Les infos sur l'utilisation de GitLab pour le projet CTA

- comment est gérée la gestion de version (git, SVN, mercurial)
  Uniquement Git via https://gitlab.in2p3.fr/ 3 repos (2 projets + 1 fork)
  - CTA-APC: cta_proposal_handling (https://gitlab.in2p3.fr/cta-apc/cta_proposal_handling)
  - CTA-APC: CTA-Simulations (https://gitlab.in2p3.fr/cta-apc/cta_simulations)
  - Fork zakharov: cta_proposal_handling (https://gitlab.in2p3.fr/zakharov/cta_proposal_handling)
- taille du dépôt (# de lignes de code)

~ 12800 lignes de Python
~ 9000 lignes html (templates et docs)
~ 12 branches (la plupart vieux et non-utilisé)

- nombre typique de contributeurs au projet

~ 4 personnes

- taille des commits (petits/gros, réguliers/irréguliers)

petits ou moyens, plutôt irréguliers

- travail sur branche ou directement sur la branche par défaut (master/main)

les deux + fork

- suivez-vous des bonnes pratiques ? (convention de nommage des branches, messages de commit standardisés, revue de code)

Pas vraiment (il y a une certaine volonté, mais il n'y a pas d'habitude)

- Projet CTA-Simulations est un nouveau petit projet créé recemment ~ 1900 lignes du code (python) ~ des fichiers binaires fits necessaires pour les simulation ~ 55 fichiers, ~18Mo (comment gerer?)

Utilisation de GitLab dans Euclid

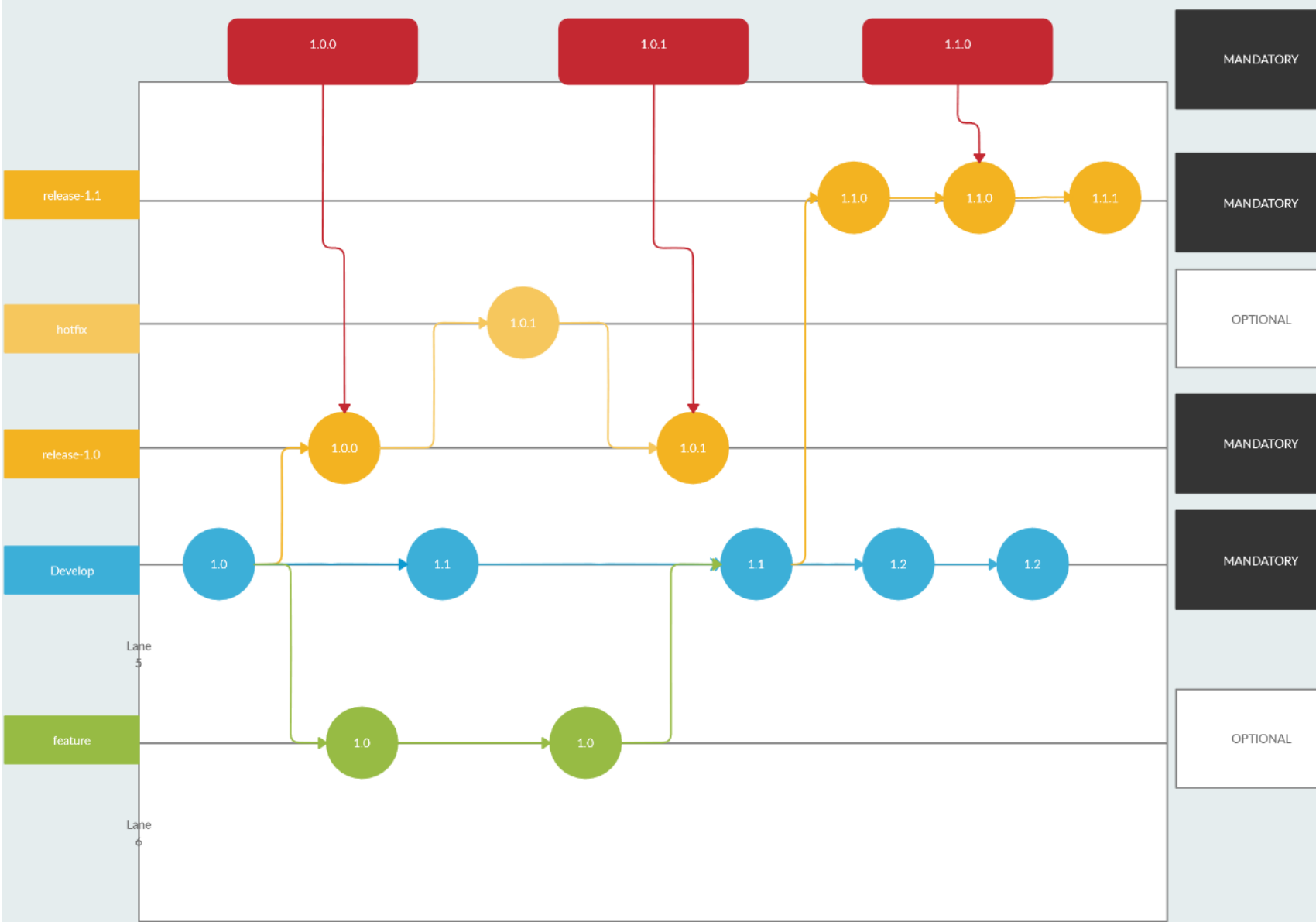
- Nombre total de projets sur gitLab: 1160.
- Projets en production : 250.
- Taille typique en prod : 1000 < N < 10000 lines (les 2/3, moins pour le reste)
- Nombre typique de contributeurs/Taille des commits : inconnu. (2 à 5 par expérience, taille variable de commit selon dev.)
- Travail avec un workflow (voir schéma)
- Bonnes pratiques vérifiées lors de “maturity assessment”, via sonar essentiellement.

Workflow basé sur le nombre de chiffre par version :

- develop/feature : 2.

- release/tag : 3.

Le CD/CI dépend de la branche.



IGOSat – Utilisation de git

Gestion de version : Git | Gitlab IN2P3, groupe IGOSat : https://gitlab.in2p3.fr/igosat

Organisation du groupe :
- Arborescence organisée selon les sous systèmes d'IGOSat (ADCS,SCI,ODB,…)
- Dans chaque sous-systèmes :
  - Logiciel de vol
  - Logiciel sol
  - CAO électronique

Taille dépôt :
GPS : ~ 4 200 LOC
OBC : ~ 10 000 LOC
GND : ~ 3 000 LOC

Nombre de personnes actives sur le groupe : 6-8 (stagiaires + chef de projet + ingé logiciel)
Nombre de personnes actives sur chaque projet : 2/3

Taille des commits : objectif d'avoir des commits les plus petits possible et le plus souvent

Gestion des branches :
- Master : garde la dernière version 'marquante', on limite les développements en direct sur cette branche
- Internship : chaque stage a sa propre branche de départ. A la fin du stage, merge sur la master
- Dev : branche pour l'ingé logiciel qui est souvent mergée sur la master
- <branch-name> : ponctuellement, on crée des branches pour résoudre des bugs particuliers / tester des nouvelles implémentations, ...

Bonnes pratiques :
Pas forcément, on essaie d'être cohérent entre le nom de la branche et son intérêt, on tag de temps en temps quand on arrive à une version remarquable ou alors pour archiver les branches (tag des branches permet de les delete mais de toujours pouvoir les recréer plus tard)

Gitlab in SVOM group

- Single repository for ECLAIRs GP pipeline / Number of maintainers: 5
- Overall code: 35.7k lines / Production code: 16.6k lines
- Language: Python (>99%), rest is Dockerfiles, .json, README.md
- Commit size / frequency: large / irregular
- One branch = one task. Then code review and (hopefully) merge on dev.
  Merge on master = one release.
- No particular formats followed for commit messages. For branch naming: e.g, “dev_number_keywords”.
- CI (pytest, pylint, SQ): in place (CEA runners & docker images) / CD: to be done

Gitlab in LISA group

- Multiple repositories but focus on LISANode pipeline / Number of maintainers: 3 (83 users)
- Overall code: 111.8k lines / Production code: 76.2k lines
- Language: C++ (83.5%), Python (16.5%), Dockerfiles
- Commit size / frequency: large / irregular
- One branch = one opened issue on Gitlab. Then merge request by CP and merge on master branch. Merge on master = one release.
- Commit messages format: e.g. “Add SumBlocks to naming.py” / Branch names format : e.g. “250-create-dynamics-toolbox-in-lisanode ”
- CI (unit tests, pylint) / CD (docker): in place

Git workflow - changeur de filtre Rubin/LSST

- dépôt de code sur Github privé
- codebase : ~50K lignes Java + Groovy + 2K Python
- contributeurs : 1-5 – 2 principaux
- commits : taille en diminution ;-) – code en refactoring actif
- branches : noms associés à des Jiras (Atlassian, equiv. Redmine)
- bonnes pratiques : code formatting + linting, pair coding, code review