

Techniques et Outils d'Ingénierie Logicielle

...

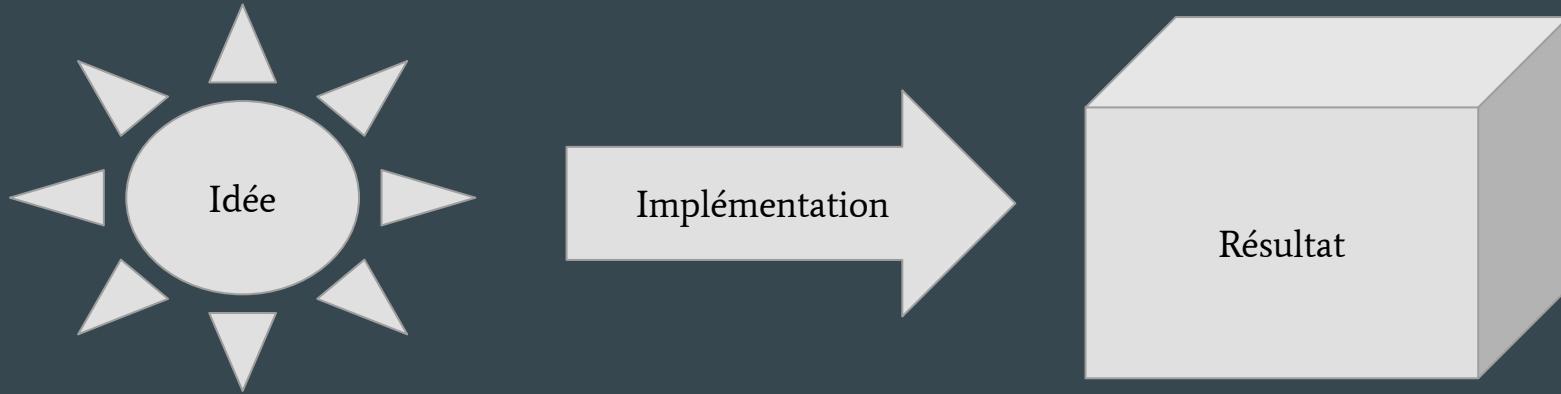
John Gliksberg

Pour récupérer les slides et exemples

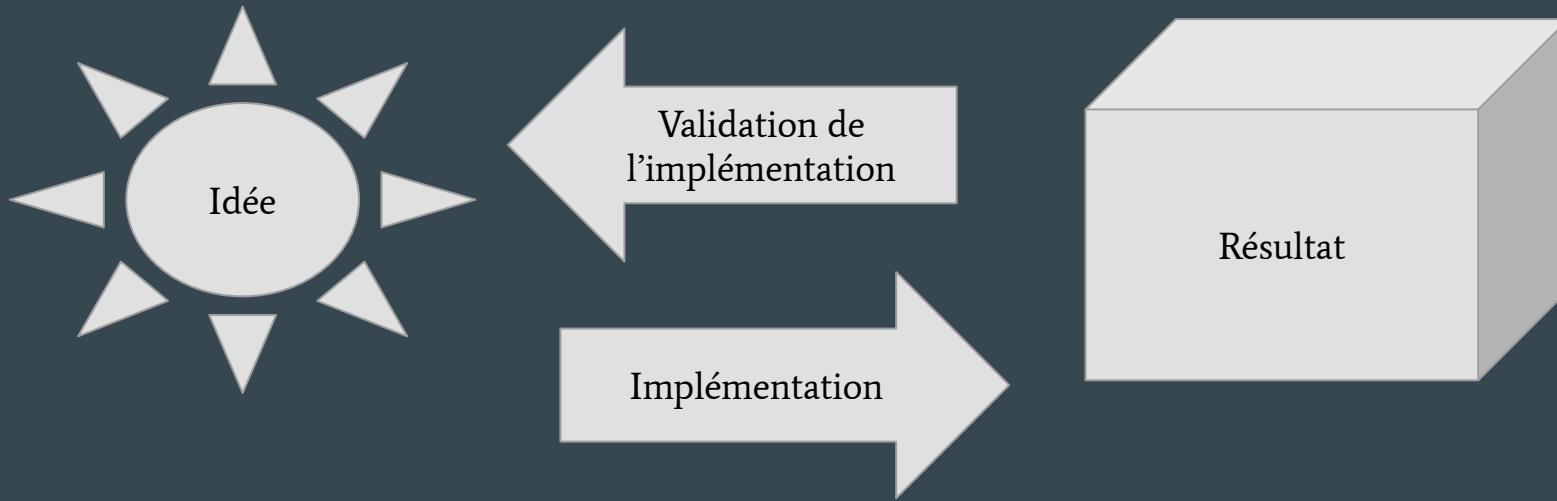
git clone <https://github.com/trosh/toi2025>

Cycle de vie du code

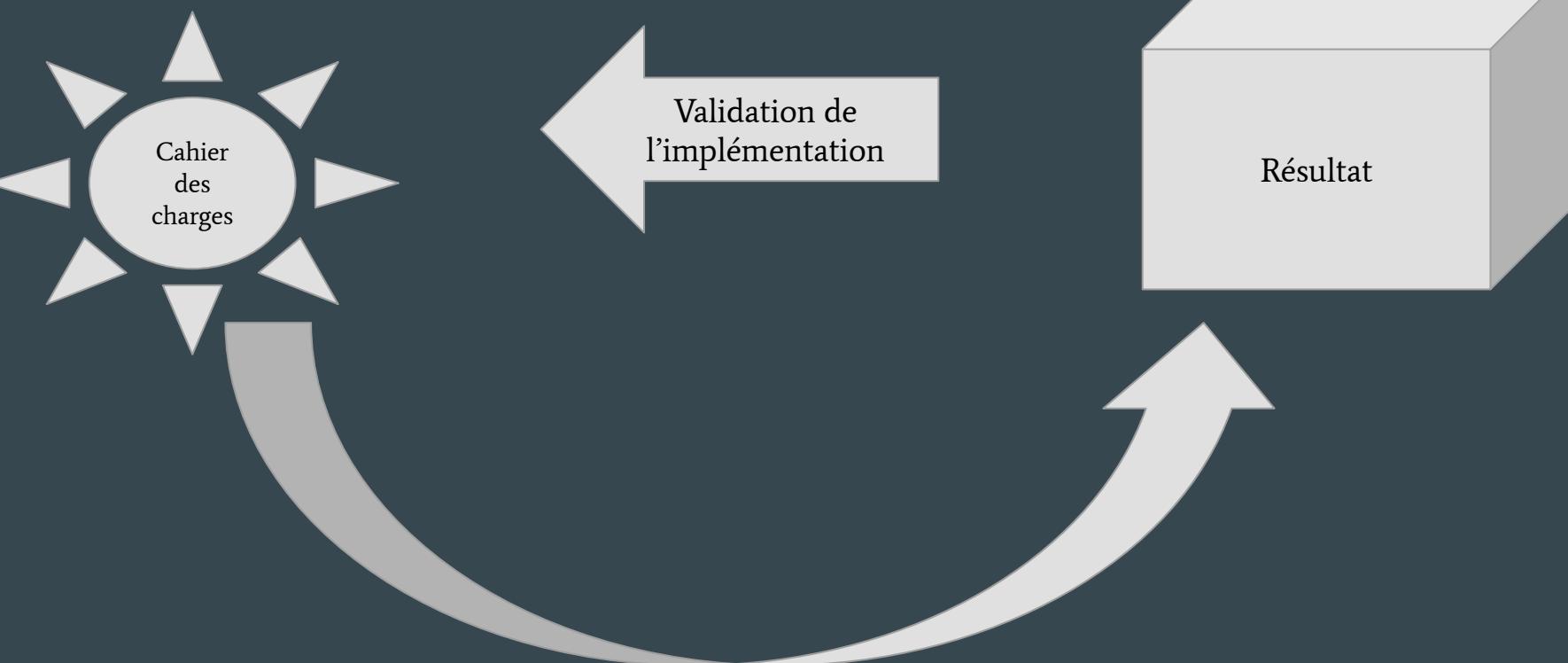
Le code répond à un besoin



On peut comparer le résultat au besoin

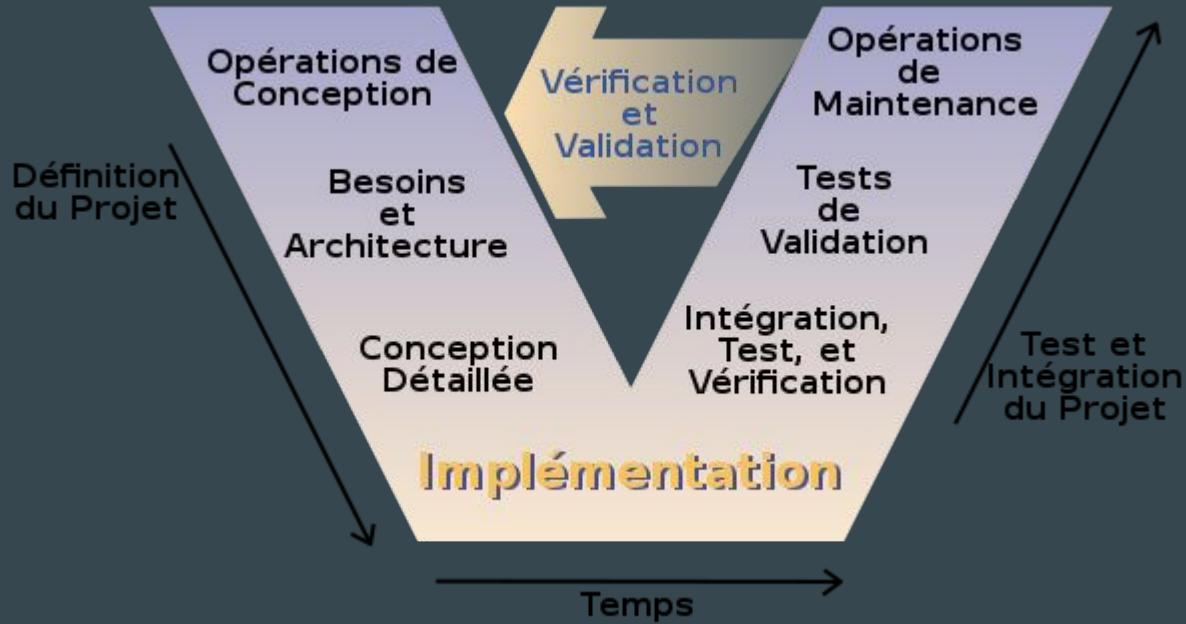


Cela crée alors un cycle



Cela crée alors un cycle

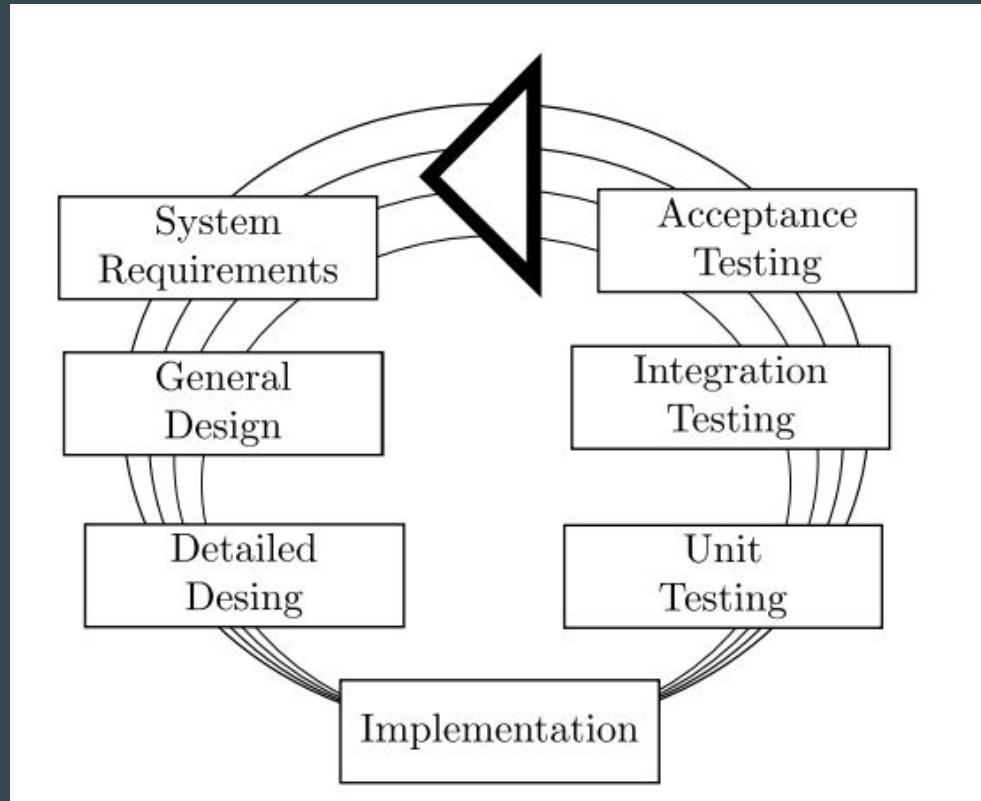
De la spécification à la validation. Le projet et avant tout affaire de communication.



Variation en mode “agile”

Répéter le cycle de manière itérative permet de:

- Raffiner le cahier des charges
- Adapter les tests
- Avancer progressivement
- **Communiquer**



La Forge



Introduction à la forge de codage

La forge de codage est une plateforme en ligne qui offre un ensemble d'outils et de fonctionnalités destinés à faciliter le processus de développement logiciel.

Elle agit comme un environnement collaboratif où les développeurs peuvent travailler ensemble sur des projets, partager des codes sources, des bibliothèques, des modules, et collaborer à la résolution des problèmes.

La forge de codage fournit également des outils de gestion de projet, de suivi des problèmes (bug tracking), de gestion de versions (version control), et de documentation, ce qui permet une gestion efficace et transparente des projets de développement.

Grâce à la forge de codage, les développeurs peuvent bénéficier d'une infrastructure solide pour organiser, suivre et gérer l'évolution de leurs projets logiciels, tout en favorisant la collaboration et la contribution de la communauté.

la

Ce que permet une forge

The screenshot shows a GitHub repository page for 'cython / cython'. The main heading is 'Commits'. It displays a list of commits from July 26, 2021, to July 23, 2021. Each commit card includes a summary, author, date, and a copy/paste icon.

- Fix compilation of "ParseTreeTransforms" in Py2. It previously tried to compare the 'directive type' to 'str', which misses Py2 str when compiling with language level 3. scoder committed on Jul 26, 2021
- Fix some imports. scoder committed on Jul 26, 2021
- Add missing attribute declaration for FunctionState class. scoder committed on Jul 26, 2021
- Fix type declaration in parser. scoder committed on Jul 26, 2021
- Add CI jobs that compile more Cython modules with "--cython-compile-all", to make sure that this keeps working and to allow comparing the performance difference. scoder committed on Jul 26, 2021

Commits on Jul 25, 2021

- Add a note on crypto currency donations (any why we shouldn't take them). scoder committed on Jul 25, 2021

Commits on Jul 23, 2021

The screenshot shows a GitHub pull request comparison between 'base: release' and 'compare: ipython711'. The status is 'Able to merge'. The comparison summary indicates 1 commit, 2 files changed, and 1 contributor. The commit details show a change in Cython/Build/IpythonMagic.py where it imports distutils.log and textwrap, and adds IO_ENCODING = sys.getfilesystemencoding() and IS_PY2 = sys.version_info[0] < 3. The diff view shows the changes with line numbers and context.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

base: release compare: ipython711 Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#) Create pull request

1 commit 2 files changed 1 contributor

Commits on Jan 1, 2020

Remove legacy imports from IPython integration since the "py3compat" ... scoder committed on Jan 1, 2020 efa8d69

Showing 2 changed files with 22 additions and 20 deletions. Split Unified

```
diff --git a/Cython/Build/IpythonMagic.py b/Cython/Build/IpythonMagic.py
--- a/Cython/Build/IpythonMagic.py
+++ b/Cython/Build/IpythonMagic.py
@@ -56,6 +56,8 @@ import distutils.log
@@ -59,6 +73,7 @@ try:
@@ -73,6 +75,6 @@ from IPython.core import display
@@ -75,6 +75,6 @@ from IPython.core import display
```

contrôle de version

Ce que permet une forge

The screenshot shows a GitHub repository interface. On the left, there's a sidebar with navigation links: Code, Issues (46), Pull requests (12), Actions, Security, and Insights. Below this is a 'Dependency Dashboard' section with a note about an open issue (#3206) and a 'Filters' dropdown. The main area displays a list of issues:

- 46 Open, 1,630 Closed
- Error while generating DuckDuckGo Forward Email Username (bug) #4226 opened 7 hours ago by zunelol 1 task done
- 2024.11.0 Crash (bug) #4223 opened 2 days ago by pcartwright81 1 task
- Not been able to create passkey on microsoft account (bug) #4222 opened 2 days ago by singhh9596 1 task
- Can't scroll back when entering email/password values too long (bug) #4209 opened 4 days ago by sagehane 1 task done
- Mobile app unable to scan some QR codes (bug) #4184 opened last week by munrobasher 1 of 2 tasks
- If the url contains port information when selecting host matching, the mobile client cannot pop up the correct auto fill option (bug) #4184 opened last week by munrobasher 1 of 2 tasks

The screenshot shows a GitHub pull request details page. At the top, it says 'aryn-ydy authored on May 28 · 7 / 7 · (Verified)'. The pull request has 1 parent commit (3e3d1fd). The code changes section shows a diff between two files: 'a/org/schabi/newpipe/fragments/list/playlist/PlaylistFragment.java' and 'a/org/schabi/newpipe/fragments/list/playlist/PlaylistDescriptionView.java'. The diff highlights several changes, including the addition of an 'ellipsizer.toggle()' call in the first file and its removal in the second. The 'Comments' section shows a single comment from 'Comment' with a note: 'You're not receiving notifications from this thread.'

suivi de bugs/tickets

Ce que permet une forge

pull/merge requests

A screenshot of a GitHub pull request interface. At the top, it shows a repository named "zzz-broadcom-pocs" with a "develop" branch and an "OPEN" status. Below this, a message states: "There is 1 issue preventing you from merging this pull request." A yellow warning icon with the number "1" indicates the presence of an issue. To the right, there are buttons for "Merge" and "Start review". The main body of the interface contains a list of users required for review: BIELLE, BENJAMIN, SRIDI, MARWA, GLIKSBERG, JOHN, Droneaud, and Yann.

A screenshot of a GitHub pull request interface for the "godotengine / godot" repository. The pull request is titled "Improve TextureProgress.set_radial_initial_angle() by removing loops #98816". The status is "Open" and it wants to merge 1 commit into the "godotengine:master" branch from the "arkology/to-infinity-and-beyond" fork. The pull request has 21 conversations, 1 commit, 20 checks, and 2 files changed. The review section shows a comment from "arkology" suggesting to replace two while loops with if and fposmodp, document the "radial_initial_angle" wrapping, and note that infinite numbers will cause an error. It also mentions that the fix for issue #60338 should be applied. The pull request has 20 approvals and 20 comments. On the right side, there are sections for "Reviewers" (KoBeWi, Mickeon, AThousandShips), "Assignees" (No one assigned), "Labels" (bug, cherrypick:4.2, cherrypick:4.3, topic:gui), "Projects" (None yet), and "Milestone" (4.4).

Ce que permet une forge

collaboration/discussions

fix up always-truthy check for 0n and support negative numbers #60324
kirkwaiblinger wants to merge 3 commits into [microsoft:main](#) from [kirkwaiblinger:always-truthy-fixes](#)

RyanCavanaugh reviewed 4 days ago

src/compiler/checker.ts

```
44512 +     return PredicateSemantics.Always;
44513 +     // handle +123, -123, -123n, etc.
44514 +     case SyntaxKind.PrefixUnaryExpression:
44515 +         const prefixUnaryExpression = node as PrefixUnaryExpression;
```

RyanCavanaugh 4 days ago

Seems like this could just be something like

```
if (operand.kind === SyntaxKind.NumericLiteral && (operator === SyntaxKind.MinusToken || operator === SyntaxKind.PlusToken)) {
    return getTruthySemantics(operand);
}
```

? We shouldn't be duplicating all the logic of the numeric case.

kirkwaiblinger 4 days ago • edited

Fair question. I wrote it this way to avoid a few bugs....

```
if (1) {} // this is intentionally exempted from the check, BUT
if (-1) {} // should report. Would NOT with the recursion.
if (+1) {} // IMO should also report, since this is clearly intended as a number, not as a shorthand
// (ditto for the following)
if (0) {} // intentionally exempted from reporting
if (+0) {} // should report IMO. This is clearly intended as a number, and not as a shorthand for '1'
if (-0) {} // see previous.
```

Note that the branches are therefore different:

```
case SyntaxKind.NumericLiteral:
```

73 Restructure routing tables in struct sw_rts, which now includes the BXI version as enum bxi_version
[fabric.h](#) ▾

File 3 of 15 | [Prev](#) | [Up](#) | [Next](#)

[SHOW BLAME](#) | Diff view:

```
0     extern int sw_alt_v3_add(struct sw *sw, struct v3_alt_tuple *v3_alt_tuple);
1     extern void sw_delete_alt_tuple(struct v3_alt_tuple v3_alt_tuple);
291    extern int sw_alt_v3_add(struct sw *sw, struct v3_alt_tuple *tuple);
292    extern void sw_delete_alt_tuple(struct v3_alt_tuple tuple);
293    extern void sw_delete_rts(struct sw_rts *rts);
```

Yann Droneaud Aug 09 ▾

As an API to be provided to external user, I would have expected sw_delete_rts() to release struct sw_rts as well, much like network_delete(), sw_delete(), node_delete() (and renamed sw_rts_delete()).

And keep the current function as sw_release_rts() or sw_clear_rts() for internal purpose.

John Gliksberg Aug 09 ▾

Done

Resolved [REPLY](#) [QUOTE](#) [EDIT](#)

```
294
295    extern struct node *node_create(struct network *network, const char *name,
296        » » » const unsigned char uuid[UUID_NODE_LENGTH],
297        » » » int nid);
298    extern struct node *node_create_ex(struct network *network, const char *name,
299        » » » const unsigned char uuid[UUID_NODE_LENGTH],
300        » » » int nid int group int pool);
```

Ce que permet une forge

bxiafm > alt_rts_nonmin → bxiafm > develop DRAFT No open tasks

Alt rts nonmin (depend de uniconf)

Overview Diff Commits Builds

All changes in this pull request 35 commits

Build	Status	Updated
BDS R&D Interconn. Env - 2nd Generation » bxiafm ... LAT... BIELLE, BENJAMIN ♦ 8f8a661ad2a [config] Add ROUTING.alt_rts...	Passed	3 days ago
BDS R&D Interconn. Env - 2nd Generation » bxiafm » PR... GLIKSBERG, JOHN ♦ c3ed5a9aac4 test-bxiafm-divio: [coverage] T...	Passed	1 week ago

Dashboard > BDS R&D Interconn. Env - 2nd Generation > bxiafm > PR-604 >

Schedule Build bxiafm-cli-1.99.0-20241101100130.el8.src.rpm 1.69 MiB view
Pipeline Syntax bxiafm-divio-1.99.0-20241101095644.el8.src.rpm 2.83 MiB view

Stage View

	Cleanup previously built RPMs and init	Retrieve sources	Launch SonarQube analysis on RHEL8	Build doc	Build SRPMs for RHEL8	Build RPMs for RHEL8/X86_64	Publish RPMs and SRPMs
Average stage times: (Average full run time: ~1h 5min)	11s	59s	38min 32s	2min 16s	16min 31s	4min 58s	4s
#28 Nov 01 10:05 No Changes	12s	38s	38min 13s	2min 16s	16min 23s	5min 4s	2s
#27 Oct 31 10:05 2 commits	10s	39s	38min 9s	2min 10s	16min 56s	5min 4s	3s
#26 Oct 28 18:01 No Changes	11s	27s	37min 57s	2min 14s	16min 17s	4min 45s	3s
#25 Oct 23 10:05 No Changes	11s	27s	37min 57s	2min 22s	16min 48s	5min 3s	9s
#24 Oct 19 10:05 1 commit	9s	32s	37min 26s	2min 17s	16min 14s	4min 53s	3s

2 commits

4226001 [fix] [TAP_simu_DopeyQ] Wrong port rank in so...
97a99% [fix] [TAP_simu_dopeyQ] bxiafm V3 is not supp...

See detail page

SonarQube Quality Gate

brie2g:bxiafm Failed
server-side processing Success

intégration continue

Ce que permet une forge

The image shows two screenshots of a project management system, likely Jira, demonstrating its features for backlog management and task tracking.

Left Screenshot: PR Backlog

This view displays a backlog of issues categorized by status:

- Waiting on reviewers (128):**
 - TypeScript #39699: Fix #4628: add compiler option for non-check static member in inheritance. Status: For Backlog Bug.
 - TypeScript #43542: Add `-metadataDecorator` option. Status: Author: Team | For Uncommitted Bug.
 - TypeScript #46120: feat(3792): Cannot 'export default' abstract, ambient class or interface. Status: For Backlog Bug.
 - TypeScript #48092: Don't erase signature type parameters in signaturesRelatedTo. Status: Author: Team | For Milestone Bug.
 - TypeScript #48172: arguments should not be allowed in class static block. Status: For Backlog Bug.
 - TypeScript #48228:
- Waiting on author (87):**
 - TypeScript #53772: Fix #50305. Status: For Backlog Bug.
 - TypeScript #48123: Use strict variance checks for strict subtype checks unconditionally. Status: For Milestone Bug.
 - TypeScript #49274: Defer indexed accesses on generic potential discriminants. Status: Author: Team | For Backlog Bug.
 - TypeScript #49563: Improve generic keyof relationship checking for generic conditionals with keyless branch types. Status: Author: Team | For Uncommitted Bug.
 - TypeScript #49730: Add declaration transform stage that combines repeated import type nodes into single import declarations. Status: Author: Team | Experiment | For Milestone Bug.
- Needs merge (23):**
 - TypeScript #54884: Add original filename header to more baselines. Status: For Uncommitted Bug.
 - TypeScript #56989: Skip parent error when reporting JSX excess property checks. Status: For Uncommitted Bug.
 - TypeScript #48838: Instantiate conditional types within contextual type instantiation when inference candidates are available. Status: For Backlog Bug.
 - TypeScript #53739: Attempt to reduce types which are about to produce a complexity error. Status: For Backlog Bug.
 - TypeScript #55547: Infer unknown at value positions of objects inferred from keyof T. Status: Breaking Change | For Uncommitted Bug.
- Not started (81):**
 - TypeScript #59166: add missing interfaces for sub-classes of PerformanceEntry. Status: For Backlog Bug.
 - TypeScript #59274: Avoid circularity when context of static property is its class. Status: For Uncommitted Bug.
 - TypeScript #59290: Not filtering return type using generic async function. Status: For Uncommitted Bug.
 - TypeScript #59297: Keep type aliases on error type when some type arguments are missing. Status: For Backlog Bug.
 - TypeScript #59302: Allow protected property to be part of union of types with the correct access level. Status: For Backlog Bug.
 - TypeScript #59376:
- Done (416):**
 - TypeScript #50759: Strip literal freshness when contextually typed by instantiable type that has return mapper constraint inferences. Status: Author: Team | For Milestone Bug.

Right Screenshot: BREO Sprint 12

This view shows a kanban board for the BREO Sprint 12:

- TO DO:** BREO-140 [INBOX] 3 sub-tasks As an Admin, I want to update an installed application.
- IN PROGRESS:** BREO-153 Application instance Upgrade, BREO-152 Application instance reconfiguration.
- DONE:** BREO-151 Use the GitRepo UID as branch name.
- Other Issues:** BREO-134 Check how to translate login page (from keycloak) using local user language, BREO-135 Translate internal Material Components labels.
- Summaries:** BREO-138 Summarizes the work done on IPSOTEK (Use Case 2), BREO-139 Summarizes the work done on DataSentics (Use Case 3).

Suivi des problèmes (bug tracking)

Le suivi des problèmes (bug tracking) est une fonctionnalité essentielle d'une forge de codage.

Permet aux développeurs de signaler les anomalies, les bogues et les problèmes rencontrés dans le code source ou dans le logiciel.

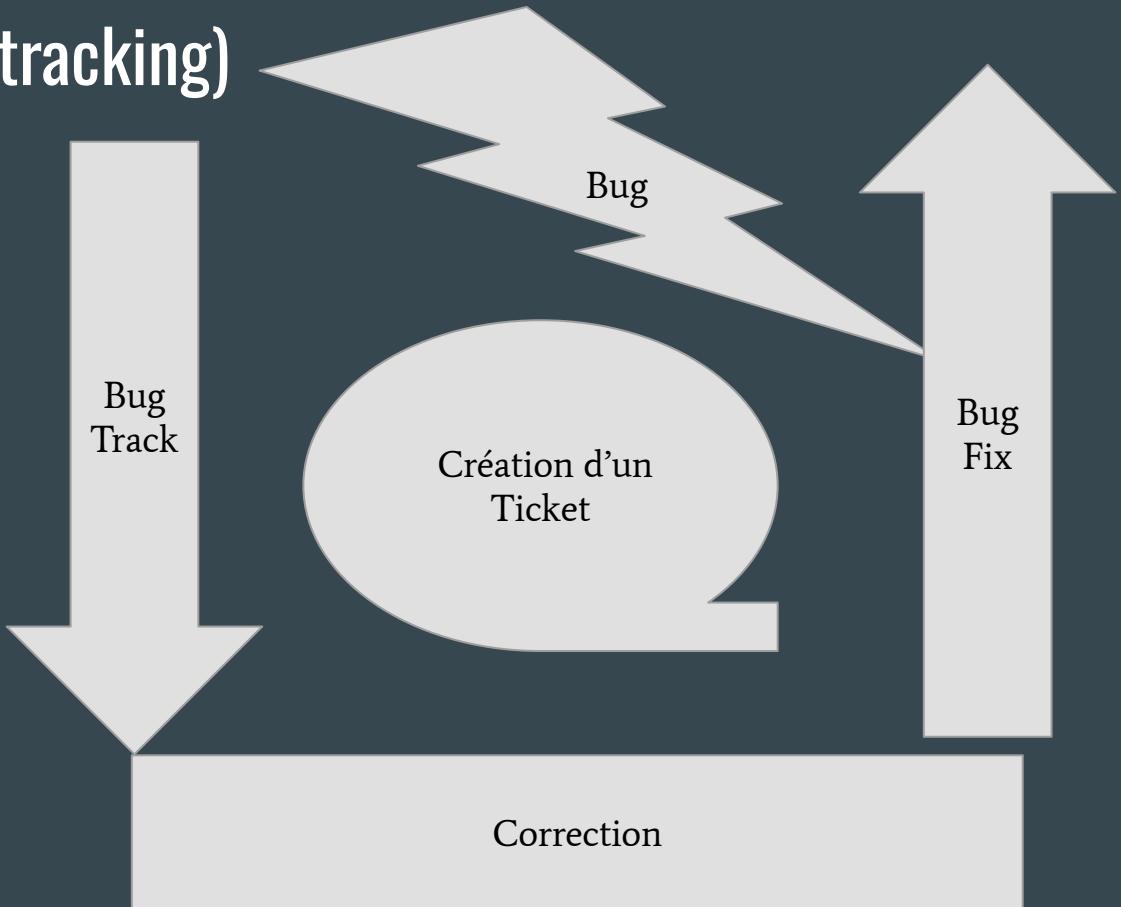
Chaque problème est enregistré dans un système de suivi avec des détails tels que la description du problème, sa gravité, sa priorité et son état.

Les développeurs peuvent attribuer des problèmes à des membres spécifiques de l'équipe pour qu'ils les résolvent.

Permet de suivre l'état de résolution des problèmes, notamment s'ils sont en cours de correction, résolus ou en attente de vérification.

Facilite la communication entre les membres de l'équipe en fournissant un historique des actions prises pour résoudre chaque problème.

Contribue à améliorer la qualité du logiciel en identifiant, documentant et résolvant efficacement les problèmes rencontrés par les utilisateurs.



Pull request (GitHub) ; Merge request (Gitlab)

- Définition : Une pull request (PR) est une demande adressée aux mainteneurs du projet pour qu'ils examinent et intègrent les modifications proposées dans une branche vers une autre (généralement de la branche de fonctionnalité vers la branche principale).
- Fonctionnalités :
 - Permet aux contributeurs de proposer des modifications au code source principal.
 - Facilite la revue de code et la discussion autour des modifications proposées.
 - Offre une plateforme pour commenter, approuver et demander des modifications sur les changements proposés.
- Utilisation :
 - Créer une PR depuis une branche de fonctionnalité vers la branche principale du dépôt.
 - Les autres membres de l'équipe peuvent examiner la PR, ajouter des commentaires, demander des modifications et approuver les changements.
 - Une fois approuvée, la PR peut être fusionnée dans la branche principale, intégrant ainsi les modifications proposées dans le code source principal.

Contenu d'une merge request (MR)

- Titre : Résumé concis des modifications proposées.
- Description : Explication détaillée des changements apportés et des problèmes résolus.
- Fichiers modifiés : Liste des fichiers affectés par la MR.
- Différences : Visualisation des modifications ligne par ligne.
- Discussion : Espace pour les commentaires, questions et discussions autour des modifications proposées.
- Liste des commits : Historique des commits inclus dans la MR.
- État de la MR : Indique si la MR est en attente de révision, approuvée, ou fusionnée.

Merge request (Gitlab)

New merge request

From `master` into `devel` [Change branches](#)

Title (required)

Test: Pull Request

Start the title with `Draft:` to prevent a merge request draft from merging before it's ready.

Description

Choose a template

[Write](#) [Preview](#)



Describe the goal of the changes and what reviewers should be aware of.

Supports [Markdown](#). For quick actions, type `/`.

Contenu d'un ticket

- Titre du bug : Un résumé concis mais descriptif du problème rencontré.
- Description : Une explication détaillée du bug, y compris les étapes pour le reproduire, le comportement attendu et observé, ainsi que toute information pertinente pour comprendre le problème.
- Priorité : Le niveau de priorité attribué au bug, qui peut être défini en fonction de la gravité de l'impact sur l'application ou sur les utilisateurs.
- Gravité : La gravité du bug, indiquant l'importance de sa résolution, souvent déterminée en fonction de son impact sur l'expérience utilisateur ou sur le bon fonctionnement de l'application.
- Environnement : Les détails sur l'environnement dans lequel le bug a été observé, tels que le système d'exploitation, le navigateur, la version du logiciel, etc.
- Références : Toute référence à des documents, des captures d'écran, des journaux ou d'autres ressources qui pourraient aider à comprendre ou à résoudre le bug.
- Affecté à : Le membre de l'équipe chargé de résoudre le bug ou d'enquêter dessus.
- État du ticket : Indique si le bug est en attente de traitement, en cours de résolution, résolu ou en attente de vérification.
- Historique des modifications : Une liste des actions prises sur le ticket, y compris les commentaires des membres de l'équipe, les mises à jour de statut et les changements de priorité ou de gravité.
- Date de création et dernière mise à jour : Les dates auxquelles le ticket a été créé et lorsque des modifications importantes ont été apportées.

Ticket sur Github

The screenshot shows a GitHub repository interface for 'besnardjb / TOI24'. The 'Issues' tab is selected. A search bar at the top right contains the placeholder 'Type to search'. Below the search bar are several icons: a magnifying glass, a plus sign, a circular arrow, a gear, and a mail icon. A user profile picture of a man with glasses is in the top right corner.

The main area is titled 'Add a title' and features a 'Title' input field. Below it is a 'Add a description' section with a 'Write' tab selected, showing a rich text editor toolbar with options like H, B, I, etc. A large text area below the toolbar has the placeholder 'Add your description here...'. At the bottom of this section, there's a note that 'Markdown is supported' and a file upload area with the instruction 'Paste, drop, or click to add files'.

To the right of the main form, there are several configuration sections:

- Assignees:** No one—[assign yourself](#)
- Labels:** None yet
- Projects:** None yet
- Milestone:** No milestone
- Development:** Shows branches and pull requests linked to this issue.
- Helpful resources:** [GitHub Community Guidelines](#)

A green 'Submit new issue' button is located at the bottom right of the main form area. At the very bottom, a note says 'Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)'.

Ticket sur Gitlab

New Issue

Title (required)

Type ?

Issue

Description

Choose a template

Write Preview

Write a description or drag your files here...

Supports [Markdown](#). For [quick actions](#), type `/`.

This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee

Unassigned

[Assign to me](#)

Due date

Select due date

Milestone

Milestone

Labels

Labels

Le contrôle des versions

- La gestion des versions, également connue sous le nom de contrôle de version, est un système qui enregistre les modifications apportées à un fichier ou à un ensemble de fichiers au fil du temps.
- Elle permet de garder une trace de l'historique des modifications, y compris qui a effectué les modifications, quand elles ont été effectuées et quelles étaient les modifications spécifiques.
- La gestion des versions est essentielle pour les projets de développement logiciel car elle facilite la collaboration entre les membres de l'équipe, permet le suivi des modifications et offre une sauvegarde en cas de besoin de revenir à une version antérieure du code.

Intégration continue

- L'intégration continue (CI) est une pratique de développement logiciel qui consiste à fusionner fréquemment les modifications de code dans un référentiel partagé.
- Chaque fusion déclenche une série d'automatisations, telles que la compilation du code, l'exécution des tests automatisés et le déploiement des modifications dans un environnement de test ou de production.
- L'objectif principal de l'intégration continue est de détecter et de corriger rapidement les problèmes d'intégration, tels que les conflits de code ou les régressions, avant qu'ils ne deviennent des problèmes plus graves.

Intégration continue

- Les outils d'intégration continue populaires incluent Jenkins, Travis CI, CircleCI et GitLab CI/CD, qui offrent des fonctionnalités pour automatiser le processus d'intégration et de déploiement.
- L'intégration continue favorise une approche agile du développement en encourageant des cycles de développement courts et des livraisons fréquentes de fonctionnalités stables.
- Elle contribue également à améliorer la qualité du code en permettant une validation rapide des modifications et en fournissant un retour d'information immédiat aux développeurs.

Intégration continue

⌚ 27 jobs for `devel` in 40 minutes and 57 seconds (queued for 2 seconds)

⟳ latest ↗ 18b72ddd2 📁

No related merge requests found.

Pipeline Needs Jobs 27 Failed Jobs 2 Tests 0

Regular Installation	Basic Testing	Regular Testing	Benchmarks	Applications	Configurations	Finalization
Initialize	MPC Default Configuration	Privatization	Lowcomm	MPI IMB-MPI 2017	Lulesh	MPC Compilation Workshare
		Simple run	MPI Simple C	MPI IMB-NBC 2017	NAS-MZ MPI	MPC Debug Messages
			MPI Simple Fortran	MPI NBC	NAS-MZ OMP	MPC Disable Lowcomm
			OpenMP Tasking		miniFe	MPC Disable MPI
						MPC Disable Threads
						MPC PMIx
						MPC Process Mode
						MPC Process Mode (no Spack)
						MPC Slurm

Système de contrôle des versions

Le contrôle des versions

- **Git**: le plus connu actuellement. Il est utilisé par les développeurs pour suivre l'historique des modifications apportées au code source durant son évolution, et offrir la possibilité d'envisager différentes versions du projet.
- **Mercurial** : créé en 2005 par une société canadienne, cette solution open-source est similaire à Git mais plus simple.
- Les anciens...
 - CVS: À l'origine développé par les laboratoires Bell Telephone et utilisant le modèle client/serveur, il a connu un certain succès avant de se voir remplacer par des alternatives plus modernes comme Git ou Mercurial.
 - SVN: étendu à d'autres types de fichiers que du code source (comme la documentation), cette solution permettait une gestion centralisée et collaborative, mais est maintenant moins utilisé par les développeurs qui préfèrent Git.

GIT

Création de GIT

Git a commencé en 2005, créé par Linus Torvalds. Il s'agissait d'un logiciel libre distribué pour faciliter la gestion des versions numériques (VCS). Torvalds lui-même travaillait sur le noyau Linux et BitKeeper, un logiciel de suivi des modifications du code source. La relation entre les deux s'est finalement arrêtée suite à une crise de licence en 2005.

Torvalds alors se lance dans son propre VCS pour répondre à cette contrainte et nomme ce nouveau logiciel Git (tout en gardant un lien avec les contenus numériques). Il est devenu très populaire, notamment grâce au système distribué qu'il propose.

“Le développement de Git a commencé le 3 avril 2005. Torvalds annonça ce projet le 6 avril et se mit en auto-hébergement la journée suivante. Le premier fusionnement de branches s'est produit le 18 avril. Torvalds a atteint ses objectifs ; le 29 avril, le jeune Git fut testé enregistrant des correctifs pour l'arbre du noyau Linux à un rythme de 0.67 par seconde. Le 16 juin, Git a géré la sortie de la version du noyau 2.6.12.”

Commandes principales de Git

Un système de gestion de versions comme GIT permet:

- git init : Initialise un nouveau dépôt Git dans le répertoire actuel.
- git clone [url] : Clone un dépôt Git distant dans un nouveau répertoire local.
- git add [fichier] : Ajoute un fichier à l'index pour être suivi par Git.
- git commit -m "message" : Crée un nouveau commit avec les fichiers ajoutés à l'index, accompagné d'un message descriptif.
- git status : Affiche l'état actuel de la branche locale par rapport au référentiel distant, ainsi que les fichiers modifiés et non suivis.
- git branch [nom] : Crée une nouvelle branche avec le nom spécifié.
- git checkout [nom_branche] : Bascule vers une autre branche spécifiée.
- git merge [branche] : Fusionne une branche spécifiée dans la branche actuelle.
- git pull : Récupère les modifications depuis le référentiel distant et fusionne automatiquement avec la branche locale.
- git push : Envoie les modifications locales vers le référentiel distant.
- git log : Affiche l'historique des commits avec leurs messages, auteurs et horodatages.
- git diff [fichier] : Affiche les différences entre le fichier actuel et la dernière version indexée.
- git reset [fichier] : Annule les modifications d'un fichier spécifique dans la zone de préparation (index).

Définir son nom et email sur Git

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your.e-mail@domain.com"
```

→ \$HOME/.gitconfig

[user]

name = ...

email = ...

Par répo :

```
git config user.email "your-e.mail@entreprise-salaire.super"
```

→ <repo>/.gitconfig

Commande : git init

Description : Initialise un nouveau dépôt Git dans le répertoire actuel.

Fonctionnalités :

- Crée un nouveau dépôt Git vide dans le répertoire local.
- Initialise le suivi des modifications apportées aux fichiers dans ce répertoire.
- Permet de commencer à versionner les fichiers et de suivre leur historique de modifications.

Utilisation :

Exécuter git init dans le répertoire de travail pour initialiser un nouveau dépôt Git.

Utilisé généralement une fois au début d'un projet pour démarrer le suivi des modifications.

man git-init

Commande : git clone

Commande : git clone [url]

Description : Clone un dépôt Git distant dans un nouveau répertoire local.

Fonctionnalités :

- Copie l'intégralité du dépôt distant dans un nouveau répertoire local.
- Crée une copie exacte du dépôt, y compris toutes les branches, les commits et l'historique des modifications.
- Établit automatiquement un lien avec le dépôt distant d'origine pour faciliter la synchronisation.

Utilisation :

- Exécuter `git clone [url]` en remplaçant `[url]` par l'URL du dépôt distant que vous souhaitez cloner.

Utilisé pour récupérer un dépôt distant sur votre machine locale afin de travailler sur le code ou de contribuer au projet.

man git-clone

Commande : git add

Description : Ajoute un fichier à l'index pour être suivi par Git.

Fonctionnalités :

- Prépare un fichier spécifique pour être inclus dans le prochain commit.
- Met à jour l'index (zone de préparation) avec les modifications apportées au fichier.
- Permet à Git de suivre les changements effectués sur le fichier et de les inclure dans le prochain commit.

Utilisation :

- Exécuter `git add [fichier]` en remplaçant [fichier] par le nom du fichier que vous souhaitez ajouter à l'index.

Utilisé pour sélectionner les fichiers spécifiques à inclure dans le prochain commit, tout en laissant les autres fichiers non suivis.

man git-add

Commande : git branch [nom]

Description : Crée une nouvelle branche avec le nom spécifié.

Fonctionnalités :

- Permet de diviser le travail en différentes branches pour développer des fonctionnalités isolées ou des correctifs.
- Crée une nouvelle référence qui pointe vers le commit actuel, initialement basée sur la branche actuelle.
- Facilite le développement parallèle en permettant à plusieurs fonctionnalités d'être développées indépendamment les unes des autres.

Utilisation :

- Exécuter `git branch [nom]` en remplaçant `[nom]` par le nom de la nouvelle branche que vous souhaitez créer.
- Utilisé pour créer une nouvelle branche à partir du commit actuel, à partir de laquelle vous pouvez travailler sur de nouvelles fonctionnalités ou des correctifs.

Commande : `git checkout [nom_branche]`

Description : Bascule vers une autre branche spécifiée.

Fonctionnalités :

- Permet de naviguer entre différentes branches du dépôt Git.
- Met à jour le répertoire de travail et l'index pour refléter l'état de la branche spécifiée.
- Utilisé pour passer d'une branche à une autre afin de travailler sur des fonctionnalités distinctes ou de récupérer du code à partir d'une branche existante.

Utilisation :

- Exécuter `git checkout [nom_branche]` en remplaçant `[nom_branche]` par le nom de la branche vers laquelle vous souhaitez basculer.
- Utilisé pour passer d'une branche à une autre, récupérer du code à partir d'une branche spécifique ou créer une nouvelle branche à partir d'une branche existante.

Commande : git merge [branche]

Description : Fusionne une branche spécifiée dans la branche actuelle.

Fonctionnalités :

- Combine les modifications apportées à une branche spécifiée dans la branche actuelle.
- Intègre les changements des commits de la branche spécifiée dans l'historique de la branche actuelle.
- Permet de fusionner le travail effectué sur différentes fonctionnalités ou branches de développement dans une seule branche.

Utilisation :

- Exécuter `git merge [branche]` en remplaçant `[branche]` par le nom de la branche que vous souhaitez fusionner dans la branche actuelle.
- Utilisé pour intégrer les modifications d'une branche spécifique dans la branche actuelle, généralement après avoir terminé le développement d'une fonctionnalité ou d'une tâche sur une autre branche.

Commande : git pull

Description : Récupère les modifications depuis le référentiel distant et fusionne automatiquement avec la branche locale.

Fonctionnalités :

- Télécharge les modifications apportées au dépôt distant depuis la branche actuelle.
- Fusionne automatiquement les modifications téléchargées avec la branche locale, mettant à jour le répertoire de travail et l'historique des commits.
- Permet de synchroniser facilement le code local avec les dernières modifications apportées par d'autres contributeurs au dépôt distant.

Utilisation :

- Exécuter git pull pour télécharger et fusionner les modifications depuis le dépôt distant dans la branche locale.
- Utilisé pour mettre à jour le code local avec les dernières modifications du dépôt distant, avant de commencer à travailler sur de nouvelles fonctionnalités ou de contribuer au projet.

Commande : git push

Description : Envoie les modifications locales vers le référentiel distant.

Fonctionnalités :

- Transfère les commits locaux vers le référentiel distant, mettant à jour la branche distante avec les modifications locales.
- Permet de partager le travail réalisé localement avec d'autres contributeurs en publiant les modifications sur le référentiel distant.
- Facilite la collaboration en permettant aux membres de l'équipe de partager leurs contributions et de synchroniser le code entre les différentes branches et dépôts.

Utilisation :

- Exécuter git push pour envoyer les modifications locales vers le référentiel distant associé à la branche actuelle.
- Utilisé après avoir effectué des commits locaux pour publier les modifications sur le référentiel distant, assurant ainsi que d'autres membres de l'équipe puissent accéder aux dernières versions du code.

Commande : git log

Description : Affiche l'historique des commits avec leurs messages, auteurs et horodatages.

Fonctionnalités :

- Affiche une liste chronologique des commits effectués dans le dépôt.
- Fournit des informations détaillées sur chaque commit, y compris le message du commit, l'auteur, la date et l'heure du commit.
- Permet de visualiser l'évolution du code au fil du temps et de suivre les modifications apportées par les différents contributeurs.

Utilisation :

- Exécuter `git log` pour afficher l'historique des commits dans le répertoire de travail.
- Utilisé pour examiner l'historique des modifications, identifier les commits spécifiques et comprendre l'évolution du code au fil du temps.

Voir aussi tig(1)

Commande : git diff [version] [-- fichier]

Description : Affiche les différences entre le fichier actuel et la dernière version indexée.

Fonctionnalités :

- Affiche les modifications apportées au fichier depuis la dernière fois qu'il a été ajouté à l'index.
- Met en évidence les lignes ajoutées et supprimées pour visualiser les changements avec précision.
- Permet de comparer les modifications locales par rapport à la dernière version validée dans le dépôt.

Utilisation :

- Exécuter `git diff [fichier]` pour afficher les différences pour un fichier spécifique.
- Utilisé pour examiner les modifications apportées à un fichier depuis la dernière fois qu'il a été ajouté à l'index, permettant ainsi de vérifier les changements avant de les valider.

Commande : git reset [version] [--<mode> | -- fichier]

Description : Annule les modifications d'un fichier spécifique dans la zone de préparation (index).

Fonctionnalités :

- Annule les modifications apportées à un fichier spécifique depuis la dernière fois qu'il a été ajouté à l'index.
- Retire le fichier de la zone de préparation (index) sans modifier les modifications locales.
- Permet de désindexer un fichier avant de le réindexer avec de nouvelles modifications.

Utilisation :

- Exécuter git reset [fichier] pour annuler les modifications d'un fichier spécifique dans la zone de préparation.
- Utilisé pour corriger les erreurs avant de valider les changements en désindexant un fichier pour le réexaminer ou pour annuler les modifications inutiles.