How an App gets into F-Droid

From source code to an app on your Phone
Walkthrough

- What’s F-Droid
- App submission
- Packaging
- Building
- Updating
- Distribution
F-Droid

• F-Droid is an App Store
• F-Droid is an installable catalogue of FOSS applications for the Android platform.
• Modeled after Debian
F-Droid cont.

- F-Droid as an ecosystem
  - F-Droid Client app
  - F-Droid server tools
  - F-Droid main repository
  - Guardianproject repo
  - f-droid.org website
  - Repomaker
  - ...

App submissions

- Rfp (Request for packaging) issue tracker
  - https://gitlab.com/fdroid/rfp
- We need
  - Link to source code
  - License
  - Description/Summary/etc.
  - Build instructions
Packaging

- Current fdroiddata maintainers:
  - @relan
  - @mimi89999
  - @Bubu
  - @Rudloff
  - Others for certain apps
Packaging cont.

- **OR:** package it yourself
- You’ll need the fdroidserver tools and either:
  - Fdroid buildserver VM (> 100 GB req.)
  - Existing android dev setup (with some caveats...)
- Create a txt (old) or yml (preferred) metadata file
- Submit a merge request to fdroiddata
Packaging cont.

- Clone fdroiddata
- Run `fdroid init` inside the fdroiddata repo
- Edit `config.py`
  - Path to `gradle` needs to be set (needs to be the correct version, this is a bit of a PITA still)
  - Android SDK and NDK paths
- `fdroid init` will also generate some signing keys for you for testing the built apps on a real device
Demo Time!
Packaging cont.

- In fdroiddata:
- `fdroid import -u https://github.com/wtcounter/wtcounter -l GPL-3.0-only -s app`
- Edit `metadata/wordtextcounter.details.main.yml`
  - Add description, summary and categories
  - Fill in commit/tag
  - Add Auto Update Mode
  - Test fdroid checkupdates
- Test the build
- Fix build errors
Packaging cont.

- Run `fdroid lint` and `fdroid re writemeta` to discover possible problems
  - Careful with `rewritemeta for yml`, it will swallow unknown keys (typoes...)
  - Run `git add <file>` first
- Finally create a merge request!
Finished metadata

| Categories: | Writing |
| License: | GPL-3.0-only |
| SourceCode: | https://github.com/wtcounter/wtcounter |
| IssueTracker: | https://github.com/wtcounter/wtcounter/issues |

**AutoName:** Word Text Counter  
**Summary:** Count words, characters, sentences, paragraphs etc in a given Text  
**Description:**  
Word Counter is a free and easy to use text tool for counting words, sentences, para[...]  

**RepoType:** git  
**Repo:** https://github.com/wtcounter/wtcounter

**Builds:**  
- **versionName:** '2.0'  
  **versionCode:** 2  
  **commit:** v2.0  
  **subdir:** app  
  **gradle:**  
  - yes  
  **prebuild:** sed -i -e '/keystore.credentials/d' build.gradle

**AutoUpdateMode:** Version v%v  
**UpdateCheckMode:** Tags  
**CurrentVersion:** '2.0'  
**CurrentVersionCode:** 2
Packaging Gotchas

- Common problems:
  - Jar or aar files inside the repo
    - Everything must be built from source or pulled from a trusted maven repository (jcenter, mavencentral and a few others)
    - Trusted here means they require a source jar to be uploaded alongside the binary
Packaging Gotchas

• Common problems (cont.):
  - Using proprietary dependencies (“usual suspects”)
    • Firebase/GCM
    • Crashlytics
    • Google play services
  - Best solution is contributing a build flavour that doesn’t need these dependencies upstream.
Packaging Gotchas

• Common problems (cont.):
  - No tags
  - No commit messages (!)
  - No license
  - Incompatible license (GPL-2.0 vs Apache2 from Android support libraries)
  - Source code is only updated occasionally
  - ...

Building

- Started with `fdroid build -v <appid:vercode>`
  - Get’s the source
  - Always resets to target commit
  - Scans for common problems
  - Applies patches/prebuild commands
  - Runs commands specified in build:
    - `Runs gradle assemble<Flavour>Release`
  - Verifies resulting apks VersionName and VersionCode match
Updating

- Server runs `fdroid checkupdates --auto` ~once a day
  - Generates new build entries if an update is detected.
- There is `UpdateCheckMode (UCM)` and `AutoUpdateMode (AUM)`
  - UCM is for detecting new versions
  - AUM generates the build entries
- Most common method is `UCM:Tags, AUM:Version %v`
Updating cont.

- Needs versions to be correctly tagged and VersionName correspond to the tag name
  - there can be a prefix like \texttt{v\%v}
- Additionally VersionName and VersionCode need to be statically set in build.gradle or AndroidManifest.xml
- Dynamically calculated Versionnames/codes are not supported for auto update yet.
  - They’d require running gradle to handle correctly
Distribution

- `fdroid build` generates unsigned apks
- They are signed with `fdroid publish`
  - This happens on a separate offline signing machine for f-droid.org
- `fdroid` generates one signing key per app, unless explicitly configured otherwise in config.py
  - Apps sharing a signature have a weaker isolation
  - This is required i.e. when one app needs to access accounts from another app
- Optional gpg signing of apks
Distribution cont.

- Now **fdroid update** can assembles the app index of all locally present signed apks.
  - It also copies together all app metadata which might come from upstream repos [https://f-droid.org/en/docs/All_About_Descriptions_Graphics_and_Screenshots/](https://f-droid.org/en/docs/All_About_Descriptions_Graphics_and_Screenshots/)
  - Also supports screenshots feature graphics and changelog entries
- Index gets signed with the repo signing key
  - It’s a `.jar` file which contains the `index-v1.json`
Distribution cont.

- The index contains the sha256sums of all apks distributed in that repo
- The index signing certificate is pinned in the client:

Repository

Address
https://f-droid.org/repo

Name
F-Droid

Description
The official F-Droid repository. Applications in this repository are built directly from the source code. (One, Firefox, is the official binary built by the Mozilla. This will ultimately be replaced by a source-built version.

Number of apps
1510

Last update
10:11

Fingerprint of the signing key (SHA-256)
43 23 8D 51 2C 1E 5E B2 D6 56 9F 4A 3A FB F5 52 34 18 B8 2E 0A 3E D1 55 27 70 AB B9 A9 C9 CC AB
Distribution cont.

- Client downloads the app index and verifies the embedded signature
- When you download an app the apk hash gets verified against the hash in the index
- Additionally android has a TOFU system for app signing keys
Questions?

Talk to us on IRC/Matrix: #fdroid / #fdroid-dev (on freenode)
Command Summary

- `fdroid import` → creates a metadata template
- `fdroid lint` → spot metadata issues
- `fdroid re writemeta` → bring metadata into canonical form (also converts between txt and yml)
- `fdroid build` → builds an unsigned apk
- `fdroid checkupdates` → checks for new versions && generates new build entries
- `fdroid publish` → signs all local unsigned apks
- `fdroid update` → creates and signs an index
Fdroid Buildserver

- A virtual machine used for building all apps in the main repo
- Libvirt or Virtualbox backed
- Based on Debian **jessie**, currently migrating to **stretch**
- Provisioned with **vagrant**
  - Installs all Android sdk tools/platforms
  - Most NDK versions
  - All gradle versions
  - Some more common dependencies
Buildserver cont.

• First you’ll need a vagrant basebox
  – Create one with
    https://gitlab.com/fdroid/basebox/
  – This will create a (mostly) vanilla Debian VM image usable by vagrant
• Then run ./makebuildserver to run all the fdroid provisioning
  – This will download lots of stuff
  – And will temporarily consume up to 100 GB of disk space
  – The final buildserver image will be around 30 GB in size
• Needs to be rebuild whenever you’re missing a dependency (new NDK, gradle versions, ...)

08.06.2018 – AfRA, Berlin
Marcus Hoffmann
Buildserver cont.

- Run builds with `fdroid build -v -server <appid:ver>`
- Will always start a fresh snapshot of the buildserver VM
- Copies currently used version of fdroidserver inside
- Copies all app source code and the metadata file
- Builds inside the VM with `fdroid build -on-server <app>`
- Copies resulting apk back if build was successful