Release process

This document describes the process by which versions of Ushahidi V3 are released. For information about the process by which Ushahidi V3 is developed, see Ushahidi v3.x, Development process.

Prepare

Set a release date

Set a tentative release date. Make sure this allows for some internal and community testing time.

Assign a release manager

Decide who is responsible for coordinating this release. This is the person who should manage the release checklist. They do not have to complete all the tasks, but are responsible for making sure someone completes them.

Ideally this should be the same person who is generating the builds. This role should be rotated around the dev team.

Clone checklist

Copy the checklist below to a github issue and use this to track the release process.

Notify Relevant Parties

Community

Notify project participants in the Skype dev chat and developers mailing list. Mention the tentative release date, and reference the github issue for tracking the release checklist.

Notifying IRC is not necessary (unless it returns to active use)

Ushahidi Team

Post the release date to the team page and google calendar.

Collect acknowledgements

Collect list of thank yous from : github pull requests, github issues, transifex, testing.

Request design assets

Request design for platform page link: http://ushahidi.com/products/ushahidi-platform (1 week advance)

Request Home Page Promo Graphic: http://ushahidi.com/ (1 week advance)

Create release candidate builds, distribute to testers

Tag the release

Releases should be generated from a tag regardless if they are a release candidate, alpha, beta, or final release.

For the client: bump first

gulp bump --type prerelease
git add bower.json package.json
git commit

This will bump the version in bower.json and package.json.
So you need to commit, tag and push afterwards.

Both client and API

Create a tag in git following semantic versioning by running the following:

git tag -as "v3.0.0-beta.1"

Note: -s creates a GPG signed tag. -a annotates the tag.
Use the changelog as the tag message.

Once you push the tag Travis-CI will generate build files and create a github release.

GPG Signatures

Builds should be distributed with a gpg signature (.asc) file. Download the generate builds from github releases and check the match the tagged code.
Generate signatures by running:

gpg -ab Ushahidi-Platform-v3.0.0-beta.1.tar.gz

These signature should be uploaded to github releases and linked on the download page

> ### If you need to manually generate build files
>
> #### For the API
>
> To create a build, run:
>
> ```
> bin/release v3.0.0-beta.1
> ```
>
> This will create .tar.gz files in build/
>
> #### For the client
>
> To create a build, run:
>
> ```
> gulp release --type prerelease
> ```
>
> This will create .tar.gz files in build/
>
> Note: the release script will bump the version in bower.json and package.json.
> So you need to commit, tag and push afterwards.

Draft release announcement

Draft a release announcement blog post for the Blog.

To draft the post, access the blog's Wordpress admin page and go to Posts > Add New. Use previous release posts as a template for your new one. Put the post into the ushahidi, code releases categories. Tag with Ushahidi 3.0

After drafting the post, share it with other folks who can help edit it, such as the community manager, other 3.x team members.

> Note: don't publish the post in this step. That should happen in the Release step.

Write release notes

Release notes live in the Readme. In future they should live on the wiki too. The release notes contain just the head line, important notices about this release and the state of the codebase. This should include known issues/bugs in the release.

Write change log

The changelog lives in the Changelog.md in the repository. The changelog should be a detailed list of everything that has changed since the last release. Include features added, bugs fixed.

Check licensing

Confirm that: a LICENCE file exists, files with licences other than the main project licence are mentioned in LICENCE, and licences comply with project policy (aka. are compatible with GNU Affero GPL). Ensure all text files contain correct licence header.

In theory this is done during development but it should be confirmed for any new libraries before release.

Review documentation

Confirm that documentation has been updated for changes made in this release.

Check that install instructions are up to date, and release can be installed by following these instructions.

Choose Candidate

Decide to ship a candidate as the final release.

Its OK if a release has bugs. Any release that has at least 1 less bug than a previous release is worth shipping. A release may introduce some bugs, you need to decide if other bug fixes and new features outweigh any new bugs.

Known bugs should be documented in the release notes.

Test and fix process

Testing

Testing starts once the first release candidate is created. Bugs found during testing should be logged in github issues.

Bug fixes

During testing developers should not land any new features on master. Developers can land bug fixes on master, provided they do not involve major changes or refactoring.

Create release candidates

New release candidate should be created regularly during testing, either after bug fixes are merged or at defined points (weekly?).

New release candidates do not need to be fully retested, however they should be tested 1. to confirm they fix the bugs they are supposed to 2. ensure major functions still work 3. any other likely issues flagged by developers.

The process for create candidate is documented above

Release

Schedule a release time

Ushahidi spans a lot of timezones. Schedule a release time and timezone. Make sure everyone involved in the release knows when this is for them.

Create final builds

The process for creating final builds is same as for creating a release candidate, as documented above.

Additionally check that

- No version control (Git) files are included in release files
- Source is exported from a tag in the version control system (Git)
- Compressed .zip and .tar.gz files are created
- Compressed distributions unpack correctly

OpenPGP keys

Confirm that the

- KEYS file contains signing key
- The key has been uploaded to regular public key servers
- Downloads are distributed with a gpg signature (.asc)

Upload files to the CDN

Upload the files to the the CDN. We use Rackspace Cloudfiles.
Upload the files to the ushahidi-downloads container (account: ushahidipro) under the V3 folder.

Review release checklist

Run through the release checklist, all key items must be complete before publishing the release announcement.

Update testing site

The current testing site: v3.ushahidi.com is hosted on Robbie Mackay's pagodabox account. Push to the git repository to update the codebase. Check it deploys successfully.

Publish Release Announcement

Go to the wordpress admin for the blog and publish the post you drafted earlier.

Notify relevant parties / Comms

Community

Notify project participants in the Skype dev chat, developers mailing list and community mailing list. Provide basic info on the new release and reference the release announcement blog post.

Notifying IRC is not necessary (unless it returns to active use)

Ushahidi team

Post to the team page and team skype chat, link to the release blog post.

Twitter, etc

Post on the Ushahidi Twitter and Facebook pages.

Host a google hangout

Host a google hangout or skype call to talk about and demo changes in the new release. Advertise this to the community via skype dev chat, the developers mailing list, and the ushahidi meetup page.

Security notices

Are there any security fixes? Document these on the security page Software Security Updates, in the blog post announcement and on the wiki.

Bask

Bask in the glow of the latest and greatest release!

Physically or virtually high-five or fist-jab contributors.

Review

Review the release process via a post-mortem, ask for feedback in the team call, guvna call, skype dev chat, and anywhere else as appropriate. Identify things that went well and we should continue to do, things that went badly that we should do differently next time, and parts of the process that have changed and for which this document needs to be updated. Make changes as appropriate.

Panic

Sometimes releases have to be pushed out more quickly, ie. when a previous release has a major bug, or for a publicly disclosed security issue. In future this process will be documented here.

Release Checklist

```
* Prepare
  * [ ] Set a release date
  * [ ] Assign a release manager
  * [ ] Clone checklist
  * Notify Relevant Parties
    * [ ] Community
    * [ ] Ushahidi Team
  * [ ] Collect acknowledgements
  * [ ] Request design assets
  * [ ] Create release candidate builds, distribute to testers
  * [ ] Draft release announcement
  * [ ] Write release notes
  * [ ] Write change log
  * [ ] Check licensing
  * [ ] Choose Candidate
* [ ] Test and fix process (continuous process - just confirm its happening)
* Release
  * [ ] Schedule a release time
  * [ ] Create final builds
    * [ ] No version control (Git) files are included in release files
    * [ ] Source is exported from a tag in the version control system (Git)
    * [ ] Compressed .zip and .tar.gz files are created
    * [ ] Compressed distributions unpack correctly
    * [ ] Downloads are distributed with a gpg signature, and the signing key is in the KEYS file.
  * [ ] Review release checklist
  * [ ] Update testing site
  * [ ] Publish Release Announcement
  * Notify relevant parties / Comms
    * [ ] Community
    * [ ] Ushahidi team
    * [ ] Twitter, etc
  * [ ] Host a google hangout
  * [ ] Security notices
* [ ] Bask
* [ ] Review
```

This document borrows heavily from:

http://oss-watch.ac.uk/resources/releasemanagementbestpractice

https://wiki.mozilla.org/index.php?title=Jetpack/Release_Process