Search Odyssey y Branching

El pod de Search Odyssey quiere cambiar la forma de trabajar, aprovechando el cambio del sistema de CVS desde Mercurial a Git, y empezar a trabajar por `branches` y con [GitFlow] como workflow.

Conceptos

Diferencia conceptual entre `forking` y `branching` viene dada por desarrollo divergente vs convergente:

Concepto de `Forking`

Se refiere al proceso de generar una copia exacta del repositorio origen a uno nuevo en ese instante temporal. Es una copia física real y diferente, la operativa surge para realizar separaciones reales y crear nuevas lógicas bajo una base común, se asume que es poco probable que vuelvan a reunirse con el parent.

Concepto de `Branching`

Se refiere a generar un copia del repositorio dentro del mismo repositorio origen, un *pointer*. Las ramas son espacios temporales sobre los que cuales realizar desarrollos nuevos o cambios. Su objetivo es volver a converger con el repositorio siempre.

La diferencia conceptual es el scope de la copia (copia separada del parent o dentro de éste) y vida (vida independiente contra tiempo de vida efímero).

La razón de la diferencia parece ser la necesidad de controlar quién puede o no realizar *push* de código a la rama principal, la práctica del *forkeo* es más común en el open source cuando los posibles colaboradores no tienen permisos sobre el repositorio original, de ahí la copia que genera otro repositorio de facto y luego la posibilidad de converger con el parent o no. En eDreams esto no aplica yo puedo desde una branch en un fork puedo lanzar una *PR* y automergeármela en *upstream*.

Diferencias

- Forking es más costoso al tener que comparar dos codebases una contra la otra, ya que el fork representa una copia literal del repositorio original (doble de espacio de almacenamiento).
- Branching sólo añade una rama sobre el árbol actual, el tamaño de la rama viene ligada literalmente a los cambios de ésta.
- Forking ofusca más ver en que anda trabajabando el equipo, al tener que moverse entre repositorios distintos en vez de sobre ramas sobre uno solo repositorio.
- En forking, al no ser un workflow colaborativo, los cambios residen en la copia de cada uno y puede llevar a mayores problemas a la hora de mergear, perecer (políticas internas de la casa para autoborrado de forks por falta de uso para liberar espacio…) o pérdida de conocimiento.
- Branching al centralizar el workflow sobre un sólo repositorio permite al actualizar sus copias

remote recibir el estado de todos los remotos de las features de sus compañeros.

Why?

> The Bitbucket team recommends branching for development teams on Bitbucket.
>
> — Bitbucket

> [...]People refer to Git's branching model as its "killer feature," and it certainly sets Git apart in the VCS community. Why is it so special? The way Git branches is incredibly lightweight, making branching operations nearly instantaneous, and switching back and forth between branches generally just as fast. Unlike many other VCSs, Git encourages workflows that branch and merge often, even multiple times in a day.
>
> — Pro Git Book by Scott Chachon and Ben Straub

Aparte de una diferencia de estilo de trabajo, de que conceptualmente los forks son para otra cosa, y el coste real es espacio en disco y tiempo de copia... ambas operativas son similares e incluso complementarias, no excluyentes.

La razón por la cual Search Odyssey encuentra interesante utilizar de forma única branches es:

- Una forma más rápida y cómoda de recorrer el código (1 repositorio, 7 ramas de feature; en vez de bajarse 7 forks y sabe Dios si habrá ramas dentro)
- Implementar GitFlow o una aproximación a éste en nuestra forma de trabajar.

Requerimientos

Para poder empezar a trabajar con `branches` sin repercutir en la productividad del equipo:

- Plan fijado de sobre como implementar la CI en las `branches`
  - Protección de ramas a `commit` de developers
    - `hotfix` branches, quién, cómo se crean y cierran
    - `release` branches
  - `master` es productivo
    - `canary` atada a `release` branches o añadimos una específica para `canary` sobre `release`
- Pipelines de la CI/CD
  - Activación automática de la CI en branches
    - En vez de activación por `commit` en `master` de los forks, que se activen también por `commit` en `branches`

    - Para reducir ejecuciones podría restringirse a ejecución manual la CI en las ramas `feature`

Refs

- [GitFlow] A Successful git branching model
- Bitbucket
- Pluralsight
- toolsqa
- /r/devops - branching vs forking
- Pro Git Book by Scott Chachon and Ben Straub