Mardis du développement technologique

Brice Goglin – EPI Runtime

2013/02/05

Faire des releases logicielles : Pourquoi, Quand et Comment ?

Agenda

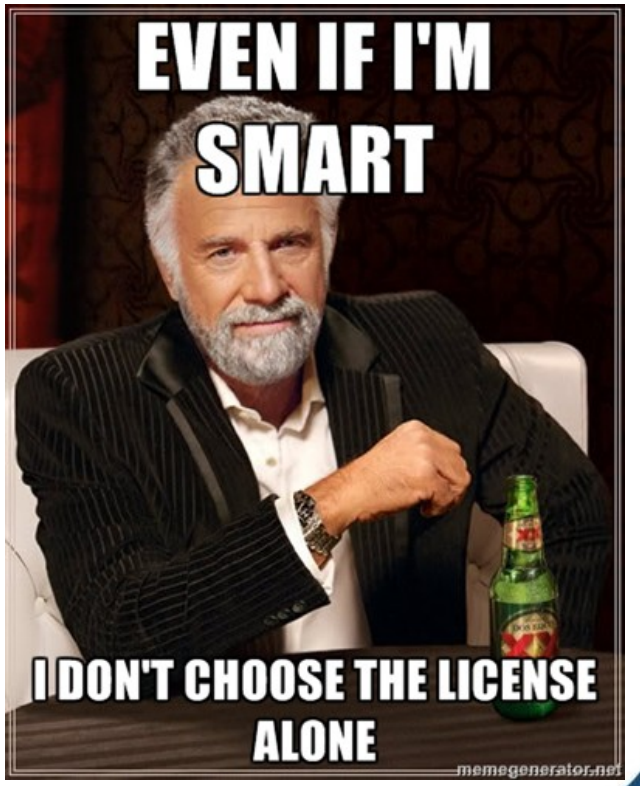
- Avant propos
- Pourquoi faire des releases ?
- Quand faire des releases ?
- Comment nommer mes releases ?
- Comment faire des releases ?
- Comment publier mes releases ?
- Après propos

1

Avant propos

Pourquoi publier du code ?

- On est ici pour faire avancer la société
  - Lui donner le fruit de notre travail
- Les résultats de recherche doivent être reproductibles
  - Logiciels disponibles en ligne ?
- Il **faut** l'accord de votre employeur
  - Le code lui appartient
  - Aller d'abord voir votre service valo
    - **Ce n'est pas à vous de décider !**



2

Pourquoi faire des releases ?

Pourquoi faire des releases ?

- Montrer qu'on est vivant
  - Aux partenaires, aux évaluateurs, aux concurrents
- Mais ça ne signifie pas forcément que ces gens vont utiliser nos releases...
- « Un logiciel sans utilisateur est un logiciel mort » – F. P.

Pourquoi faire des releases ?

- Il y a des gens qui veulent tester votre code
  - Quand on lit un article, s'il est intéressant, on a envie de regarder plus en détail
- Si un lecteur veut regarder mais n'y arrive pas, il va nous citer négativement
  - « Ca marche pas », « le code n'est pas disponible », « pas portable », « pas utilisable en production », …
    - Et ses lecteurs vont le croire sans prendre la peine d'essayer
- Le code doit être disponible
  - Et facile à utiliser !
    - Et marcher à peu près...

Mais ?! Mon SVN est public !

- L'utilisateur ne peut pas prendre le SVN facilement
  - « Quelle révision du SVN dois-je prendre ? »
  - « Quelle branche ? »
  - « Cette branche est-elle cassée en ce moment ? »
- Les releases permettent de clarifier tout ceci
  - Les releases sont testées avant publication
  - Leur contenu ne changera plus
  - On peut marquer développement, beta, expérimental, ...
    - L'utilisateur se doutera bien que ça ne marche pas parfaitement
    - Mais il pourra voir dans le ChangeLog que la feature qu'il cherche est bien là
      - Et jouer avec

3 Quand faire des releases ?

Planifier les releases

- Releases régulières tous les 6 mois ?
  - Comme Ubuntu
  - Bien pour les utilisateurs
    - Ils savent à quoi s'attendre et quand
  - Pas facile pour les développeurs
    - Ils ne dorment plus quand la deadline approche
  - Pas facile à tenir dans le monde de la recherche
    - On a d'autres deadlines à gérer...

Planifier les releases

- Releases basées sur les features ?
  - Bien théoriquement
  - Sauf quand les features prennent 2 ans de retard
    - Rappel : logiciel sans releases régulières = logiciel mort
    - Faire des points réguliers avec les développeurs pour mettre à jour la liste des features prévues dans la prochaine release

Éviter les grands délais

- Si gros délai entre les releases, les utilisateurs rapportent des bugs sans rapport avec le code que les développeurs manipulent tous les jours
  - Et les réponses des développeurs les frustrent
    - « C'est corrigé dans le trunk mais il est pas stable là, ça sera ok dans v1.8 dans 3 mois »
    - « Ah mais tu utilises une version antédiluvienne, forcément ça peut pas marcher aussi... »
    - Note : Il faut choyer les utilisateurs qui font des bug reports
- Et les features de la nouvelle version sont testées très tard, donc amènent plein de nouveaux bugs
  - Les utilisateurs ne seront jamais contents

Bien gérer les releases majeures

- Essayer d'avoir du testing tôt
  - Faire des release candidates dès que possible
    - C'est normal si une RC1 n'est pas parfaite
    - Faire 5 release candidates consécutives n'est pas anormal
  - Voire des beta avant (ou nightly snapshots)
- En parallèle, faire des releases stables pour corriger rapidement les bugs de la version majeure précédente
  - Ne pas attendre un « gros » bug fix pour releaser les 15 petits qui attendent depuis 1 an
    - Faire une release stable quand il y a un bug fix important OU un long délai depuis la release précédente
  - Un nightly snapshot en ligne peut aider entre temps

S'adapter aux événements

- Faire des releases juste avant les gros coups de pub
  - Présentation en conf
  - Démo sur un salon
  - Communiqué de presse
  - Tout ce qui va amener des gens à regarder soudainement
    - Ils vont vouloir essayer les nouvelles super features
  - S'ils viennent et ne peuvent pas tester, ils ne reviendront pas avant longtemps
    - Et vont vous citer négativement (bis)
- Profiter aussi des coups de pub des autres
  - Profiter du buzz lors de la sortie d'un nouveau matériel qu'on supporte déjà

Communiquer pour planifier

- Avec les collègues
  - Pour juger de la stabilité des branches avant les releases
  - Se synchroniser en amont pour éviter les imprévus
- Avec les utilisateurs
  - Pour juger de l'importance des bugs
    - Grave ou pas ? Rare ou pas ?
      - Justifie une release stable immédiatement ou pas ?
  - Diffuser la roadmap pour les tenir au courant ?

Comment NE PAS planifier les releases ?

- Trop serré
  - Si je sors 1.3.2 et 1.4 en même temps, les utilisateurs n'en testeront qu'une seule
  - Si je sors une RC, je laisse du temps avant la version finale pour que les gens testent la RC
    - Une semaine ?

Comment NE PAS planifier les releases ?

- Trop tôt
  - Releaser une interface de programmation non mature va causer des problèmes plus tard
    - Il faudra casser l'ABI, renommer des fonctions, ...
    - Marquer un bout d'API comme expérimental ?
      - Ne pas trop en abuser, sinon les utilisateurs ne l'utiliseront jamais
- Juste avant le week-end ?
  - S'ils n'ont pas le temps de tester avant le week-end, ils auront peut-être oublié d'ici lundi...

4 Comment nommer mes releases ?

Nommer mes releases

- Alpha, Beta, Dev, RC, Stable, …
  - Ces noms influent sur l'attrait des utilisateurs
    - Qui va vraiment installer une version Alpha ?
    - Linus appelle toutes ses preleases « RC »
      - Mais ils ont clarifié quels patchs sont acceptables dans les RC
- Utiliser des identifiants clairs de version
  - Que ce soit des numéros ou dates
  - Pour que les bug reports soient utilisables
    - Savoir facilement quelle version l'utilisateur a prise
      - Pour pouvoir reproduire le problème

Numéroter mes releases

- Le numéro de version doit être représentatif de la réalité
  - Ne pas attendre d'avoir TOUT implémenté avant de faire la version 1.0
    - Sinon la version 1.0 n'arrivera JAMAIS
      - Un projet de recherche n'est jamais terminé ?
  - Faire 1.0 qui **marche** même si **peu de features**
    - Puis faire 1.1 ou 2.0 quand les autres grosses features seront prêtes

Exemple de numérotation 1 : Une 1.0 arrivée beaucoup trop tard

1.0.0	October 8th 2012	ac61ddb6ff7fa261c76b3f9172057cad	3598f
Older releases:			
0.9.8	February 21st 2012	9f4ed565a333be68959ae19c41737b38	148db
0.9.7	July 29th 2011	7576a0b991075becc4e43b3fe16227b6	f4360
0.9.6	April 2nd 2011	9ae1a4066cb02cda0adbfede28072019	df1fa
0.9.5	February 1st 2011	794fb4e653948d9008c492d9eef0bd03	a6272
0.9.4	December 16th 2010	ed4aab1b04b1718b3c1ac8daa06a4399	4aad
0.9.3	October 20th 2010	a26b52713da37aba7492e63126643e4d	96de
0.9.2	August 30th 2010	e1e7a3254096aca26559958964d7f005	08789
0.9.1	July 6th 2010	6bf08346536d6e975555925341ab2062	af3a6
0.9.0	June 14th 2010	2b10c1daee8a260023b41d81c19f3f52	21ad5
0.8.901	May 15th 2010	64fce0f69689c5da19100fa70d2fff3e	09b0d
0.8.1	March 22nd 2010	f9849456e624b333f7100602a731de3a	cdee1
0.8.0	March 3rd 2010	8543e2aba5272d8caff1c828bae48ffe	069f9

Exemple de numérotation 2 : Une 1.0 presque trop tard

1.0.1	29.03.2012	[illegible]-1.0.1.tar.gz	
1.0.0	28.03.2012	[illegible]-1.0.0.tar.gz	
0.9.2	06.10.2011	[illegible]-0.9.2.tar.gz	
0.9.1	13.05.2011	[illegible]-0.9.1.tar.gz	
0.9-win32	12.05.2011	[illegible]-win32-build.zip	s
0.9	12.05.2011	[illegible]-0.9.tar.gz	
0.4-win32	25.08.2010	[illegible]-win32-build.zip	s
0.4	25.08.2010	[illegible]-0.4.tar.gz	
0.2	04.07.2009	[illegible]-0.2.tar.gz	
0.1	02.04.2009	[illegible]-0.1.tar.gz	

Numéroter mes releases

- Ne pas sous-estimer l'impact psychologique du numéro de version
  - 0.5 = « à moitié prêt/utile/utilisable/stable/... »
  - 0.10 = pire
    - Ca semble 10 % utilisable/stable/...
    - Et ca ne convergera jamais vers 1.0
      - Donc ne sera jamais prêt !
- Mozilla a abandonné ces numéros compliqués et publie simplement Firefox 19, 20, 21, … toutes les 6 semaines
  - Spécifique à l'évolution rapide du web ?

5 Comment faire des releases ?

Comment faire une release ?

- Elle doit être **facile** à installer (et à utiliser)
  - Pas besoin de dépendances super récentes
    - autotools, doxygen, ...
  - L'utilisateur fait configure && make et ça marche
- Il existe plein d'outils
  - make dist et distcheck avec automake
  - CMake/CPack
- Noter que certains systèmes de build ne sont pas auto-suffisants (CMake, Waf, …)
  - L'utilisateur devra les installer pour compiler

Que mettre dans les releases ?

- Inclure les dépendances ou pas ?
  - En général, non
  - Si la dépendance est rare et difficile à compiler ? Bof
  - Si la dépendance a une ABI instable ? Peut être pratique
- Attention aux licences
  - Indiquer dans votre fichier COPYING la licence des dépendances que vous avez inclues
  - Demander de l'aide au SED et/ou au SREV
- Ne pas réinventer la roue sous prétexte d'éviter d'ajouter une dépendance !

Quoi activer dans les releases ?

- **TOUT !**
  - Les utilisateurs utilisent quasiment toujours la configuration par défaut
    - Le reste sera très rarement testé
- Ne pas mettre plein d'options de configuration
  - Si c'est trop compliqué, les utilisateurs ne sauront pas configurer
    - Linux a 10 000 options, peu de gens les utilisent, et les distributions les activent toutes
  - Les utilisateurs diront que votre logiciel ne marche pas bien
    - Uniquement parce que la config par défaut n'est pas optimale

Quoi activer dans les releases ?

- Et mon code « expérimental » ?
  - L'activer par défaut
    - Et faire des release candidates pour le tester
  - Si ça casse tout, c'est trop expérimental, ça ne devrait pas être dans la branche SVN principale
    - Ca veut dire quoi « expérimental » ?
- Ne pas distribuer du vieux code non maintenu
  - Si jamais quelqu'un le teste, vous allez être embêté pour lui expliquer que ça ne sera jamais corrigé...

A quoi faire attention **AVANT** une release ?

- Ne pas casser l'ABI
  - Sauf si **vraiment** nécessaire
    - Et donc ne pas le faire 15 fois
    - Grouper tous les changements dans une grosse release 2.0 ?
- Et si un bug fix impose de casser l'ABI ?
  - Ca ne devrait pas arriver...
  - Faire une release majeure ?
    - La synchroniser avec les autres gros changements en cours
- Bien réfléchir à son API/ABI à l'avance
  - Bien choisir les noms de fonctions, les uniformiser, ...

A quoi faire attention **AVANT** une release ?

- Vérifier que ça marche !
  - Sur des vraies machines, pas juste sur votre portable...
- Normalement c'est le moment où vous vous dites qu'il faudrait vraiment finir par aller voir
  - http://pipol.inria.fr
  - http://hydra.bordeaux.inria.fr
  - http://ci.inria.fr
- Cf séminaires SED passés (et probables futurs)

A quoi faire attention **APRES** une release ?

- Bien documenter les releases dans le VCS
  - Tagger les releases
    - Ca permet de bissecter facilement
- Changer le numéro de version du VCS juste après la release
  - Permet de distinguer la release 1.5 officielle et le SVN trunk qui pourrait être buggué 2h plus tard
    - Utile pour comprendre les bug reports

6 Comment publier mes releases ?

Comment publier une release ?

- Tarball ?
  - Tous les OS n'aiment pas
    - .zip ou .exe pour Windows
  - Certains Unix préfèrent .tar.gz à .tar.bz2
- Indiquer clairement la date de sortie
  - « Il date de quand foo-1.0 déjà ? »
  - « Est-ce que mon SVN est plus vieux que ta version 1.3 ? »
- Indiquer les MD5/SHA1 et signer les fichiers
- Remplir les champs quand on publie sur gforge
  - Utiliser une page séparée pour MD5/SHA1/signature ?

Bien documenter mes releases

- Ne pas confondre ChangeLog et NEWS
  - ChangeLog surtout utile aux développeurs (git log)
  - NEWS est utile aux utilisateurs
    - NEWS clair et court listant les gros changements et les bug fixes
    - Remercier les contributeurs (rapports, patchs, tests, …)
      - Rappel : Il faut choyer les utilisateurs qui font des bug reports
      - Et ça permet de montrer que le logiciel est vivant !
    - Intégrer les NEWS des branches précédentes dans la dernière
- Afficher la version VCS correspondant aux releases ?
- Mettre le NEWS bien en évidence partout
  - Tarball, mails d'annonce, web, …
  - On doit pouvoir trouver les changements des dernières releases sans être abonné à la liste -announce@

Comment annoncer une release ?

- Pas juste ajouter un tarball sur gforge
- Mettre une news en haut du site web
  - Lien vers mail d'annonce et/ou le NEWS
    - Et vers la page de download
  - Le site web doit montrer que le logiciel est vivant
    - Mettre aussi des news à propos d'autre chose que les releases
      - Publication, démo, ...
- Mail sur les mailing lists
  - Vérifier que les gens ne sont pas juste sur -devel@ si on écrit à -announce@
  - Inclure le NEWS et lien vers la page download

Annoncer une release (encore mieux)

- News sur votre site web, celui de l'équipe, …
  - Voire « Scotch breaks the 32bits barrier » sur inria.fr
  - Il faut que ça google bien
- Élargir la diffusion pour les versions majeures
  - Diffuser aux projets connexes, …
  - Tweeter, blogger, …
    - Voire teaser les versions futures
- Installer sur les clusters du coin et faire annoncer par les admins
  - Vos collègues/voisins sont des bons testeurs
    - C'est moins grave si ce sont eux qui trouvent des bugs...

7

Après propos

Je perds combien de temps à faire tout ça ?

- Passer un peu de temps à documenter le processus de release
- Intégration continue pour faciliter les tests
  - Et la construction des releases
- Automatiser pour simplifier la diffusion
  - Cf le séminaire sur XSLT

Je gagne quoi à faire tout ça ?

- Mon logiciel vivant avec plein de releases a-t-il des utilisateurs ?
- Stats de téléchargement
  - Celles de gforge sont assez fausses
  - Utiliser un service dédié ? (G....e Analytics, Piwik, ...)
- L'activité sur les mailing listes est aussi un bon indicateur
  - Même si ce sont des bug reports
    - Un utilisateur ne rapporte pas un bug s'il n'est pas motivé pour utiliser votre logiciel
- Et les citations
  - Notamment les positives

Au boulot !

(et merci de votre attention !)

Brice.Goglin@inria.fr

EPI Runtime - BSO