a' was never used; consider replacing with '_' or removing it
var a, b: Int
^
```

# Html страничка с логами



[Ansi2Html](#)

A Swift library to convert text with ANSI escape codes to HTML

Public

C++ ★ 1

06

# Нюансы сборки и запуска



# Запуск



`swift run`

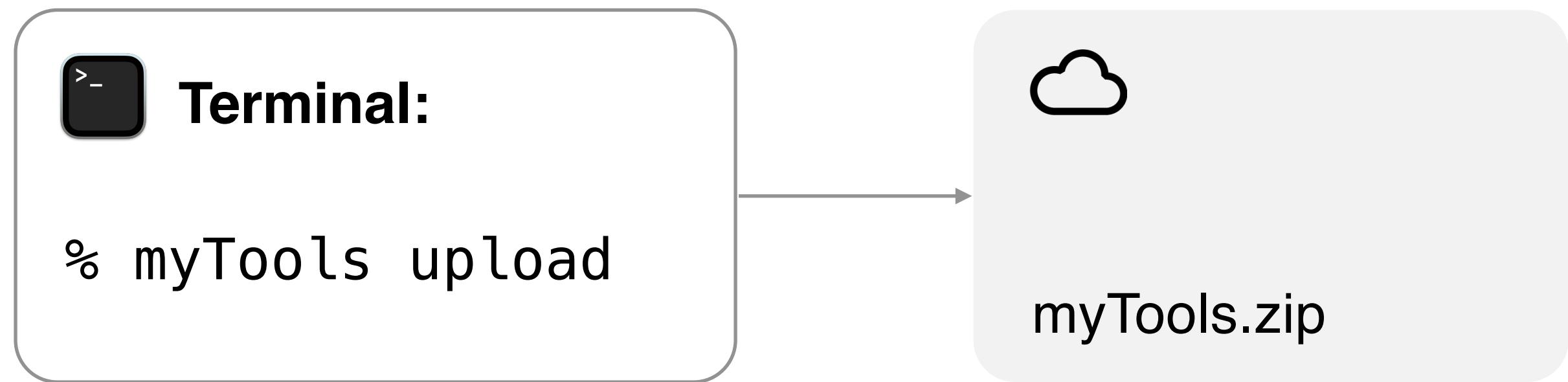


Предсобранный  
бинарник  
в репозитории

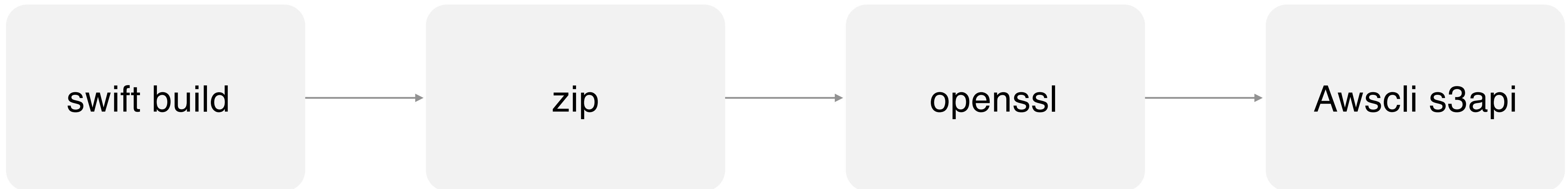


Предсобранный  
бинарник в облаке

# Upload



# Upload



.runnerEnv



```
EXECUTABLE = AutoRuBuildTools
URL[arm64] =
CHECKSUM[arm64] =
URL[x86_64] =
CHECKSUM[x86_64] =
```

# Runner

.runnerEnv



```
EXECUTABLE = AutoRuBuildTools  
URL[arm64] =  
CHECKSUM[arm64] =  
URL[x86_64] =  
CHECKSUM[x86_64] =
```

Runner

.runnerCache

```
myTools_v1  
myTools_v2  
myTools_v3
```



07

# Build tools в Авто.py



# CI: Build AdHoc

## Steps

- tools issue-info --auth-token \*my token\*
- tools pr-info --auth-token \*my token\*
- tools xcode-archive --configuration Debug --colored-logs --path \*путь до проекта\*
- ...
- tools lint --config \*путь до конфига\*
- tools import-check --xcodebuild-log-path \$PWD/.ci\_build/xcodebuild.log --path \*путь до проекта\*
- tools warnings-check --warnings-logs-file-path .ci\_build/warnings.log --path \*путь до проекта\*

# CI: Build AdHoc

## Steps

- tools issue-info --auth-token \*my token\*
- tools pr-info --auth-token \*my token\*
- tools xcode-archive --configuration Debug --colored-logs --path \*путь до проекта\*
- ...
- tools lint --config \*путь до конфига\*
- tools import-check --xcodebuild-log-path \$PWD/.ci\_build/xcodebuild.log --path \*путь до проекта\*
- tools warnings-check --warnings-logs-file-path .ci\_build/warnings.log --path \*путь до проекта\*

# CI: Build AdHoc

## Steps

- **tools init --path ... --auth-token ...**
- **tools issue-info --auth-token \*my token\***
- **tools pr-info --auth-token \*my token\***
- **tools xcode-archive --configuration Debug --colored-logs --path \*путь до проекта\***
- ...
- **tools lint --config \*путь до конфига\***
- **tools import-check --xcodebuild-log-path \$PWD/.ci\_build/xcodebuild.log --path \*путь до проекта\***
- **tools warnings-check --warnings-logs-file-path .ci\_build/warnings.log --path \*путь до проекта\***

# CI: Build AdHoc

## Steps

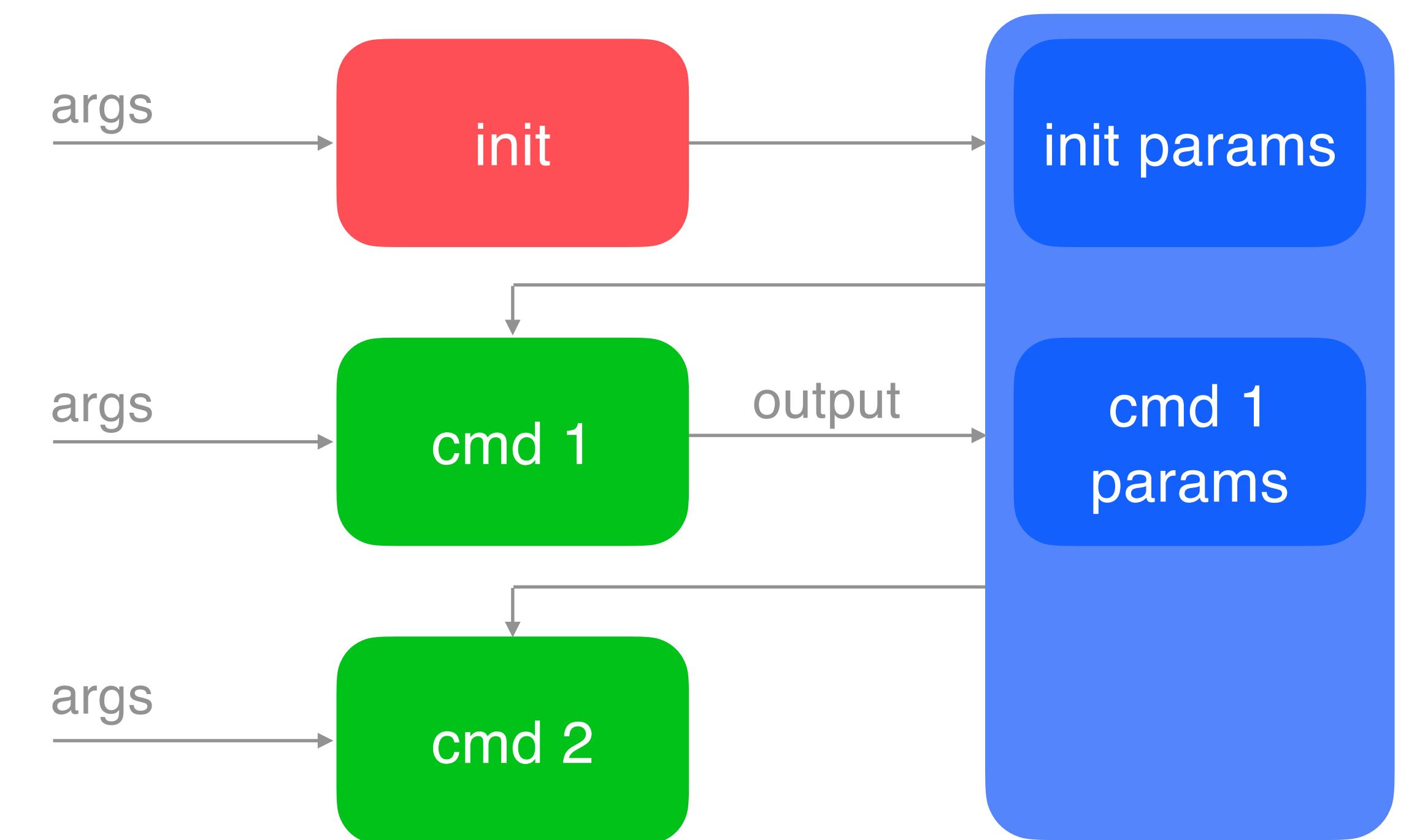
- **tools init --path ... --auth-token ...**
- tools issue-info
- tools pr-info
- tools xcode-archive --configuration Debug --colored-logs
- ...
- tools lint --config \*путь до конфига\*
- tools import-check --xcodebuild-log-path \$PWD/.ci\_build/xcodebuild.log
- tools warnings-check --warnings-logs-file-path .ci\_build/warnings.log

# Tools init

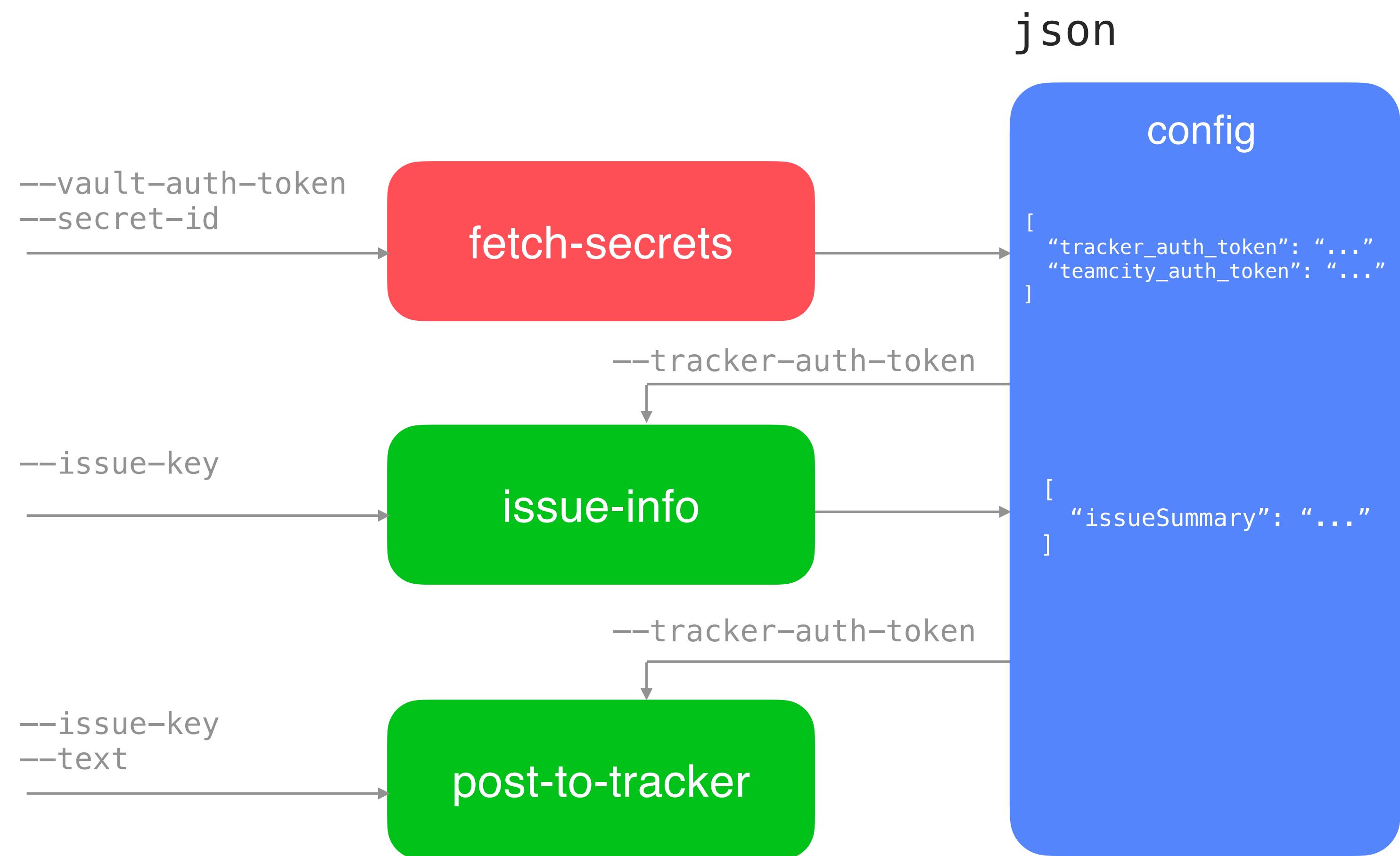
a.yaml

```
- tools init
--root-path "$PWD"
--build-number "$BUILD_NUMBER"
--project-path "$PWD/AutoRu.xcodeproj"
--workspace-path "$PWD/AutoRu.xcworkspace"
--scheme AutoRu
--derived-data-path "$PWD/.ci_build/derived_data"
--profiles-path "$PWD/Profiles/adhoc"
--archive-path "$PWD/.ci_build/AutoRu.xcarchive"
--export-path "$PWD/.ci_build/export"
--branch "$BUILD_BRANCH"
```

json



# Command output



# Command output

```
public protocol AsyncParsableCommandWithOutput: AsyncParsableCommand {  
    associatedtype Output  
  
    mutating func run() async throws -> Output  
}
```

# CommonBuildTools

## Часть общих команд

**xcode-archive**

Собирает архив проекта

**xcode-test**

Запускает все или указанные тесты в схеме

**upload-build-to-tf**

Загружает .ipa в TestFlight

**post-to-telegram**

Постит сообщение в указанный Telegram чат/канал

**upload-symbols-to-appmetrica**

Загружает dsym-ы в AppMetrica

**upload-file-to-s3**

Загружает произвольный файл на s3

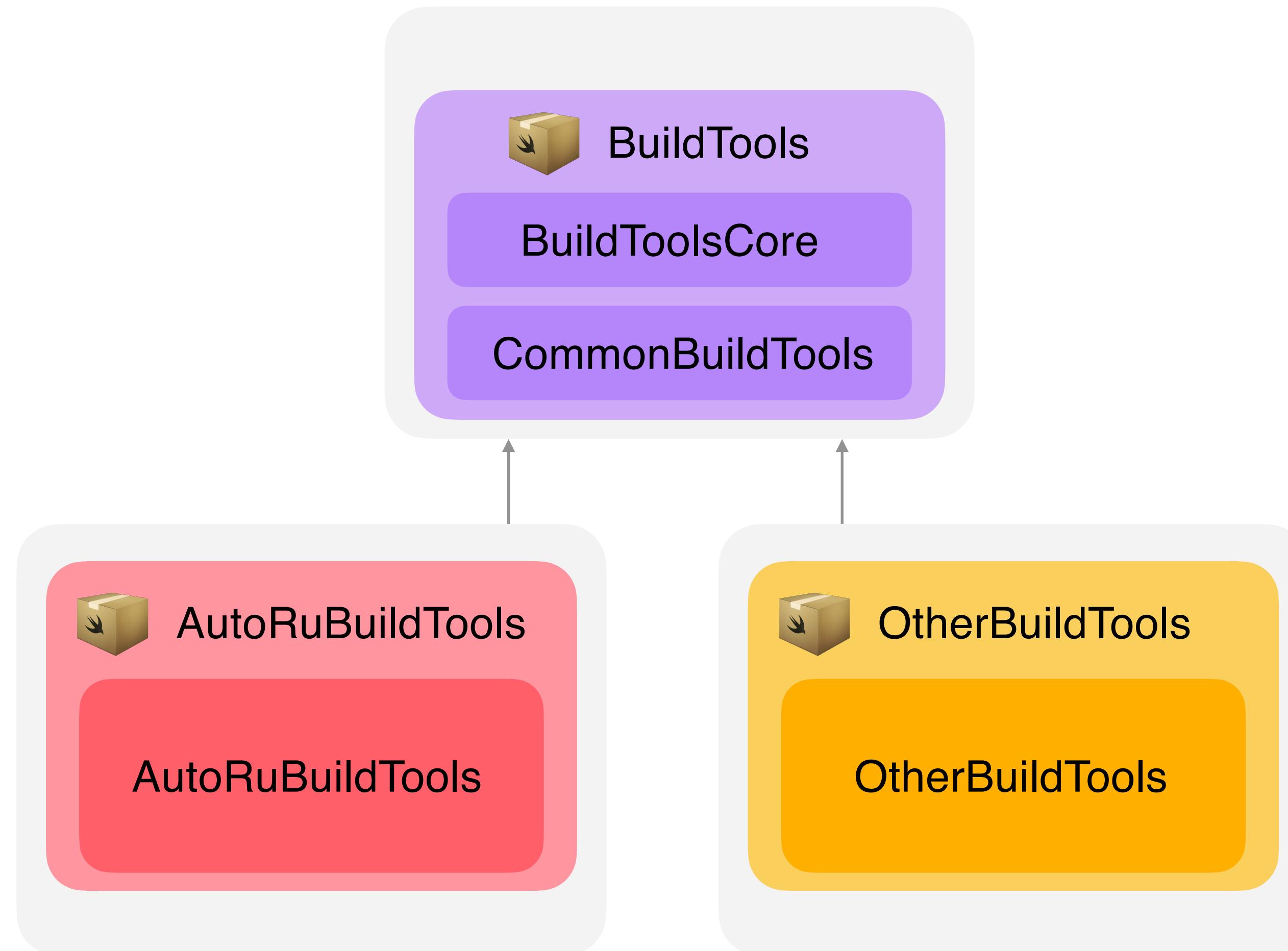
**destination**

Находит симулятор с указанными параметрами, или создает его, если не найден

**update-xcconfig-value**

Обновляет значение в xcconfig по ключу

# Build tools



```
public protocol CommandGroup {  
    @CommandsBuilder  
    static var commands: Commands { get }  
}  
  
import BuildToolsCore  
  
@main  
struct RootCommandGroup: CommandGroup {  
    static var commands: Commands {  
        CommonBuildTools.self,  
        MyProjectCommand.self,  
    }  
}
```

# Выводы

01

Открывается много новых возможностей для взаимодействия с кодом проекта и его структурой

02

Удобный API для разработки CLI интерфейса

03

Лёгкий онбординг для iOS разработчиков

**auto.ru**



**Спасибо за внимание!**

@ShamrinIgor