

Optimiser la collaboration dans les projets open source : comment maintenir un dépôt GitHub clair et compréhensible

Mémoire de fin d'études

Bouchet Ulysse

Efrei Paris - M2-APP-BDML

CEA Paris-Saclay - DRT/LIST/DIASI/SIALV/LVA

2024

Table des matières

Table des matières	1
Abstract	3
Introduction	4
I. Contexte du mémoire	5
1. Présentation des organismes	5
2. Présentation de la mission	5
3. Objectif du mémoire	7
II. Fondements théoriques	8
1. Qu'est-ce que l'open source ?	8
a. Principes clés de l'open source	8
b. Principaux avantages de l'open source	9
c. Inconvénients de l'open source	10
2. Que sont Git et GitHub ?	10
a. Intérêt et fonctionnement de Git	11
b. La plateforme numéro un : GitHub	13
III. État de l'art	15
1. Kubernetes	15
a. Qu'est-ce que Kubernetes ?	15
b. Les bonnes pratiques de son dépôt GitHub	15
2. Visual Studio Code	17
a. Qu'est-ce que VS Code ?	17
b. Les bonnes pratiques de son dépôt GitHub	18
3. Rust	19
a. Qu'est-ce que Rust ?	19
b. Les bonnes pratiques de son dépôt GitHub	19
4. Compte-rendu	20
a. Fichier README.md structuré	20
b. Licence et code de conduite	20
c. Documentation complète	20
d. Guidelines de contribution détaillées	21
e. Issues et pull requests organisées	21
f. Utilisation de GitHub Projects	21
g. Utilisation de GitHub Actions	21
h. Forums ou applications de chat pour regrouper les développeurs	21
IV. Comparaison avec Pixano	22
1. Qu'est-ce que Pixano ?	22
2. Le dépôt GitHub de Pixano	25

✓ Fichier README.md	25
☹ Licence et code de conduite	25
✓ Documentation	26
✓ Guidelines de contribution	26
✓ Issues et pull requests	26
☹ GitHub Projects	27
✓ GitHub Actions	27
✗ Forums ou chats	27
3. Récapitulatif des conseils pour améliorer le dépôt	28
V. Bilan	29
Remerciements	30
Annexe	31
Lexique	32
Références	39



Abstract

Open source software has become a cornerstone of modern computing, fostering rapid innovation, effective global collaboration, and crucial transparency. Major projects like Google Kubernetes, Apache Spark, Microsoft Visual Studio Code, Rust, Node.js, and TensorFlow highlight the immense potential of open source. Central to these projects' success is the use of Git for version control and GitHub as the primary platform, hosting over 100 million users and billions of contributions annually. However, with more than 420 million repositories available on GitHub as of 2023, many projects lack the resources to maintain clear, accessible repositories, hindering their growth and innovation.

This thesis aims to identify best practices for maintaining a clear and comprehensible GitHub repository, which can significantly enhance usability and collaboration in open source projects. The study focuses on analyzing successful projects like Kubernetes, Visual Studio Code, and Rust, comparing their repository management practices with those of the Pixano project. The methodology includes examining aspects such as structured README.md files, licensing, codes of conduct, detailed contribution guidelines, organized issues and pull requests, and the use of GitHub Projects and Actions.

Key findings indicate that well-maintained repositories share common traits: comprehensive documentation, clear contribution guidelines, efficient issue and pull request management, and the strategic use of GitHub features. Applying these practices to the Pixano project revealed areas for improvement, such as better integration of community forums or chat applications for developer interaction.

The results of this study offer actionable insights for developers aiming to optimize their GitHub repositories, ensuring they are more accessible, collaborative, and conducive to innovation. By enhancing repository management, open source projects can attract more contributors, streamline development processes, and achieve greater impact in the technology community. This research contributes to the broader understanding of effective open source project management, supporting the ongoing growth and success of the open source ecosystem.

Introduction

Aujourd'hui, l'*open source* joue un rôle crucial dans le paysage de l'informatique. *Google Kubernetes*, *Apache Spark*, *Microsoft VS Code*, *Rust*, *Node.js*, *Tensorflow*, mais également bien d'autres projets d'envergure ont fait le choix de l'*open source*, permettant une innovation plus rapide, une collaboration mondiale efficace et une transparence plus que jamais nécessaire.

Dans le domaine de l'informatique, l'*open source* est aujourd'hui intrinsèquement lié à *Git*, une technologie permettant la gestion collaborative de code, et la plateforme *GitHub* s'est aujourd'hui imposée comme numéro un de l'écosystème. Avec plus de 100 millions d'utilisateurs et plusieurs milliards de contributions chaque année, il est important pour chaque projet de définir des directives claires pour empêcher les développeurs d'empiéter sur le travail de leurs pairs.

Les projets les plus établis, ou soutenus par de grandes structures, possèdent déjà souvent le nécessaire pour une collaboration facile. Cependant, avec plus de 420 millions de projets disponibles en 2023 sur la plateforme, il est impossible pour chacun d'entre eux de posséder ces moyens, laissant ainsi de nombreux *dépôts* mal documentés et peu accessibles pour les nouveaux utilisateurs ou contributeurs, limitant donc leur potentiel de croissance et d'innovation.

Ce mémoire vise à explorer les meilleures pratiques pour maintenir un dépôt GitHub clair et compréhensible, en analysant différents projets de grande envergure. En examinant plusieurs aspects tels que la documentation et les fonctionnalités de GitHub utilisées, nous chercherons à répondre à la question suivante : comment mettre en place un dépôt GitHub facilitant l'utilisation et la collaboration dans des projets en *open source* ?

Dans un premier temps, nous présenterons le contexte de ce mémoire, avant d'analyser les fondements théoriques de l'*open source*. Ensuite, nous établirons l'état de l'art des projets les plus actifs afin de déterminer les bonnes pratiques pour la gestion d'un dépôt, puis nous les mettrons en parallèle avec celles appliquées pour le projet *Pixano* développé en entreprise, en cherchant des propositions pour optimiser son développement. Pour finir, nous dresserons un bilan en espérant contribuer à la création de projets *open source* plus accessibles, collaboratifs et réussis.

Attention

Ce mémoire s'adresse principalement aux personnes initiées au développement collaboratif et utilise parfois des termes assez techniques. Cependant, un lexique très complet est proposé en annexe. Les termes inclus dans le lexique sont indiqués en italique lors de leur première utilisation.

I. Contexte du mémoire

Ce mémoire est rédigé dans le cadre de mes études d'ingénieur en informatique, avec une spécialisation en *Big Data* et *Machine Learning* à l'*Efrei* Paris, en formation par alternance au CEA, sous l'encadrement de ma professeure référente Mme Stefani El Kalamouni, et de mon maître d'apprentissage, M. Jaonary Rabarisoa.

1. Présentation des organismes

L'Efrei, anciennement École française d'électronique et d'informatique, est une école d'ingénieurs privée située à Villejuif. Fondée en 1936, elle propose des cursus variés en technologies de l'information, sécurité et réseaux, *systèmes embarqués*, ou encore en science des données. L'Efrei met l'accent sur l'internationalisation, l'innovation et l'entrepreneuriat et propose de nombreux échanges et partenariats avec des universités et entreprises à travers le monde.

Le CEA, ou Commissariat à l'énergie atomique et aux énergies alternatives, est un organisme public français de recherche, également qualifié en tant qu'*EPIC* (Établissement public à caractère industriel et commercial en France). Fondé en 1945, il a joué un rôle clé dans le développement du nucléaire en France, mais ses activités se sont élargies, incluant aujourd'hui les domaines de la défense, des technologies de l'information, de la santé et de l'environnement.

2. Présentation de la mission

En tant qu'alternant, j'ai rejoint le CEA pour une durée de deux ans, de septembre 2022 à septembre 2024. Ma mission se déroule au sein du LVA (Laboratoire de Vision et d'Apprentissage pour l'analyse de scène), sur le site *Nano-Innov* du CEA Paris Saclay.



fig. 1 : Diagramme représentant la position du LVA au sein du CEA
(définition des acronymes dans le lexique en annexe)

Le LVA est l'un des plus gros laboratoires de Nano-Innov : une quarantaine de chercheurs, stagiaires, thésards, sont répartis sur différents projets touchant au machine learning, comme la détection de *montes* de vaches dans des élevages ou bien le *tracking* des joueurs et du ballon dans des sports collectifs. La plupart de ces projets ont un besoin en commun

: celui d'annoter des données afin de créer des *datasets* pour entraîner leurs *modèles* de machine learning.

C'est pour répondre à ce besoin que le LVA a lancé le développement de Pixano, un outil open source d'annotation de données, dont les fonctionnalités sont créées sur mesure selon les besoins des chercheurs du laboratoire. Le développement initial de Pixano a débuté en 2019; cependant, l'architecture et les technologies alors utilisées rendaient son maintien, sa prise en main et surtout son utilisation difficile pour les chercheurs en IA. Il a alors été décidé en 2022 de se baser sur l'expérience et les connaissances acquises lors du développement de Pixano pour en développer une nouvelle version, plus pratique pour ses utilisateurs.



fig. 2 : Logo de Pixano

C'est pour contribuer au développement de cette nouvelle version de Pixano que j'ai été recruté par le CEA, rejoignant ainsi une équipe composée d'ingénieurs chercheurs et d'externes à l'entreprise pouvant être des développeurs ou des designers d'UI/UX. Cette équipe est menée par mon maître d'apprentissage, l'un des développeurs originels de Pixano. Nous nous consacrons ainsi au développement de nouvelles fonctionnalités, tout en restant à l'écoute de nos utilisateurs afin de corriger les bugs rapidement.



3. Objectif du mémoire

En définissant le sujet de ce mémoire, je n'avais pas pour but de révolutionner le monde de l'open source et de la collaboration en proposant des idées nouvelles; mon objectif était plutôt de l'ordre du développement personnel. En effet, l'open source est un sujet qui m'intéresse tout particulièrement, du fait des valeurs qu'il défend. Je souhaitais ainsi acquérir de nouvelles connaissances dans ce domaine ainsi que sur le développement en collaboration efficace, qui sera une compétence plus que nécessaire pour mon avenir en entreprise. Toutefois, les conclusions que je tirerai pourront servir à d'autres développeurs qui, comme moi, sont intéressés par l'open source et le travail en collaboration.

Ainsi, j'ai pour objectif d'étudier ce qui se fait de mieux en la matière, de comparer cela avec ce qui est fait d'un point de vue plus interne avec Pixano, pour mieux comprendre le fonctionnement du développement collaboratif, et proposer des solutions pour améliorer la collaboration dans Pixano.

II. Fondements théoriques

1. Qu'est-ce que l'open source ?

L'open source, parfois appelé *code source* ouvert, est un mouvement né à la fin des années 1990 qui prône la liberté dans l'accès, la modification et la redistribution libre du code source d'un *logiciel*. L'open source est en opposition aux *logiciels propriétaires*, ou *closed source*, pour lesquels seul l'auteur a accès au code source, et les utilisateurs doivent, souvent pendant l'installation, accepter une licence leur empêchant une utilisation non conforme du logiciel. Des exemples célèbres de logiciels propriétaires sont la suite Office ou Photoshop.

Attention

Les logiciels open source ne sont pas à confondre avec les logiciels libres, bien que les deux notions soient très proches et même interchangeable dans la plupart des contextes. Le logiciel libre, plus ancien, a vu une partie des membres de sa communauté s'en séparer pour devenir le mouvement open source. La principale différence entre les deux mouvements se situe dans les valeurs qu'ils prônent. Cette citation de Richard Stallman, initiateur du logiciel libre, compare habilement les deux concepts : "L'open source est une méthodologie de développement; le logiciel libre est un mouvement de société".

a. Principes clés de l'open source

L'*Open Source Initiative*, organisation principale de soutien au mouvement open source, définit les 10 principes suivants, à respecter pour tous les logiciels souhaitant obtenir une licence open source :

- Le logiciel doit pouvoir être redistribué librement;
- L'accès au code source doit être libre et facile;
- La modification et les créations dérivées doivent être autorisées;
- L'intégrité du code source de l'auteur doit être respectée;
- La licence ne doit pas discriminer des personnes ou des groupes;
- La licence ne doit pas discriminer des champs d'application;
- La licence doit être appliquée à tous ceux qui reçoivent le logiciel;
- La licence ne doit pas être spécifique à une distribution spécifique;
- La licence ne doit pas restreindre la distribution d'autres logiciels;
- La licence doit être neutre vis-à-vis de la technologie.

Ainsi, contrairement aux idées reçues, l'open source n'impose pas la gratuité du logiciel.

En savoir +

Pour mieux comprendre ces principes, vous êtes invités à lire l'article The Open Source Definition, cité parmi les références en annexe.

b. Principaux avantages de l'open source

Au-delà de l'avantage évident pour les utilisateurs d'avoir en général accès à un logiciel de façon gratuite, l'open source apporte bien des atouts aux logiciels qui font ce choix.

Pour commencer, l'open source permet de développer un code de meilleure qualité. En effet, du code créé par une petite équipe de développeurs sera souvent de moins bonne qualité que du code rédigé par des milliers de développeurs ayant des expériences dans des technologies, industries ou projets variés. De plus, les bugs sont identifiés plus rapidement, étant donné que le code est constamment en train d'être lu par plusieurs personnes. Même le code écrit par une seule personne est souvent de meilleure qualité lorsqu'il est passé en open source; en effet, comme personne ne veut que ses futurs employeurs ou collaborateurs remarquent de mauvaises habitudes dans son code, l'auteur s'appliquera plus particulièrement à la production d'un code propre. Qui plus est, l'open source permet aux développeurs novices de s'améliorer en apprenant du code proposé par leurs pairs plus expérimentés.

D'autres avantages de l'open source sont la flexibilité et la possibilité de customisation apportées naturellement. En effet, un code source accessible que l'on peut modifier permet de l'adapter pour mieux répondre à nos besoins plus spécifiques. C'est grâce à cela que de nombreuses *extensions* sont nées pour compléter l'éditeur de code VS Code, permettant par exemple de l'utiliser pour développer des programmes dans d'autres *langages* que ceux supportés nativement.

La sécurité est encore un avantage important de l'open source. Le code source étant ouvert, il est plus facile pour les experts en sécurité d'identifier et de corriger les vulnérabilités avant que des utilisateurs puissent en être victimes. Les logiciels open source peuvent également être audités par des tiers indépendants, ce qui renforce la confiance en leur sécurité. De plus, les projets open source ne peuvent pas utiliser de n'importe quelle manière les données de leurs utilisateurs pour par exemple les revendre, comme Google ou *Meta*, puisque la communauté pourrait rapidement repérer ces abus, ruinant ainsi la réputation du logiciel et de son propriétaire. Ces qualités poussent d'ailleurs de plus en plus les gouvernements à favoriser des projets open source pour des logiciels sensibles.

Naturellement, rien n'est parfait et l'open source ne déroge pas à la règle, possédant également des inconvénients.

En savoir +

Les avantages cités ci-dessus sont loin d'être une liste exhaustive. D'autres avantages peuvent être la liberté d'utilisation, l'indépendance vis-à-vis des fournisseurs, le support communautaire, la documentation riche ou encore la pérennité du logiciel.

c. Inconvénients de l'open source

Bien qu'évoquée en tant qu'avantage, la sécurité peut également se retrouver en inconvénient de l'open source. En effet, s'il est facile pour les experts d'identifier les failles du code, cela l'est également pour les personnes malveillantes, qui pourront les exploiter plutôt que les corriger. En pratique, les logiciels propriétaires et open source possèdent généralement le même nombre de vulnérabilités.

Les logiciels open source sont souvent disponibles en tant que simple dossier comprenant le code source, avec quelques instructions pour l'installation et la configuration. Cela peut être un obstacle pour les utilisateurs non techniques ou pour les petites entreprises sans personnel informatique dédié. La documentation peut également être insuffisante ou mal organisée, rendant difficile la résolution des problèmes ou l'apprentissage de nouvelles fonctionnalités.

Le développement en collaboration avec des inconnus, sans organisation, peut également s'avérer être une tâche assez complexe. Comment s'assurer que Luigi, en Italie, et John, aux États-Unis, travaillent de manière indépendante sur le même fichier ? C'est pour répondre à ce problème que *Linus Torvalds* a créé Git.

2. Que sont Git et GitHub ?

En 1991, Linus Torvalds, ingénieur finlandais, publie le premier prototype de *Linux*, un *noyau de système d'exploitation* devenu aujourd'hui célèbre et très apprécié dans la communauté informatique. Il contribue encore à ce jour au développement du noyau, mais c'est en 2005 qu'il va révolutionner encore une fois le monde du développement collaboratif grâce à son nouvel outil : Git.



fig. 3 : Linus Torvalds

En effet, il n'est alors plus seul à développer Linux, le projet est depuis devenu énorme. Pour gérer les différentes versions du code, lui et ses collaborateurs utilisent *BitKeeper*. Cependant, ce logiciel est propriétaire, ce qui n'est pas au goût de la communauté open source, parmi laquelle Linus se place en figure de proue. En quelques mois seulement, il réussit à développer une version fonctionnelle de Git, qui va alors rapidement s'imposer comme système n°1 de contrôle de versions de code source.

En savoir +

À l'instar de Linux, et de la grande majorité des projets qui utilisent Git, ce dernier est également open source et son développement continue à ce jour !

a. Intérêt et fonctionnement de Git

Git est un outil qui permet de savoir qui a travaillé sur quel fichier, quand et comment. Basé sur un *système de contrôle distribué* et décentralisé, il permet la gestion des versions de projets de toute taille, avec une rapidité et une efficacité remarquables. Grâce à Git, la collaboration entre plusieurs développeurs situés dans différents endroits est rendue possible, et même très simple !



fig. 4 : Logo de Git

L'une des principales caractéristiques de Git est son architecture distribuée. Contrairement aux systèmes de contrôle de version centralisés, où il existe un seul *référentiel central* : Git permet à chaque développeur de cloner l'intégralité du dépôt, y compris son historique complet, sur son propre ordinateur.

La *décentralisation* présente plusieurs avantages, comme la fiabilité : le code source étant copié sur les machines des utilisateurs, une perte de données sur le serveur central n'entraînera pas la perte du projet complet.

De plus, les opérations courantes telles que les *commits* (pour enregistrer les modifications effectuées), les *diffs* (pour comparer différentes versions d'un fichier) ou les historiques sont extrêmement rapides étant donné qu'elles s'effectuent en local, sans communication avec le serveur.

Enfin, les développeurs peuvent continuer à travailler sur le code et faire des commits, même lorsqu'ils n'ont pas accès à internet. Les modifications peuvent être synchronisées par la suite sur le serveur central, une fois la connexion établie.

Git fonctionne en capturant des “*snapshots*” du projet à chaque point de commit. Plutôt que de stocker les différences entre les versions (comme c'est le cas pour certains autres systèmes de contrôle de version), Git stocke une image complète de l'ensemble du projet. Si les fichiers n'ont pas été modifiés, Git ne stocke pas à nouveau le fichier, mais crée simplement un lien vers le fichier précédent déjà stocké. Cette méthode permet de rendre Git très performant et efficace en termes de stockage.

Un autre aspect crucial de Git est sa gestion des *branches*. Les branches permettent aux développeurs de travailler sur des fonctionnalités ou des correctifs séparément du code principal. Cela facilite le développement en parallèle et permet d'intégrer les nouvelles fonctionnalités ou les correctifs uniquement lorsqu'ils sont prêts et testés. La création, la fusion et la suppression des branches sont des opérations rapides et peu coûteuses grâce à la manière dont Git gère les commits.

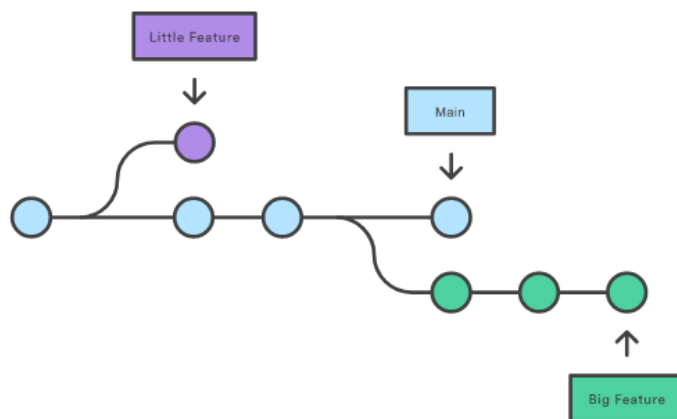


fig. 5 : Schéma représentant différentes branches d'un dépôt Git

En savoir +

Voici les commandes les plus courantes de Git :

- `git init` : Permet d'initialiser un nouveau dépôt.
- `git clone` : Permet de cloner un dépôt déjà existant.
- `git add` : Permet d'ajouter des fichiers ou des modifications à l'index.
- `git commit` : Permet de sauvegarder les modifications dans l'historique du projet.
- `git status` : Permet de vérifier l'état des fichiers dans le répertoire de travail.
- `git branch` : Permet de gérer les branches.
- `git merge` : Permet de fusionner des branches.
- `git pull` : Permet de récupérer les modifications du serveur.
- `git push` : Permet d'envoyer les modifications au serveur.

La combinaison de ces commandes et fonctionnalités fait de Git un outil puissant et flexible pour le contrôle de version, adapté aussi bien aux projets open source qu'aux développements en entreprise. Il a ainsi révolutionné la manière dont les développeurs collaborent sur le code, apportant une structure et une efficacité sans précédent au processus de développement logiciel.

b. La plateforme numéro un : GitHub

Bien que Git soit un outil puissant et efficace pour la collaboration, il ne permet pas de couvrir tous les besoins des développeurs. En effet, il manque une interface plus conviviale que le *terminal de commande*, et surtout un moyen d'héberger les dépôts open source de tous les projets. C'est pour répondre à ce problème que les développeurs du monde entier utilisent la plateforme GitHub.

Fondée en 2008, GitHub est une plateforme de développement collaboratif qui s'est rapidement imposée comme la référence en matière d'hébergement de projets open source et de collaboration entre développeurs. Son succès repose sur une combinaison d'outils robustes, de fonctionnalités communautaires et d'une interface utilisateur accueillante, qui facilitent le travail collaboratif et la gestion de projets.




fig. 6 : Logo de GitHub

GitHub propose de nombreuses fonctionnalités pour faciliter le développement collaboratif. Par ailleurs, GitHub n'est pas exclusif aux projets open source : les dépôts peuvent être publics ou privés selon les besoins du projet.

L'une des fonctionnalités les plus utiles de GitHub est sa gestion des "issues". Une issue représente un bug ou une tâche à effectuer, à laquelle des développeurs peuvent être assignés, facilitant la gestion des projets. La fonction de "project board" permet également une organisation visuelle des tâches à l'aide de tableaux, permettant planification et suivi efficace des progrès.

Une autre fonctionnalité importante est celle des "pull requests". Au cœur de la collaboration sur GitHub, elles permettent aux développeurs de proposer des modifications au code, qui peuvent ensuite être examinées, validées et fusionnées dans le projet.



principal par les autres développeurs. Grâce à des outils intégrés, il est possible de commenter des lignes spécifiques du code afin de s'assurer de sa qualité avant son intégration.

GitHub propose également *GitHub Actions*, qui permet l'automatisation de tests et de validation du code, après chaque commit ou pull request. GitHub s'intègre également avec de nombreux services tiers comme *Slack* ou *Trello*, afin de facilement s'intégrer dans les flux de travail existants. Enfin, GitHub propose GitHub Pages, qui permet aux utilisateurs de déployer des sites web reliés à un dépôt, par exemple pour héberger la documentation de ces projets.

Attention

Il existe d'autres plateformes similaires à GitHub, comme GitLab ou Framagit, qui sont plus orientées en tant que solution locale pour les entreprises. Ainsi, GitHub reste le leader incontesté dans la communauté open source. La preuve : même Git a choisi GitHub pour héberger son projet ! Ce mémoire se concentre donc sur cette plateforme en particulier.

Ainsi, GitHub n'est pas seulement un outil de gestion de code, mais une plateforme complète de développement collaboratif qui a transformé la manière dont les développeurs travaillent ensemble. Son intégration avec Git, combinée à ses fonctionnalités de gestion de projets, de collaboration et d'automatisation, en fait un atout indispensable pour toute équipe de développement moderne. Que ce soit pour des projets open source ou des développements d'entreprise, GitHub facilite la création de logiciels de haute qualité de manière collaborative et efficace.

Maintenant que les fondements théoriques sur l'open source, Git et GitHub ont été présentés, nous allons étudier l'état de l'art des projets open source, en analysant les dépôts GitHub de certains des plus gros projets actuels.

III. État de l'art

Nous allons maintenant étudier les dépôts GitHub de différents projets majeurs, afin de déterminer les bonnes pratiques de la gestion d'un projet open source.

En savoir +

Sur chaque dépôt GitHub, on retrouve un fichier README.md. Ce fichier permet généralement de présenter rapidement le projet et comment l'utiliser, mais propose aussi des liens vers une documentation plus complète et une ligne directrice pour contribuer au projet.

1. Kubernetes

a. Qu'est-ce que Kubernetes ?

Kubernetes est un système open source de gestion de *conteneurs*, initialement développé par Google, qui permet l'automatisation du déploiement, de la mise à l'échelle et de la gestion des applications conteneurisées. Avec plus de 3000 contributeurs, dix ans d'ancienneté et une centaine de commits par semaine, sa popularité et son adoption mondiale en font un excellent exemple de gestion réussie d'un projet open source.

b. Les bonnes pratiques de son dépôt GitHub

En parcourant le dépôt GitHub de Kubernetes, on constate assez facilement que plusieurs pratiques sont mises en place pour faciliter son utilisation, mais aussi son développement en collaboration.

Tout d'abord, on peut découvrir deux choses : la licence open source et le code de conduite. La licence décrit les conditions d'utilisations et les droits d'auteur, de réutilisation, ou de redistribution du projet. Quant au code de conduite, il définit dans un aspect moral comment les membres de la communauté Kubernetes devraient se comporter. Cela couvre politesse, respect des autres quelque soit leur origine, genre, orientation sexuelle, âge, etc.

En parcourant le fichier *README.md*, on retrouve d'abord un lien vers la documentation du projet, un aspect crucial pour tout projet open source. Kubernetes en propose une très complète : allant de l'installation jusqu'à des conseils pour déployer l'outil en production, en passant par une aide à la configuration.

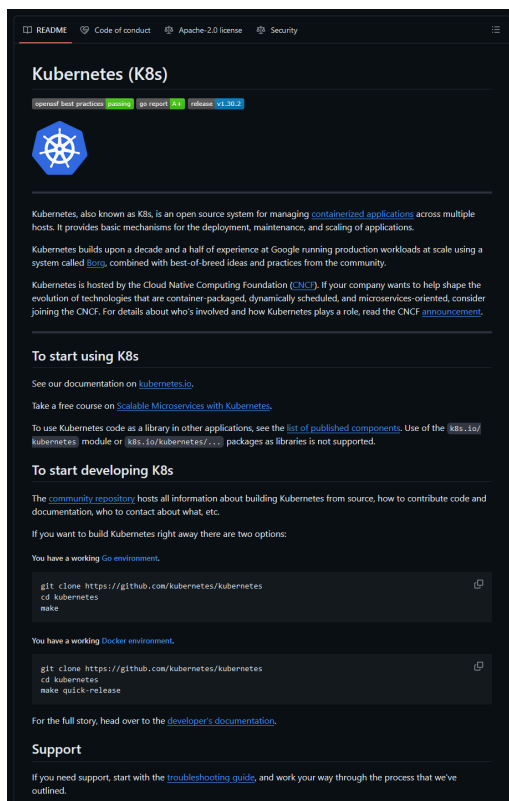


fig. 7 : Fichier README.md de Kubernetes, avec des onglets redirigeant vers le code de conduite et la licence

Ensuite, on retrouve les lignes directrices de contribution : Kubernetes fournit un guide détaillé pour les contributeurs, sous forme de texte mais aussi de vidéos. On y trouve des instructions sur les pré-requis, de l'aide à la configuration de l'environnement de développement, des contraintes sur le style de codage, mais aussi des explications sur les tests et le processus de soumission des pull requests.

Kubernetes propose également une gestion efficace des issues et des pull requests, grâce à un système bien organisé utilisant des tags et attribuant des niveaux de priorité distincts en fonction de l'ampleur du problème. De plus, des développeurs spécifiques sont désignés pour régler certains bugs, ou bien pour examiner le code proposé par les nouvelles pull requests. L'utilisation de *bots* permet de mettre en place un suivi plus efficace des issues, mais également d'automatiser les tests sur la qualité et la fonctionnalité du code.

Pour faciliter le suivi des issues et des pull requests, Kubernetes utilise *GitHub Projects*, qui permet de mettre en place des *dashboards* répartissant les issues selon leur statut : *Backlog*, *In Progress* ou encore *Done*. Grâce à cela, il est facile de visualiser ce qui est fait, ce qui est en train d'être fait, ou ce qui reste à faire.

The screenshot displays the Kubernetes Workloads dashboard, updated on Mar 13. It is organized into four columns representing different stages of issue resolution:

- Freezer (8 issues):** Issues that are blocked or on hold. Examples include "Stricter checking/validation of statefulset pvcs" and "Need pattern for updating controllers that works with extensions (TPR/Aggregated)".
- Backlog (31 issues):** Issues that are planned for future work. Examples include "Modifying nodeSelector on StatefulSet doesn't reschedule Pods" and "TTL support for ConfigMaps tied to job completion".
- In Progress (2 issues):** Issues currently being worked on. Examples include "Facilitate ConfigMap rollouts / management" and "Deployment enters creation hot-loop when rs field is mutated by API server".
- Done (306 issues):** Issues that have been resolved. Examples include "Support MinReadySeconds on StatefulSet" and "Environment variables containing colon not allowed".

Each issue card provides details such as the issue number, the person who opened it, and various tags indicating the issue's nature (e.g., bug, feature), priority, and the area of the code it affects.

fig. 8 : Dashboard présentant les issues de Kubernetes


Enfin, Kubernetes met en place une structure de gouvernance ouverte pour le projet : des comités et des groupes de travail définis sont responsables de différents aspects du projet, et les décisions sont prises de manière collaborative, lors de réunions auxquelles les contributeurs peuvent participer.

Grâce à toutes ces pratiques, Kubernetes maintient la qualité et la sécurité de son code, tout en favorisant l'engagement et l'activité de sa communauté.

2. Visual Studio Code

a. Qu'est-ce que VS Code ?

Visual Studio Code, couramment appelé VS Code, est un éditeur de code source gratuit, léger et puissant, développé par Microsoft. Il est conçu pour offrir une expérience de développement rapide et fluide, tout en prenant en charge un large éventail de langages de programmation grâce à son écosystème d'extensions. Depuis son lancement en 2015, VS



Code est devenu l'un des éditeurs de code les plus populaires au monde, apprécié pour ses fonctionnalités avancées, sa flexibilité et son interface utilisateur intuitive. Aujourd'hui, 2000 contributeurs sont à l'origine d'environ 200 commits par semaine pour continuer à améliorer l'éditeur.

b. Les bonnes pratiques de son dépôt GitHub

Comme pour Kubernetes, après exploration du dépôt GitHub de VS Code, on peut identifier plusieurs bonnes pratiques qui facilitent à la fois l'utilisation et la contribution au projet.

On y retrouve d'abord la licence et le code de conduite, dont les termes sont similaires à ceux de Kubernetes.

Comme celui de ce dernier, le fichier README.md de VS Code est particulièrement bien structuré. Il contient un lien vers le site officiel du projet, sur lequel on peut trouver la documentation, un lien de téléchargement, mais aussi un blog et une FAQ (Foire aux Questions) pour mieux comprendre ce que représente le logiciel.


VS Code se distingue par un accent fort mis sur les extensions et leur développement. Ainsi, le dépôt propose une documentation complète dédiée aux développeurs d'extensions, incluant des exemples de code, des instructions détaillées sur l'API de VS Code, et des guides pour la publication d'extensions sur le *marketplace* de VS Code. Cette approche encourage la création d'un écosystème riche et diversifié, contribuant à la popularité de l'éditeur.

Une base de lignes directrices pour contribuer se trouve également directement dans le fichier README.md. Comme pour Kubernetes, on y trouve des guides de configuration, une définition des standards de code appliqués au projet, et des instructions pour soumettre des pull requests.

Pour la gestion de ces dernières, l'approche de VS Code est similaire à celle de Kubernetes, utilisant des tags et des bots. On remarque tout de même que les tags sont moins utilisés que pour les issues de Kubernetes; en revanche, on peut constater plus d'interactions humaines, les développeurs de VS Code répondant à la majorité des issues.

Pour compléter son workflow, VS Code se sert de GitHub Actions, afin d'effectuer de nombreux checks et tests pour s'assurer pleinement que les modifications apportées au code n'aient pas de répercussions néfastes et inattendues.

Enfin, on remarque que VS Code utilise la fonctionnalité "Wiki" de GitHub, afin de détailler de nombreuses informations à propos du projet, regroupant la *roadmap* et le



fonctionnement de la gestion du projet, des guides pour contribuer, ou encore une documentation du projet.

Ainsi, comme Kubernetes, VS Code réussit à maintenir un haut niveau de qualité de code tout en proposant un développement et une collaboration efficace.

3. Rust

a. Qu'est-ce que Rust ?

Rust est un langage de programmation système, open source, conçu pour être sûr, concurrentiel et performant. Initialement développé par *Mozilla Research* et maintenant maintenu par la communauté, Rust vise à offrir des performances similaires à celles du C et du C++ tout en garantissant la *sécurité mémoire* sans avoir besoin d'un *garbage collector*. Grâce à son système de gestion de la mémoire unique, Rust est particulièrement adapté aux systèmes embarqués, aux applications “*low latency*” et aux projets nécessitant une grande sécurité. Depuis 2010, Rust a rapidement gagné en popularité et compte aujourd'hui presque 5000 contributeurs totaux, pour environ 500 commits par semaine.

b. Les bonnes pratiques de son dépôt GitHub

Une fois encore, en parcourant le dépôt GitHub de Rust, il est possible d'identifier plusieurs bonnes pratiques pour faciliter l'utilisation et la contribution au projet.

On retrouve comme toujours la licence et le code de conduite, ainsi que le fichier README.md, bien structuré et très informatif. Il présente les points forts du langage, ainsi que des liens vers la documentation officielle, le guide de contribution, et le forum de la communauté.

Comme d'habitude, la documentation est complète et accessible et le guide de contribution est détaillé. Pour ce qui est des issues et des pull requests, Rust combine le meilleur des pratiques de Kubernetes et de VS Code, en utilisant des tags complets et des bots, mais aussi avec des développeurs très actifs pour répondre aux problèmes et relire le code proposé.

Enfin, Rust utilise également GitHub Actions et GitHub Projects pour fluidifier son workflow. Toutes ces pratiques lui permettent lui aussi de conserver la qualité de son code et l'engagement de sa communauté.

4. Compte-rendu

En comparant les différents dépôts étudiés, ainsi que d'autres dépôts populaires comme Spark, Node.js ou Tensorflow, il est possible de dégager des pratiques communes pour créer un dépôt propre, accessible et facilitant la collaboration.

a. Fichier README.md structuré

Le fichier README.md est la première impression que les utilisateurs et les contributeurs potentiels auront du projet. Un README bien structuré donne une présentation claire et concise du projet, ce qui est crucial pour attirer et retenir l'intérêt des utilisateurs et des contributeurs.

Un bon fichier README.md doit contenir les informations suivantes (ou des liens redirigeant vers celles-ci) :

- Une introduction ou une brève description du projet;
- Des instructions pour installer et configurer le projet;
- Des exemples ou des cas d'utilisation du projet;
- Des directives pour contribuer au projet;
- Et d'autres liens utiles liés au projet (site officiel, documentation, réseaux sociaux...)

b. Licence et code de conduite


La licence et le code de conduite définissent le cadre légal et éthique pour l'utilisation et la contribution au projet. Cela assure une utilisation correcte et encourage un comportement respectueux au sein de la communauté.

La licence doit détailler les droits d'utilisation, de modification et de distribution du projet, afin de protéger les auteurs mais aussi les utilisateurs.

Le code de conduite doit établir les attentes de comportement pour tous les participants, pour s'assurer d'un environnement inclusif et respectueux.

c. Documentation complète

Une documentation complète et accessible est essentielle pour que les utilisateurs puissent comprendre et utiliser efficacement le projet. Elle sert également pour les contributeurs qui souhaitent s'impliquer dans le développement.



Une documentation complète comporte par exemple une documentation pour l'utilisateur, une documentation pour le développeur, et une documentation de l'API pour les projets en possédant.

d. Guidelines de contribution détaillées

Des *guidelines* claires facilitent la contribution en définissant les attentes et les processus à suivre. Elles permettent aux nouveaux contributeurs de s'intégrer plus facilement et assurent une qualité et une cohérence du code.

Elles peuvent comprendre des pré-requis, des standards de codage, ou des informations sur les processus de soumission de pull requests.

e. Issues et pull requests organisées

Une gestion efficace des issues et des pull requests permet de suivre les bugs, les fonctionnalités demandées et les contributions de manière structurée. Cela améliore la communication et la collaboration au sein de la communauté.

Grâce à des tags ou des bots, il est possible de facilement organiser ses issues et pull requests pour que le travail de relecture des développeurs assignés soit plus facile.

f. Utilisation de GitHub Projects

GitHub Projects offre une vue d'ensemble du statut des tâches et des priorités. Cela aide à planifier et à gérer le développement de manière efficace, à l'aide de dashboards.

g. Utilisation de GitHub Actions

GitHub Actions permet d'automatiser les workflows, les tests et les déploiements. Cela assure une intégration continue et une livraison continue (CI/CD), améliorant ainsi la qualité et la stabilité du code.

On peut par exemple mettre en place de simples checks pour vérifier le formatage du code, ou bien des tests plus poussés pour vérifier sa fonctionnalité.

h. Forums ou applications de chat pour regrouper les développeurs

La communication et la collaboration en temps réel sont essentielles pour le succès d'un projet open source. Il est conseillé de mettre en place des forums et des applications de chat pour permettre aux développeurs de discuter des idées ou de résoudre les problèmes.

IV. Comparaison avec Pixano

Après avoir détaillé les bonnes pratiques du développement collaboratif sur GitHub, nous pouvons les comparer avec celles mises en place autour de l'outil qui faisait l'objet de ma mission en entreprise, à savoir le développement de Pixano. En effet, alors que ma mission initiale était de produire un code robuste et lisible pour Pixano, j'ai aussi eu pendant ces deux années l'opportunité d'observer et de participer à la mise en open source du logiciel. Dans cette section, je vais donc présenter et critiquer l'état actuel du dépôt GitHub de Pixano.

1. Qu'est-ce que Pixano ?

Pixano est un outil d'exploration et d'annotation de données développé par le Laboratoire de Vision et d'Apprentissage pour l'analyse de scène (LVA) du CEA depuis 2019. Il existait une première version de Pixano, mais celle-ci était peu intuitive à utiliser et à configurer. Ainsi, en utilisant l'expérience et les compétences acquises lors du développement de cette version, une nouvelle version a commencé à être développée en 2022.

En savoir +

L'annotation de données est un processus crucial dans le développement d'intelligences artificielles. En étiquetant ou en catégorisant des données brutes avec des informations supplémentaires, il devient possible d'entraîner un modèle d'IA à répondre à une certaine tâche.

Après deux ans de développement, cette nouvelle version, bien plus pratique que l'ancienne, couvre déjà solidement l'annotation d'images, tandis que l'annotation de vidéos est presque prête à être publiée. Il est également possible de parcourir rapidement ses datasets pour en avoir une vue d'ensemble.

Health demo with metadata Dashboard Dataset











image	id	split	image_type	metadata	
	SF9oZWMjWjWPwMmuFJc4qNp	all	peau	haute	→
	CKJEKNSYRnqNDYkV6Z7gV	all	microscope	cellules	→
	DUnBeJjCWyz9BOnfufT3H	all	os	bras	→
	MpreQxti5w2dlk8Q53nsUz	all	peau	melanome	→
	P9CWwvcMVan2KqaQEKoggD	all	microscope	peau	→
	XFDZDX9RPFUXDRmeFjotn	all	irm	cerveau	→
	Y4SHyoVFn2kWS2LM29aW7	all	microscope	spermatozoide	→
	ZuVcYwBfdYPPRPKR4+EG	all	os	fracture	→
	cXerKyZch469foU4K3VvsG	all	os	machoire	→
	jgJUDIiFGwwM2HwNKYuEQ	all	microscope	red_blood_cells	→

fig. 9 : Exploration d'un dataset dans Pixano

Plusieurs outils sont disponibles pour supporter plusieurs types d'annotations : des *boîtes*, des *polygones*, des *masques* et des *keypoints*. De plus, des filtres graphiques comme la luminosité ou le contraste existent pour faciliter le travail sur des images peu visibles pour l'œil humain.

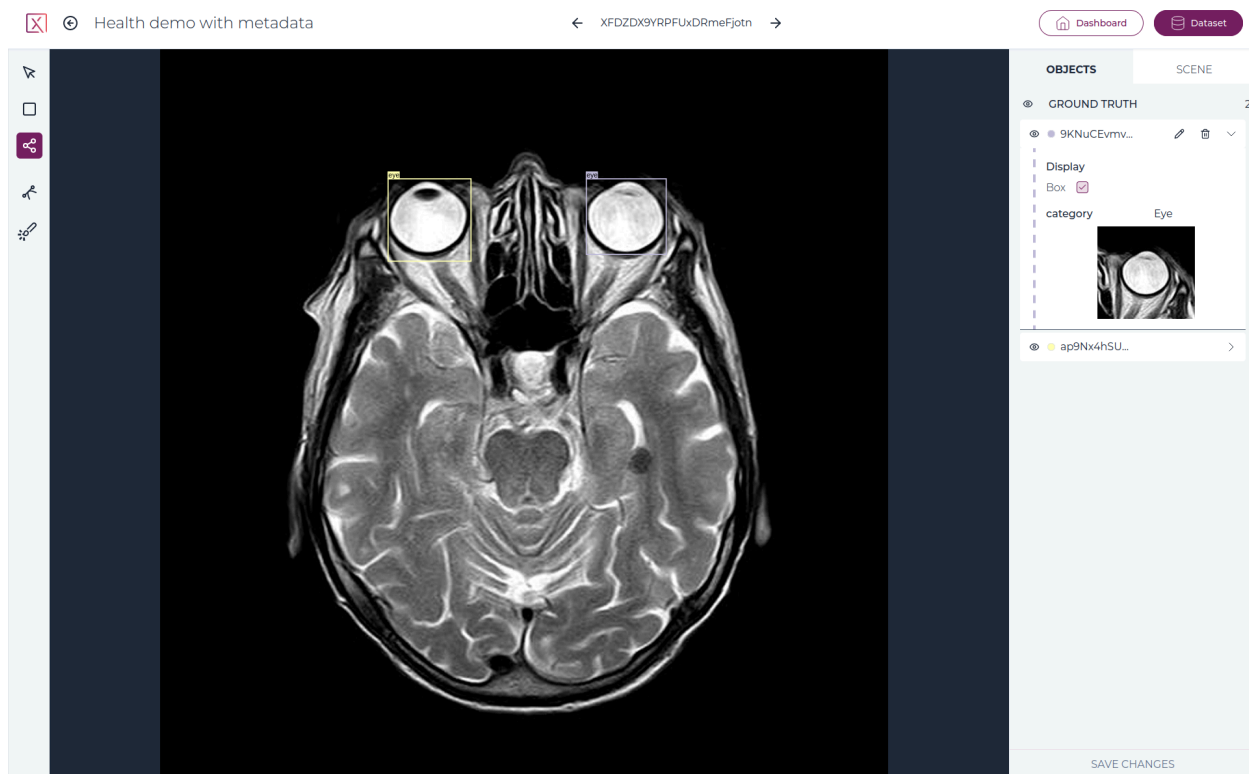


fig. 10 : Annotation d'une image dans Pixano

À l'origine développé, Pixano était développé pour répondre à un besoin : celui de proposer une plateforme d'annotation de données afin de contribuer au développement de modèles d'IA efficaces. Dans sa nouvelle version, les chercheurs du LVA apportent souvent des retours et des demandes de fonctionnalités, mais Pixano ne s'arrête pas là, et cherche à toucher le plus de monde possible. Aujourd'hui, quelques externes utilisent Pixano, et son dépôt GitHub est suivi par plus de 40 personnes.

2. Le dépôt GitHub de Pixano

Observons maintenant le dépôt GitHub de Pixano afin de le comparer aux bonnes pratiques conseillées. Nous allons pour chaque pratique déterminer ce qui est fait par Pixano, mais surtout conseiller des pistes d'amélioration.

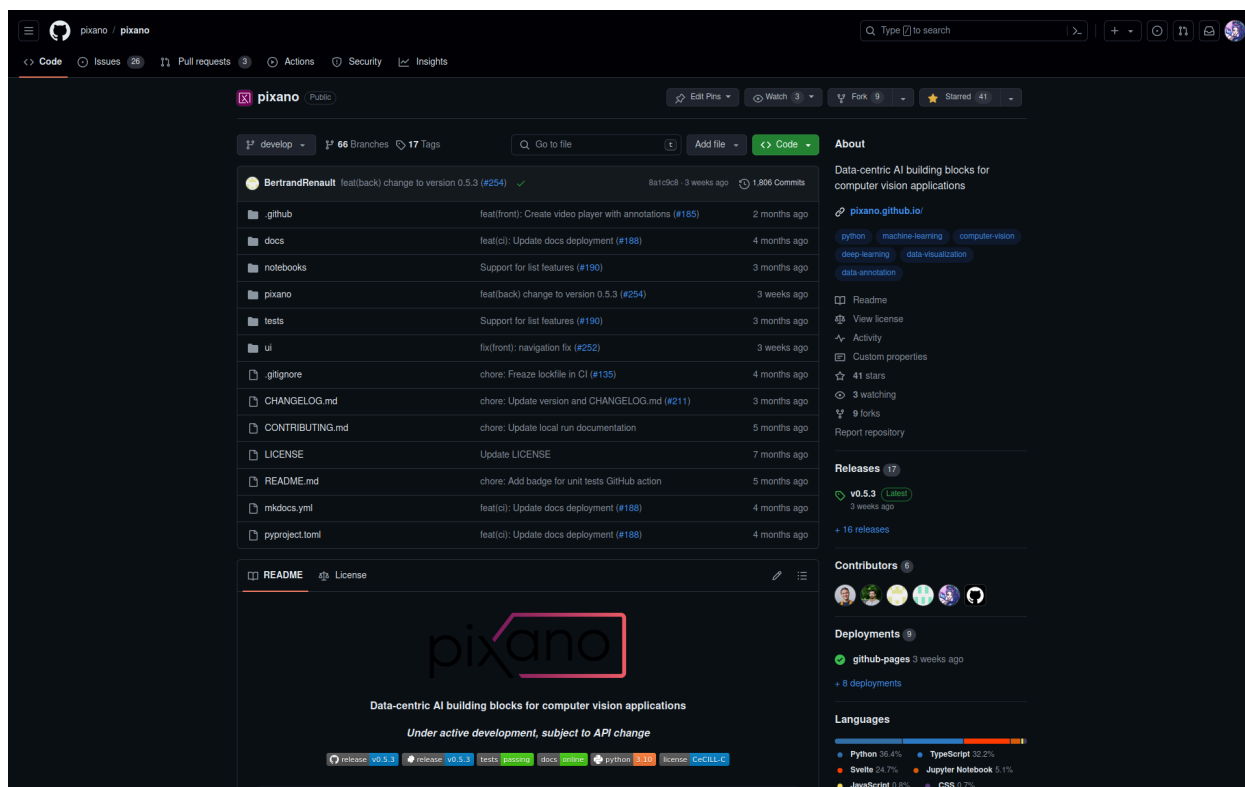


fig. 11 : Dépôt GitHub de Pixano

✓ Fichier README.md

Dans le fichier README.md de Pixano, on retrouve une description de l'outil, des instructions basiques pour l'utiliser, ainsi que des liens vers les guidelines de contributions et la licence open source du projet.

En revanche, on regrette l'absence d'un lien plus explicite vers la documentation que le tag présente actuellement.

😞 Licence et code de conduite

Une licence complète est présente sur le dépôt de Pixano. Par contre, on retrouve également une version courte de cette licence au début de beaucoup de fichiers du code

source, mais pas tous. Est-ce nécessaire de rappeler la licence dans chaque fichier ? Si oui, il faudrait alors s'assurer que tous les fichiers possèdent cette licence pour rester cohérent.

Aucun code de conduite n'est défini dans le dépôt. Il serait préférable d'en concevoir un, afin de s'assurer que tout contributeur ou utilisateur respecte les valeurs défendues par le CEA.

✓ Documentation

La documentation de Pixano est en très bonne voie ! Les instructions et guides sont bien détaillés, et la documentation de l'API est très complète.

Toutefois, il reste quelques pistes d'amélioration. Certains guides redirigent vers des *notebooks Jupyter*, il serait préférable d'ajouter le contenu de ces notebooks directement dans la documentation, pour tout regrouper au même endroit.

De plus, la documentation de l'interface utilisateur de Pixano est regroupée intégralement dans un seul article. Étant plutôt complète, elle mériterait d'avoir une section du site dédiée, afin de pouvoir la séparer en plusieurs articles.

✓ Guidelines de contribution

Les guidelines de contribution de Pixano sont bien définies et très claires. Elles proposent des pré-requis, conventions de codage, instructions pour effectuer des tests, ainsi que des guides pour soumettre des issues ou des pull requests. Il n'y a rien à reprocher à ce sujet, si ce n'est l'absence de code de conduite.

✓ Issues et pull requests

Pixano est encore bon élève à ce niveau ! On retrouve des tags variés et explicites qui permettent de retrouver facilement les issues et pull requests touchant au *front-end*, au *back-end*, à la documentation, etc.

De plus, les développeurs sont réactifs et à l'écoute des problèmes rencontrés par les utilisateurs externes. Chaque pull request doit d'ailleurs être relue par deux contributeurs, et validée par des tests automatiques avant d'être fusionnée.

Enfin, Pixano utilise un bot inclus dans GitHub pour automatiser des opérations basiques sur les issues. Même si dans l'idéal un bot "fait-maison" serait plus efficace, la quantité d'issues ouvertes chaque jour ne justifie pas de déployer autant de ressources pour en développer un, d'autant plus que Pixano est géré par une petite équipe (3 employés à

temps plein), et que d'autres fonctionnalités plus importantes restent encore à être développées.

GitHub Projects

Pixano utilise GitHub Projects de manière efficace, en utilisant plusieurs catégories pour ranger ses issues : Backlog, Todo (next stable release), Priority (next beta release), In Progress et Done. Grâce à cela, les développeurs peuvent avoir un aperçu global de l'état d'avancement de Pixano.

Cependant, le dashboard est privé, les contributeurs externes ne peuvent donc pas y accéder. Il serait intéressant de l'ouvrir au public, afin d'informer plus ouvertement les utilisateurs de la roadmap du projet, mais aussi pour permettre aux potentiels contributeurs externes de savoir ce qu'ils pourraient développer. Attention toutefois à nettoyer le tableau avant de l'ouvrir au public, puisque certains problèmes ne sont plus d'actualité, ou dupliqués.

GitHub Actions

Pixano utilise GitHub actions pour effectuer une série de tests et de checks de formatage lors de chaque pull request. Ainsi, le code reste fonctionnel et lisible à tout moment.

Forums ou chats

Pixano utilise aujourd'hui Talkspirit, outil interne au CEA pour s'organiser entre développeurs. Il est possible mais difficile d'ajouter des contributeurs externes dans les groupes de discussion.

Ce n'est pas un problème s'il existe un groupe privé. Cependant, la création de groupes ou forum ouverts au public, voire l'utilisation de réseaux sociaux, permettrait d'obtenir des retours de personnes extérieures, des propositions de support et de façon plus générale une organisation du projet facilitée.

3. Récapitulatif des conseils pour améliorer le dépôt

Selon les pratiques que nous avons définies, Pixano possède un dépôt d'assez bonne qualité. Sur 8 pratiques, 5 sont validées, soit plus de la moitié. En revanche, il reste des améliorations conséquentes à faire pour les 3 autres pratiques.

Résumons donc les conseils que Pixano devrait suivre pour améliorer son dépôt GitHub :

1. Ajouter un lien plus explicite vers la documentation dans le README
2. Définir un code de conduite en accord avec les valeurs du CEA
3. Ajouter le contenu des notebooks guides directement dans la documentation
4. Déplacer la documentation de l'interface dans une section dédiée
5. Ouvrir au public l'accès au tableau GitHub Projects, après l'avoir nettoyé
6. Créer des groupes de discussion, forums, ou réseaux sociaux pour faciliter la discussion avec les utilisateurs et les contributeurs

En suivant ces conseils, Pixano pourra rendre plus efficace son développement, tout en étant plus ouvert aux retours de sa communauté.

V. Bilan

Dans ce mémoire, plusieurs aspects critiques ont été analysés à travers l'étude de projets open source de grande envergure comme Kubernetes, Visual Studio Code, Rust, mais aussi le projet lié à ma mission, Pixano. L'analyse des bonnes pratiques adoptées par ces projets a permis d'identifier des éléments clés pour la gestion efficace d'un dépôt GitHub. Ces éléments incluent une documentation complète, des guidelines de contribution détaillées, une organisation rigoureuse des issues et des pull requests, ainsi que l'utilisation de fonctionnalités GitHub comme GitHub Projects et GitHub Actions pour automatiser les tâches.

En suivant les conseils identifiés dans ce mémoire, les projets open source peuvent non seulement améliorer leur gestion interne mais aussi accroître leur attractivité et leur accessibilité pour de nouveaux contributeurs. Les pratiques exemplaires observées dans les projets étudiés offrent une feuille de route précieuse pour toute équipe souhaitant optimiser son dépôt GitHub.

Sur un aspect plus personnel, rédiger ce mémoire m'aura été très enrichissant. L'open source est un sujet dont la philosophie me passionne, et étudier ses bonnes pratiques m'a permis de développer mes connaissances à ce sujet. En analysant les différents projets, j'ai bien sûr appris de nouvelles pratiques qui ne m'étaient jamais venues à l'esprit, comme la définition d'un code de conduite, mais j'ai aussi pu poser de façon claire d'autres pratiques que je connaissais déjà, pour mieux les comprendre. Grâce à cela, je pourrai créer d'emblée un dépôt efficace pour mes futurs projets, aussi bien professionnels que personnels.

Ma mission en alternance m'a fait découvrir le plaisir de travailler sur un projet long terme en collaboration. Je suis particulièrement reconnaissant envers mes collègues qui m'ont guidé pendant ces deux années, en m'initiant à la gestion de branches et à la création d'issues et de pull requests.

J'espère que mes lecteurs auront eu, comme moi, le plaisir de découvrir ou de redécouvrir le principe d'open source, et que les bonnes pratiques que j'ai pu dégager leur seront utiles.



Remerciements

Je remercie mon tuteur en entreprise Jaonary Rabarisoa pour l'encadrement et le support qu'il m'a consacré pendant ces deux années d'alternance.

Je remercie mes professeurs référents, Mohamed Belaoued en 1ère année et Stefani El Kalamouni en 2nde année, pour leur accompagnement pendant mes études.

Je remercie mes collègues Angélique Loesch, Bertrand Renault et Virginie Valaire pour l'aide quotidienne qu'ils m'ont apportée pendant mon alternance.

Je remercie mes relecteurs Elina Leung Kam, Ariane Bonneau et Jean-Yves Thuret pour leur aide et leurs conseils dans la rédaction de ce mémoire.

Enfin, je vous remercie aussi, cher lecteur, pour l'attention que vous aurez porté à mon travail.



Annexe

Lexique

Note : la plupart de ces définitions ont été générées par une IA (GPT-4o). Elles ont évidemment été relues, corrigées et validées par moi-même avant d'être ajoutées au lexique.

Annotation de données : *Processus de marquage ou de labellisation des données avec des informations supplémentaires, souvent utilisé dans le contexte de l'apprentissage automatique. Un type d'annotation commun est une boîte encadrant l'objet et lui donnant un nom ou une classe.*

Apache : *Organisation à but non lucratif qui soutient plusieurs projets de logiciels open source, dont Apache Spark, un moteur de traitement de données rapide et généraliste.*

API : *Application Programming Interface : ensemble de règles et de protocoles qui permet à différentes applications ou services de communiquer entre eux.*

Application conteneurisée : *Application qui est emballée avec toutes ses dépendances et configurations nécessaires dans un conteneur.*

Application "low latency" : *Application qui nécessite un temps de réponse extrêmement court entre l'entrée et la sortie.*

Back-end : *Partie de l'application ou du site web qui fonctionne en coulisses et gère la logique métier, les bases de données, les serveurs et les API. Le back-end est responsable du traitement des requêtes, de la gestion des données, de la sécurité et de la performance de l'application.*

Backlog : *Liste priorisée de tâches, de fonctionnalités ou de travaux à effectuer dans un projet.*

Big Data : *Terme désignant de vastes ensembles de données complexes et volumineux, souvent provenant de sources diverses, qui sont difficiles à traiter et à analyser avec les outils et techniques traditionnels.*

BitKeeper : *Système de gestion de version distribué, utilisé principalement dans les années 2000.*

Boîte (annotation) : *En annotation de données, une boîte fait référence à une "boîte englobante" (bounding box) utilisée pour délimiter et identifier des objets spécifiques dans des images.*

Bot : *Programme automatisé conçu pour effectuer des tâches spécifiques de manière autonome.*

Branche (Git) : *Environnement de développement séparé au sein d'un projet de code. Les branches permettent de travailler sur de nouvelles fonctionnalités, des corrections de bugs ou des modifications sans affecter le reste du projet.*

C : *Langage de programmation de bas niveau, développé dans les années 1970 par Dennis Ritchie au Bell Labs. C est connu pour sa performance, sa flexibilité et son efficacité.*

C++ : Langage de programmation de haut niveau, développé par Bjarne Stroustrup dans les années 1980 comme une extension du langage C. C++ ajoute des fonctionnalités de programmation orientée objet, ce qui permet aux développeurs de créer des applications modulaires et réutilisables.

CEA : Commissariat à l'énergie atomique et aux énergies alternatives. Organisme public français de recherche scientifique, technique et industrielle dans les domaines de l'énergie, de la défense, des technologies de l'information, de la santé et de l'environnement.

CI/CD : Continuous Integration/Continuous Deployment. Ensemble de pratiques de développement logiciel visant à automatiser et à améliorer le processus de livraison des applications.

Closed source : Type de logiciel dont le code source est propriétaire et non accessible au public. Les utilisateurs ne peuvent ni voir, ni modifier, ni distribuer le code source.

Code source : Ensemble des instructions et déclarations écrites dans un langage de programmation, que les développeurs écrivent pour créer des logiciels et des applications.

Commit (Git) : Enregistrements qui capturent les modifications apportées au code source à un moment donné. Chaque commit inclut un message descriptif, un identifiant unique et les changements de code effectués.

Conteneur : Environnement d'exécution isolé qui contient tout ce dont une application a besoin pour fonctionner, y compris le code, les bibliothèques, les dépendances et les fichiers de configuration.

Dashboard : Tableau de bord permettant la visualisation d'informations importantes regroupées en un seul endroit.

Dataset : Ensemble structuré de données, souvent utilisé pour l'entraînement et l'évaluation des modèles d'apprentissage automatique. Un dataset peut contenir des données de différents types, comme des images, du texte, des enregistrements audio ou des données tabulaires.

Décentralisation (système / architecture) : Modèle d'architecture où les composants du système sont répartis sur plusieurs nœuds ou sites, sans un point de contrôle central. Chaque nœud peut fonctionner indépendamment et collaborer avec les autres pour atteindre les objectifs du système.

Dépôt (GitHub) : Répertoire où est stocké un projet de code source sur GitHub. Il contient tous les fichiers et l'historique des versions du projet, permettant la collaboration et la gestion des modifications.

DIASI : Département Intelligence Ambiante et Systèmes Interactifs

Distribution (système / architecture) : Architecture dans laquelle les composants d'un système logiciel sont distribués sur plusieurs ordinateurs interconnectés par un réseau. Chaque composant effectue une partie spécifique des tâches globales, permettant une meilleure répartition des charges, une amélioration de la performance et une plus grande robustesse du système.



DRT : *Direction de la Recherche Technologique. L'une des principales directions du CEA.*

Efrei : *École d'ingénieurs généraliste en informatique et technologies du numérique située à Villejuif, offrant des formations allant du niveau Bachelor au niveau Master.*

EPIC : *Établissement public à caractère industriel et commercial. Type d'établissement public en France ayant une mission de service public, mais fonctionnant dans un cadre commercial.*

Extension : *Module ou complément qui ajoute des fonctionnalités supplémentaires à un logiciel ou une application. Les extensions peuvent être installées pour personnaliser et étendre les capacités de programmes tels que les navigateurs web, les éditeurs de texte, et les suites bureautiques.*

Front-end : *Partie visible et interactive d'une application ou d'un site web avec laquelle les utilisateurs interagissent directement. Cela inclut tout ce qui est affiché à l'écran, comme les interfaces utilisateur, les mises en page, les graphiques et les formulaires.*

Garbage collector : *Composant d'un système de gestion de la mémoire dans certains langages de programmation (comme Java et C#) qui automatise la récupération de la mémoire allouée mais non utilisée par les programmes. Il aide à prévenir les fuites de mémoire en libérant l'espace occupé par les objets qui ne sont plus accessibles.*

Git : *Système de contrôle de version décentralisé utilisé pour suivre les modifications apportées aux fichiers dans le développement de logiciels. Il permet aux développeurs de collaborer et de gérer les versions de leur code source.*

GitHub : *Plateforme de développement collaboratif et de gestion de versions basée sur Git. Elle permet aux développeurs de stocker, partager et gérer leurs projets de code source.*

GitHub Actions : *Service d'intégration et de livraison continues (CI/CD) fourni par GitHub, permettant d'automatiser les flux de travail de développement.*

GitHub Projects : *Fonctionnalité de GitHub qui permet aux utilisateurs de planifier et de suivre le travail sur leurs dépôts à l'aide de tableaux de projet.*

Google : *Multinationale américaine spécialisée dans les services et produits liés à Internet, notamment les technologies de publicité en ligne, le moteur de recherche Google, le cloud computing, les logiciels et le matériel.*

Guidelines : *Ensemble de recommandations ou de directives conçues pour orienter les actions et les décisions dans un contexte spécifique.*

IA (Intelligence Artificielle) : *Domaine de l'informatique qui se concentre sur la création de systèmes capables d'effectuer des tâches qui nécessitent normalement l'intelligence humaine. Ces tâches incluent la reconnaissance vocale, la compréhension du langage naturel, la prise de décision, la résolution de problèmes, ou la reconnaissance d'images.*



Jupyter : Environnement open source pour la création et le partage de notebooks. Le projet Jupyter supporte plusieurs langages de programmation dont Python, R, et Julia.

Keypoints (annotation) : Points d'intérêt spécifiques dans une image utilisés pour l'annotation de données. Les keypoints sont souvent utilisés pour capturer des caractéristiques particulières d'un objet, comme les articulations d'un corps humain (coudes, genoux) dans la détection de poses, ou des points de repère faciaux (yeux, nez, bouche) dans la reconnaissance faciale.

Kubernetes : Système open source de gestion de conteneurs, initialement développé par Google, qui permet l'automatisation du déploiement, de la mise à l'échelle et de la gestion des applications conteneurisées.

Langage de programmation : Ensemble de règles et de syntaxe utilisé pour écrire des programmes informatiques. Ces langages permettent aux développeurs de communiquer avec les ordinateurs et de créer des logiciels, des applications, et des systèmes.

Linus Torvalds : Ingénieur logiciel finlandais principalement connu pour avoir initié le développement du noyau Linux en 1991. Il est également le créateur de Git, un système de contrôle de version distribué. Linus Torvalds est une figure emblématique de la communauté open source et continue de superviser le développement du noyau Linux.

Linux : Famille de systèmes d'exploitation open source basés sur le noyau Linux, développé par Linus Torvalds et publié pour la première fois en 1991. Linux est utilisé dans une grande variété d'applications, des serveurs et superordinateurs aux dispositifs embarqués et aux smartphones (via Android).

LIST : Laboratoire pour l'Intégration des Systèmes et des Technologies

Logiciel : Ensemble des programmes, procédures et règles, ainsi que de la documentation associée, relatifs au fonctionnement d'un système informatique. Les logiciels peuvent être classés en plusieurs catégories, dont les logiciels d'application et les logiciels système.


Logiciel propriétaire : Logiciel qui est la propriété de son éditeur et dont l'utilisation, la modification et la distribution sont réglementées par une licence restrictive. Le code source de ces logiciels n'est pas disponible pour le public.

LVA : Laboratoire de Vision et d'Apprentissage pour l'analyse de scène.

M2-APP-BDML : Master 2 en Apprentissage, filière Big Data & Machine Learning.

Machine learning : Sous-domaine de l'intelligence artificielle qui se concentre sur le développement d'algorithmes et de modèles permettant aux ordinateurs d'apprendre à partir de données et de faire des prédictions ou des décisions sans être explicitement programmés pour chaque tâche.

Markdown : Langage de balisage léger conçu pour formater du texte de manière simple et lisible en utilisant une syntaxe facile à écrire et à lire.



Marketplace : Dans le cas de VS Code, plateforme en ligne où les utilisateurs peuvent découvrir, installer et gérer des extensions pour leur éditeur de code.

Masque (annotation) : En annotation de données, un masque est une représentation binaire ou multiclasse de l'image où chaque pixel est marqué pour indiquer à quelle classe ou quel objet il appartient.

Meta : Meta (anciennement Facebook) est une multinationale américaine spécialisée dans les technologies et les services en ligne. L'entreprise est principalement connue pour ses plateformes de réseaux sociaux Facebook, Instagram, et WhatsApp, ainsi que pour son développement dans le domaine de la réalité virtuelle avec Oculus et le Metaverse.

Microsoft : Multinationale américaine de technologie informatique, connue pour son système d'exploitation Windows, la suite bureautique Office, et des produits comme Azure, Surface, et Xbox.

Modèle (machine learning) : Algorithme ou système mathématique entraîné sur des données pour effectuer des prédictions ou prendre des décisions sans être explicitement programmé pour chaque tâche spécifique.

Monte (de vache) : Une vache qui monte sur une autre. Phénomène observé notamment chez les vaches d'élevage lorsqu'elles sont en chaleur.

Mozilla Research : Division de la Fondation Mozilla dédiée à la recherche et à l'innovation dans les technologies web et open source.

Nano-Innov : Centre de recherche et d'innovation situé en France, spécialisé dans les nanotechnologies et les nouvelles technologies, regroupant diverses institutions et entreprises pour collaborer sur des projets de R&D.

Node.js : Environnement d'exécution JavaScript côté serveur basé sur le moteur V8 de Google Chrome. Il permet de créer des applications web évolutives et performantes.

Notebook : Document interactif qui combine du texte, des visualisations, et du code exécutable.

Noyau de système d'exploitation : Partie centrale et essentielle d'un système d'exploitation, responsable de la gestion des ressources matérielles de l'ordinateur et de la communication entre le matériel et les logiciels.

Open source : Type de logiciel dont le code source est librement accessible, modifiable et redistribuable par n'importe qui. Il favorise la collaboration et le partage au sein de la communauté des développeurs.

Open Source Initiative : Organisation à but non lucratif dédiée à la promotion et à la protection des logiciels open source. Elle certifie les licences open source et promeut les bonnes pratiques en matière de développement open source.



Pixano : Outil open source d'annotation de données, développé par le laboratoire LVA du CEA.

Polygone (annotation) : En annotation de données, un polygone est utilisé pour tracer des formes plus précises autour des objets dans des images, au lieu de simples boîtes rectangulaires.

README.md : Fichier texte rédigé en Markdown et généralement inclus dans le répertoire racine d'un projet de code source. Il fournit des informations essentielles sur le projet, telles que son objectif, comment l'installer et l'utiliser, les dépendances nécessaires, et des instructions de contribution.

Référentiel central : Emplacement unique et centralisé où les données, les configurations ou les documents d'un projet sont stockés et gérés.

Richard Stallman : Programmeur et militant américain, fondateur du mouvement du logiciel libre, initiateur du projet GNU et de la Free Software Foundation. Il est un fervent défenseur des libertés des utilisateurs de logiciels.

Roadmap : Plan stratégique qui décrit les étapes, les objectifs et les jalons importants à atteindre sur une période donnée pour un projet ou un produit.

Rust : Langage de programmation open source, conçu pour être sûr, concurrent et performant. Il est particulièrement apprécié pour ses garanties de sécurité de la mémoire et ses performances comparables à celles du C++.

Sécurité mémoire : Ensemble de caractéristiques et de garanties offertes par des langages de programmation pour prévenir les erreurs de gestion de la mémoire.

SIALV : Service Intelligence Artificielle Langage et Vision.


Slack : Plateforme de communication et de collaboration d'équipe, offrant des fonctionnalités de messagerie instantanée, de partage de fichiers et de gestion de projets.

Snapshot : Copie de l'état des fichiers ou du système à un moment précis.

Spark : Apache Spark, moteur de traitement de données open source conçu pour la vitesse et l'efficacité, utilisé pour le traitement et l'analyse de grandes quantités de données.

Système d'exploitation : Ensemble de programmes qui gère les ressources matérielles et logicielles d'un ordinateur. Il fournit des services aux applications et assure la gestion des tâches fondamentales telles que la gestion des fichiers, la mémoire, les processus et les périphériques.

Système de contrôle : Ensemble de logiciels et de pratiques utilisés pour suivre, gérer et enregistrer les modifications apportées au code source et à d'autres fichiers d'un projet au fil du temps. Ces systèmes permettent aux développeurs de collaborer plus efficacement, de maintenir un historique complet des modifications, et de revenir à des versions antérieures si nécessaire.



Système embarqué : *Système informatique spécialisé qui fait partie d'un dispositif plus large et est conçu pour réaliser des tâches spécifiques. Les systèmes embarqués sont souvent utilisés dans des appareils comme les voitures, les électroménagers, les équipements médicaux et les appareils industriels.*

TensorFlow : *Bibliothèque open source de machine learning développée par Google, utilisée pour diverses tâches d'apprentissage automatique et d'intelligence artificielle, y compris les réseaux neuronaux profonds.*

Terminal de commande : *Interface utilisateur qui permet aux utilisateurs d'interagir avec le système d'exploitation via des lignes de commande textuelles.*

Tracking : *Processus de suivi d'objets en mouvement dans une vidéo. Le tracking d'objets implique la détection et le suivi continu d'objets spécifiques à travers une séquence d'images vidéo pour analyser leurs déplacements et comportements.*

Trello : *Outil de gestion de projet basé sur des tableaux, où les utilisateurs peuvent créer des cartes pour représenter des tâches et les organiser en listes sur des tableaux.*

UI (User Interface) : *Interface utilisateur d'une application ou d'un site web, qui comprend tous les éléments graphiques et interactifs avec lesquels un utilisateur peut interagir.*

UX (User Experience) : *Expérience utilisateur globale lorsqu'une personne utilise une application ou un site web. L'UX englobe tous les aspects de l'interaction de l'utilisateur avec le produit, y compris la facilité d'utilisation, l'efficacité, la satisfaction et le plaisir.*

VS Code : *Visual Studio Code, l'éditeur de code open source développé par Microsoft, connu pour sa légèreté, ses fonctionnalités puissantes et ses nombreuses extensions.*

Références

100 millions of developers and counting :

<https://github.blog/2023-01-25-100-million-developers-and-counting/>

5 raisons pour lesquelles l'avenir de l'Open Source est... open ! :

<https://www.lesechos.fr/idees-debats/cercle/5-raisons-pour-lesquelles-lavenir-de-lopen-source-est-open-132979>

Apache Spark (dépôt GitHub) :

<https://github.com/apache/spark>

CEA :

<https://www cea.fr/>

Comprendre Git en 7 minutes :

<https://www.jesuisundev.com/comprendre-git-en-7-minutes/>

Efrei :

<https://www.efrei.fr/>

Efrei (Wikipédia - FR) :

https://fr.wikipedia.org/wiki/EFREI_Paris

EPIC (Wikipédia - FR) :

https://fr.wikipedia.org/wiki/Établissement_public_à_caractère_industriel_et_commercial_en_France

Git :

<https://git-scm.com/>

Git (dépôt GitHub) :

<https://github.com/git>

Git (Wikipédia - FR) :

<https://fr.wikipedia.org/wiki/Git>

GPT-4o (pour la rédaction du lexique) :

<https://chatgpt.com>

Kubernetes (communauté) :

<https://www.kubernetes.dev/>

Kubernetes (dépôt GitHub) :

<https://github.com/kubernetes/kubernetes>



Kubernetes (documentation) :

<https://kubernetes.io/docs/home/>

Logiciel libre et logiciel open source : quelles différences ? :

<https://interhop.org/2021/01/18/opensource-libre-difference>

Linus Torvalds (Wikipédia - FR) :

https://fr.wikipedia.org/wiki/Linus_Torvalds

Node.js (dépôt GitHub) :

<https://github.com/nodejs/node>

Open Source (Wikipedia - EN) :

https://en.wikipedia.org/wiki/Open_source

Open Source (Wikipédia - FR) :

https://fr.wikipedia.org/wiki/Open_source

Open Source Initiative :

<https://opensource.org/>

Pixano (documentation) :

<https://pixano.github.io/>

Pixano (dépôt GitHub) : <https://github.com/pixano/pixano>

Richard Stallman (Wikipédia - FR) :

https://fr.wikipedia.org/wiki/Richard_Stallman

Rust :

<https://www.rust-lang.org/>

Rust (dépôt GitHub) :

<https://github.com/rust-lang/rust>

Rust (documentation) :

<https://doc.rust-lang.org/book/index.html>

Tensorflow (dépôt GitHub) :

<https://github.com/tensorflow/tensorflow>

The difference between free and open source software :

<https://www.digitalocean.com/community/conceptual-articles/free-vs-open-source-software>



The Open Source Definition :

<https://opensource.org/osd>

The Pros and Cons of Open Source Software Development :

<https://www.freecodecamp.org/news/what-is-great-about-developing-open-source-and-what-is-not>

The state of open source and rise of AI in 2023 :

<https://github.blog/2023-11-08-the-state-of-open-source-and-ai/>

VS Code :

<https://code.visualstudio.com/>

VS Code (dépôt GitHub) :

<https://github.com/Microsoft/vscode>

What is open source ? :

<https://opensource.com/resources/what-open-source>