Engineer

Base de code

le code source maintenu, revu et livré comme un ensemble

Ce que c’est

Une base de code, codebase en anglais, parfois écrit en deux mots, code base, est l’ensemble du code source qu’une équipe modifie, revoit et livre comme un tout. Quand les ingénieurs parlent de « la base de code », ils désignent l’endroit où aboutit chaque changement d’une application : le code lui-même, ses tests, les scripts qui le construisent et le vérifient, les modifications de base de données et les documents qui expliquent comment les pièces s’assemblent.

La méthode Twelve-Factor App, destinée à construire des services web, donne la définition de travail la plus utile, car elle rend la frontière vérifiable. Dans un système de gestion de versions centralisé comme Subversion, une base de code est un dépôt. Dans un système distribué comme Git, c’est tout ensemble de dépôts partageant un commit racine, le premier de l’historique, celui qui n’a pas de parent. Deux dépôts issus du même premier commit constituent une seule base de code, quelle que soit l’ampleur de leur divergence. Deux dépôts commencés séparément sont deux bases de code, même s’ils contiennent des fichiers presque identiques.

Le même document associe cette définition à une seconde règle : une base de code, de nombreux déploiements. Chaque déploiement est une instance de l’application en cours d’exécution, et plusieurs coexistent généralement. La production en est une, la préproduction une autre, et la copie locale de chaque développeur compte aussi. Toutes exécutent la même base de code, souvent à des commits différents : un développeur a du travail que la préproduction n’a pas encore reçu, et celle-ci a du travail que la production n’a pas encore reçu.

Le contenu découle de cette définition. Le code source et tout ce qui permet d’en faire un programme opérationnel ont leur place dans la base, y compris un manifeste nommant chaque dépendance et sa version. Les dépendances elles-

mêmes restent généralement à l'extérieur, récupérées par un gestionnaire de paquets. Les valeurs qui changent d'un déploiement à l'autre restent aussi dehors, et surtout les identifiants : un mot de passe enregistré une fois dans un commit accompagne ensuite chaque copie de l'historique.

Une base de code peut contenir plusieurs programmes. Un monorepo garde un frontend, une API et le code de base de données côte à côte dans un dépôt, pour qu'un seul commit puisse modifier les trois. Selon la règle du commit racine, cela reste une base de code. Le choix opposé répartit une application entre des dépôts aux historiques séparés, et chaque dépôt est alors sa propre base de code. Twelve-Factor est catégorique sur le résultat : plusieurs bases de code signifient un système distribué, dont chaque composant est une application à part entière.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'une base de code ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Définir d'abord le système sur le papier pour que ses éléments s'articulent et que son coût soit connu avant d'écrire du code.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Un plan d'infrastructure pour 14 départements — 14 départements de recherche sur un même plan d'infrastructure

Un plan d'infrastructure pour 14 départements Contactez-nous →

Pourquoi c'est important

La frontière décide de ce qui peut changer ensemble. Dans une base de code, une modification d'API et de l'écran qui l'appelle peut s'écrire, se revoir et se fusionner comme une unité. Entre deux bases de code, la même modification devient deux pull requests, deux publications et une période où un côté est en avance sur l'autre. Les deux approches fonctionnent ; leur prix dépend de l'endroit où la limite a été tracée.

La base de code est aussi l'unité d'application des règles. Une suite de tests, un linter, un outil de formatage, une règle de message de commit et un exécuteur de tâches prennent tous « la base de code » pour périmètre, et rien de ce qui reste à l'extérieur n'est contrôlé par eux.

Les décisions précoces se figent dans une base de code plus vite qu'ailleurs. Chaque ligne écrite après un choix le tient pour acquis : une règle de séparation en couches ou un niveau de qualité fixé dans les premières semaines donne donc

sa forme à tout ce qui suit. Revenir dessus plus tard exige de réécrire le code construit autour de ce choix.

Pour les personnes, la base de code est ce que reçoit un nouveau membre de l'équipe. Si elle contient tout ce qu'il faut pour construire, tester et exécuter l'application, un clonage et une commande suffisent à démarrer. Si le savoir se partage entre le code et la mémoire de quelqu'un, la première semaine sert à découvrir ce qui se trouve où.

À quoi cela ressemble en pratique

Go, TypeScript, SQL, Python et shell partagent le dépôt d'une plateforme, où la base de code est un monorepo polyglotte et chaque langage arrive avec sa propre façon de construire, tester et exécuter le lint. La réponse a été un point d'entrée unique : une couche go-task passée à 578 cibles nommées, dont 50 dans le fichier racine et 528 dans des fichiers organisés en espaces de noms en dessous. Elle couvre les modes de développement natif, Docker et HTTPS, la qualité du code, la gestion de base de données et le déploiement. Chacun peut lancer `task --list` et voir toute la chaîne d'outils. Le nombre est aussi un avertissement : 578 cibles dépassent ce que quelqu'un peut garder en tête, et ce sont donc le nommage et les espaces de noms qui gardent l'ensemble utilisable.

Un autre projet a commencé par un dépôt vide le 25 novembre 2025, et trois décisions de ses premières semaines tenaient encore neuf mois et plusieurs milliers de commits plus tard : une pile en couches, l'accès aux données par des fonctions stockées et une exigence de zéro avertissement instaurée avant qu'il y ait beaucoup de code auquel l'appliquer. Le raisonnement derrière cette dernière décision est au cœur de cet article : une règle introduite au commit 5 000 est un projet de nettoyage, alors que la même règle au commit 50 est simplement la façon dont le dépôt fonctionne.

Les frontières internes d'une base de code peuvent constituer la conception elle-même. La contribution de la fondatrice à l'architecture dans le cadre de NextMariner concerne une plateforme maritime construite en trois couches : le frontend Next.js ne s'occupe que de la présentation ; une API Go au rôle délibérément restreint, construite avec Huma sur Fiber, assure le routage, la validation et l'orchestration ; et la logique métier réside dans les fonctions PostgreSQL. Quand une règle change, elle change dans une seule couche, en SQL. Les frontières ont été documentées et imposées en revue, car une convention que personne ne fait respecter cesse d'en être une.

La base de code de ce site trace une autre frontière. Ses pages de projets sont écrites dans un dépôt séparé, exportées dans celui-ci par un script et enregistrées dans des commits, pour que le site se construise de façon autonome. Dans cette base de code, ces pages sont des fichiers générés, et les consignes de travail précisent qu'elles ne sont jamais modifiées à la main.

Où les problèmes surviennent

Le code copié diverge

Deux applications ont besoin de la même fonction auxiliaire, et le plus rapide est de la copier de l'une à l'autre. Un an plus tard, deux versions ont des corrections différentes, et un bug corrigé dans l'une reste présent dans l'autre. Twelve-Factor considère que plusieurs applications partageant le même code enfreignent sa règle et donne la solution : déplacer la partie commune dans une bibliothèque, l'importer par le gestionnaire de dépendances et laisser chaque base de code déclarer la version qu'elle utilise.

Un secret entre dans l'historique

Un commit enregistre un fichier de configuration contenant une chaîne de connexion ou une clé d'API. Cette information reste dès lors dans l'historique que chaque clone emporte, même après la suppression de la ligne dans un commit ultérieur. Le test Twelve-Factor mérite d'être appliqué avant chaque commit : pourrait-on publier cette base de code en open source aujourd'hui sans exposer un seul identifiant ? Tout ce qui ferait échouer ce test appartient à l'environnement de chaque déploiement.

Un fichier généré est modifié à la main

Un fichier écrit par un script ressemble à tout autre fichier source, et quelqu'un y corrige directement une faute. À l'exécution suivante, le script remet l'ancien texte et la correction disparaît sans un mot. Signalez clairement les chemins générés, gardez le générateur comme seul outil autorisé à les écrire et envoyez chaque correction vers ses données d'entrée.

Les frontières internes s'effacent

Sous la pression d'une échéance, une requête de données est placée directement dans le gestionnaire d'une requête entrante parce que c'est le chemin le plus court, et les couches prévues au départ ne décrivent plus le code. Chaque raccourci est petit. Ensemble, ils transforment un changement de deux

lignes en un changement qui touche toutes les couches. Une règle écrite aide, la revue aide davantage et un contrôle qui fait échouer la compilation quand une couche accède à l'intérieur d'une autre aide le plus.

Une règle ne couvre que la moitié du code

Un linter installé pour le frontend laisse de côté les scripts shell et le SQL, et ces fichiers accumulent précisément les problèmes qu'il devait empêcher. Une exigence applicable à toute la base de code doit nommer chacun de ses langages : c'est le coût caché d'un dépôt polyglotte.

Un fork lance un second produit

Une équipe copie un dépôt pour lancer un produit apparenté. Les deux copies gardent le même commit racine : selon Twelve-Factor, elles sont donc toujours une seule base de code, désormais au service de deux applications, et la divergence précédente commence à l'échelle d'un dépôt entier. Comparer les commits racine montre dans quelle situation on se trouve :

```
git -C app-a rev-list --max-parents=0 HEAD
git -C app-b rev-list --max-parents=0 HEAD
```

Une sortie identique indique une racine et une base de code communes ; des sorties différentes indiquent deux historiques séparés.

Pour aller plus loin

- I. Base de code — la règle d'une base de code pour de nombreux déploiements et le test du commit racine partagé qui la sous-tend.
- II. Dépendances — déclarer et isoler chaque dépendance dans un manifeste, pour que la base de code indique exactement ce qu'elle importe.
- III. Configuration — le test de publication en open source pour déterminer ce qui doit rester hors d'une base de code.
- gitsubmodules — comment Git intègre un dépôt dans un autre en laissant chacun garder son historique.
- git-rev-list — la référence pour lister les commits, y compris les commits racine d'un historique.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'une base de code ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/codebase/