Eco Conception

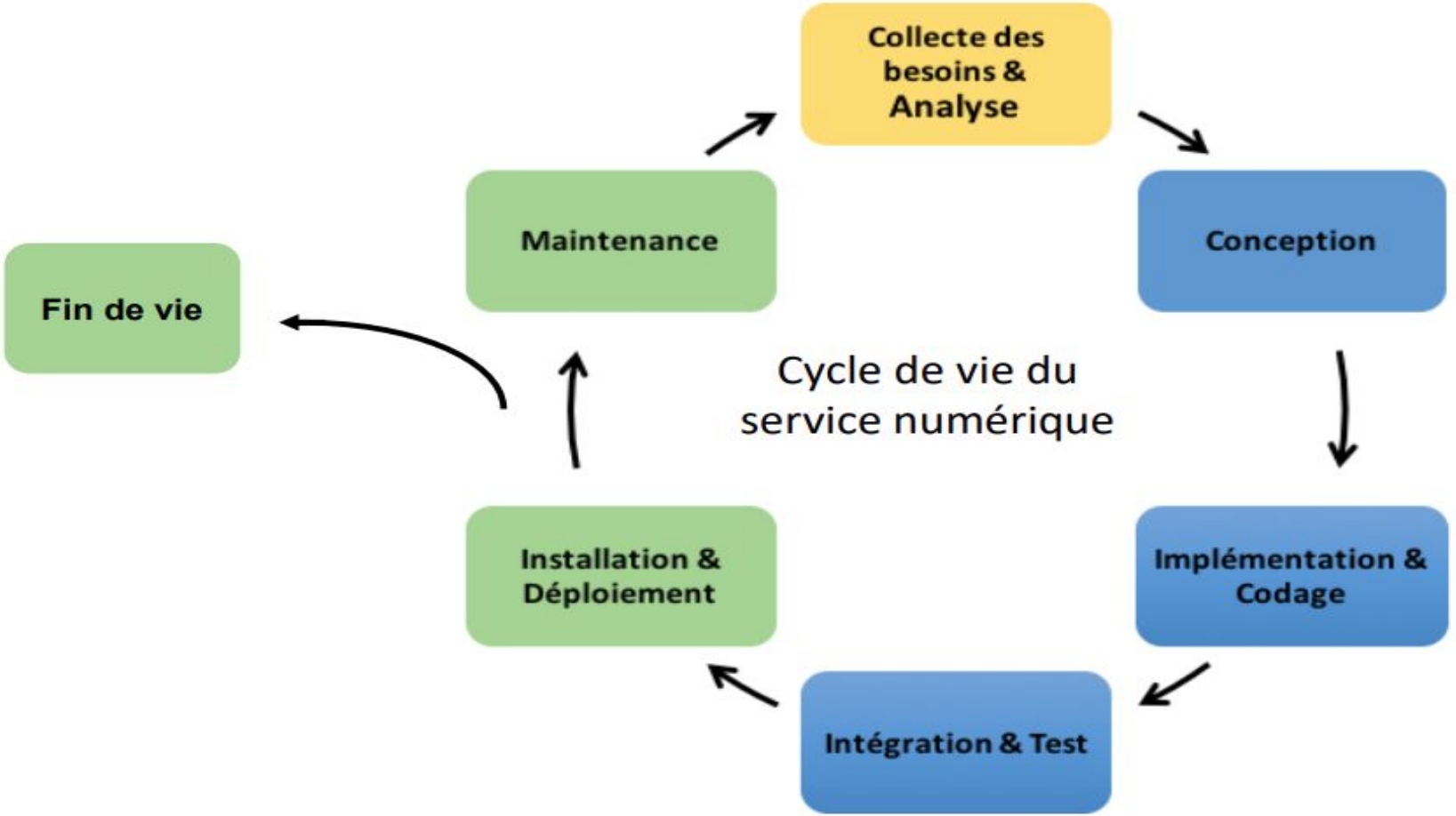
Team Sud

Daumas Caroline
Duffau Jean-Charles
Haage Hugo
John-Rose Delmerie
Laclau Tristan
Lozzi Christophe

FMS
FACYLITIES MULTI SERVICES

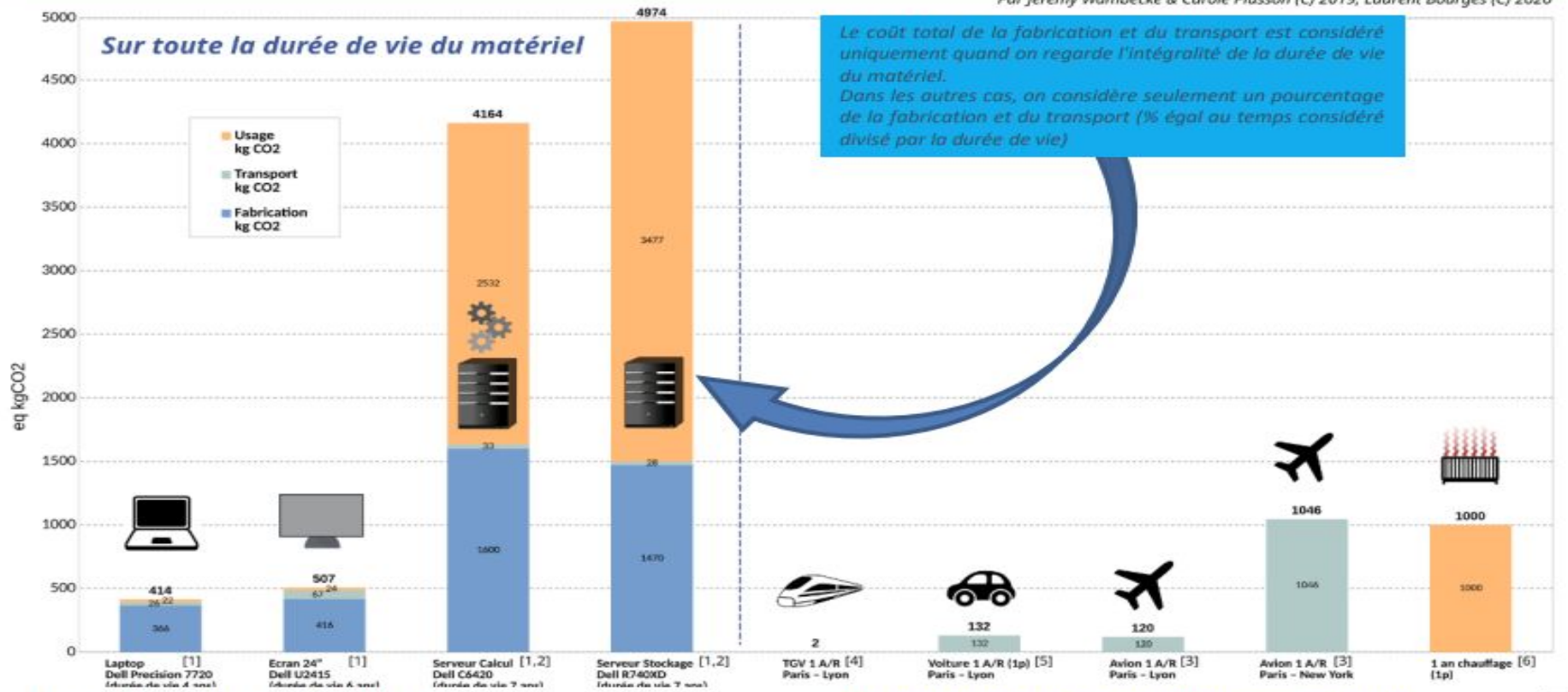
Sommaire

- Qu’est ce que l’éco-conception ?
- Spécification
- Conception
- Réalisation : Implémentation et codage
- Réalisation : Intégration et tests
- Production : Installation et déploiement
- Maintenance


Collecte des besoins & Analyse
Conception
Implémentation & Codage
Intégration & Test
Installation & Déploiement
Maintenance
Fin de vie
Cycle de vie du service numérique

Comparatif d'émissions CO2

Par Jérémy Wambecke & Carole Plasson (C) 2019, Laurent Bourgès (C) 2020



[1] Données Fiches Dell (usage corrigé pour usage FR) : (https://www.dell.com/learn/us/en/uscorp1/corp-comm/environment_carbon_footprint_products)
[2] Usage à partir de la consommation moyenne (Berthoud et al. 2020) d'un nœud = 275W (C6420), 375W (R740XD) (https://hal.archives-ouvertes.fr/hal-02549565)
[3] https://eco-calculateur.dta.aviation-civile.gouv.fr/
[4] https://ressources.data.sncf.com/explore/dataset/emission-c02-tgv/table/
[5] Trajet de 473km, pour une voiture émettant 0,140 kg CO_2/km
[6] https://www.insee.fr/fr/statistiques/fichier/1281320/ip1445.pdf
Facteur d'impact : 0,108 kgCO_2/kWh (FR)

Qu’est ce que l’éco-conception ?

L'éco-conception d'un service numérique consiste à intégrer des contraintes environnementales dans tous les processus de développement, afin de réduire les impacts environnementaux du service numérique pendant son cycle de vie :

- avant le développement : définition des besoins et analyse
- pendant le développement : conception, codage, tests, mise en production
- après le développement : exploitation de l’application, maintenance, fin de vie

De façon générale, plus la prise en compte des aspects environnementaux intervient tôt dans le cycle de vie, plus l'effet est important.

Spécifications

Intégrer les contraintes environnementales dans les spécifications implique une remise en question des besoins.

- En réduisant la couverture et la profondeur fonctionnelle de l'application, on abaisse les impacts environnementaux associés. On diminue ainsi mécaniquement l'infrastructure nécessaire à son exécution.
- Par ailleurs, à niveau ergonomique constant, plus l'application est pauvre fonctionnellement, plus elle sera simple à utiliser. Il faut donc réduire le plus possible la couverture fonctionnelle de l'application, en la centrant sur le besoin essentiel de l'utilisateur.
- Détecter une fonctionnalité non essentielle est possible au moment de l'analyse de l'expression du besoin. La méthode MoSCoW, des ateliers, des wireframes (maquettes fonctionnelles) ou des prototypes avec tests utilisateurs permettent de vérifier l'utilité d'une fonctionnalité en amont de son développement.

Spécifications

- Les dimensions de chaque fonctionnalité doivent être définies précisément et dans leur ensemble. Il peut s'agir d'un taux de compression pour les images de l'interface graphique, du temps de réponse maximum pour une requête HTTP/SQL, du nombre d'items affichés dans une liste, etc.
- Éviter les "obésiciel".En effet, plusieurs études démontrent que 70 % des fonctionnalités demandées par les utilisateurs ne sont pas essentielles et 45 % ne sont jamais utilisées.
- Site statique ou dynamique (dépends de qui maintient le service).
- Utiliser un cache HTTP? (best-practices/BP_099_fr.md at main · cnumr/best-practices (github.com))

Conceptuellement : minimiser le nombre de clics/requêtes pour les fonctions principales.

Conception

Choix du langage

- Prendre le temps de choisir un langage en considérant l'impact énergétique et la pérennité (comparaison temps/mémoire/énergie)
  - Privilégier pour les langages compilés pour les traitements lourds, de haute performance ou de temps réel
  - Privilégier les langages interprétés pour les traitements moins contraints afin de faciliter la maintenance et la durabilité

Conception

Gestion des données

- Anticiper et prendre en compte les données pour diminuer leur impact environnemental
  - Privilégier le traitement côté serveur pour éviter de transférer les jeux de données sur les postes des utilisateurs (se connecter à une base de données que si nécessaire)
  - Tenir compte du volume des données à manipuler pour concevoir les traitements et éviter les transferts volumineux entre les différentes couches
  - Supprimer les dépendances non utilisées

Conception

Limites les appels aux API HTTP

- Fixer des quotas afin d'inciter les utilisateurs à définir une stratégie de mise en cache des réponses et éviter des appels systématiques
- Limiter le nombre d’appels distants en regroupant plusieurs requêtes pour n’en faire qu’une seule (charge réseau, impact d’une latence réseau élevée, coûts dans le cadre de licences à la requête)

Conception

Expérience utilisateur

- Simplifier le parcours permet de fluidifier l'expérience et réduit une empreinte environnementale élevée
- Concevoir en "mobile first" : en commençant par une version desktop pour une application, le contenu devra être adapté pour une version mobile. Rendre tout le contenu en mode responsive design, l'expérience sera détériorée et l'impact environnemental accru par le chargement de contenu inutile. De plus, cette approche sera pensée pour des différents types de terminaux voir avec des connexions réseau non optimale

Réalisation : Implémentation et codage

L'***implémentation*** est le moment où on code, où on met en œuvre les technologies nécessaires. Cette phase coïncide également avec la réalisation des visuels, icônes, images mais également avec la création des contenus, la collecte et la mise en forme des données, qui devront être accessibles via le service.

Réalisation : Implémentation et codage

OPTIMISER VOTRE CODE 👩‍💻

Que vous soyez développeur front ou back, vous avez un rôle à jouer. Le collectif Green IT propose une **liste de recommandations** qui donnent de nombreuses pistes. Comme utiliser la version la plus récente du langage. Ex :

- Fournir une alternative textuelle aux contenus multimédias.
- Limiter le nombre de CSS.
- Écrire des sélecteurs CSS efficaces.
- Limiter le nombre de requêtes HTTP.
- Rendre les éléments du DOM invisibles lors de leur modification.
- Assurer la compatibilité avec les anciens appareils et logiciels.
- Optimiser les requêtes aux bases de données.
- ….

Il est important de n’optimiser que ce qui a le plus d’impact (Loi de Pareto)

Réalisation : Implémentation et codage

OPTIMISER VOTRE CODE 👩‍💻

- Fournir une alternative textuelle aux contenus multimédias.

  Le texte, même mise en forme en HTML/CSS, utilise beaucoup moins de bande passante que des formats multimédias comme l'audio ou la vidéo. Fournir aux utilisateurs une alternative textuelle à ces contenus leur permet s'ils le souhaitent de lire plutôt que d'écouter ou de visionner, et donc de transférer moins de données…

Réalisation : Implémentation et codage

OPTIMISER VOTRE CODE 👩‍💻

- Limiter le nombre de CSS.

  Limiter le nombre de CSS pour ne pas multiplier les requêtes HTTP et pour simplifier le rendu côté navigateur. Utiliser une feuille de style commune pour tous les éléments communs, quel que soit l'affichage, et un fichier par résolution cible ou media querie…

Réalisation : Implémentation et codage

OPTIMISER VOTRE CODE 👩‍💻

- Optimiser les requêtes aux bases de données.

Les requêtes effectuées pour récupérer et enregistrer des données ont une influence importante sur la consommation de ressources. Il est donc important de valider, au moins pour celles qui coûtent le plus, qu'elles soient bien optimisées.

- Ramener moins de données et se limiter au nécessaire.
- N'utiliser que les champs qui sont nécessaires dans les tables ou documents utilisés.
- Ajouter des index sur les champs utilisés comme clés. Leur ajout peut complètement changer la puissance nécessaire pour exécuter la requête.

Réalisation : Implémentation et codage

OPTIMISER VOTRE CODE 👩‍💻

- Assurer la compatibilité avec les anciens appareils et logiciels.

  Pour ne pas déclencher l’obsolescence du terminal de l’utilisateur les pages doivent être utilisables - pas de mises en page cassées, de boutons inactifs, d'erreurs affichées ou autre problème empêchant la lecture ou la navigation, etc.

  Le mieux est de faire une étude sur le terrain des utilisateurs et de leurs équipements.

Réalisation : Implémentation et codage

OPTIMISER VOTRE CODE 👩‍💻

- Rendre les éléments du DOM invisibles lors de leur modification.

  Lorsqu'un élément du DOM (Document Object Model) doit être modifié par plusieurs propriétés, chaque changement de style ou de contenu va générer un repaint ou un reflow. Aussi est-il généralement plus économe de : - rendre l'élément invisible (passer la propriété display à none) (1 reflow) ; - modifier toutes les propriétés de l'élément et rendre l'élément à nouveau visible (1 reflow).Soit 2 reflow au maximum.

  Le ***reflow*** se produit quand un navigateur doit réarranger et redessiner tout ou partie d'une page web, par exemple, après une mise à jour sur un site interactif.

Réalisation : Implémentation et codage

OPTIMISER VOTRE CODE 👩‍💻

- Limiter le nombre de requêtes HTTP.

  Pour chaque fichier, le navigateur émet un GET HTTP vers le serveur. Il attend sa réponse, puis télécharge la ressource dès qu'elle est disponible. Selon le type de serveur web que vous utilisez, plus le nombre de requêtes par page est important, moins vous pourrez servir de pages par serveur. Diminuer le nombre de requêtes par page est crucial pour réduire le nombre de serveurs HTTP (voire des serveurs d'application et de base de données) nécessaires au fonctionnement du site, et donc les impacts environnementaux associés.

Réalisation : Implémentation et codage

OPTIMISER VOTRE CODE 👩‍💻

- Écrire des sélecteurs CSS efficaces.
- Privilégier les sélecteurs basés sur des ID ou des classes. Ils seront ainsi filtrés plus rapidement, économisant des cycles CPU à la machine interprétant les règles.

Réalisation : Implémentation et codage

GESTION DE VERSION 👩‍💻

Utilisation d’ un outil de gestion de version, mais :

- Évitez ou limitez d'y stocker les paquets binaires et les jeux de données non indispensables.
- Ne pas placer en gestion de version les produits de compilation ni les fichiers de sortie.

Réalisation : Implémentation et codage

CHOISIR LA SIMPLICITÉ 👩‍💻

Évitez la complexité. Un code simple à comprendre et à mettre en œuvre sera aussi plus durable et plus facile à faire évoluer.

C'est la base du principe KISS : Keep It Simple and Stupid.

Réalisation : Implémentation et codage

ANALYSER LE CODE 👩‍💻

- les analyses statiques permettent, entre autres, de déterminer la difficulté de maintenance d'un logiciel

  (exemples d'outils : Sonarqube, codacy)

- les analyses dynamiques permettent de quantifier l’usage des ressources (exemples outils : gcov, gprof, perf, valgrind)

  Ces analyses de logiciel - qui font partie des bonnes pratiques classiques - sont indispensables afin de réduire l'impact d'un service numérique.

Réalisation : Implémentation et codage

MESURER POUR RÉDUIRE 

Utilisez des outils de mesure pour évaluer les performances de votre code. Que ce soit la quantité de données échangées, le temps de chargement des pages, le temps d'exécution, le nombre de requêtes ... tout est matière à optimisation.
L'écoconception est une démarche d'amélioration continue sans fin. La mesure est indispensable à cette démarche afin de savoir quoi et comment améliorer, et d'éviter la tentation de la fonctionnalité en plus ou l'effet rebond.

Exemples d'outils : **Firefox web tools** (trafic, requêtes, JavaScript), **Apache JMeter** (scénarios web), **ecoindex** (web), profiler intégré et adapté au langage, **CodeCarbon**, Green Algorithms, KDE eco.

Réalisation : Intégration et tests

Vient ensuite la phase d'***intégration*** qui va consister à la fois à assembler les différentes briques logicielles réalisées pour qu'elles fonctionnent ensemble, et à insérer le service numérique dans un écosystème existant.

Une fois qu'un premier prototype est opérationnel, il va subir des séries de ***tests*** afin de valider sa qualité.

Réalisation : Intégration et tests

LA QUALITÉ AU SERVICE DE LA DURABILITÉ 

La mise en place de tests permet de détecter les anomalies et d'assurer un important niveau de qualité du service. Ceci va **éviter** les régressions lors de ses évolutions et améliorer la satisfaction des utilisateurs. Si votre service numérique contient trop de bugs ou d'instabilités, il risque de ne pas être utilisé et d'être difficile à maintenir dans le temps : tous les impacts environnementaux engendrés par sa production auront été générés pour rien. Néanmoins ***focalisez-vous sur les tests les plus essentiels*** et ***méfiez-vous de leur automatisation*** qui est elle-même une ***source de gras numérique***.

Production : Installation et déploiement

CHOISIR UN HÉBERGEUR ÉCO-RESPONSABLE 

Préférer un hébergeur qui combine des serveurs économes à une alimentation sans énergie fossile (pétrole, gaz, charbon) basé principalement sur des sources d’énergies renouvelables.

Production : Installation et déploiement

UTILISER UN SERVEUR ASYNCHRONE 

Les serveurs (web, d’applications, etc.) tels que Nginx, node.js ou Gwan sont conçus pour utiliser le minimum de ressources possible. Grâce à leur fonctionnement en mode asynchrone, ils ne sont pas tenus de créer un processus ou un thread pour chaque requête. Ils évitent ainsi de gaspiller leurs ressources.

Production : Installation et déploiement

INSTALLER LE MINIMUM REQUIS SUR LE SERVEUR 

Désinstaller tous les services qui ne sont pas indispensables au bon fonctionnement du site. Cette mesure supprimera nécessairement des daemons (agents et services fonctionnant en permanence en mémoire), qui sont consommateurs de ressources, notamment en cycles CPU et en mémoire vive.

Production : Installation et déploiement

SOBRIÉTÉ NUMÉRIQUE 

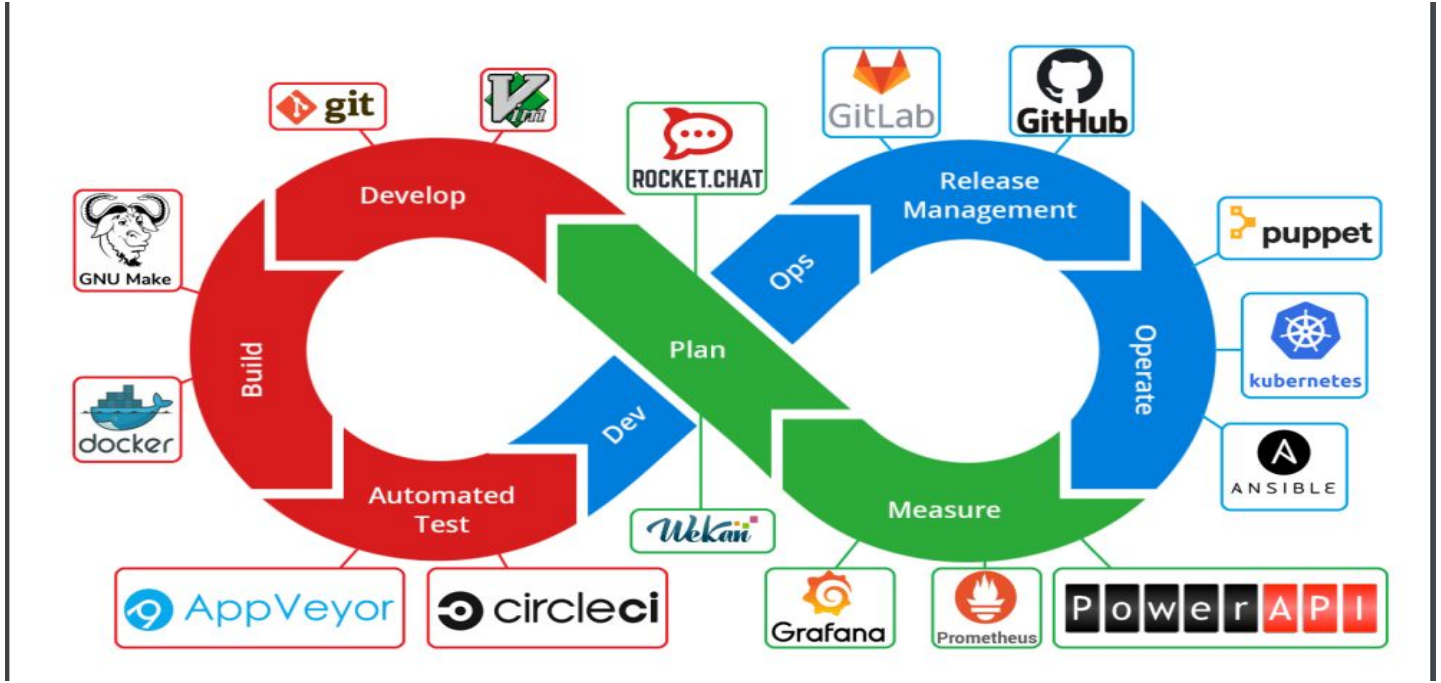
- privilégier un hébergement mutualisé, labellisé COC (Code of Conduct), au plus près des données et des utilisateurs.
- privilégier la virtualisation (moins de serveurs physiques), en minimisant l'empreinte mémoire et disque.
- éviter la multiplication des machines virtuelles, des instances du service.

Production : Installation et déploiement

AMÉLIORATION CONTINUE EN FONCTION DES USAGES 🤦

- Exploiter la supervision (et les alertes) pour observer les pics (CPU, disque, réseaux, consommation électrique, etc…)
- Modifier le service numérique pour l'adapter en fonction de l'usage observé.
- Réduire les fréquences et volumes des sauvegardes.
- Exemples d'outils de supervision: top, zabbix, prometheus, grafana

Production: Exemples d'outils utilisés pour l'amélioration continue du service numérique



Maintenance

Les actions entreprises au cours de la conception et réalisation de l'application doivent être suivie par une maintenance adéquat, afin d'assurer leur pérennité dans le temps, ainsi, voici quelques recommandations pour conserver un service aussi éco-responsable que possible:

- Mesurer le taux d'utilisation lors des maintenances pour désactiver les fonctionnalités non utilisées, et éventuellement supprimer les données obsolètes.
- Partager les principes de l'éco-conception afin de la faire perdurer, par exemple en créant une documentation et/ou en formant les futurs utilisateurs du service pour les aider à respecter les normes mises en place.
- Vérifier les jeux de données du service, mettre en œuvre un tri, automatique ou non, régulièrement. Ne pas oublier de gérer la durée de vie du contenu et d'en informer l'utilisateur.
- Dans le cas d'une application native, informer l'utilisateur des mises à jour et lui laisser le choix sur l'installation des mises à jour ou non.

Sources

- https://collectif.greenit.fr/ecoconception-web/115-bonnes-pratiques-eco-conception_web.html
- https://ecoresponsable.numerique.gouv.fr/publications/referentiel-general-ecoconception/
- https://www.greenit.fr/
- https://blog.codewise.fr/pollution-numerique-developpeur
- https://lowtechlab.org/fr/actualites-blog/faire-un-site-low-tech
- https://alexsoyes.com/eco-conception-web/
- https://www.academie-nr.org/ (MOOC)