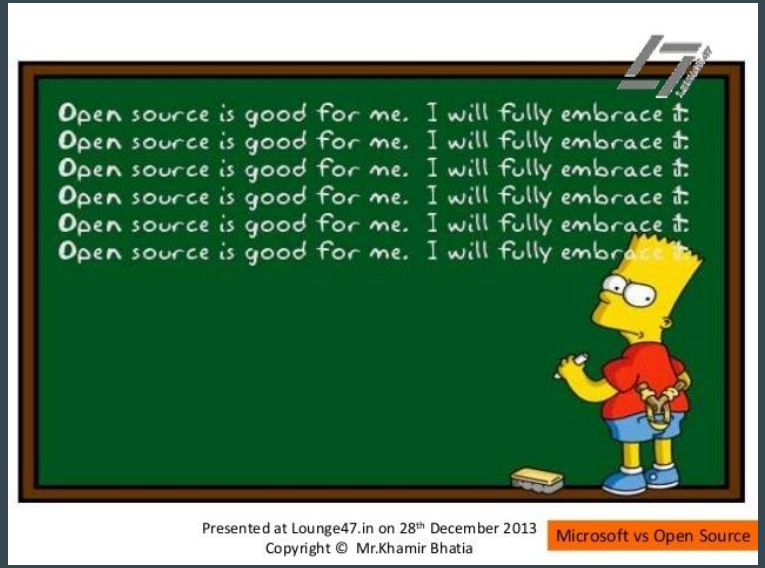
L'Open Source au secours du développeur (et de l'architecte) ?



Un peu de contexte ....

- Développeur un jour ... développeur toujours
  - Débuts à l'âge de 9 ans
  - Passion avant un métier

- De développeur à architecte
  - 13 ans de développement
  - 7 ans d'architecture / développement

- D'intellij à powerpoint

Pourquoi se lancer dans l'Open Source ?

- Continuer à développer
- Continuer à progresser
- Partage et communauté
- Participer à quelque chose d'utile
- Différent d'un side project
- Reverser un peu de ce que l'on a pris au monde Open Source

- ~~Gloire et beauté~~
- ~~Célébrité et renommée~~
- ~~Argent~~

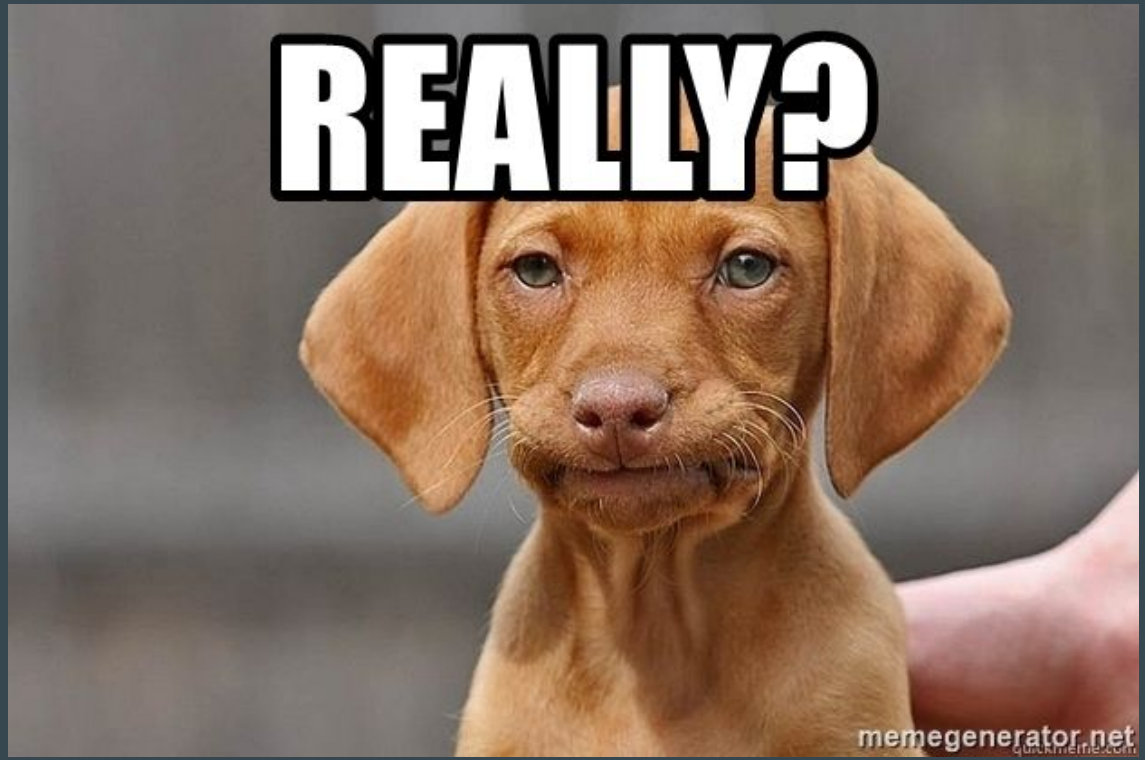
Par où commencer ?

- Se documenter
  - Internet ....
    https://daily.dev/blog/how-to-contribute-to-open-source-projects-as-a-beginner
  - Conférence donnée par Helen Scott (Jet Brains)

- Utiliser des “indexeurs” de projets ouverts à participation
  - https://up-for-grabs.net
  - https://www.codetriage.com
  - https://www.firsttimersonly.com/

Le choix du premier projet

Un projet connu et utilisé régulièrement



Un projet connu et presque utilisé régulièrement

- Syndrome de l'imposteur ou marche trop haute ?
  - SpringBoot
  - OpenJDK
  - Jenkins

- JenkinsPipelineUnit ... Trop professionnel
  - Groovy
  - Candidat idéal pour une démarche Open Source "professionnelle"

💡 Ne pas viser tout de suite la lune, accepter de commencer modestement 💡

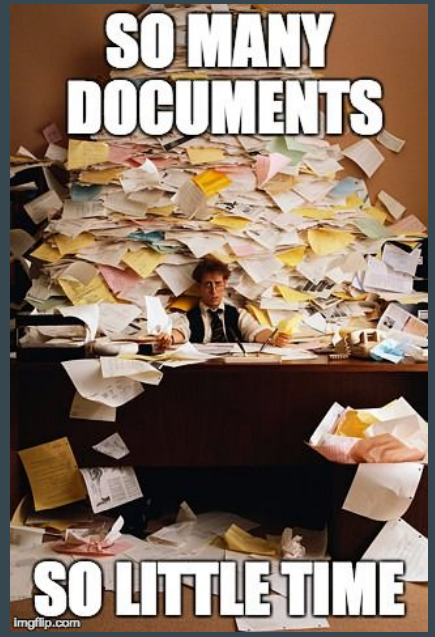
Un projet connu fonctionnellement et utilisé régulièrement

- Jenkins reste un bon choix
- Java 8 (pas encore 11)
- Fonctionnement connu : utilisation quotidienne
- Plugins plus “abordables” : moins complexe que le core

💡A défaut de participer à un projet utilisé dans ses développements, participer à un projet qui est une application utilisée tous les jours peut aider à “rentrer” dans le projet 💡

Comprendre le projet

- Les documentations projet sont inégales
  - Peu de documentation de conception / d'architecture
- Privilégier les projets dont les issues sont sur GitHub
  - Avis personnel (préférence par rapport à JIRA)



💡 Regarder les tests unitaires (en règle générale les projets Open Source ont pas mal de tests) aide à comprendre comment le projet doit être utilisé 💡

La première contribution

- Lire les différents éléments inhérents à la contribution (CONTRIBUTING.MD)
  - règles de développement (checkstyle, nom des branches, ...)
  - code de conduite
- Ansicolor-plugin
- “Il n’y a pas de petites contributions”
  - Correction d’un lien du README vers le CONTRIBUTING

💡 Lire les documentation pour la contribution et l’organisation de la communauté facilite l’intégration et la prise en compte des futures PR 💡

🎉

- 🔓 Verrou débloqué
- 🚀 Envie d’aller plus loin !

Le début de l’aventure

La recherche du Graal

- Géré dans GitHub
- Pas trop gros mais actif
- Communauté active
- Peu de code (facilité de prise en main)
- Un peu de chance ... (GSoC)

➡️ conventional-commits-plugin

💡Essayer de trouver un projet d'une taille pas trop grosse facilite sa prise en main 💡

💡Une communauté réduite est simple à contacter et identifier (qui fait quoi, notamment les maintainers)💡

La première issue / Pull Request (PR)

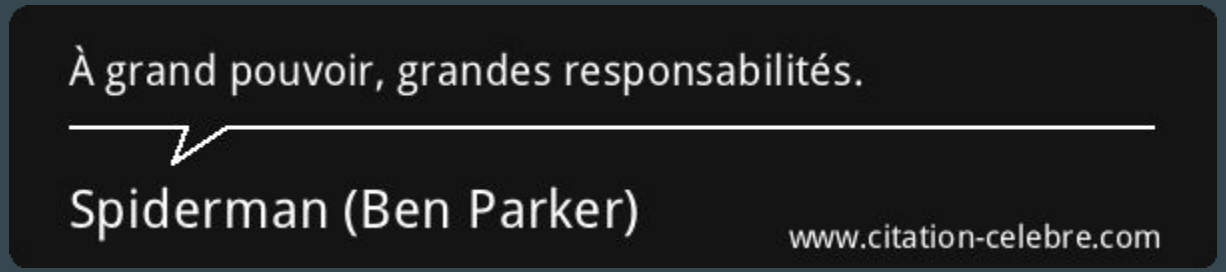
- Fork du projet (et non clone) : penser à mettre à jour l'upstream pour faciliter les merges
- PR au plus tôt (dès la modification de la documentation par exemple)
- Coder pour le plaisir
  - retrouver la raison première d'être développeur
  - pas de biais d'entreprises (budgets, rétro planning, managers, réunions, ...)
- A ce jour une petite dizaines de PR

💡Il ne faut pas hésiter à créer la PR très tôt en indiquant clairement ce que l'on veut faire, cela évite de coder pour rien mais surtout permet de montrer à l'ensemble de la communauté ce que l'on compte faire et avoir leurs retours très tôt.💡

Rythme de travail

- Ne pas reproduire le stress du monde professionnel
- Prendre le temps pour développer feature après feature
- Une feature : 1 à 2 semaines
- Rythme de review : 3 à 5 jours

➡️ Un projet Open Source est géré par des anonymes qui font cela sur leur temps personnel comme vous et moi !



💡 Avant de se lancer dans l'aventure de participer à un projet Open Source il faut s'assurer que l'on a le temps pour cela une fois déduites toutes les activités habituelles (travail, sommeil, famille, ...). 💡

💡 S'engager sur des implémentations de features et ne pas le faire par manque de temps pénalise le projet mais aussi les personnes qui auraient pu travailler sur ces features. 💡

Organisation

- Principalement le WE et pause déjeuner
- Accepter le rythme de travail des autres lors de reviews de PR
  - en profiter pour faire une pause / review des autres PR

➡️ 3 à 4 heures par semaine

💡Ne pas cloner son projet sur son ordinateur pro pour ne pas être “tenté” 💡

💡Un seul projet pour débuter dans le l’Open Source et pour ne pas avoir à switcher et avoir des moments de pause 💡

Bonus apporté par l'Open Source

- Améliorer mon niveau d'anglais
- Apprendre de nouvelles façons de développer / de s'organiser
- Aider d'autres développeurs
- Petite fierté personnelle : au bout de quelques semaines je suis devenu maintainer

Pari gagné ?

✅ Reprendre un rythme de développement en Java

✅ Continuer à apprendre ... même en Java 8

✅ Vaincre (un peu) le syndrôme de l'imposteur

✅ Sentiment d'être utile

😎

Et la suite ?

- Projet en légère perte de vitesse
  - étudiant cherche du travail
  - réalité pour un projet Open Source, il peut être abandonné
  - ~~essayer d'être un~~ maintainer principal ✅
- Se lancer dans d'autres projets
  - Plugin maven pour Jenkins
  - Projet Java 11 / 17
- Hacktoberfest 😉

Dans tous les cas continuer cette aventure !

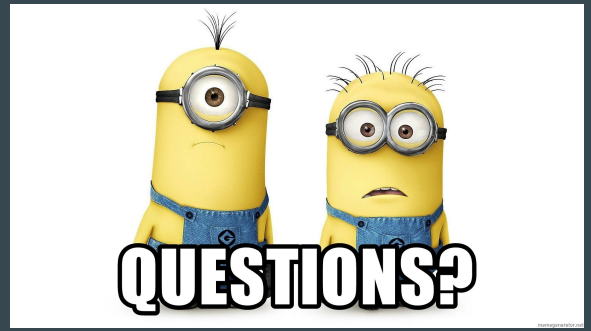
Stéphane Philippart

"Il n'y a pas de petites contributions"

🏷️ Architecte logiciel à Harmonie Mutuelle

🏷️ Co-créateur de TADx (meetups Agile, Dev, DevOps)

🏷️ Commiteur Open Source



@wildagsx

🌐 https://philippart-s.github.io/blog

https://github.com/philippart-s/

https://www.linkedin.com/in/philippartstephane/

➡️ https://github.com/jenkinsci/conventional-commits-plugin/issues/116