iOS Release Process (WIP)

> This is a work in progress - Omar Miskinyar to document with Connor Barattini

Team Members

- Connor Barattini

References

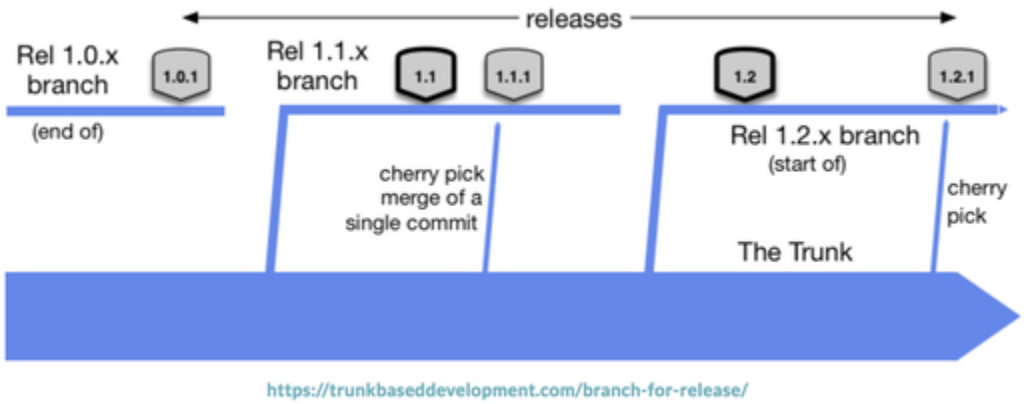
- Mobile Team Process
- iOS App Release Guide

Current Coverage

Platform	Source Code	Automated Build Software	Build Failure Mechanism	Unit Test Coverage	Integrated Tests	Functional Tests/Automation Software	Automation Source Code	Automation Failure Mechanism	Release Manager Role	Confluenc
iOS	Git=iOS		N/A	iOS=7.5%	N/A	N/A	ios-turncar automation tests		• Connor Barattini • Sweta Shinde	iOS Team

Branch Strategy Trunk Based Development

- There is only one branch, there are no other branches, so there is no merging. No merging equals no merge conflicts.



Release Process for iOS

(Step 1) Update Build and Version Number for Release

1. Update version number under project.pbxproj in Turn

(Step 2) Complete Final Merge into Github

1. Submit final branch/feature set into the Turn-iOS git repository for review
    a. This can be complete manually or through Sourcetree.
    b. List code reviewers to approve code.
        i. Code analysis (Peer Review) **APPROVED** `trunk/master`
2. Merge final branch/feature set after approved of code review.

(Step 3) Build Staging App for QA Testing

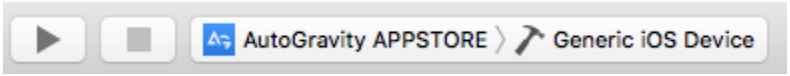
1. In AppCenter, build latest development branch under Turn Staging or Athon Staging. (Depending on build)
    a. Under TurnAthlon or TurnCar Apps Staging (variant) Build Select develop branch
    b. This will build the application on the last merge to the development branch.
2. Move ticket to *Ready for QA* under the Turn Project
    a. Add a comment specifying the build number and version. (Example: *Ready for QA in AppCenter build 2.1.1(10) for Turn Staging*)

(Step 4) App goes through QA Testing

1. If bugs are report, those must be fixed and sent through Steps 2 & 3 again
2. INFRA ticket
    a. INFRA Ticket Must be Cleared
        i. Creating INFRA Tickets
            1. This can happen anytime from the first commit preferably on 'Staging' branch aka 'Release Branch' but should be done when at this point
                a. This needs to happen in **unison with the commit where the app version** is updated
                b. There is code in the app that says if the version does not match in DB (supported versions table - utms .version_support) Update Fails
                    i. Suggestion cleaner versioning to help bypass?
        ii. QA & Stakeholder Must Approve
            1. INFRA ticket to deploy to production is created
            2. Creating INFRA tickets are required before any production environment change/release (script, new build, service restart)
            3. The INFRA ticket should contain a reference/description to all the tickets/features included in the release
            4. QA team managers should approve the INFRA tickets then an official approval should be provided by a director level or higher
            5. The team agreed to post the INFRA ticket to the QA slack channel to provide visibility then ask for confirmation from the QA team

Submit to Apple

1. **Submit to Test flight** (30 mins or 8 hours)
2. **Submit For Review** Apple (2 hours or 48 hours)
3. **Release to the app store** (30 mins or 6 hours)

- **Review/Release Apple Store Release Notes**
    1. Production releases from the *staging* *branch once all testing has passed* ***release candidate***
    2. Use **git pull** to make sure the latest code is pulled in, including last-minute bug fixes.
    3. Bump the app version and extension versions match the new release target
    4. **IMPORTANT:** Make sure you to have a corresponding version number entered into the utms.version_support table for your app. If this entry is not present, the app will be forced down an endless forced upgrade loop.
        a. If this entry isn't already present, someone with Prod DB access will need to enter it.
        b. Choose the ***APP STORE*** scheme by selecting <YOUR APP APP STORE> from the scheme drop-down
        c. 

        d. It is important to note that prior to the building you need both the Distribution Certificate and Distribution Provisioning Profile for your app installed on your machine.If you do not have the AutoGravity Distribution certificate installed, talk to a member of the team that has the .p12 key installed and have them extract it to send to you.
        e. In Keychain Access you should see something like:
        f. 

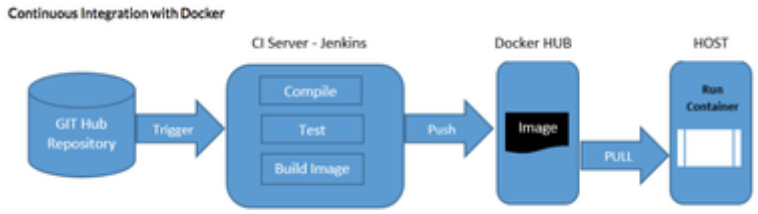
    5. Once you have the certificate key installed, you'll need to verify you have the iOS distribution provisioning profile installed from the developer center.
    6. Once both are properly installed, you should see the following for app signing:
    7. We can now Archive the app, by selecting **Product** > **Archive** from the Xcode menu.
    8. While archiving, ensure things are ready in iTunes Connect.
    9. Log in to iTunes Connect, and select the app you are building for
    10. Add a new version corresponding to the one you are creating
    11. Once the new version is created, click that version and ensure you have entered What's New copy. This can usually be obtained

from someone on the product team

12. After archive completes, choose the **Upload to App Store...** button in the Xcode Organizer
    - Follow through the Xcode upload prompts by selecting **Next**
13. Once the upload has completed, the build will need to "Process" in iTunes Connect before you are able to select it for submission (This process can take anywhere from 5 minutes to a few hours)
14. When the build finishes processing in iTunes Connect, go to the **Build** section and Add the version you just uploaded, and "Save" the updated settings
15. You can now select "**Submit For Review**" for this version of the app
16. On the prompt asking about advertising, you'll need to select Yes, and then tick the 2 bottom checkboxes
17. Continue through the flow until there are no more options
18. Your app has been submitted for review
19. Scrum Master or Release Manager Sends out *Release Notes*

Hotfix and Unplanned Releases

1. Branch off `master` Verify Fix Cherry-Pick back to Release/Production
2. Code analysis (Peer Review) **APPROVED**
3. Jenkins triggers a build
    a. Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests Manual/QA Verify Fix **PASSES**
    b. Event Notification
        i. Slack to notify events/status
        ii. This should go up on Screen/Board if developers are co-located and email notification should be sent out
4. Deploy fix to Staging Docker



    a. Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests Manual/QA Verify Fix **PASSES**
    b. Event Notification
        i. Slack to notify events/status
        ii. This should go up on Screen/Board if developers are co-located and email notification should be sent out

- **Deploy to Production Cherry-Pick to Release**

1. Promote Bumped Version Release.YYYYMMDD.[SequenceNumber], HotFix.YYYYMMDD.[SequenceNumber] Semantic
    a. The sequence number is the production build sequence number + 1
2. Dependency Analysis
    a. Microservices
        i. Release Manager checks for dependencies
    b. Feature Flag On/Off
    c. Library dependency analysis
3. ~~INFRA Ticket Approved~~
4. Mobile iOS Stakeholders Approval Apple Store Production
    a. **Submit to Test flight** (30 mins or 8 hours)
    b. **Submit For Review** Apple (2 hours or 48 hours)
    c. **Release to the app store** (30 mins or 6 hours)
5. Release Notes
    a. Scrum Master or Release Manager Sends out *Release Notes*