Mobile CI/CD at TomTom

Dmytro Vorobiov

TOMTOM

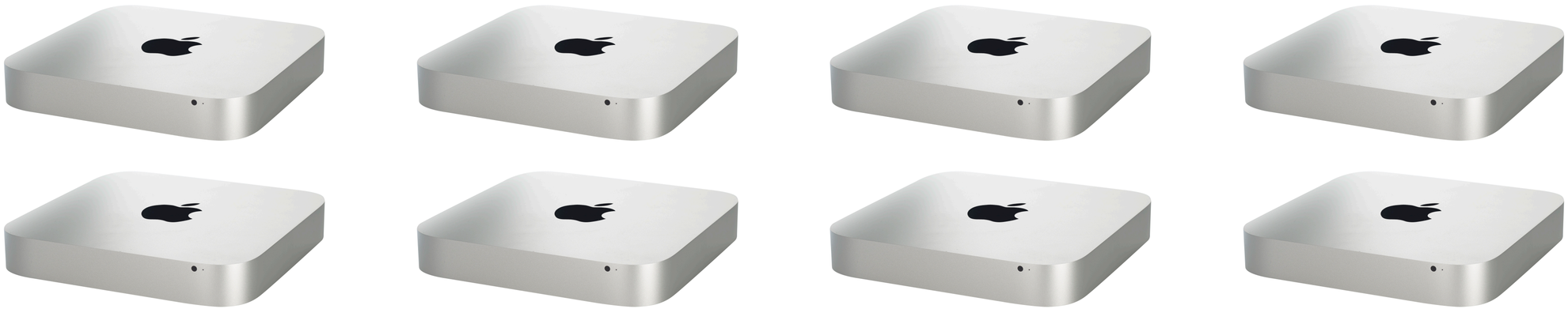
Agenda

1. Intro
2. Old setup
3. How we evolved: Fastlane integration
4. Benefits and examples
5. Q&A

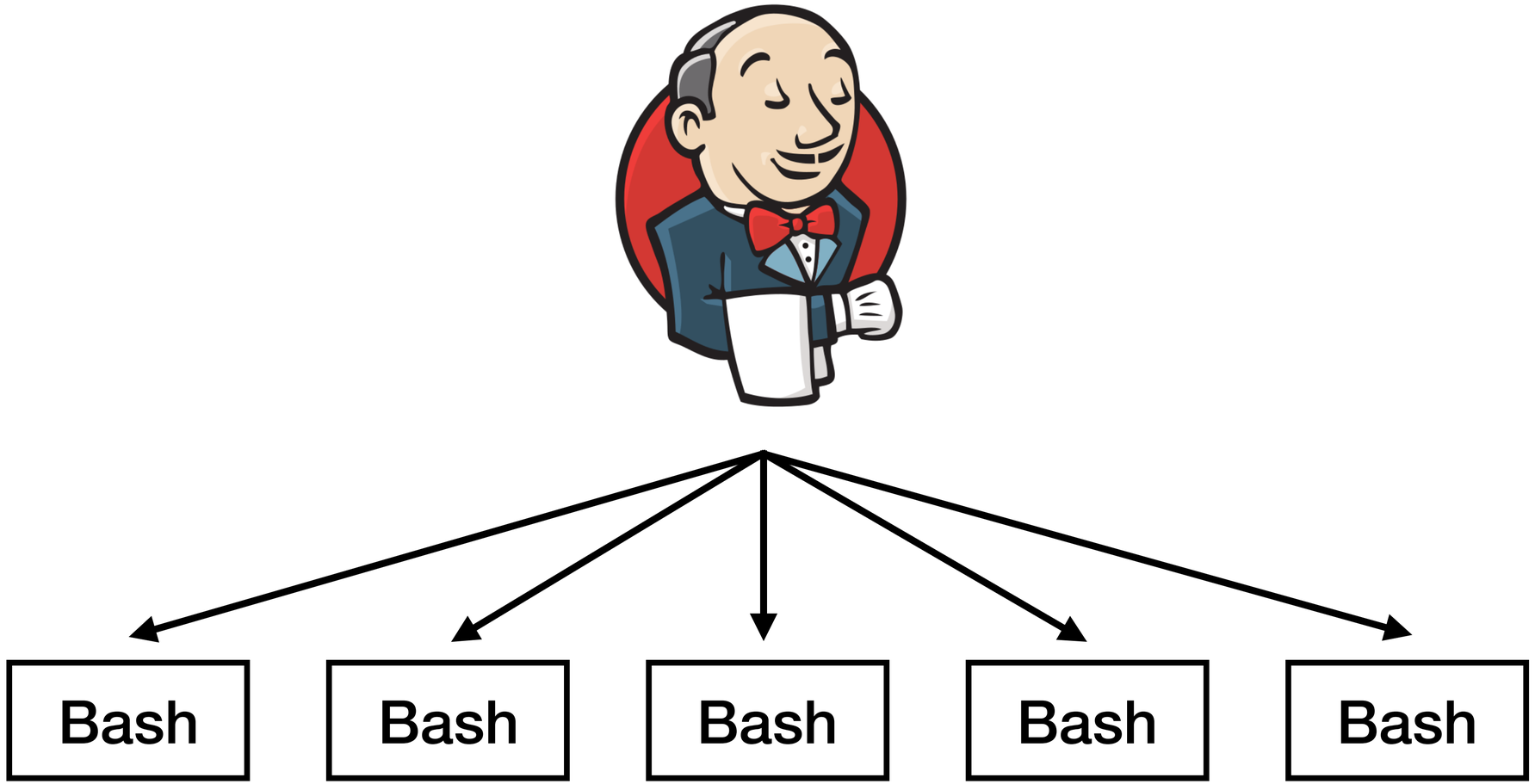
Our team

- 2 apps
- 4 locations
- 36 people in total

Our CI



3 years ago...



3 years ago...

xcodebuild build

3 years ago...

```
# xcodebuild will ignore the icon sets unless the timestamp is recent
touch "${TT_SOURCE_ROOT}/Resources/Images.xcassets/AppIcon.appiconset/"*.png

for CONFIGURATION in ${TT_IPA_BUILD_CONFIGURATIONS} ; do
    check_ccache_for_configuration ${CONFIGURATION}

    for BUNDLE_ID in ${TT_APP_BUNDLE_IDS} ; do
        BUILD_SUBPATH="build/${CONFIGURATION}-${TT_BUILD_PLATFORM}"
        PRODUCTS_SUBPATH="${BUILD_SUBPATH}/Products"
        INTERMEDIATES_SUBPATH="${BUILD_SUBPATH}/Intermediates"
        APP_SUBPATH="${PRODUCTS_SUBPATH}/${TT_APP_DISPLAY_NAME}.app"

        PLIST_SUBPATH="${TT_SOURCE_ROOT}/Classes/izmir-Info.plist"
        /usr/libexec/PlistBuddy -c "Set :CFBundleVersion ${TT_APP_BUILD_NUMBER_STRING}" "${PLIST_SUBPATH}"
        /usr/libexec/PlistBuddy -c "Set :CFBundleShortVersionString ${TT_APP_VERSION_STRING}" "${PLIST_SUBPATH}"
        /usr/libexec/PlistBuddy -c "Set :CFBundleDisplayName ${TT_APP_DISPLAY_NAME}" "${PLIST_SUBPATH}"
        /usr/libexec/PlistBuddy -c "Set :CFBundleIdentifier ${BUNDLE_ID}" "${PLIST_SUBPATH}"

        if [ "$CONFIGURATION" == "Debug" ]; then
            CODE_SIGN_IDENTITY="iPhone Developer"
            PROVISIONING_PROFILE=""
        elif [ "$CONFIGURATION" == "Release" ]; then
            CODE_SIGN_IDENTITY=""
            PROVISIONING_PROFILE=$(for f in ~/Library/MobileDevice/Provisioning\ Profiles/* ; do grep -e "${BUNDLE_ID}" "$f" >/dev/null && security cms -D -i "$f" | perl -pe
                's/(data|date)>/string>/g' | plutil -convert json -r - -o -  | perl -MJSON::PP -we 'my $bundleID = shift @ARGV; use strict; my $profile =
                decode_json(join("",<>)); my $UUID = $profile->{UUID}; die "Missing UUID for profile\n" unless $UUID; my $testflight = $profile->{Entitlements}->{"beta-reports-
                active"} || 0; my $development = $profile->{Entitlements}->{"get-task-allow"}; my $appID = $profile->{Entitlements}->{"application-identifier"}; my $name =
                $profile->{Name}; do { warn "Matched provisioning profile $UUID ($name)\n" ; print $UUID; exit 0 } if !$testflight && !$development && $appID =~ m/^\w+\.
                $bundleID/; exit 1' "${BUNDLE_ID}" && break ; done || true)
            test -z "${PROVISIONING_PROFILE}" && errorlog "buildProblem: could not find matching Ad-Hoc profile" && teamcitylog "buildProblem: could not find matching Ad-Hoc
                profile" && exit 1
        else
            errorlog "buildProblem description='Wrong build configuration'"
            teamcitylog "buildProblem description='Wrong build configuration'"
            exit 1
        fi

        xctool \
            -workspace ${TT_SOURCE_ROOT}/${TT_PROJECT_NAME}.xcworkspace \
            -scheme ${TT_PROJECT_NAME} \
            -sdk iphoneos \
            CODE_SIGN_RESOURCE_RULES_PATH='${TT_SOURCE_ROOT}/Resources/ResourceRules.plist' \
            -showTasks \
            -configuration ${CONFIGURATION} \
            -reporter pretty \
            DEBUG_INFORMATION_FORMAT=dwarf-with-dsym \
            TT_BUILD_IS_INTERNAL=${CF_BUILD_IS_INTERNAL} \
            TT_ENABLE_CRASHLYTICS=${CF_ENABLE_CRASHLYTICS} \
            POLYGLOT=${TT_ENABLE_POLYGLOT} \
            CODE_SIGN_IDENTITY="${CODE_SIGN_IDENTITY}" \
            PROVISIONING_PROFILE="${PROVISIONING_PROFILE}" \
            clean build \
            CONFIGURATION_TEMP_DIR="${TT_SOURCE_ROOT}/${INTERMEDIATES_SUBPATH}" \
            CONFIGURATION_BUILD_DIR="${TT_SOURCE_ROOT}/${PRODUCTS_SUBPATH}" \
            | grep -v -E '✓.*(Compile|Copy|Remove)|Libs.*Resources|SceneRendererData|\.png|write-file|chmod|Mapping of.* found in'

        if ! test ${PIPESTATUS[0]} -eq 0
        then
            teamcitylog "buildProblem description='Generate IPA: xctool failed.'"
            exit 1
        fi

        SYMBOLS_JSON_PATH="${TT_SOURCE_ROOT}/symbols_${CONFIGURATION}.json"
        if [ -e "${SYMBOLS_JSON_PATH}" ]; then
            teamcitylog "publishArtifacts '${SYMBOLS_JSON_PATH}'"
        fi

        mkdir -p Products
        PRODUCT_BASENAME="${TT_APP_DISPLAY_NAME}-${BUNDLE_ID}-${BUILD_NUMBER}-${CONFIGURATION}"
        rm -rf Payload && mkdir Payload && ln -s "${TT_SOURCE_ROOT}/${APP_SUBPATH}" Payload && zip -1qr "Products/${PRODUCT_BASENAME}.ipa" Payload
        tar -C ${TT_SOURCE_ROOT} -jcf "${TT_SOURCE_ROOT}/${PRODUCT_BASENAME}.dSYM.tar" ${APP_SUBPATH}.dSYM
        teamcitylog "progressMessage description='Generated product ${PRODUCT_BASENAME}.'"
    done
done
```

3 years ago...

- 1000 lines of bash code
- No version control
- Hard to maintain
- Easy to break

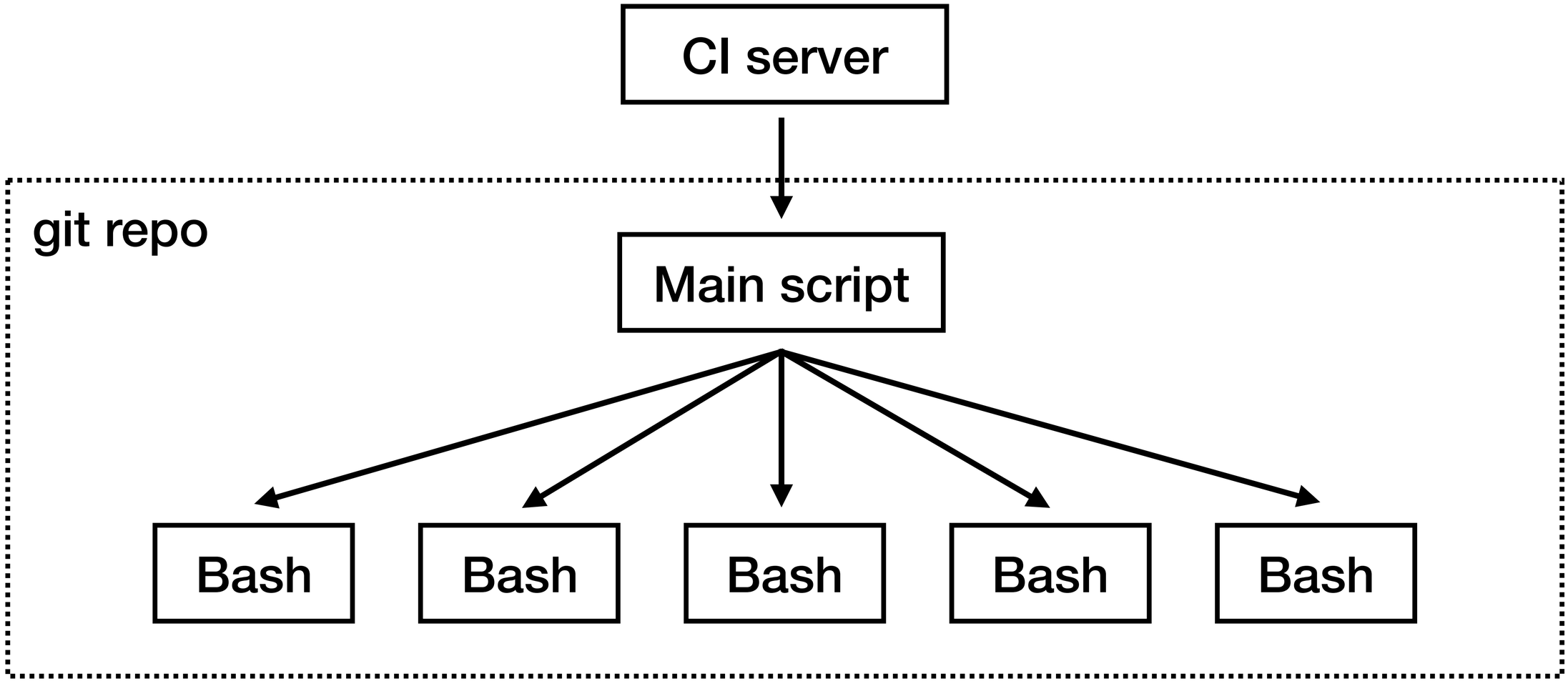
3 years ago...

That's enough!

Journey begins

- Move scripts to our repo
- Simplify setup on CI server as much as possible

One script to rule them all



One script to rule them all

One script to rule them all

- Swift is not a scripting language

One script to rule them all

- Swift is not a scripting language
- Breaks with every Xcode update


fastlane

fastlane

scan

Pilot

gym

deliver

snapshot

sigh

match


fastlane


deliver

fastlane

scan

Pilot

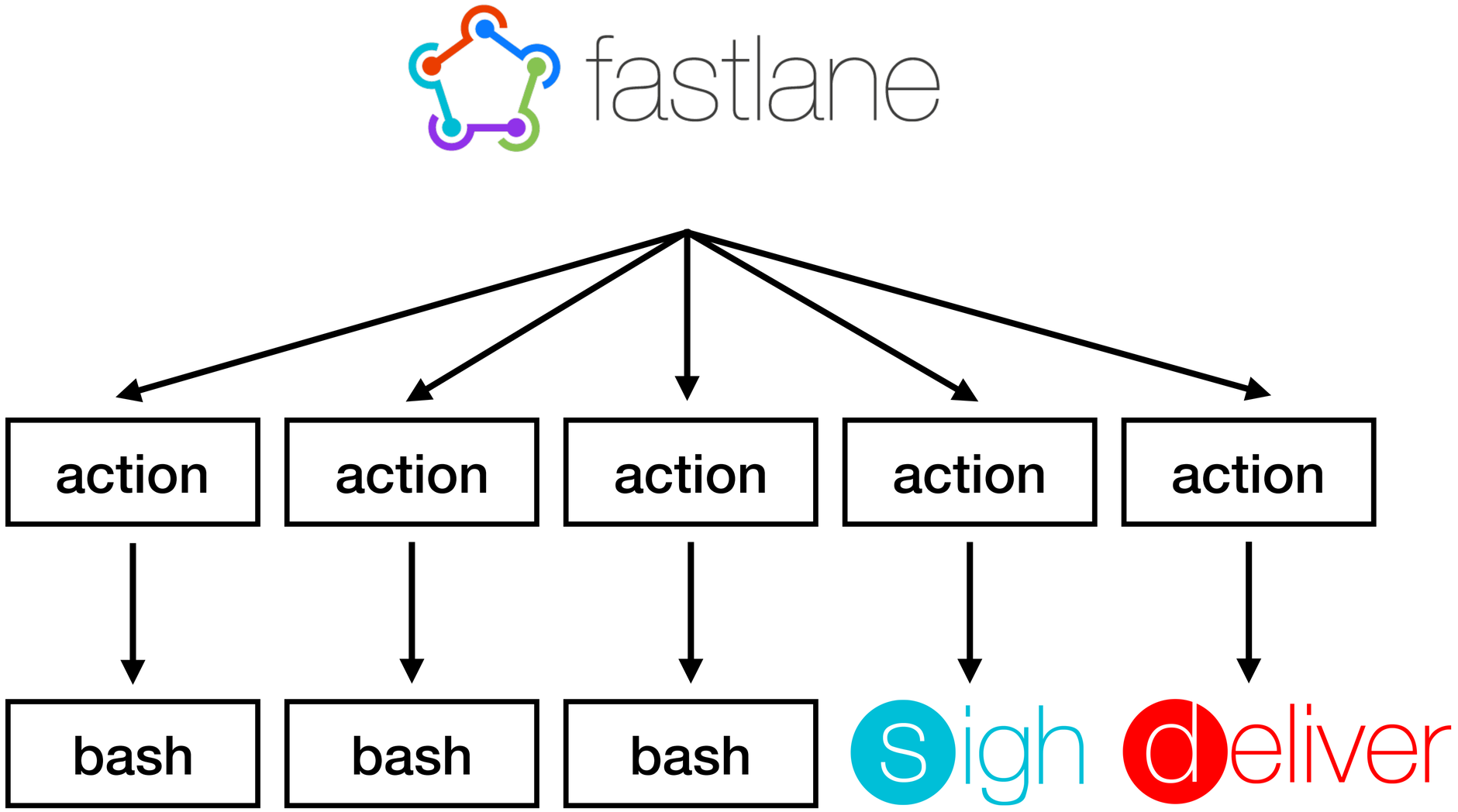
gym

deliver

snapshot

sigh

match


fastlane
action
action
action
action
action
bash
bash
bash
sigh
deliver


```
module Actions
    class TtIpaAction < Action
        def self.run(params)
            sh './Tools/install/generateIpa.sh'
        end
    end
end
```


```
module Actions
    class TtAppstoreAction < Action
        def self.run(params)
            config = params[:config]

            team_id = config[Config::DEVELOPMENT_TEAM]
            ipa = TTCommon.ipa_path_for_itunesconnect(config, team_id)

            other_action.appstore(
                app_identifier:    config[Config::BUNDLE_ID],
                username:          config[Config::ITUNES_AUTOMATION_USERNAME],
                team_id:           team_id,
                ipa:               ipa,
                app_version:       config[Config::APP_VERSION],
            )
        end
    end
end
```


```
desc 'Generate ipa and do all related actions if needed (like resigning ipa for the App Store distribution).'
lane :build do
    if config[Config::MARK_ICON]
        tt_icon
    end

    tt_ipa

    if config[Config::RESIGN]
        tt_resign
    end

    if config[Config::ENTERPRISE_RESIGN]
        tt_enterprise_resign(config: config)
    end
end
```


```
desc 'Do various post-build actions, like uploading to iTunesConnect, etc. Primarly used on CI.'
lane :postbuild do
    if config[Config::RUN_ANALYSIS]
        tt_analysis
    end

    if config[Config::CRASHLYTICS_BETA_UPLOAD]
        tt_crashlytics_beta(config: config)
    end

    if config[Config::TESTFLIGHT_PUBLISH]
        tt_testflight(config: config)
    end

    if config[Config::APPSTORE_PUBLISH]
        tt_appstore(config: config)
    end

    if config[Config::SLACK_ON_SUCCESS]
        tt_slack(config: config)
    end
end
```

Config

```
base:
    # Server environment to use.
    # Values: staging, preproduction, production
    environment: staging

    # Is build internal or not.
    # Internal builds includes logging, debug menu and other debug features.
    internal_build: true

    # Send crash reports to Crashlytics.
    crashlytics: false

    # Possible values: "default" or Xcode version string: "10.0", "9.4.1", etc.
    xcode_version: "default"

    ...
```

Config

```
tc_app_a_testflight:
    include_config: base

    internal_build: false
    build_configuration: release
    resign: true
    testflight_publish: true
    slack_on_success: true
```

Config

```
tc_app_a_testflight:
    include_config: base

    internal_build: false
    build_configuration: release
    resign: true
    testflight_publish: true
    slack_on_success: true
```

On TeamCity:

Custom script: *

Enter build script content:

```
bundle exec fastlane ci_job config:tc_app_a_testflight
```

On TeamCity

App A Debug

App A Smoke Tests

App A TestFlight

App A AppStore

App B Debug

App B Unit Tests

App A Crashlytics

App A TestFlight

On TeamCity

App A Debug

Enter build script content:

```
bundle exec fastlane ci_job config:tc_app_a_debug
```

App A Smoke Tests

Enter build script content:

```
bundle exec fastlane ci_job config:tc_app_a_smoke_tests
```

App A TestFlight

Enter build script content:

```
bundle exec fastlane ci_job config:tc_app_a_testflight
```

App A AppStore

Enter build script content:

```
bundle exec fastlane ci_job config:tc_app_a_appstore
```

App B Debug

Enter build script content:

```
bundle exec fastlane ci_job config:tc_app_b_debug
```

App B Unit Tests

Enter build script content:

```
bundle exec fastlane ci_job config:tc_app_b_unit_tests
```

App A Crashlytics

Enter build script content:

```
bundle exec fastlane ci_job config:tc_app_b_crashlytics
```

App A TestFlight

Enter build script content:

```
bundle exec fastlane ci_job config:tc_app_b_testflight
```

Dev perspective

```
./install.sh
```

Dev perspective

```
./install.sh

./install.sh environment:preproduction internal_build:false
```

Dev perspective

```
./install.sh

./install.sh environment:preproduction internal_build:false

./install.sh config:tc_app_a_smoke_tests
```

Dev perspective

```
./install.sh

./install.sh environment:preproduction internal_build:false

./install.sh config:tc_app_a_smoke_tests

bundle exec fastlane ci_job config:tc_app_a_smoke_tests xcode:10.0
```

Benefits

- Simple setup
- Good scalability
- Easy to adjust for any needs
- Faster on-boarding
- No dependencies on specific CI server

Questions?

Contact info: https://dvor.me