PROGRAMMEZ!

Le magazine des développeurs

04/2020 N°239 22e année

Interview exclusive

Bjarne Stroustrup

le créateur de C++

Informatique quantique

Q# / QISKIT

Écologie

Être un dev responsable

Captain Blood

Renaissance d'un jeu mythique

M 04319 - 239S - F: 6,50 € - RD

Le seul magazine écrit par et pour les développeurs

Printed in EU - Imprimé en UE - BELGIQUE 7 € - Canada 9,80 $ CAN - SUISSE 13,10 FS - DOM Surf 7,50 € - TOM 1020 XPF - MAROC 55 DH

N°2
DISPONIBLE

Retour vers le passé :

redécouvrez les ordinosaures et les technologies des années 1970 à 2000 !

Commandez directement sur **programmez.com**

6,66 € *(+frais de port*)* **36 pages**

*Revue trimestrielle. Editée par Nefer-IT. *Avec frais de port : 7,66 €*

NOUVEAU : abonnement 1 an !

Explosera ou explosera pas ?

Le mois de mars fut particulier. Durant plusieurs jours, on se demandait quelles seraient les mesures du gouvernement pour lutter contre le Coronavirus. On avait pu constater comment la Chine, et plus près de nous, l'Italie, tentaient de juguler la propagation. L'arrêt de plusieurs régions sensibles chinoises stoppa de nombreuses chaînes de fabrication d'électronique, mettant dans l'incertitude l'approvisionnement de composants.

Puis finalement, dès fin février, une première (timide) reprise fut annoncée par Foxconn. Et d'ailleurs, on a pu constater une reprise progressive de l'activité chinoise. Mais l'approvisionnement des écrans et de la mémoire pourrait encore être particulièrement tendu plusieurs semaines / mois.

L'enjeu pour les constructeurs sera la reprise des marchés européens et américains et comment la demande va rebondir, ou pas. Business is business.

Programmez! s'adapte aussi aux circonstances. Parfois, nous avons rebooté notre planning et les dates des magazines plusieurs fois par jour, selon les indications et contre-indications des prestataires. Initialement, ce numéro ne devait pas être disponible en version papier car durant les premières heures de confinement, nous n'avions aucune visibilité avant que la situation ne se stabilise. C'est comme passer du kernel panic au mode sans échec de l'OS.

ÉVÈNEMENT

C'est véritablement un petit événement pour nous. Bjarne Stroustrup, le créateur de C++, nous a accordé une longue interview. Évènement, car Bjarne donne assez peu d'interviews, et encore moins à la presse française. A l'occasion de la disponibilité de C++ 20, il nous paraissait pertinent d'en savoir plus sur la création du langage, les évolutions successives, sa vision de la programmation, et, bien entendu, nous parlerons de C++ 20.

A lire impérativement !

De nombreux événements et conférences ont été annulés ou reportés. Nous avons mis à jour notre rubrique agenda. Nous mettons régulièrement à jour la situation sur www.programmez.com

Prenez soin de vous et de vos proches.

La rédaction.

François Tonic - ftonic@programmez.com

SOMMAIRE

Brèves ——— 4
Agenda ——— 6
Roadmap 2020 ——— 8
Captain Blood, le retour ——— 11

Interview
Bjarne Stroustrup ——— 14

Chronique ——— 19

Dossier DevOps partie 2 ——— 20

Informatique quantique ——— 35
Éco-conception ——— 49

Abonnez-vous ! ——— 42
Commitstrip ——— 82

NIVEAU 100


Instapy 58

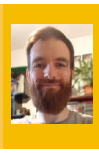

Java 14.....65




Démarrer dans l'open source 61


gRPC *partie 2* 68

NIVEAU 200


Vagrant 72


Tests & gRPC 75

NIVEAU 300


Concurrence en Java *partie 3* 77

MESSAGE PRIORITAIRE :
Programmez! n°240, prochain numéro
date stellaire : 30.04.2020

Dossier tests logiciels : l'éternel retour des tests
Panorama des outils

Coronavirus partout, salons nulle part

Le mois dernier, on vous parlait du MWC annulé à cause de l'épidémie du Coronavirus. Cela ne vous aura pas échappé, le MWC n'était que le premier d'une longue série : de nombreux salons et rassemblements ont annoncé qu'ils repoussaient ou annulaient afin de limiter les risques de propagations. En France, les salons Big Data Paris, IT Partners, Cloud Datacenter, IoT World, Paris Blockchain Week Summit encore Orange Business Summit sont ainsi tous reportés à la fin d'année. Outre-Atlantique, Facebook a annoncé l'annulation de sa conférence F8, également le cas de la Google I/O. Certains évènements se tiendront exclusivement en ligne, comme l'Adobe Summit, Dell World ou la Google Cloud Next. L'année 2020 ne sera pas idéale pour networker.

Ransomware : l'autre épidémie

Au début du mois de mars, on apprenait ainsi que le groupe lyonnais de lingerie Lise Charmel s'était placé en redressement judiciaire afin de se remettre une attaque au ransomware ayant paralysé le groupe au mois de novembre 2019. Fin février, c'étaient les systèmes informatiques de la région Grand Est qui se retrouvaient paralysés par un ransomware ayant chiffré 80 serveurs. Malheureusement pour cette épidémie-là, il faudra faire un peu plus que se laver les mains.

Lets encrypt joue dans la cour des grands

Let's Encrypt

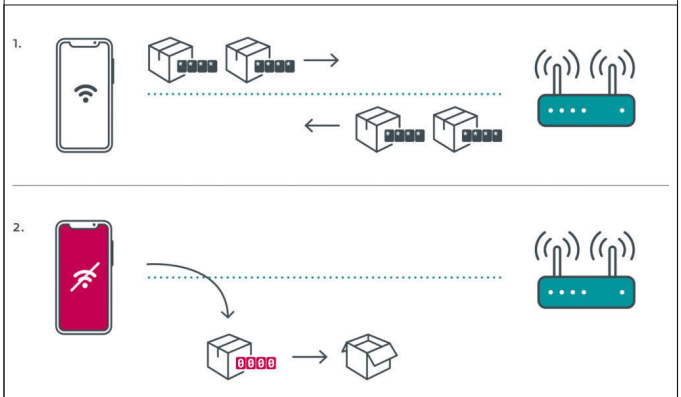
Let's Encrypt a fait du chemin et la petite autorité de certification gratuite à laquelle personne ne croyait il y a quelque temps a annoncé récemment avoir émis plus d'un milliard de certificats SSL et être devenue un acteur majeur dans son domaine. Et comme les autres grandes autorités de certifications, Let's Encrypt se mélange aussi les pinceaux : l'autorité a ainsi annoncé que suite à un bug dans son outil d'émission des certificats, 3 millions de certificats non conformes aux bonnes pratiques avaient été émis par erreur et devaient être remplacés sans délai. Au total, Let's Encrypt a renouvelé 1,7 million de certificats en l'espace de 48h, et a pris la décision de laisser expirer naturellement le million restant pour éviter de perturber l'écosystème. On ne les voit pas grandir hein.

Linux se prépare pour 2038

L'actualité récente offre de nombreuses pistes à explorer pour l'apocalypse qui nous pend au nez, mais on pourra déjà rayer l'apocalypse Linux de la liste des candidats potentiels. Le système faisait en effet face à un potentiel bug de l'an 2000 : son système d'affichage du temps, issu du vénérable Unix et conçu pour des systèmes 32bits, risquait en effet de s'arrêter complètement le 19 janvier 2038 à 3h14 et 8 secondes. Un problème ennuyeux au vu du nombre de systèmes critiques fonctionnant sur les dérivés d'Unix, mais la version 5.6 du noyau Linux commence à proposer les premiers correctifs visant à permettre à un système 32bits de continuer à fonctionner après l'heure fatidique.

Kr00k : un air de Krack

À l'occasion de la conférence RSA, qui avait lieu à San Francisco à la fin du mois de février (et ce malgré le coronavirus, croyez-le ou non) des chercheurs d'Eset ont présenté une nouvelle faille de sécurité affectant les implémentations Wifi sur les puces Broadcom et Cypress. Au total, plus d'un milliard d'appareils seraient concernés par la vulnérabilité, qui permet dans certaines conditions à des attaquants de décrypter le trafic protégé par le chiffrement WPA2. Si la faille est jugée moins grave que la faille Krack, qui avait fait beaucoup parler d'elle il y a quelques années, des correctifs sont néanmoins en cours de diffusion pour limiter la casse.



Microsoft fait marche arrière sur Bing

Office 365

À la fin du mois de janvier, Microsoft annonçait tout fièrement qu'ils allaient forcer les utilisateurs d'Office 365 ProPlus à installer l'extension « Microsoft Search in Bing », qui fait de Bing le moteur de recherche par défaut du navigateur, que ce soit sur Edge, Chrome ou Firefox. Sans surprise, l'annonce n'a pas vraiment suscité un enthousiasme débordant, et, face à la fronde, Microsoft a choisi d'annuler purement et simplement le déploiement automatique de cette extension de navigateur. Celle-ci reste disponible de façon optionnelle pour ceux qui souhaitent la déployer, mais elle n'obligera plus les utilisateurs à passer automatiquement Bing en moteur de recherche par défaut. Comme quoi, parfois ça fonctionne de râler.

tangente
n° 193

Katherine Johnson
Une femme d'exception

Suites numériques
Des dynamiques étonnantes !

Art persan
Des pavages apériodiques

tangente
l'aventure mathématique

Les polynômes cachés du quotidien

De l'approximation au traitement du signal
Défis et énigmes, de la Renaissance à aujourd'hui

Traduire les maths

Des lieux mythiques : Bagdad, Tolède, Alexandrie
Des interprètes inspirés : Boèce, Adélard de Bath

L'émergence des probabilités

Fermat, Descartes, Huygens…
Une application insoupçonnée au football

DOM - LUX - BELG : 7,30 € SUISSE : 12,20 CHF CANADA : 11,99 $ CAN
TUNISIE : 7,20 DTU MAROC : 70 DH / ISSN 0987-0806 / AVRIL-MAI 2020

Avec le soutien du CNL Centre National du Livre

M 05421 - 193 - F: 6,80 € - RD

Avertissement : avec le Coronavirus et les restrictions ordonnées par le gouvernement, de nombreux évènements et conférences pour les développeurs ont été annulés et reportés. Nous faisons un point sur la situation connue au moment du bouclage de Programmez! Nous mettons régulièrement à jour sur www.programmez.com

27 & 28 mai : Big Data Paris

Le salon dédié au big data revient fin mai avec plusieurs dizaines de conférences et d'exposants.

Juin

2 : FlutterConf Paris / Paris

La première grande conférence autour de Flutter arrive début juin. Vous y découvrirez les dernières avancées en matière d'applications mobiles natives multiplateforme (Android et iOS), pour le web ou encore le desktop (Windows, Mac et Linux). Site : https://flutter-conf.paris/fr_FR/

4-5 : Best of web 2020/Paris

Best of web, c'est un évènement sur deux jours. Une journée de workshops et une journée de conférences préparée par 16 meetups web parisiens. La journée-conférence est composée à 50 % du Best Of des talks meetups de l'année et 50 % d'inédits provenant d'une CFP. Les tickets sont spécifiques à chaque journée. Site : http://bestofweb.paris

12 : DevFest Lille/Lille

La DevFest revient à Lille. L'appel aux speakers est lancé. C'est le plus grand évènement dans le nord, avec +800 personnes attendues. Site : https://devfest.gdglille.org

Juillet

1-3 : Devoxx France/Paris

Devoxx France sera en été. La 9e édition s'annonce très bien. La première nouvelle c'est qu'il n'y a plus de places à vendre... depuis l'ouverture. Jeudi et vendredi matin, il y aura 5 à 6 plénières, sur un format de 20mn, proche de ce que font les conférences TED. Le thème de cette année est "Tech 4 good, Tech 4 evil". Comment notre métier et la technologie peuvent-ils améliorer (ou non) la vie au quotidien de nos clients ? Une personne de la Fondation de l'Abbé Pierre, un chercheur, une CEO dans l'humanitaire... Sans dévoiler le programme organisé par Benoit Lafontaine, Katia Aresti et Charlotte Abdelnour, on peut s'attendre à de belles choses ! +920 sujets reçus pour 240 sessions !

Bref, c'est l'évènement incontournable des dévs.

Septembre

1er : JFTL / Montrouge

La journée des tests logiciels revient pour la 12e année. Le but est de montrer, expliquer les tests, et pourquoi ils sont si importants. 1 000 personnes sont attendues. Site : http://www.cftl.fr/JFTL/accueil/

23 & 24 : Cloud Expo & IoT World

Octobre

30 : Agile Tour Brussels/Bruxelles

C'est l'évènement agilité en Belgique. +200 personnes attendues. Site : http://www.agiletourbrussels.be

Evenements en ligne

Plusieurs salons ont annoncé une transformation en évènement en ligne. Les dates restent à préciser :

- AWS Summit Paris
- GitHub Satellite 2020
- Microsoft Build
- WWDC 2020 : juin

Reporté à 2021

- MixIT/Lyon : De nombreux thèmes sont abordés : design, technologie, makers, éthique dans l'IT, style de vie, travail en équipe, etc. https://mixitconf.org
- RGP Level 3 : 20&21 mars 2021
- BreizhCamp 2020
- Google I/O
- Facebook F8
- Serverless Days Paris
- RivieraDev
- Newcrafts

LES MEETUPS PROGRAMMEZ!

21 avril : meetup#9

Angular & Ivy

19 mai : meetup#10 *(Meetup en partenariat avec Red Hat)*
16 juin : meetup#11 *(Sujet à venir)*

A PARTIR DE 18H30

Où : Infeeny 5 rue d'Uzès Paris
Métro : station Grands Boulevards (lignes 8 & 9)

Informations & inscription : programmez.com

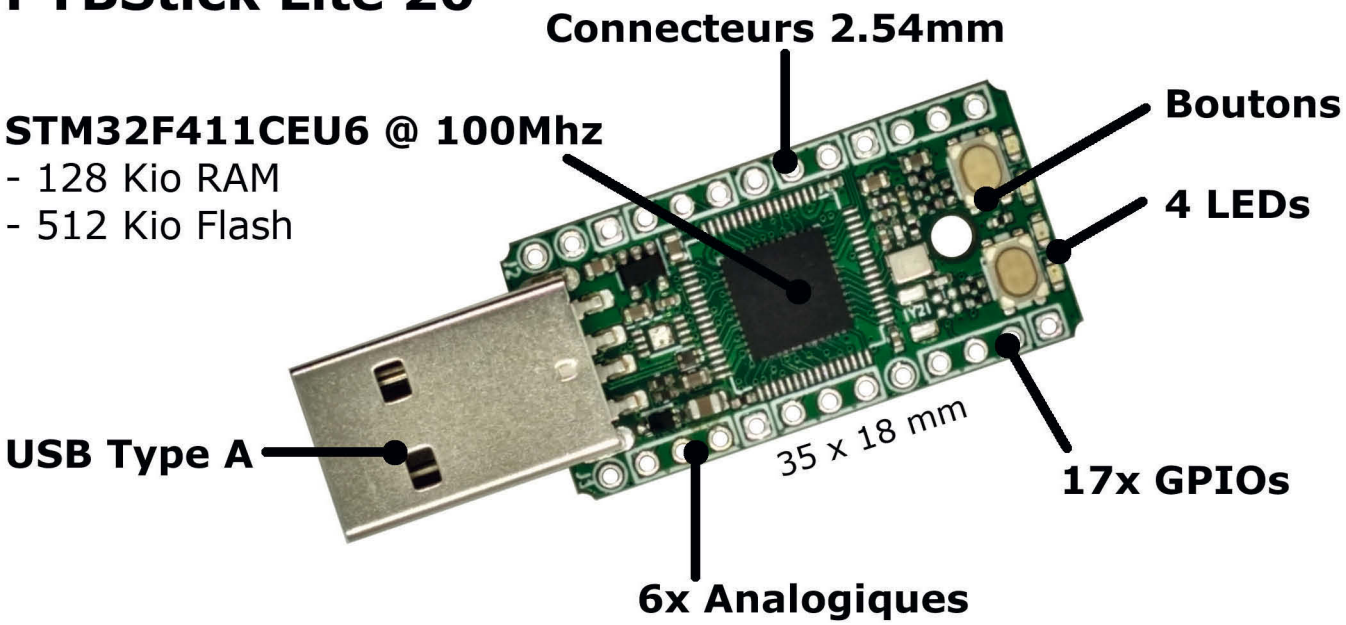
KubeCon

La grande conférence européenne sur Kubernetes est reportée à juillet ou août. Pas de nouvelles connues.

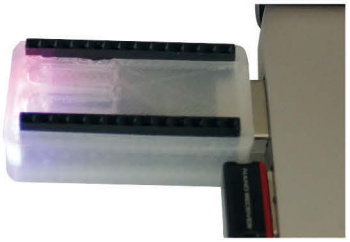
Merci à Aurélie Vache pour la liste 2020, consultable sur son GitHub :
https://github.com/scraly/developers-conferences-agenda/blob/master/README.md

Printemps 2020

PYBStick Lite 26



Programmez!, Garatronic et MC Hobby s'associent pour vous offrir une plateforme d'apprentissage **surpuissante**, **compacte**, **abordable** et **documentée en Français**!

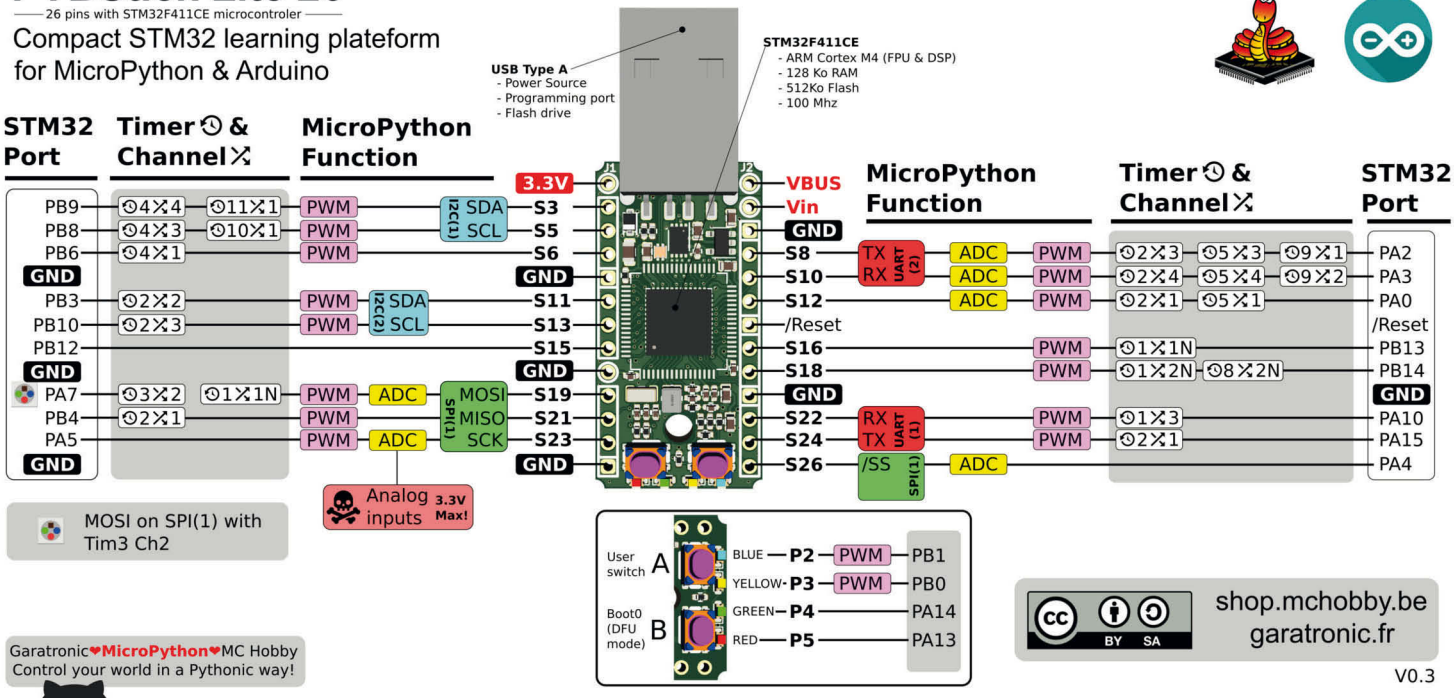


Fabriqué en France

PYBStick Lite 26

26 pins with STM32F411CE microcontroler

Compact STM32 learning plateform for MicroPython & Arduino



shop.mchobby.be
garatronic.fr

V0.3

Garatronic♥MicroPython♥MC Hobby
Control your world in a Pythonic way!

https://github.com/mchobby/**pyboard-drive**r/tree/master/**PYBStick**

Roadmap des langages

Chaque mois, Programmez! vous propose un panorama des sorties des versions des langages, frameworks, etc.

DÉJÀ DISPONIBLE

Rust 1.41

En ce qui concerne le langage lui-même, une des nouveautés les plus importantes est l'assouplissement de certaines restrictions lors de l'implémentation des traits. On trouve aussi des améliorations sur le gestionnaire des packages Cargo. Avec cargo install, vous pouvez installer des crates binaires dans votre système. La commande est souvent utilisée par la communauté pour installer des outils CLI populaires écrits en Rust.

Java 14

Voir notre article

TypeScript 3.8

Cette mouture se fait principalement remarquer par une nouvelle fonctionnalité permettant d'importer/exporter des types seuls. On notera aussi le support de l'ECMAScript Private Fields.

```
class Person {
  #name: string

  constructor(name: string) {
    this.#name = name;
  }

  greet() {
    console.log(`Hello, my name is ${this.#name}!`);
  }
}

let jeremy = new Person("Jeremy Bearimy");

jeremy.#name
//     ~~~~~
// Property '#name' is not accessible outside class 'Person'
// because it has a private identifier.
```

Flutter 1.12.13

Cette version propose une belle panoplie de nouveautés et améliorations. Citons :

- Le support du mode sombre iOS 13 ;
- Version alpha de Flutter pour macOS ;
- Flutter pour le web arrive en bêta ;
- La version de DART évolue aussi. Flutter supporte la v2.7 ;
- Meilleure stabilité des API pour Java, Kotlin, Objective-C, Swift.

Attention : les équipes mettent en garde contre les changements qui vont casser le code existant.

Angular 9

La base technique est celle de la v8. Cette version propose tout de même des évolutions sur le framework, Angular Material et la CLI. On retrouve par défaut Ivy (compilateur + runtime). Ivy promet beaucoup sur l'optimisation de la taille des projets. Les équipes promettent aussi des améliorations sur la gestion des erreurs, le debug, l'internationalisation, etc.

C++20

Cette version du langage met en avant deux nouveautés : les modules et les coroutines. Les modules constituent une nouvelle alternative aux fichiers d'en-tête qui apportent un certain nombre d'améliorations clés, notamment en isolant les effets des macros et en permettant des compilations évolutives, explique Herb. Il ajoute qu'en 35 ans c'est la première fois que C ++ ajoute une nouvelle fonctionnalité permettant aux utilisateurs de définir une limite d'encapsulation nommée. Les coroutines sont elles aussi une fonctionnalité à remarquer. Une coroutine est une unité de traitement qui s'apparente à une fonction (ou routine), avec cette différence que si une sortie du corps d'une fonction met fin à l'exécution de celle-ci, la sortie de la coroutine suspend seulement son traitement qui peut ensuite reprendre, l'état du traitement à la sortie étant conservé. Une coroutine peut être vue comme un morceau de programme qui conserve son état, mais qui n'a pas de thread d'exécution. Les coroutines peuvent notamment être utilisées pour des itérateurs et des générateurs.

Kotlin 1.3.70

Cette version est une mise à jour mineure. Plusieurs améliorations et nouveautés sont tout de même présentes :

- Meilleur support de gradle.kts. Les développeurs pouront améliorer le support des fichiers et réduire la charge CPU durant la synchronisation ;
- Sur Kotlin/JavaScript : optimisation sur les navigateurs. Des tâches vont changer de nom comme browserWebPack. Donc méfiance.

BIENTÔT DISPONIBLE

Symfony 5.1

Mai

Une des grosses nouveautés est l'apparition du mode safe http Preference via Prefer: Safe http. Cette fonction sera implémentée dans le composant HttpFoundation. On verra aussi l'apparition des propriétés typées dans PropertyInfo. Elles sont apparues dans PHP 7.4.

2e semestre 2020/2021

Python 3.9.0

Eté/automne

Les premières versions alpha de la 3.9 ont été distribuées fin janvier. La phase bêta est attendue vers mai/juin. Désormais, il faudra s'attendre à une mise à jour annuelle du langage. Bref : moins de nouveautés, mais plus d'évolutions et d'améliorations. Parmi les nouveautés attendues : des améliorations sur le garbage collector et la résurrection d'objets (pouvant provoquer des blocages), évolution de la stabilité ABI pour éviter les incompatibilités sur les différents systèmes.

Swift 5.2

?

La prochaine version du langage sera la 5.2. La première bêta est sortie en mars. Plusieurs focus ont été annoncés : réduction de la taille du code et optimisation de l'usage mémoire en exécution, nouvelle architecture diagnostic pour mieux aider le développeur, Subscripts pourront être déclarés en argument default.

PHP 8.0

Fin 2020 ?

La prochaine grande version de PHP doit arriver fin 2020 si tout va bien. Tout n'est pas encore fixé, mais nous connaissons les premières nouveautés et évolutions du langage :

- Forte possibilité d'une casse de compatibilité avec les versions 7.x : sur ce point, on attend les précisions ;
- Les types Union vont arriver avec PHP 8, qui permettront de définir plusieurs types pour les arguments reçus par une fonction, ainsi que pour la valeur qu'elle retourne. En plus du type self, le type static va devenir un type de valeur retournée valide ;
- Le compilateur JIT de PHP 8 promet d'importantes améliorations de performances.

.NET 5 : PREMIÈRES PRÉVERSIONS !

.Net 5 devrait sortir officiellement à l'automne prochain. L'objectif de cette version est d'unifier les différents .Net en ciblant le multiplateforme. Il inclura ASP.Net Core, Entity Framework Core, WinForms, WPF, Xamarin, ML. Net. Pour la première fois, la plateforme utilisera les Base Class Libraries pour toutes les apps. Depuis la mi-mars, les premières préversions sont disponibles. Deux installeurs sont récupérables :
.Net 5.0 preview SDK
.Net 5.0 preview runtime
+
ASP.Net Core updates pour .Net 5 preview

Un SDK encore incomplet

Le SDK inclut le runtime 5.0, ASP.Net Core runtime 5.0 et le Desktop runtime 5.0. Il supporte C# 8, F# 5 (préversion) et VB 15.5. On peut le déployer sur Linux, macOS, Windows. Les architectures ARM sont supportées dans les versions Linux et Windows. Le runtime est disponible sur les mêmes plateformes. Pour le SDK ? Vous devrez utiliser les versions les plus récentes de Visual Studio 2019.
À noter que le support ARM64 sur Windows est encore préliminaire et sera amélioré au fur et à mesure. L'éditeur annonce l'intégration de WPF et Windows Forms dans une prochaine préversion, sans date précise.

Les focus de la preview 1

La préversion 1 doit apporter :
- Une amélioration des performances des expressions régulières (via le moteur regex),
- Une génération JIT plus véloce,
- De nouveaux outils pour monitorer l'event pipe (via dotnet-trace),
- La rationalisation des différents référentiels de code .Net sur Github.

Attention : pour bien préparer vos codes, Microsoft conseille de migrer vos codes .Net Core vers la version 3.1. La migration de la 3.1 vers .Net 5 sera facilitée. Cependant, Microsoft ne promet pas une migration transparente et sans effort. On devrait y voir plus clair dans les prochaines semaines, au fur et à mesure des préversions.
Pour récupérer les installeurs et binaires :
https://dotnet.microsoft.com/download/dotnet-core/5.0

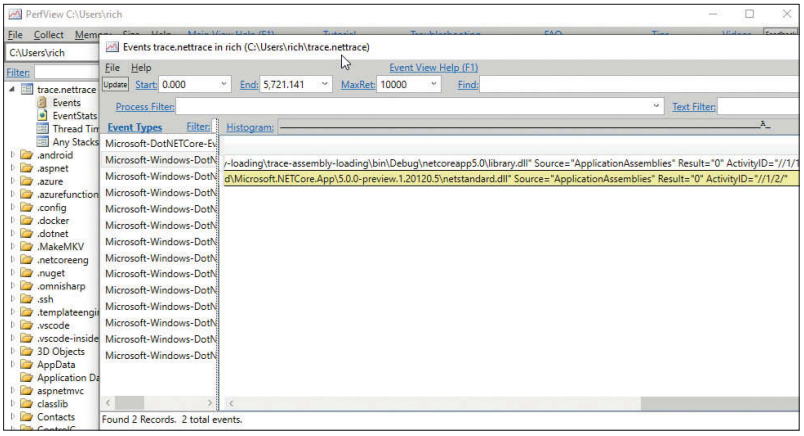
.Net 5 & Visual Basic

Voir notre chronique en page 19

Rappels sur les supports des versions actuelles

Versions	Support	Fin du support
.Net Core 1.0 & 1.1	Fin de vie	Juin 2019
.Net Core 2.2	Fin de vie	Décembre 2019
.Net Core 2.1	LTS	Aout 2021
.Net Core 3.0	Fin de vie	Mars 2020
.Net Core 3.1	LTS	Décembre 2022

LTS est le support long terme.

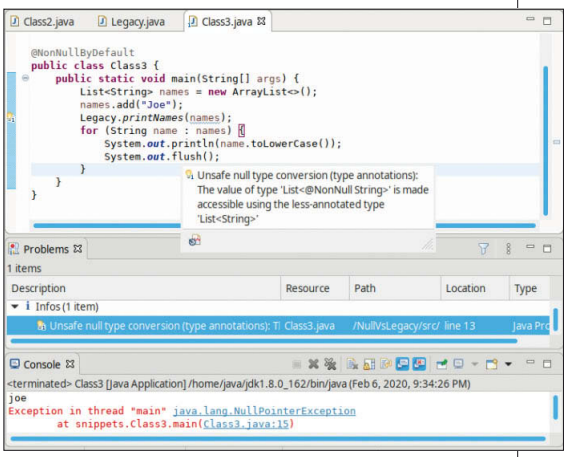


.NET 5 a des objectifs que Microsoft qualifie 'de haut niveau'. Notamment :

- Expérience du SDK .NET unifiée :
  - BCL (bibliothèque de classes de base) unique dans toutes les applications .NET 5. Aujourd'hui, les applications Xamarin utilisent le Mono BCL, mais viendront à utiliser le .NET Core BCL, améliorant ainsi la compatibilité entre les modèles d'applications.
  - Le développement mobile (Xamarin) est intégré dans .NET 5. Cela signifie que le SDK .NET prend en charge les mobiles. Par exemple, vous pouvez utiliser «dotnet new XamarinForms» pour créer une application mobile.
- Applications natives prenant en charge plusieurs plateformes: projet Single Device qui prend en charge une application pouvant fonctionner sur plusieurs appareils. Par exemple Windows Desktop, Microsoft Duo (Android) et iOS à l'aide des contrôles natifs seront pris en charge.
- Applications Web prenant en charge plusieurs plateformes: projet Single Blazor qui prend en charge une application pouvant fonctionner dans les navigateurs, sur les appareils mobiles et en tant qu'application de bureau native (Windows 10x par exemple).
- Cloud Native Applications : hautes performances, fichier unique (.exe) <50 Mo de microservices et prise en charge de la création de plusieurs projets (API, frontaux Web, conteneurs) à la fois localement et dans le cloud.
- Améliorations continues, telles que : algorithmes plus rapides dans la BCL, meilleure prise en charge des conteneurs lors de l'exécution, prise en charge de HTTP3.

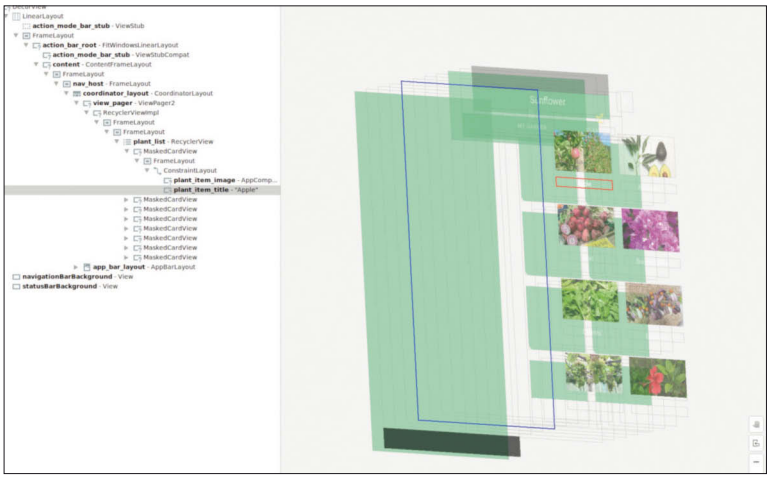
Eclipse 2020-03 (version 4.15)

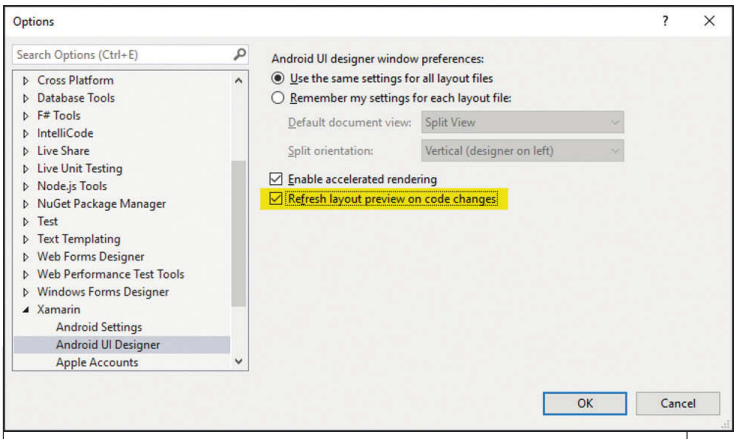
La fondation Eclipse a lancé la version 4.15 le 18 mars. Les évolutions sont nombreuses : amélioration sur la complétion du code, simplification de syntaxe des expressions lambda et des méthodes références, avertissement de compilation pour le null dans les codes anciens, nouvelles API dans CTabItem, retrait du support de WebKit1, amélioration du thème sombre.
Prochaine version : mi-juin.



Android Studio 4.0

Les équipes de Google préparent la version 4.1 de son IDE dédié aux développements Android : Android Studio. La v4.1 supportera Jetpack Compose. Il s'agit d'une approche plus « moderne » pour créer des interfaces utilisateurs. Le toolkit va tirer parti des possibilités de Kotlin. Le code devrait être plus concis et plus clair. L'interopérabilité avec Java est pleinement assurée. Il sera aussi possible de faire une assertion de code quand on déploie une version debug de son app. Le runtime Android ne permet pas ce type de chose, pas par défaut du moins.
Attention : la 4.1 n'est qu'au début de son développement. La v4 sera la prochaine version. Elle proposera des améliorations sur le CPU Profiler et sur la partie build. L'inspecteur live pour les layouts aura droit à quelques améliorations : meilleure hiérarchie dynamique layout, une vue 3D (pratique pour voir les différentes couches), les vues multiples, support des fichiers scripts Kotlin DSL, un nouvel éditeur pour le MotionLayout (animation). Cette fonction est très pratique quand on veut créer des animations rapidement sur un écran. On aura droit à une nouvelle version du plugin Gradle ou encore aux templates live Kotlin.
La v4 s'annonce comme une version majeure.





Visual Studio 2019 – v16.5

Microsoft vient de libérer la version 16.5 de Visual Studio. Le développement mobile .Net est à l'honneur avec la fonction XAML Hot Reload pour les forms Xamarin. Cette fonction doit aider le développeur Xamarin quand il crée les formations. Intérêt : pas de build de son application à chaque modification des forms ! Les modifications peuvent se faire en live !
Microsoft souligne encore une amélioration de la productivité en .NET grâce à l'arrivée de nouvelles fonctions de refactoring et par le fait qu'IntelliSense prend désormais en charge la complétion pour les méthodes d'extension non importées.
Sur la partie C++, cette version améliore le travail avec CMake : possibilité de supprimer, d'ajouter des fichiers source et des cibles dans un projet CMake sans le faire manuellement dans les scripts CMake.
De nombreux changements ont été faits dans F#, Azure tools, debug, les tests, JavaScript et TypeScript, UWP, etc.

Tous les détails : https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes#16.5.0

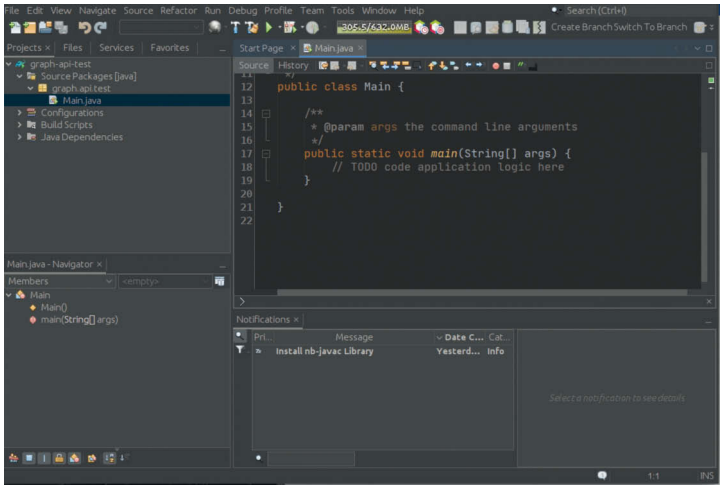
Pour voir les prochaines fonctionnalités supportées : https://docs.microsoft.com/en-us/visualstudio/productinfo/vs-roadmap

L'IDE est disponible en édition Community, Professional et Enterprise. Visual Studio for Mac sera mis à jour vers avril ou juin avec la version 8.6. Les principales évolutions : support des projets Blazor, gRPC, Azure Functions v3.

NetBeans 11.3

La fondation Apache a déployé la version 11.3 (NetBeans 11 feature update 3) le 24 février dernier. Cette version intègre de nombreuses évolutions et nouveautés, telles que :

- Records (JEP 359) : syntaxe plus compacte pour déclarer des classes ;
- Formatage du code : pratique et rapide à faire pour harmoniser son code ;
- Conversion d'un bloc texte en string ;
- Mise à jour de Gradle ;
- Disponibilité des thèmes Dark Metal et Dark Nimbus ;
- Mises à jour des modules ASM, PostgreSQL, JUnit, PHP, HTML, JSF, avertissements de compilation, etc.



Tous les détails de la version : https://cwiki.apache.org/confluence/display/NETBEANS/Apache+NetBeans+11.3

Frédéric Larivé
Développeur fullstack web/.net chez **RadioShop.** *Développeur amateur de jeux et créateur du fan game The Bloodian Chronicles inspiré par l'arche du Captain Blood, jeu vidéo de SF sorti en 1988 sur Atari ST, Amiga, Amstrad CPC, Commodore 64...*
http://www.bloodianchronicles.com - contact@bloodianchronicles.com

L'étrange destinée du Captain Blood

En 1988, j'avais 12 ou 13 ans à l'époque et j'avais un Amstrad CPC... C'était la grande époque des ordis grand public, les illustres Atari ST, Amiga, Amstrad CPC, Commodore... Une époque que les moins de 40 ans n'ont sans doute pas connue... Les démo makers rivalisaient de génie, les crackers faisaient la loi et on s'échangeait des disquettes pour tester les meilleurs jeux du moment... C'était l'époque de 3615 Joystick, des pokes, de Pixel, d'Amstrad 100%, de TILT, des magazines aujourd'hui disparus... Les CDROM n'existaient même pas et Internet était encore inconnu du grand public.

1988, la révélation

Un beau jour, je suis tombé par hasard sur un jeu vidéo nommé L'arche du Captain Blood. C'était un jeu de science fiction très original qui ne ressemblait à rien qu'on ait déjà vu... Le scénario était génial, le gameplay très innovant et les graphismes inspirés de HR Giger absolument sublimes... La musique d'intro était signée Jean-Michel Jarre... Une véritable œuvre d'art... Made in France ! 1

Les années 80 ont marqué le début de l'industrie du jeu vidéo. Les jeux étaient généralement développés par de petites équipes avec peu de moyens... Il n'y avait pas de marketing, pas d'analytics, on pouvait tout se permettre, la créativité débridée des auteurs de cette époque était juste limitée par les nombreuses limitations techniques des machines. C'était le début de la fameuse French Touch.

Je vais tâcher de vous raconter l'étrange épopée de ce jeu extraordinaire, mon combat pour refaire revivre le souvenir de ce jeu hors du commun et les incroyables rencontres qu'un développeur passionné et acharné peut faire !

2009, le déclic

J'habite à Montpellier et je suis développeur depuis plus de 20 ans, mais pas du tout dans le jeu vidéo...

J'ai toujours adoré développer des prototypes de jeux sur mon temps perso... J'ai commencé sur Amstrad avec OCP Art Studio en basic, sur Amiga avec Deluxe Paint en amos, puis sur PC avec 3ds max, Photoshop et Unity en C#.

Avec le temps j'ai appris à m'occuper un peu de tout, graphisme, programmation, scénario, gameplay...

En 2009, je suis retombé sur des screenshots de l'arche de Captain Blood sur Internet et cela m'a rappelé beaucoup de bons souvenirs... La musique marque nos souvenirs de façon indélébile, les jeux vidéo aussi... Sur l'écran d'intro du jeu, on voyait le nom des auteurs : Scénario Philippe ULRICH / Code & Graphisme Didier BOUCHON. Je n'ai jamais oublié ces deux noms. Le studio se nommait ERE Informatique et le jeu était créé sous le label EXXOS (sorte de divinité inspiratrice qu'on retrouvera sur de nombreux jeux).

Après avoir créé une trentaine de jeux entre 1983 et 1988, Ere Informatique sera finalement racheté par Infogrames. Philippe Ulrich finira par quitter Infogrames avec d'autres créatifs pour cofonder Cryo Interactive. L'arche du Captain Blood a reçu de nombreuses récompenses prestigieuses et a connu deux suites officielles qui n'ont pas connu le même succès : Commander Blood en 1994 et Big Bug Bang en 1996. 2

J'ai alors décidé sur un coup de tête, par nostalgie et par passion, de m'amuser à modéliser le pont de l'arche sous 3ds max. Au bout de quelques semaines j'avais redessiné plusieurs écrans et créé plusieurs nouveaux.

J'ai décidé de faire vivre tout cela, en maquettant un début de jeu en XNA (C#).

Au bout de quelques mois, j'avais un début de démo jouable de ce qui pourrait être une suite à l'arche du Captain Blood. Je suis tombé par hasard sur le site perso de Philippe Ulrich qui avait quitté le jeu vidéo après l'explosion de la bulle Internet dans les années 2000. Je lui ai envoyé un email pour lui faire part de mon projet en espérant pouvoir lui présenter ma maquette.

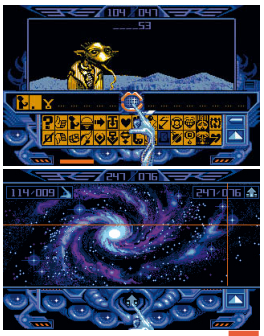
2010, la rencontre

Quelques jours plus tard, j'avais une réponse très positive de Philippe Ulrich et de Didier Bouchon (ils avaient adoré ma maquette) et un rendez-vous était fixé pour se rencontrer à Paris. Pouvoir rencontrer deux grands noms du jeu vidéo français était un véritable rêve de gosse. Ils avaient bercé mon adolescence de leurs créations atypiques et avaient sans doute fait naître ma vocation de développeur.

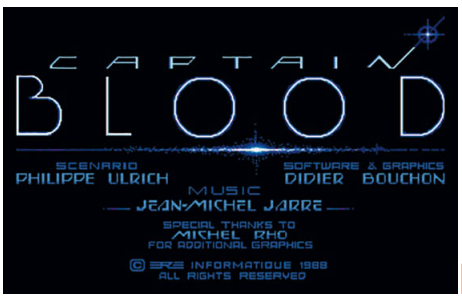
Nous nous sommes rencontrés devant le grand Rex à Paris. A la table d'un restaurant nous avons partagé des souvenirs et des anecdotes autour de l'arche du Captain Blood et des jeux des années 90.

De retour à Montpellier, nous avons travaillé sur le gameplay du futur Captain Blood, nous avions imaginé un vaste monde ouvert, un jeu social et participatif (c'était l'époque ou les jeux sociaux de Zynga cartonnaient sur facebook).

Philippe Ulrich avait inventé un concept de biogame (jeu du vivant) où le joueur se voyait offrir une planète en héritage et devenait un messie, une sorte de bienfaiteur pour les peuples aliens qui y habitaient.



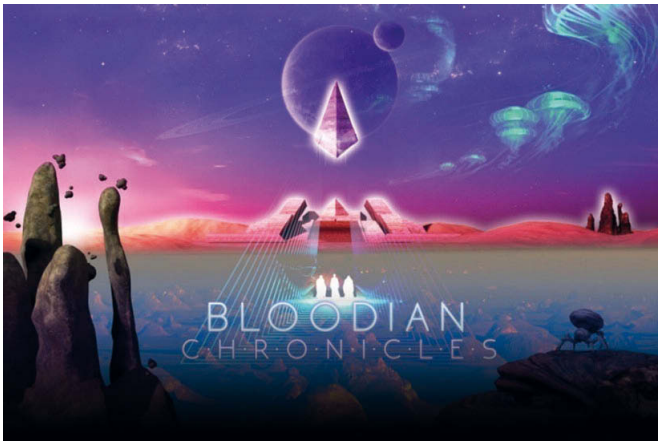
1



2

3

4



2011, Captain Blood Legacy

Nous avions trouvé un nom : Captain Blood Legacy (l'héritage du Captain Blood) et il était question de monter un petit studio indépendant entre Montpellier et Paris : Twikon Biogames. Pour cela, il fallait trouver des business angels, car réaliser un jeu de cette ampleur demandait une équipe et une infrastructure technique. 3

Après 2 ans de recherche et malgré les nombreuses relations de Philippe Ulrich et de Didier Bouchon, nous n'avons pas réussi à trouver d'investisseurs prêts à nous soutenir. Nous avions une fenêtre avec MyMajorCompany qui voulait un projet phare pour se lancer dans le financement participatif de jeux vidéo, ou même avec KickStarter, mais pour diverses raisons cela ne s'est pas fait. Le crowdfunding n'est pas une solution miracle, on n'a le droit qu'à une et unique chance, il faut se préparer longuement et trouver des contreparties à offrir aux bakers. Nous faisions des points par Skype régulièrement pour discuter du gameplay, des tendances et cherchions à faire un jeu original, étrange et différent des autres. Les idées fusaient dans tous les sens, il y avait tellement matière à créer de nouvelles choses, de nouveaux concepts, de nouvelles races d'aliens… Cette volonté de créativité et de différence, loin du format bien borné et marketé des jeux commerciaux du moment, nous a sans doute desservi dans notre recherche d'appuis financiers.

De 2010 à 2013, j'ai réalisé toutes sortes d'artworks sous 3ds max et Photoshop pour essayer de mettre en image les concepts de gameplay de Captain Blood Legacy.

En 2011, un trailer vidéo de Captain Blood Legacy a été présenté lors du Salon International du jeu vidéo de Montpellier, le MIG. J'ai eu le plaisir d'assister à une conférence passionnante sur le retro gaming made in France à laquelle participaient Philippe Ulrich (Captain Blood), Eric Chahi (Another World), Frederik Raynal (Alone in the dark) et Paul Cuisset (Flashback). Nous espérions trouver des soutiens lors de cette présentation, mais ce ne fut malheureusement pas vraiment le cas. 4

Notre petite équipe a fini par se dissoudre et à sonné le glas de Captain Blood Legacy. Je n'ai plus aucune nouvelle de Philippe Ulrich ni de Didier Bouchon depuis 2013, mais ce fut néanmoins une belle rencontre que je ne regrette pas.

2016, The Bloodian Chronicles

En 2016, j'ai décidé de démarrer un nouveau projet from scratch en solo. Je ne pouvais pas rester sur l'échec de Captain Blood Legacy. J'ai décidé de repenser complètement le projet, car étant seul, il fallait redimensionner le projet et partir sur un objectif plus atteignable. Il fallait tout repenser : le scénario, les personnages, le gameplay, c'était comme repartir d'une page blanche 5. J'ai commencé par écrire le scénario en partant de la fin. Je voulais une fin palpitante, un vrai coup de théâtre, avec plein de personnages étranges, attachants et charismatiques. Je voulais que le jeu soit relativement grand public avec une science-fiction recherchée mais accessible. D'où l'idée de partir sur un anti-héros : une jeune fille d'une douzaine d'année qui se réveille après 800 ans de cryogénisation et qui a perdu la mémoire.

En plus de la SF, j'adorais les point and click des années 90 comme Monkey Island, Indiana Jones… C'est pourquoi j'ai décidé de me lancer dans un point and click inspiré de l'univers de Captain Blood. J'ai commencé à apprendre à utiliser Unity et c'est comme ça que j'ai démarré The Bloodian Chronicles il y a 4 ans.

Techniquement, c'est un point and click en 2,5D, c'est-à-dire des personnages en 3D temps réel et un fond en 2D.

Le fond est un mélange de rendus 3D montés et améliorés ensuite sous Photoshop. Pour les bitmaps de fond je suis parti sur

5

6

7

une taille 2048x1024. Cela me permet de couvrir les différentes plateformes sans stretching (réduire la taille d'une image bitmap ne dégrade pas sa qualité visuelle, mais agrandir physiquement un bitmap dégrade visuellement l'image). 6

Tout faire tout seul est un travail de titan. Il y a la partie création graphique, il faut imaginer un univers, le modéliser en 3D et le peaufiner sous Photoshop, l'intégrer sous forme d'assets dans le jeu

Il y a les étapes suivantes :

- la modélisation des personnages en low poly ;
- le mapping (comment la texture va être projetée sur l'objet) ;
- le texturing (créer les textures sous Photoshop) ;
- le rigging, (définir le squelette pour l'animation) ;
- le skinning (comment le squelette va déformer l'objet) ;
- la création d'animation (j'utilise beaucoup de fichiers de motion capture .bvh trouvés sur le web) ;
- la création de morph targets (morphing) pour les anims du visage, des mains ;
- l'import dans Unity, la récupération des anims dans Mecanim, le module d'animation de Unity ;
- La création d'un controleur en C# qui va permettre de manipuler le personnage par code (marche, ferme les yeux, arrête toi…)

Heureusement on peut trouver beaucoup de ressources gratuites (attention quand même au type de licence) sur le web… On trouve pléthore de modèles 3D (des aliens, des armes, des vaisseaux…), parfois déjà en low poly mappés et texturés, d'autres fois non… Si ce n'est pas le cas, j'utilise ZBrush (Decimation Master et UV Master) pour réduire le nombre de polys du modèle et lui appliquer des coordonnées de mapping. Ensuite, je passe par un petit outil nommé xNormal pour créer les textures de normal (normal maps) à partir du modèle hi-poly. La normal map sera appliquée au modèle low-poly et simulera tous les détails de la version hi-poly…

Voici quelques-uns de mes sites de ressources préférés :

- Sketchfab.com
- Cadnav.com
- Cgtrader.com
- Unity Asset Store

Parfois j'utilise des modèles 3D ou des images trouvées sur internet (deviant art par exemple).

J'essaie au maximum de contacter les auteurs et de leur demander leur autorisation d'utiliser leurs œuvres dans mon jeu en créditant leur travail. La plupart acceptent avec plaisir.

Sinon, je les utilise en les détournant suffisamment de leur aspect original, si bien qu'ils sont méconnaissables.

Dans tous les cas, je n'utilise rien de façon brute. J'essaie toujours d'apporter de la valeur ajoutée, d'ajouter ma touche personnelle à tout ce que j'intègre dans le jeu afin de le rendre étrange, original et visuellement homogène.

Conclusion

Je ne prétends pas faire une suite officielle à l'arche du Captain Blood, mais plutôt un fan game amateur inspiré par cet univers. Je ne reprends aucun personnage de Captain Blood, je m'inspire des concepts clés qui ont fait le succès du jeu comme l'UPCOM, l'interface de communication universelle à base d'idéogrammes qui permet de communiquer avec toutes les races d'aliens de l'univers.

Le jeu est un hommage au jeu original et à ses créateurs pour qui j'ai toujours eu le plus grand respect. L'arche du Captain Blood est un jeu unique, la chaîne Arte à d'ailleurs créé le documentaire Retro Gaming Made In France, une série documentaire en 10 épisodes qui retrace l'histoire des premiers grands jeux vidéo made in France et de leurs auteurs. Le 1er épisode était consacré à… L'arche du Captain Blood bien sûr !

https://www.arte.tv/fr/videos/086752-001-A/retro-gaming-made-in-france-1-10/

L'arche du Captain Blood est un jeu qui disparaît doucement de la mémoire collective. Il y a aujourd'hui de moins en moins de personnes qui se souviennent de ce jeu et c'est bien dommage. Les temps ont changé, les joueurs ne sont plus les mêmes, les codes ont changé, ils n'ont plus les mêmes attentes et la même patience que les joueurs des années 80/90. Aujourd'hui il faut soigner son on boarding pour éviter que le joueur ne passe à autre chose au bout de 2mn. The Bloodian Chronicles sera entièrement gratuit et sera disponible d'abord pour Windows, ensuite pour Android et IOS. Grâce à Unity, j'ai un code unique portable sur de nombreuses plateformes (via Mono). Le jeu sera divisé en 2 épisodes séparés (2 apps différentes sur les différents stores) d'une douzaine de chapitres chacun. Aujourd'hui je termine le chapitre 4. Je pense diffuser une première beta d'ici quelques mois des 4 premiers chapitres afin d'avoir des retours de joueurs. Elle se sera disponible en téléchargement sur mon blog : http://www.bloodianchronicles.com 7. Vous pourrez y trouver des tutoriels pour débuter sous Unity, et une sorte de carnet de bord (dev blog) de l'avancement du jeu… N'hésitez pas à me contacter si le projet vous intéresse et à me dire ce que vous en pensez !

Bjarne Stroustrup créateur de C++

Depuis plusieurs mois, Programmez! souhaite créer une nouvelle série d'articles, et plus spécifiquement, des interviews des créateurs de langages. Pour inaugurer cette rubrique, nous vous proposons une interview exceptionnelle du créateur de C++ : Bjarne Stroustrup. Christophe Pichaud, contributeur C++ bien connu dans le magazine, s'est longuement entretenu avec Bjarne le 24 février dernier, quelques jours après la disponibilité officielle de C++ 20.

Christophe PICHAUD
Lead Software Architect chez Infeeny
christophep@cpixxi.com | www.windowscpp.com
infeeny | econocom

Comment naît un langage de programmation ? Comment avez-vous créé C++ ? Expliquez-nous son histoire.

Bjarne Stroustrup : C++, comme la plupart des langages que j'aime le mieux, a commencé avec un besoin, un problème. Je ne voulais pas particulièrement concevoir un langage de programmation. Je voulais construire un système, mais j'ai décidé que je ne pouvais pas construire ce système - une version d'Unix pour fonctionner comme un système distribué - avec les langages existants. J'avais besoin d'un langage capable de faire des choses de bas niveau comme des gestionnaires de mémoire, des pilotes de périphériques et des planificateurs de processus. J'avais aussi besoin d'un langage qui pouvait faire des choses de haut niveau comme spécifier les composants d'un programme et comment ils communiquaient. Aucun langage ne pouvait faire les deux, j'ai donc construit C++. Je n'ai jamais pu construire mon système distribué, mais d'autres ont construit de tels systèmes en C++.

Nous sommes une communauté de plusieurs millions de développeurs C++, en Amérique du Nord, en Australie, en Europe, en Asie. Nous avons traversé les époques et nous ne sommes pas là par hasard. Nous sommes des survivants. Même quand Java est sorti, nous avons survécu et ce sans marketing, sans organe de presse.

C++ doit être bon dans l'abstraction et dans la génération du code, le code bas-niveau, bon dans la gestion du hardware. Il permet de faire aussi bien du C que des templates. C++ n'est pas un full langage OOP au sens strict du terme, cependant il comporte de nombreuses fonctionnalités qui en font un bon candidat.

Les MFC n'utilisaient pas la STL. Dans mon apprentissage, j'ai appris le C++ puis la STL. Dans les MFC, il y a avait des macros car le compilateur ne supportait toutes les fonctionnailités

des templates. Comment avez-vous réagi quand vous avez vu la première version de la STL écrite par Stepanov pour HP ?

Quand j'ai vu la STL pour la première fois, nous avons discuté et j'ai trouvé cela bizarre. Puis, j'ai fait le compte de tout ce qui était utilisé comme fonctionnalités. Tout y était utilisé. Les templates sont complexes mais d'une puissance formidable. Au fur et à mesure que je découvrais son travail, je lui ai demandé de proposer la STL au comité ISO.

Êtes-vous surpris par le succès du C++ ?

Bjarne Stroustrup : Oui définitivement. J'étais - et je suis toujours - assez étonné. Il n'y a jamais eu un moment où C++ n'ait pas eu plusieurs concurrents mieux financés et fortement commercialisés. De nombreux aspects de la conception n'étaient pas à la mode, juste des réponses à des défis réels. Pourtant, la communauté C++ a grandi et continue de croître. C++ répond à des besoins réels.

C++ est une norme ISO. Est-ce important qu'il le soit ? Est-ce une garantie de maintenir un écosystème et d'assurer le fonctionnement sur plusieurs plateformes ?

Bjarne Stroustrup : Pour C++, le comité de normalisation ISO (WG21) est essentiel. Nous n'avons pas de riche propriétaire, et, pour collaborer, les organisations concurrentes ont besoin d'un forum ouvert comme le WG21. Où des personnes d'organisations concurrentes peuvent-elles s'asseoir et résoudre ensemble des problèmes si ce n'est pas à la WG21 ? Avoir un comité des normes ne garantit rien. Seule la qualité de la norme donne le succès. Il est crucial que la norme soit basée sur le consensus, afin que tous les fournisseurs de compilateurs et de bibliothèques de normes trouvent utile de s'y conformer. A Prague, le vote pour le C++ 20 était de 79-0 en sa faveur! Cela signifie que C++ 20 deviendra officiel à l'automne. La plupart des parties de C++ 20 sont déjà livrées dans plusieurs compilateurs et une grande partie est déjà utilisée en production. C++ 20 n'est donc pas de la science-fiction.

Cette idée d'associer le C++ à un comité ISO vous est venue comment ?

Je n'ai pas eu l'idée. IBM et HP m'ont dit de faire cela, car ils ne faisaient pas confiance à mon employeur (AT&T). Il fallait donc que je fasse cet effort de standardisation, car ils avaient des milliards en investissement sur C++ ; je n'avais pas le choix. Au début je ne savais pas trop ; le langage n'était pas complètement terminé en termes de design. Cela prend des années pour concevoir un langage. Puis je me suis rendu compte que c'était une bonne idée.

Le C++ a évolué avec de nombreuses fonctionnalités, est-ce une bonne chose ?

On pioche dans différents langages et il y a du fonctionnel, du procédural, de l'orienté objet, etc. Il n'y a pas de langage parfait et unique. On peut toujours programmer comme le C++ à la Papa et aussi avec le style Moderne.

Il est difficile d'apporter des améliorations à un langage de programmation. Nous l'avons vu avec Sun et Oracle pour Java. Le cas Kotlin faisant courir le risque de casser la compatibilité. Quel est votre sentiment avec C++ ?

Bjarne Stroustrup : La stabilité est une caractéristique importante, en particulier pour les personnes qui construisent des systèmes qui doivent durer des décennies. Le C++ est devenu un langage beaucoup plus expressif et performant que ce que j'avais dans les années 80, mais nous avons fait très attention à maintenir un grand degré de compatibilité. Tout le monde veut un langage plus petit et plus simple. Ils veulent également plus de fonctionnalités. De plus, la plupart des développeurs C++ insistent pour que leur ancien code ne soit pas cassé. Il est impossible de livrer les trois. J'essaie de maintenir la compatibilité tout en ajoutant des moyens de simplifier l'utilisation de C++.

C++ a évolué au fil des ans avec une croissance rapide au cours de la dernière décennie. C'est une bonne chose ?

Bjarne Stroustrup : Oui. De nouveaux défis émergent et de nouvelles idées sont nécessaires pour les relever. Une langue change pour s'adapter à un monde en évolution. Cela est vrai pour le langage naturel et artificiel. De plus, j'ai toujours su que le C++ n'était pas encore le mieux adapté à mes idéaux. Par exemple, en 1981, j'ai écrit qu'une programmation générique était nécessaire. J'ai supposé que les macros pouvaient être utilisées pour répondre à ce besoin, mais cela ne s'est pas mis à l'échelle, j'ai donc implémenté des modèles. Cependant, les modèles ne rencontraient qu'une partie de mes idéaux pour la programmation générique. Ce n'est qu'avec des «concepts» que nous nous rapprochons.

Un autre exemple est la concurrence et le parallélisme où nous avons dû normaliser un niveau de support de threads et de verrous fortement typé. En tant que langage de programmation de systèmes, C++ devait fournir un support direct à ce que les systèmes d'exploitation offraient. Ensuite, nous devons offrir un meilleur support standard pour la programmation sans verrouillage, et ce n'est que plus tard que nous pourrons aborder les modèles de plus haut niveau d'algorithmes simultanés et parallèles. C++ 20 fournit des algorithmes parallèles et C++ 23 est susceptible de fournir un modèle général de concurrence.

Au cours de l'année, C++ a également fourni de nombreuses fonctionnalités pour simplifier la programmation. «Simplifiez les choses simples!» est un commentaire courant. Dans le C++ moderne, nous devons écrire beaucoup moins de code pour gérer les ressources, pour exprimer des algorithmes généraux, pour écrire des boucles simples, etc., que dans les anciennes versions de C++. Nous ne pouvons pas simplifier le langage sans casser le code, mais nous pouvons simplifier l'utilisation du langage.

Vous avez reçu de nombreuses récompenses et divers honneurs. Quelle est la récompense dont vous êtes le plus fier ?

Bjarne Stroustrup : Le prix Charles Stark Draper de la US National Academy of Engineering (https://en.wikipedia.org/wiki/Charles_Stark_Draper_Prize). C'est l'une des plus hautes distinctions honorifiques au monde

pour un ingénieur. Il a été fondé pour compenser le fait qu'il n'y a pas de prix Nobel d'ingénierie. Très peu d'informaticiens ont reçu «The Draper». C'est pour tous les domaines de l'ingénierie. Les ingénieurs ont toujours apprécié mon travail et je suis fier d'avoir fait quelque chose d'utile à grande échelle.

Pouvez-vous nous donner votre sentiment lorsque les jeunes développeurs considèrent le C++ comme un vieux matériel hérité par rapport à Rust, Kotlin ?

Bjarne Stroustrup : Tout le monde aime la nouveauté et nous aimerions tous ignorer les complexités désordonnées du monde réel, dont beaucoup sont liées au travail passé. Je suis quelque peu amusé quand quelqu'un m'accuse d'emprunter quelque chose à Rust que j'ai inventé des décennies plus tôt. Chaque langue réussie deviendra «héritée» et devra faire face à des idées plus anciennes et à un code plus ancien. L'alternative est l'échec.

Quelles sont les fonctionnalités les plus importantes pour vous ?

Bjarne Stroustrup : La fonctionnalité unique la plus importante en C++ est la classe avec des constructeurs et des destructeurs pour gérer la gestion des ressources. C'est une fonctionnalité qui distingue C++ de la plupart des autres langages et elle est fondamentale pour la programmation C++. Après cela, je vais pointer vers des modèles avec des concepts pour prendre en charge la programmation générique. C'est généralement une erreur de se concentrer sur les fonctionnalités de chaque langue. Je caractérise souvent le C++ comme ceci :

- Un système de type statique avec un support égal pour les types intégrés et les types définis par l'utilisateur ;
- Sémantique de valeur et de référence ;
- Gestion systématique et générale des ressources (RAII) ;
- Prise en charge d'une programmation orientée objet efficace ;
- Prise en charge d'une programmation générique flexible et efficace ;
- Prise en charge de la programmation au moment de la compilation ;
- Utilisation directe des ressources de la machine et du système d'exploitation ;
- Prise en charge de la concurrence par le biais de bibliothèques (si nécessaire implémentée en utilisant intrinsèques).

C'est pour C++, bien sûr. D'autres langages sont conçus pour différents domaines d'applications et différentes populations de développeurs, ils ont donc des critères de conception différents.

SECONDE PARTIE DE L'INTERVIEW

Au-delà des questions génériques, nous avons posé des questions plus spécifiques sur le langage et les fondations de C++.

La sémantique de déplacement est-elle une bonne chose (et qu'en est-il du flou &&) ?

Bjarne Stroustrup : Déplacer la sémantique est une très bonne chose dans la mesure où elle complète le modèle de gestion des ressources de C++ en permettant aux ressources d'être déplacées d'une étendue à l'autre, avec peu ou pas de surcharge, et sans tripoter les pointeurs ou la gestion explicite des ressources. Cela simplifie considérablement le code et complète les développements commencés par l'optimisation de la valeur de retour (que j'ai implémentée en 1984). J'utilise presque exclusivement la sémantique de déplacement implicitement dans les instructions de retour où elle est sûre et implicite une fois que vous avez défini les constructeurs de déplacement et les affectations de déplacement. Je suis très méfiant à propos de l'utilisation explicite de && et std :: move () car une telle utilisation est sujette aux erreurs et généralement inutile. Toujours au cœur de C++; exprimez ce que vous pensez!

Avons-nous besoin d'améliorations des classes ?

Bjarne Stroustrup : Les classes avec constructeurs et destructeurs (si nécessaire) sont au cœur de C++. Je pense qu'ils sont assez bons et n'ont pas besoin d'améliorations significatives.

Pouvons-nous nous présenter les concepts de C++ 20 ?

Bjarne Stroustrup : Quand j'ai conçu les modèles, je voulais trois choses :

- Généralités - pour permettre aux gens de faire plus que ce que j'imaginais ;
- Frais généraux zéro ;
- Interfaces précisément spécifiées.

Je ne savais pas comment faire les trois en même temps, donc nous n'avons eu que les deux premiers. Cependant, les deux premiers ont suffi pour que les modèles deviennent un succès majeur. Ils ont en revanche également causé beaucoup de problèmes et de confusion. «Concepts» nous donne ce dernier point : des interfaces spécifiées avec

précision sans surcharges d'exécution ni restrictions sur ce que vous pouvez exprimer. C'est simplement une logique de prédicat au moment de la compilation sur les propriétés des types et des valeurs.
Enfin, nous pouvons dire juste : trier (v); et vous obtenez un message d'erreur décent si v n'est pas triable. Nous pourrions déclarer sort () comme ceci :
void sort (sortable_range auto &);
Cette auto est redondante, mais de nombreux membres du comité des normes ont estimé que le laisser de côté serait source de confusion, car ils ne seraient pas en mesure de voir que sort () était un modèle.

Les coroutines sont au cœur de C++ 20, que faut-il en retenir ?
Bjarne Stroustrup : Les coroutines faisaient partie de la conception originale de C++. De plus, la toute première bibliothèque C++ (lorsque C++ était encore appelée «C avec classes») fournissait des coroutines. Je suis très content de les revoir. Les coroutines simplifient de nombreuses formes de code autrement compliquées en permettant la sauvegarde automatique de l'état d'un calcul entre les calculs. La représentation d'exécution des coroutines C++ 20 est très petite et efficace de sorte qu'elles peuvent être utilisées pour gérer des centaines de milliers de calculs asynchrones dans un seul programme. Ils sont déjà largement utilisés sur Facebook et Microsoft.

Avons-nous besoin d'améliorations sur les classes dérivées?
Bjarne Stroustrup : Non, je ne pense pas que nous devons ajouter aux mécanismes de classes dérivées.

Quand le constructeur fait appel à Close() alors que vous l'aviez oublié…. Qu'en pensez-vous ?
Bjarne Stroustrup : Probablement ma fonctionnalité C++ préférée. C'est la clé de la gestion générale des ressources. La gestion de la mémoire ne suffit pas. Nous devons gérer les ressources autres que la mémoire telles que les descripteurs de fichiers, les verrous et les connexions à la base de données.

Je ne suis pas un gars d'exception mais j'aime tout attraper… (…) Et les exceptions ?
Bjarne Stroustrup : Eh bien, je suis "un gars d'exception" et ça ne me dérange pas "d'attraper (...)". L'exception utilisée avec RAII simplifie énormément le code et élimine des classes entières d'erreurs à des coûts très mineurs. Parfois, il fournit des accélérations par rapport au code jonché de tests de codes d'erreur. Il empêche les flux de contrôle exceptionnels de compliquer le code source.
La plupart des problèmes avec les exceptions proviennent de personnes qui utilisent try-catch simplement comme une forme si-alors-sinon ou dans une base de code qui est un tel gâchis de pointeurs et de gestion manuelle des ressources que les exceptions deviennent une source d'erreurs et de coûts d'exécution. Une autre classe de problèmes survient dans les systèmes qui n'essaient pas de détecter les erreurs et dans les petits systèmes où toute forme de prise en charge à l'exécution peut évincer les fonctionnalités nécessaires.

for-range est assez cool. Dites-en nous plus.
Bjarne Stroustrup : Que dire de plus? Traverser une plage est l'une des actions les plus courantes dans notre code, nous devons donc avoir un moyen simple et relativement sûr de l'exprimer. La plupart des erreurs courantes avec les boucles C traditionnelles ne peuvent pas être faites avec range-for et il est plus facile à optimiser. Avec std :: span (), il élimine la plupart des erreurs de style de dépassement de tampon.

Les fonctions lambda, je les apprécie beaucoup pour les parties STL.
Bjarne Stroustrup : Oui. Ils sont devenus la clé des rappels de toutes sortes, y compris dans la spécification d'arguments pour des algorithmes paramétrés avec des prédicats et d'autres opérations. Les lambdas sont souvent plus rapides et plus pratiques que les alternatives. Notez qu'en C++, un lambda est simplement une notation pratique pour définir et instancier un objet fonction.

La surcharge de l'opérateur est assez bonne ; c'est merveilleux même. Est-ce indispensable ?
Bjarne Stroustrup : En effet. Je ne pourrais pas me passer de surcharge (pour la fonction et les opérateurs). Elles sont l'une des clés de la programmation générique et de nombreuses formes de code élégant.

Public, privé et protégé : la séparation entre les choses peut être merveilleuse, et parfois friend (un ami) vient à la rescousse !
Bjarne Stroustrup : En effet, même si je ne trouve pas beaucoup d'utilisations pour «protégé» de nos jours.

Avec l'expérience, je vois les concepts comme une pure beauté de l'abstraction, avec la fonction de spécialisation. Avons-nous besoin d'améliorations des concepts?
Bjarne Stroustrup : Nous devons polir quelques coins difficiles sur l'utilisation des contrats, mais sinon je pense que nous sommes bien pour l'instant. Par exemple, j'aimerais pouvoir contraindre les arguments de modèles dans les définitions de variables :
paire <intégrale, dérivée_de <Forme> *>
p = {27, nouveau cercle {p, 30}};
Avec des concepts, beaucoup de nos conceptions deviendront plus simples et plus faciles à utiliser. Si nous n'avons pas besoin de contraindre, nous pouvons simplifier :
paire p = {27, nouveau cercle {p, 30}}; //
p devient une paire <int, Circle *>
Nous avons la déduction d'argument modèle depuis C++ 17.

Même chose pour la notion de virtuel. Je ne pourrais plus m'en passer en programmation. Vous avez la même approche ?
Bjarne Stroustrup : Et bien souvent, nous le pouvons. Les hiérarchies de classe et les fonctions virtuelles sont excellentes, mais quelque peu surutilisées.

A New-York, vous étiez à l'affiche de 66 Minutes avec le titre « The engine of everything ». Qu'est ce que cela vous fait de savoir que C++ est si populaire depuis 40 ans et ce sans publicité, sans organe de presse ni marketing ?
Bjarne Stroustrup : Évidemment, je me sens heureux quand je vois que le C++ est si largement utilisé pour bien faire. J'ai aussi un peu peur parce que c'est une grande responsabilité. C++ est vraiment un peu partout. La plupart du temps, il est invisible dans le cadre d'un système dont nous

dépendons, par exemple notre téléphone, notre téléviseur, nos films et nos jeux, notre voiture, notre banque, notre appareil photo, la ferme qui produit notre nourriture. Bien que ce soit un honneur et non une publicité pour C++, me voir sur un écran vidéo de trois étages à Times Square était un peu effrayant.

Quand les techniciens pensent au C++, ils pensent au C et aux pointeurs. Quand ils se rendent compte que les pointeurs intelligents peuvent faire le travail, avec RAII, ils sont sans voix. Tout ce qu'ils ont appris avec Java et NET est inutile.

Bjarne Stroustrup : Eh bien, ces autres langages ont leur place, mais oui, il est triste que beaucoup aient une vision complètement déformée du C++. J'ai écrit «A Tour of C++» pour aider les gens à se faire une idée du C++ moderne. Il a l'avantage évident d'être mince. Si vous avez déjà maîtrisé la programmation, vous pouvez la lire sur un week-end.

Quel est votre sentiment avec ces langages soi-disant productifs comme Java et .NET / C #?

Bjarne Stroustrup : Je ne fais pas de comparaisons. Ils sont difficiles à bien utiliser. Je pense que beaucoup gagneraient à apprendre le C++ moderne. Malheureusement, une grande partie de l'enseignement du C++ est bloquée dans les années 1980. Le comité des normes C++ (WG21) dispose désormais d'un groupe d'étude sur l'éducation, qui vise à fournir des directives d'enseignement facilement accessibles.

Comment avez-vous prévu d'écrire les lignes directrices Core C++ avec Herb Sutter sur isocpp github ? Pouvez-vous nous donner l'aspect sous-scène de cela ? Ce pourrait être un livre d'Addison Wesley C++ Series mais c'est gratuit. Était-ce voulu ?

Bjarne Stroustrup : Je travaillais sur un ensemble de lignes directrices pour Morgan Stanley basé sur les sections de conseils dans mes livres (par exemple, «A Tour of C++») dans le but de les rendre largement disponibles. Il m'est venu à l'esprit que je ne pouvais pas être le seul à faire ça. Après tout, C++ 11 et C++ 14 amenaient beaucoup de nouvelles personnes au C++ moderne et ils avaient besoin de lignes directrices. J'ai donc demandé autour de moi et Herb avait commencé un travail similaire chez Microsoft. Évidemment, pour pouvoir collaborer librement et partager notre travail en temps opportun, nous en avons fait un projet open source. Les directives et la petite bibliothèque de support (GSL) sont sur Github:

- https://github.com/isocpp/CppCoreGuidelines
- https://github.com/microsoft/gsl

Le support d'analyse statique (essentiel) est principalement dans Visual Studio avec un peu de support dans Clang Tidy. C'est un projet ambitieux. Pour commencer, nous voulons garantir un style de C++ qui soit sûr pour les types et pour les ressources. Au-delà de cela, nous aimerions éliminer l'utilisation complexe inutile et les problèmes de performances courants. Les Core Guidelines sont une autre bonne façon de voir le C++ moderne. Mais ne lisez pas tout d'un coup. Il est destiné à être utilisé avec un analyseur statique, mais l'introduction peut être très utile et les directives individuelles peuvent être utilisées comme une FAQ. De toute évidence, les gens sont encouragés à contribuer. Ce n'est pas seulement Herb et moi. Nous avons de nombreux contributeurs et plusieurs éditeurs réguliers.

Certaines personnes en marketing ont dit il y a quelques années que le C++ était «dangereux et non sécurisé». Je leur explique que les pointeurs intelligents, auto, nullptr, vector, string font le travail et ce n'est pas logique d'émettre une telle idée. Quelle est votre opinion sur ce point ?

Bjarne Stroustrup : La sûreté et la sécurité sont des propriétés du système. Certaines personnes semblent obsédées par la possibilité d'insécurités linguistiques, mais chaque langue capable de manipuler directement le matériel en aura. Tout langage pouvant utiliser directement l'API des systèmes d'exploitation en possède. Si je voulais pénétrer dans un ordinateur, je ne commencerais pas à jouer avec C++, il y a des moyens plus faciles. Cela dit, la plupart des choses dont les gens se plaignent peuvent être évitées dans le C++ moderne ; il suffit de regarder les directives de base C++ (et de les appliquer). La fonctionnalité que vous mentionnez fait partie des fonctionnalités qui vous aident.

C++ est un diamant. Pour les personnes qui ont besoin d'allouer de la mémoire dynamique, il convient, pour les personnes qui ont besoin d'une mémoire constante, il convient, pour créer toutes sortes de logiciels. Avec le succès de l'outillage natif et spécialement du backend LLVM ou GCC, une série de langages adoptent le style natif car c'est celui qui fonctionne bien et qui fonctionne le mieux. Que ressentez-vous avec l'écosystème des langages de programmation natifs ?

Bjarne Stroustrup : Difficile à dire car il n'y a pas qu'un seul écosystème de ce type. C'est vraiment agréable de voir tous ces nouveaux langages construits sur une infrastructure C++.

Certaines entreprises investissent dans Java, Kotlin, d'autres Rust, d'autres Go, d'autres Swift. Que pensez-vous de la cette diversité?

Bjarne Stroustrup : Il est bon de voir les développements dans les langages de programmation et les techniques de programmation. Je souhaite juste que plus de gens se rendent compte qu'il n'y a pas de langage qui soit le meilleure pour tout et pour tout le monde. Le mieux est qu'ils essaient d'apprendre le mieux possible et qu'ils profitent du meilleur de ce qu'ils connaissent dans les langages pour développer au mieux. Le C++ est souvent «celui à battre». Il y a une raison à cela : C++ est suffisamment bon, efficace et stable pour beaucoup de choses. Tout langage qui survit pendant quelques décennies aura certains des problèmes de C++.

Tout sur mon PC portable est fait en C / C++ : Windows, Sysinternals Suite, Office (Word, Excel, PowerPoint, Outlook), Chrome, VLC, Winamp, Gimp, Acrobat Reader, SQL Server, mes jeux. TOUT. Y a-t-il de meilleures réalisations? C++ domine le monde, non ?

Bjarne Stroustrup : Personnellement, ce dont je suis le plus fier, c'est du rôle du C++ en science et en ingénierie… Pensez au CERN, le projet du génome humain, les Mars Rovers. De plus, C++ se propage dans de nouveaux domaines d'application. Par exemple, grâce à TensorFlow, c'est le fondement de la plupart des IA / ML.

François Tonic

Visual Basic : VB est mort, vive C# (et les autres langages)

Le 11 mars dernier, l'équipe .Net de Microsoft a officialisé ce que de nombreux développeurs attendaient depuis bien longtemps. Le sort de Visual Basic ne faisait guère de doute. Le message officiel est clair : « nous ne prévoyons pas de faire évoluer Visual Basic en tant que langage ». R.I.P. Visual Basic.

Microsoft prépare activement l'arrivée de .Net 5 pour cet automne. Rappelons que .Net 5 est le nom officiel de .Net vNext. Il s'agit de .Net Core avec comme objectif de fournir un seul runtime et framework. Cela permettra de remettre un peu d'ordre dans le monde .Net qui était tout sauf clair avec la multiplication des éditions…

Support mais plus d'évolutions prévues

L'équipe .Net est très claire :

- Oui Visual Basic sera supporté par .Net 5 pour assurer la compatibilité,
- Non il n'y pas d'évolution prévue du langage à l'avenir.

La non évolution du langage signifie que toutes les nouveautés et améliorations de .Net ne seront pas implémentées dans VB. Or, comme le dit le post officiel : "Les futures fonctionnalités de .Net Core requièrent des changements dans le langage qui pourront ne pas être supportés par Visual Studio". Bref, le langage deviendra obsolète par défaut. Jusqu'à présent, VB recevait les évolutions du framework. Vous pouvez continuer à utiliser VB et uniquement .Net Framework si cela vous convient. Attention aussi, certaines technologies ne sont pas supportées par .Net Core comme WebForms, Workflow ou WCF. Si vous souhaitez continuer à supporter les évolutions de .Net, il faudra porter / migrer ces technologies non standard.

L'objectif du support de VB dans .Net 5 n'est pas d'inciter de nouveaux projets mais uniquement d'assurer la compatibilité avec les apps et les codes legacy.

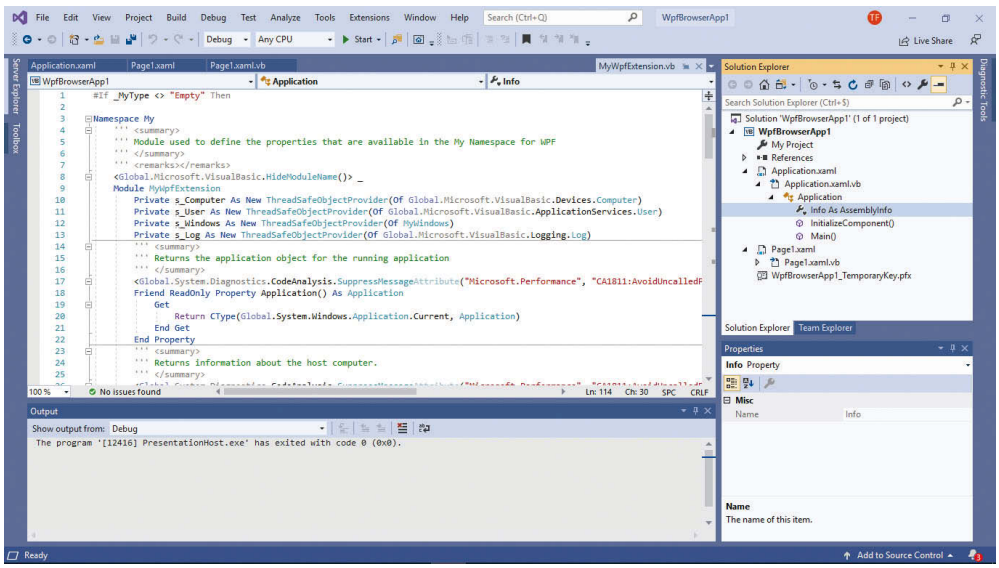
.Net 5 supporte les apps VB suivantes :

- Class Library ;
- App console ;
- Windows Forms ;
- WPF ;
- Worker Service ;
- ASP.Net Core Web API.

Migration

Visual Studio permet de migrer les vieux codes VB 6 et VB.Net. Des outils tiers permettent aussi de réaliser cette migration, comme par exemple : Telerik Code Converter. Sauf si vos vieux codes sont absolument nécessaires pour certains usages ou des matériels anciens, pensez à migrer vos codes ou à les supprimer au fur et à mesure. Attention : certaines couches techniques « anciennes » nécessitent un portage vers des technos récentes donc prudence si ces projets sont importants.

Quid de Visual Basic for Applications (VBA) ? VBA est un cas à part. Il ne sera pas impacté pour cette annonce.



Petite histoire de VISUAL BASIC

1988 : rachat de Tripod avec un prototype d'outils drag&drop créé par Alan Cooper, le père de VB.

Mars 91 : QuickBasic et le prototype Ruby (de Tripod) fusionnent pour lancer un nouveau projet : Thunder. Sortie de VB 1.0.

Septembre 92 : sortie de Visual Basic pour DOS !

Novembre 92 : VB 2 est disponible avec support d'ODBC Level 1, formulaires MDI, etc.

Juin 93 : Visual Basic 3.0 sort. Enorme succès.

Octobre 96 : VB 4

Avril 97 : VB 5

Octobre 98 : VB 6

Février 2002 : première version de Visual Basic .Net.

Novembre 2007 : apparition de l'édition Visual Basic .Net 2008 Express.

Mars 2020 : Visual Basic ne devrait plus évoluer.

Olivier Bouzereau
Journaliste et coordinateur de projets collaboratifs OW2

Partie 2

DevOps : encore et toujours ?

© Metroworks

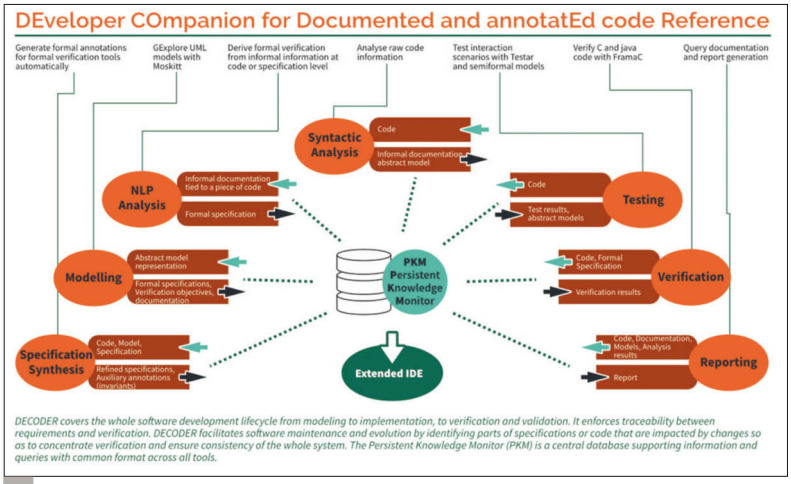
Comprendre les codes externes, maîtriser les dépendances

Pour révéler une partie des futures pratiques DevOps, cet article retrace l'enjeu et les objectifs de deux projets de recherche et d'innovation en cours : Decoder approfondit l'analyse des codes sources, Fasten vérifie les dépendances au niveau des fonctions. Il évoque les premiers résultats obtenus après un an de coopération, tous deux ayant démarré en janvier 2019 et étant financés par la Commission Européenne durant trois ans, jusqu'en décembre 2021.

Dans un premier temps, l'article décrit ***Decoder*** qui cherche à comprendre la finalité des codes sources et leurs propriétés dans le but d'améliorer l'efficacité du développement, des tests et de la maintenance. Considérant le code comme une forme de langage naturel, il s'appuie sur des outils NLP (Natural Language Processing) et des modèles semi-formels abstraits. Qui participe ? Sept entités dont l'institut CEA LIST, leader technique, l'UPV (Université Polytechnique de Valence), Cap Gemini, Sysgo, Tree Technology, l'association open source OW2 et Technikon, coordinateur du projet.

Dans un second temps, le projet connexe ***Fasten*** prépare une gestion intelligente des dépendances pour devenir un véritable logiciel d'aide à la décision focalisé sur la mise à jour des programmes. Le projet est porté par une équipe de chercheurs de l'Université hollandaise TU Delft associée aux universitaires de Milan et d'Athènes, aux ingénieurs des PME Endocode, SIG, et XWiki et de l'association OW2.

Ces deux projets génèrent des programmes open source, implémentés et vérifiés par plusieurs cas d'usage, afin d'aboutir à une réutilisation la plus large possible. Ils feront l'objet d'une campagne de bêta-test dans les mois à venir. Dès lors, toute équipe DevOps pourra fournir son retour d'expérience et contribuer à l'amélioration des outils.



1 *Le projet Decoder veut couvrir tout le cycle de développement d'applications, de la modélisation à la validation, en passant par l'implémentation et les vérifications.*

Un moniteur de connaissances persistant

Decoder est l'acronyme de "compagnon du développeur pour une référence au code documentée et annotée"- en anglais DEveloper COmpanion for Documented and annotatEd code Reference. Le composant central de l'architecture ***Decoder*** n'est autre qu'une base de données persistante, baptisée PKM (Persistent Knowledge Monitor). Alimenté par de nombreux outils distincts, ce moniteur de connaissances est chargé de répondre aux requêtes émanant de chaque membre de l'équipe DevOps, ou de tout autre participant actif durant le cycle de vie de l'application (architecte, testeur, mainteneur, etc.).

Certains outils sont développés à partir de rien tandis que d'autres résultent d'extensions d'outils préexistants. La modélisation est prise en charge par l'outil Moskitt d'UPV qui traite les modèles UML. Des outils de requêtage fournissent les requêtes avancées pour naviguer dans les API tierces. Côté vérification, l'inspection de codes passe par les outils FramaC (langage C) et OpenJML (Java). Le projet Testar génère des scenarii d'interaction pour les tests. Enfin, l'édition de rapports et documents pourra confirmer si un code contient bien les propriétés prévues au stade de la modélisation.

Plusieurs outils NLP sont chargés d'extraire des informations de la documentation informelle, à partir de rapports de bugs, de commentaires ou de forums de développeurs. Ce qui compte ici, c'est la possibilité de relier ces informations non structurées en langage naturel à des informations plus structurées au niveau du code ou même au niveau des spécifications, en particulier pour faire une vérification formelle. La synthèse des spécifications, pour sa part, est destinée à générer des annotations formelles pour aider les outils d'analyse comme FramaC à effectuer une activité de vérification. Le consortium du projet construit la base de données et établit un format commun que tous ces outils peuvent comprendre pour communiquer aisément avec le moniteur PKM, soit pour y écrire, soit pour en extraire des informations.

Les outils ***Decoder*** ont un impact lors de chaque phase de développement du logiciel, maintenance et évolutions de code comprises. Ils interviennent dès la définition du modèle, puis lors de sa mise en œuvre grâce à l'IDE augmenté, et enfin au niveau des vérifications et de la validation (figure 1).

Pour soutenir la maintenance et l'évolution des logiciels, ils vont retracer la partie du code ou de la spécification qui est affectée par un changement récent, afin que les développeurs puissent se concentrer sur les modifications nécessaires. Parallèlement, le moniteur PKM joue un rôle central, toutes les informations disponibles sur le code y étant stockées.

Comprendre l'évolution du code à chaque étape

Reste à combler le fossé entre les documents informels - rédigés en anglais, en français ou dans toute autre langue - et les documents formels tels que le code et les spécifications formelles lorsqu'elles existent. Le projet ***Decoder*** souhaite travailler dans les deux sens ; il veut extraire des informations de phrases en langage naturel, établir une correspondance avec les parties pertinentes du code ou des spécifications, et générer des spécifications formelles à partir d'informations informelles.

2 *L'équipe DevOps pourra visualiser tous les impacts d'une évolution de code.*

En outre, il veut extraire des informations du code ou des spécifications formelles pour les présenter dans un format plus informel ou plus facile à lire par l'homme, comme le ferait un générateur de documents semi-automatisé.

C'est dans ce cadre qu'interviennent les modèles semi-formels abstraits (ASFM). Ils apportent un langage naturel graphique pour décrire via un schéma ce qu'une fonction est censée faire. Face à une structure de données complexe, il est souvent plus efficace de raisonner à l'aide d'un diagramme. L'idée consiste à fournir des diagrammes ASFM utiles au débogage, chaque diagramme représentant ce qui se passe lors de l'exécution d'un fragment de code. Chaque membre de l'équipe DevOps peut ainsi visualiser tout ce qui change, après chaque évolution du code (figure 2).

Quatre cas d'usage pour améliorer les outils

Quatre cas d'utilisation vont évaluer les outils ***Decoder*** et démontrer leur utilité pour l'industrie. Le premier concerne l'analyse

de pilotes du système d'exploitation Linux ; il s'agit de comprendre rapidement si des pilotes développés par des tiers sont éligibles ou non à l'intégration au sein de systèmes embarqués plus ou moins critiques.
De son côté, OpenCV cherche à bâtir une connaissance de l'API de cette bibliothèque fréquemment utilisée et à mieux comprendre comment elle est utilisée dans les applications qui en dépendent.
MyThaiStar, le cas d'usage de Cap Gemini, se concentre sur la conception et la validation de l'interface utilisateur.
Enfin, le cas d'usage Java analyse divers projets de la base de code OW2. L'idée est d'évaluer l'ensemble des outils pour vérifier leur facilité d'utilisation par des personnes qui ne sont pas directement impliquées dans le projet ***Decoder***.

Un meta modèle PKM

Une partie importante du travail a porté sur la conception du moniteur PKM, avec un métamodèle décrit comme un ensemble de diagrammes UML pour mieux comprendre toutes les informations à placer dans le moniteur, et comment elles sont reliées les unes aux autres.
L'implémentation a démarré par la conception d'un serveur PKM. Le format de fichier ouvert Json (JavaScript Object Notation) est exploité comme canal principal d'échange d'informations entre le serveur PKM et les outils qui vont dialoguer avec lui.
Un premier schéma Json a été élaboré ; il est fondé sur le métamodèle UML déjà conçu. D'autres formats Json pourront traiter les résultats d'analyse statique (comme le format Sarif). Les membres du projet souhaitent réutiliser plusieurs technologies éprouvées pour produire des outils aussi interopérables que possible. Au niveau back-end, l'institut CEA LIST a exploré plusieurs gestionnaires de données orientés documents, dont MongoDB, CouchDB et OrientBD, ce dernier gérant une base de données multimodèle. L'outil Testar d'UPV fournit sa partie modèle graphique, mais pourrait également être exploité pour stocker des documents complets.

L'extraction d'informations

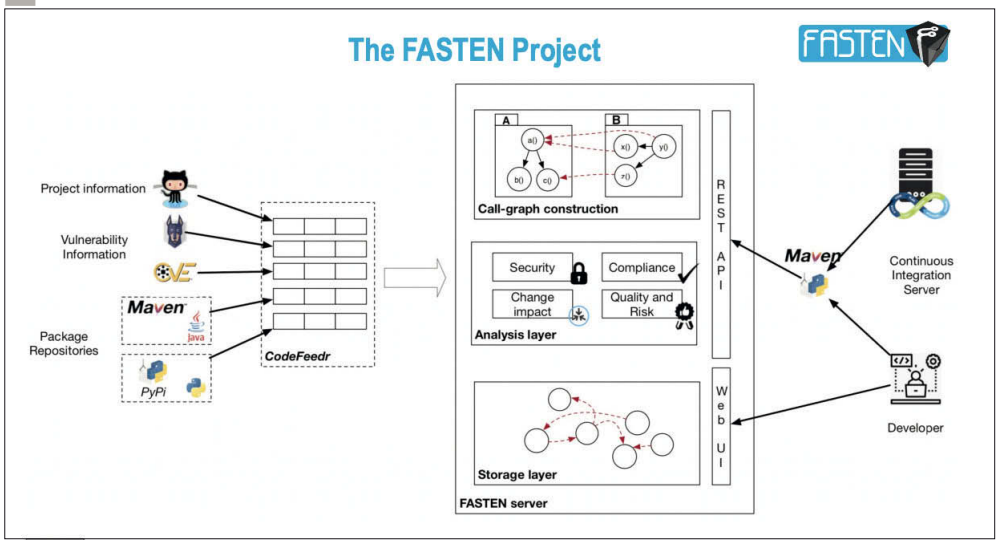
Afin d'améliorer la précision de l'outil, il faudra rassembler un ensemble de données pour la partie NLP du projet. L'équipe ***Decoder*** prévoit d'utiliser plusieurs jeux de données - regroupant du code et des documents associés - déjà disponibles via GitHub et dans les cas d'usage du projet. En outre, un corpus de formation du projet DeepAPI pourra également servir à entraîner les outils, notamment pour établir des correspondances entre des phrases en langage naturel et des séquences d'appels en code Java.
Ces ensembles de données rassemblés procurent suffisamment d'éléments pour commencer l'expérimentation dans les deux sens, du code au langage naturel, pour extraire les principales caractéristiques du code ; et dans l'autre sens, du langage naturel au code. La première expérience consiste à considérer les langages de programmation comme un type particulier de langue étrangère et à faire une traduction plus ou moins standard à l'aide de réseaux de neurones. Le but ? Voir comment relier la sémantique du code à son homologue en langage naturel pour obtenir une vue précise des similitudes entre les deux.

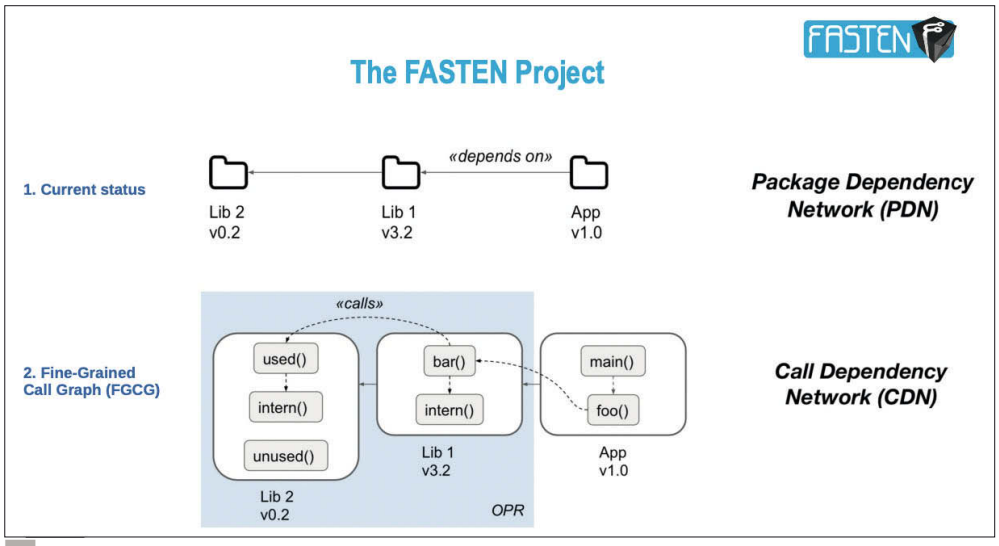
Un atelier sur l'expérience utilisateur

Dernier point saillant du travail accompli, Capgemini a organisé un atelier pour définir ce qui devrait être inclus dans le client PKM ainsi que le type de requêtes les plus intéressantes à adresser à la base de données. Cet atelier a permis de définir les rôles principaux que l'utilisateur du moniteur PKM remplirait en termes de développement logiciel, et d'avoir une première idée des scénarios d'usage et du type de contrôles à confier aux utilisateurs du logiciel client. Cette première étape intéressan-

Fasten propose de gérer les dépendances via un réseau de graphes d'appels de fonctions

3

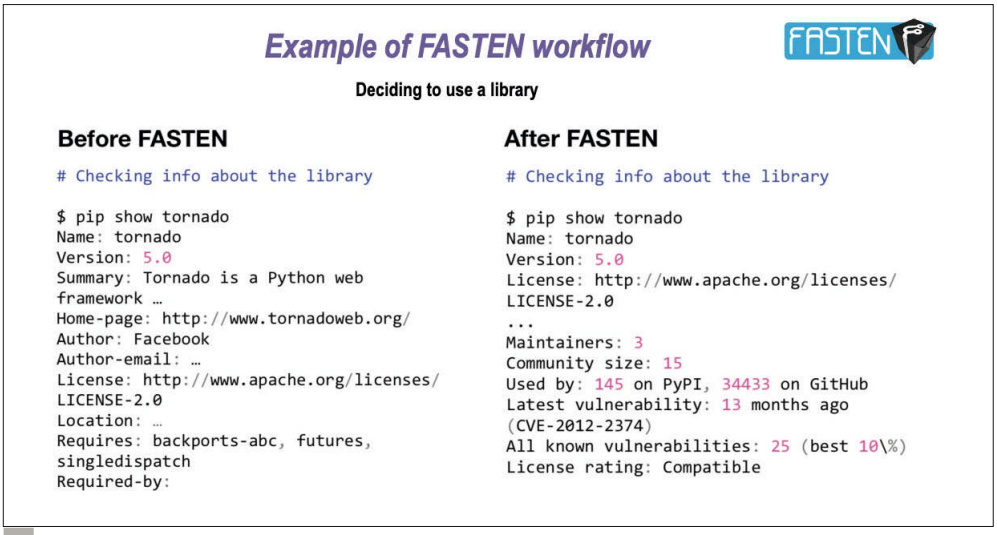




4 *Les outils Fasten procurent à l'équipe DevOps des analyses d'impact, d'usage, de conformité, de risques ou de qualité*

te doit être affinée. La mise en œuvre du moniteur PKM reste un défi important à relever, mais le projet semble sur la bonne voie. Le schéma est un élément important pour garantir que chaque outil dispose d'informations suffisantes pour ses besoins. Le choix final de la base de données est moins épineux en dépit de son impact sur les performances des réponses aux requêtes à venir. Changer le moteur sera moins difficile que changer le schéma, puisque des données Json seront échangées entre la base de données et ses clients.

Les travaux sur la langue graphique de l'ASFM ont démarré, de même que ceux en relation avec les cas d'usage, désormais bien définis.



5 *Fasten livre un ensemble d'informations pour choisir d'utiliser ou pas une bibliothèque de fonctions externes*

Fasten veut dompter les dépendances

Le second projet de recherche européen répond à une problématique quotidienne des équipes DevOps, concernant le packaging d'applications.

Avec leurs codes sources ouverts et accessibles à tous, les logiciels open source séduisent un nombre croissant de développeurs. Mais plusieurs risques doivent être identifiés et évalués préalablement. Lorsqu'un développeur exploite des outils tiers ou des bibliothèques de fonctions libres pour gagner du temps, il multiplie les dépendances. Plusieurs frameworks vont dépendre de bibliothèques externes et de composants qui évoluent séparément, sans coordination centralisée.

Plusieurs questions surgissent dès lors, par rapport aux fonctionnalités intégrées ; ces questions ont trait à la confiance, à la sécurité ou aux notifications fournies aux parties prenantes, à chaque évolution de code téléchargeable gratuitement via Internet.

Comment vérifier que les fonctions fournies correspondent toutes à ce que l'on attend, sans la moindre brèche de sécurité ? Et comment garantir qu'une mise à jour de dépendance n'empêchera pas l'application finale de fonctionner ?

Des conséquences coûteuses

En 2016, d'innombrables sites web ont été bloqués par le retrait de la bibliothèque *left-pad*, présente dans plusieurs frameworks JavaScript tels *node*, *babel* ou *react*. Autre exemple durant l'été 2017, la brèche d'Equifax – qui a provoqué une fuite massive de données confidentielles relatives aux paiements électroniques – a été estimée à 4 milliards de dollars, dont 750 millions en amendes réclamées par les clients américains, sans compter l'impact sur la réputation de l'entreprise canadienne.

L'équipe DevSecOps doit se prémunir de tels problèmes provoqués par une incompatibilité technique, ou juridique comme une licence open source incompatible. En particulier, les conséquences du moindre changement de code devraient être connues par l'équipe en charge du maintien en condition opérationnelle des applications et par l'équipe de cybersécurité IT, ce qui n'est pas le cas aujourd'hui.

Une meilleure façon de gérer les dépendances s'avère donc nécessaire et c'est justement l'objectif du projet ***Fasten*** (Fine-Grained Analysis of Software Ecosystems as Networks).

Une analyse des fonctions dépendantes

L'objectif de ***Fasten*** consiste à renforcer les écosystèmes de logiciels via un conditionnement plus intelligent des applications. L'implémentation de ses outils d'analyses est en cours de développement pour les langages Java, Python et C/C++.

Au cœur du projet, une base de connaissances regroupe des graphes d'appels agissant au niveau des fonctions des logiciels et non seulement au niveau des bibliothèques qui les contiennent. L'équipe DevOps peut accéder à cette base de connaissances via son gestionnaire de packages en place. Elle découvre ainsi quelles fonctions précises d'une bibliothèque ont des dépendances vers quelle autre fonction d'une autre bibliothèque, et ainsi de suite (figure 3).

Le réseau constitué à partir des graphes d'appels de dépendances permet d'obtenir de précieuses analyses d'impacts et également des analyses d'usages (figure 4). Un développeur peut ainsi savoir combien de clients seront réellement affectés par le dernier changement de son code. De même un exploitant sera-t-il rassuré avant de mettre à jour telle ou telle bibliothèque. Il saura aussi si un code sous licence libre GPL, par exemple, est en lien avec son application, via une quelconque dépendance.

L'équipe DevOps pourra donc vérifier les contraintes en relation avec l'usage de nouvelles dépendances, sur les plans technique et juridique, chaque licence apportant des droits, mais aussi des obligations.

En particulier, les ingénieurs sauront combien de mainteneurs sont impliqués dans l'évolution d'un code externe, combien de membres compte sa communauté, combien d'utilisateurs ont déjà mis en production une certaine bibliothèque et quelles sont les vulnérabilités connues (figure 5). De nombreuses autres variables pourront ainsi être collectées, au niveau des fonctions, puis combinées pour décider d'installer ou pas la dernière version d'un framework ou d'une bibliothèque.

Pour en savoir plus sur :

DECODER : www.decoder-project.eu

FASTEN : www.fasten-project.eu

Philippe Charrière
(pcharriere@gitlab.com),
Technical Account Manager
chez GitLab

Partie 2

Cueillette des champignons avec GitLab CI

(ou comment avoir du feedback sur son code source avec un runner)

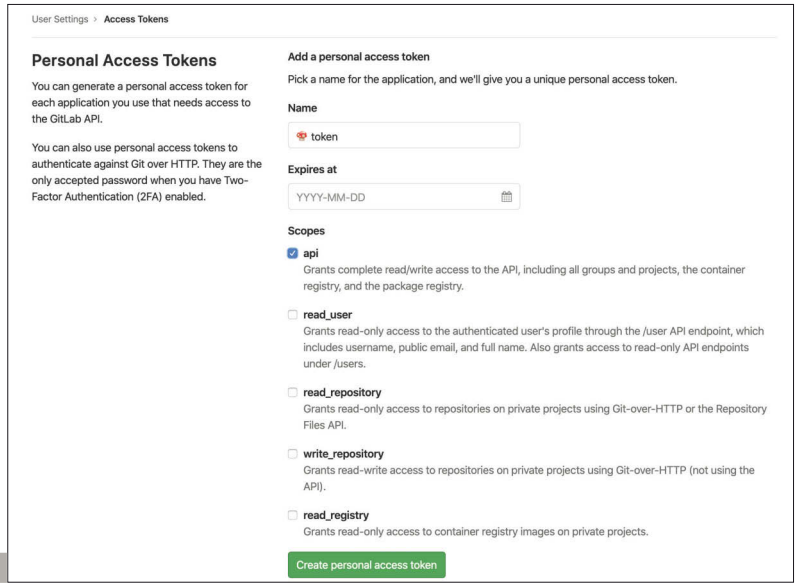
Dans cet article nous allons découvrir que l'on peut utiliser un GitLab runner installé sur son poste en local pour exécuter la CI d'un projet distant hébergé sur GitLab.com (donc sans utiliser les shared runners, mais vos propres ressources), nous verrons aussi comment utiliser ses propres outils dans une image Docker et se servir de l'API GitLab pour apporter du feedback au développeur sur d'éventuelles vulnérabilités présentes dans son code. Suite et fin.

Ajouter du feedback dans la merge request

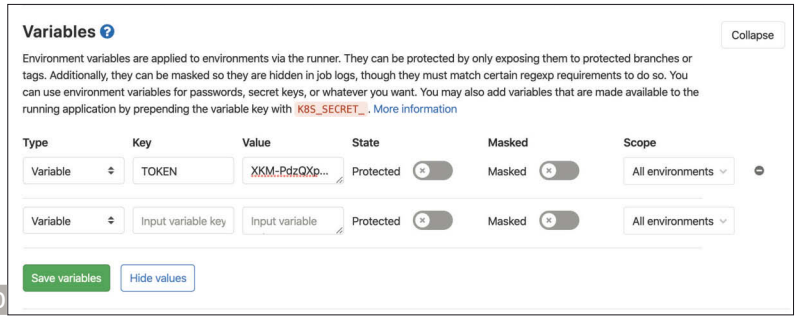
J'aimerais donc qu'à chaque fois que j'introduis "malencontreusement" des champignons dans mon code, le script de CI (qui est déclenché à chaque modification) ajoute une note dans la merge request.

Génération et utilisation d'un token

Pour cela nous allons devoir utiliser l'API GitLab (https://docs.gitlab.com/ee/api/notes.html#create-new-merge-request-note), et il nous faut donc un "**Personal Access Tokens**". Vous pouvez le créer dans Users Settings, sélectionnez uniquement le scope **api**, puis cliquez sur "**Create personal access token**" 9



9



10

Conservez (copiez) votre token quelque part et allez le renseigner dans les CI Variables de votre projet. Retournez dans les "**Settings > CI/CD**" de votre projet, et dans la rubrique "**Variables**" ajouter une variable TOKEN (champ **Key**) avec pour valeur le token que vous avez généré juste avant (champ **Value**): 10
Maintenant, nous pouvons modifier notre script de CI.

Allons modifier notre script de CI (.gitlab-ci.yml)

Ajoutons un helper pour utiliser l'API GitLab

Nous allons faire cela sur une nouvelle branche, donc avant toute chose, nous allons créer cette branche:

```
git checkout -b mr-mushroom-feedback
```

Et maintenant modifions le fichier **.gitlab-ci.yml** en ajoutant une nouvelle section:

```
.notes: &notes |

function addNoteToMergeRequest() {
  GITLAB_API_URL="https://gitlab.com/api/v4/projects"
  PROJECT_ID=$CI_MERGE_REQUEST_PROJECT_ID
  MR_IID=$CI_MERGE_REQUEST_IID
  curl -d "body=$1" --request POST --header "PRIVATE-TOKEN: $TOKEN" \
  $GITLAB_API_URL/$PROJECT_ID/merge_requests/$MR_IID/notes
}
```

J'ajoute donc une nouvelle section **.notes: &notes** avec une fonction **addNoteToMergeRequest** qui utilise l'API GitLab pour ajouter une note à une Merge Request.
La variable d'environnement **CI_MERGE_REQUEST_PROJECT_ID** (fournie par GitLab) permet de récupérer l'ID du projet de la MR en cours et **CI_MERGE_REQUEST_IID** (fournie par GitLab) c'est l'identifiant de la MR en cours.
J'ai pu obtenir cette dernière grâce au filtre de pipeline **only: merge_requests** qui fournit le contexte nécessaire pour obtenir **CI_MERGE_REQUEST_IID**.
J'ai donc tout ce qu'il faut pour utiliser l'API de note de GitLab (https://docs.gitlab.com/ee/api/notes.html#create-new-merge-request-note). Il suffit d'utiliser la fonction de cette manière pour ajouter une note à la MR en cours:

```
addNoteToMergeRequest " 👋 Hello World 🌍 "
```

Puis je mets à jour la section **before_script**:

```
before_script:
 - *notes
 - *find-mushrooms
```

Utiliser notre nouvel helper
Pour cela je modifie le job initial de la façon suivante:

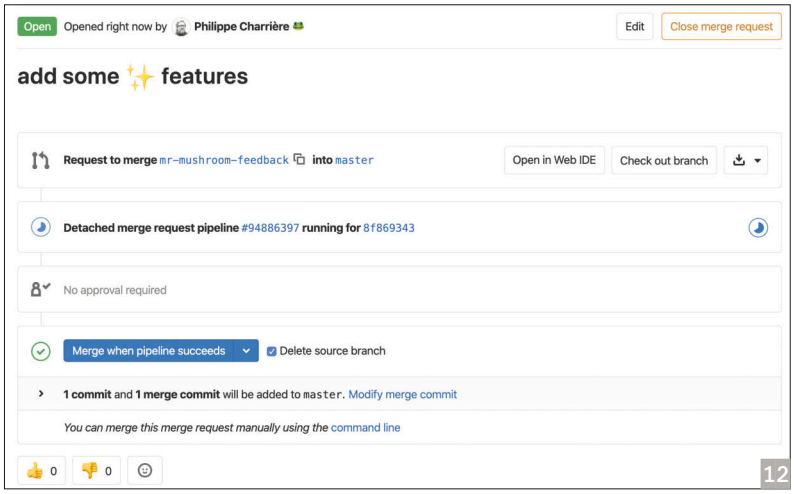
```
🍄 mushrooms-picking:
 stage: 🌲 forest-trip
 image: k33g/my-little-node-ci-tools:latest
 only:
  - merge_requests
 script: |
  mushrooms_markdown_report
  # Display the mushrooms report
  cat mushroom.report.md
  # Send the report to a note in the MR
  NOTE=$(cat mushroom.report.md)
  addNoteToMergeRequest "$NOTE"

 artifacts:
  paths:
  - mushroom.report.md
  expire_in: 1 week
```

- J'ai enlevé le filtre **only: master** (le rapport ne sera généré qu'avec une MR, et puis normalement vous n'êtes pas censé modifier directement la branche **master**.
- Ensuite je copie le contenu du rapport dans une variable: **NOTE=$(cat mushroom.report.md)**
- Pour enfin passer ce contenu en paramètre à ma fonction: **addNoteToMergeRequest "$NOTE"**

Le code final doit ressembler à ceci:

```
# Mushrooms Picking
stages:
 - 🌲 forest-trip

## My tools
.find-mushrooms: &find-mushrooms |

 function mushrooms_markdown_report() {
  RESULTS=$(grep -rwni " 🍄 " * || echo "") node --eval '
    let mushrooms = process.env.RESULTS.split("\n")
     .map(item => item.split(":"))
     .map(item => `- File: ${item[0]} (line ${item[1]}) Extract: \`${item[2].trim()}\``)
     .join("\n")
    let report = `### Mushrooms Picking \n${mushrooms}`
    console.log(report)
  ' > mushroom.report.md
 }

.notes: &notes |
```

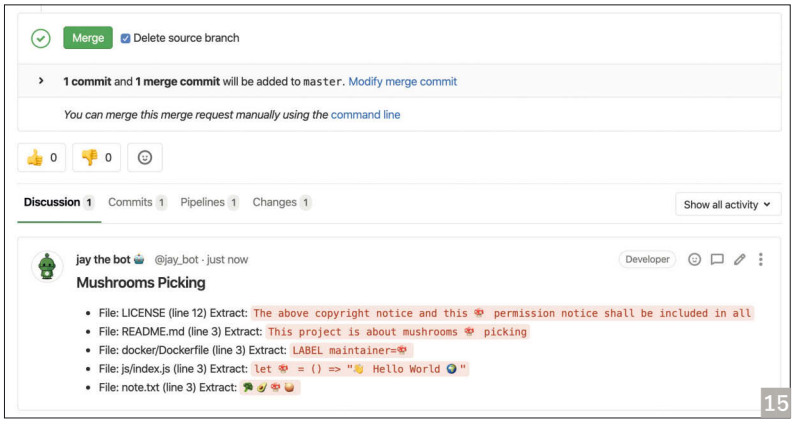
11



12

13

14



15

```
function addNoteToMergeRequest() {
  GITLAB_API_URL="https://gitlab.com/api/v4/projects"
  PROJECT_ID=$CI_MERGE_REQUEST_PROJECT_ID
  MR_IID=$CI_MERGE_REQUEST_IID
  curl -d "body=$1" --request POST --header "PRIVATE-TOKEN: $TOKEN" \
  $GITLAB_API_URL/$PROJECT_ID/merge_requests/$MR_IID/notes
}

before_script:
 - *notes
 - *find-mushrooms

🍄 mushrooms-picking:
 stage: 🌲 forest-trip
 image: k33g/my-little-node-ci-tools:latest
 only:
  - merge_requests
 script: |
  mushrooms_markdown_report
  # Display the mushrooms report
  cat mushroom.report.md
  # Send the report to a note in the MR
  NOTE=$(cat mushroom.report.md)
  addNoteToMergeRequest "$NOTE"

 artifacts:
  paths:
  - mushroom.report.md
  expire_in: 1 week
```

Il est maintenant temps de "pousser" notre code vers GitLab.com:

```
git add .
git commit -m "add some ✨ features"
git push --set-upstream origin mr-mushroom-feedback
```

La branche a été créée, vous pouvez donc créer une Merge Request en cliquant sur le bouton "**Merge Request**": 11

Puis la soumettre: 12

Et une fois le pipeline exécuté, vous allez voir apparaître une note avec la liste des "vulnérabilités 🍄 " dans la note: 13

Et si vous poussez un nouveau code avec de nouveau 🍄 vous verrez la progression dans une nouvelle note: 14

Et en utilisant un bot avec son propre token pour utiliser l'API c'est encore mieux : 15

Voilà, c'est tout pour aujourd'hui. Vous avez entre les mains de quoi améliorer le feedback que vous pouvez "envoyer" dans une Merge Request. J'ai encore quelques astuces de ce genre, mais ce sera pour une prochaine fois.
Vous pouvez retrouver le code source des exemples par ici: https://gitlab.com/cook-books/have-fun-with-gitlab-ci/mushrooms-picking

Grégory Guillou
Leader technique de l'équipe Easyteam DevOps. Grégory travaille avec Kubernetes depuis 2016 et développe en Go depuis 2018. Il a d'abord construit et maintenu des clusters, avant d'intégrer des solutions de gestion opérationnelle incluant le déploiement continu, la supervision des applications et des solutions de ChatOps avec Slack, Teams et GitHub. Il est un contributeur régulier du blog https://natives.easyteam.fr.

Etendre Kubernetes avec Kubebuilder

Partie 2

Kubernetes est un orchestrateur de containers. Mais c'est surtout une API extensible et riche ! Les fournisseurs cloud ont intégré leur réseau, leurs outils de supervision, leur solution de stockage, leurs équilibreurs de charge niveau 4 et niveau 7 pour offrir des services à valeur ajoutée à Kubernetes. Les éditeurs de logiciels comme Elastic, Confluent ou Redis Labs proposent des opérateurs qui gèrent leurs solutions sur vos clusters. Des projets comme Linkerd, Knative ou Argo-Flux étendent Kubernetes avec un "Service Mesh", une infrastructure "Serverless" et une solution "GitOps". Kubernetes est décrit comme une solution "API-centric" et des centaines d'organisations utilisatrices et de fournisseurs développent des composants qui intègrent ou étendent Kubernetes. Les composants de Kubernetes deviennent eux-mêmes, peu à peu, des extensions de l'API. Vous allez découvrir comment étendre Kubernetes et une application avec un opérateur appelé Emojis, programmé avec Kubebuilder et Go.

Modifier le type "Emoji" et régénérer la CRD

branch: operator/04-specify-emoji-type

La CRD est créée par Kubebuilder en se basant sur les types inclus dans le fichier api/v1alpha1/emoji_types.go, modifiez-le comme suit :

- Dans la struct EmojiSpec, remplacez "Foo" par "Description" comme ci-dessous

```
// Description allows to add a description to an Emoji.
Description string `json:"description,omitempty"`
```

- Dans la struct "EmojiStatus", ajoutez les champs "Supported", "LastPublishedTime" et "LastPublishedOutput" en précisant qu'il s'agit d'un type "time". Notez que vous pouvez utiliser des annotations pour mettre des contraintes sur les données contenues par les champs. "time.Time" ne peut pas être utilisé, utilisez "metav1.Time"

```
// Defines if the Emoji is part of the 100 registered Emojis
// +optional
Supported *bool `json:"supported,omitempty"`
```

```
// Information when was the last time the Emoji was successfully published.
// +optional
LastPublishedTime *metav1.Time `json:"lastPublishedTime,omitempty"`

// Information when the last time the Emoji was published what has been the output.
// +optional
LastPublishedOutput string `json:"lastPublishedOutput,omitempty"`
```

- Pour que status soit publié comme une sous-ressource d'Emojis, vous devrez ajouter l'annotation en dessous de "+kubebuilder: object:root=true"

```
// +kubebuilder:subresource:status
```

Vous pouvez d'ores et déjà créer des ressources exemples dans le répertoire "operator/config/samples". Lancez la commande "make install" pour modifier le fichier de manifeste de la CRD.

Implémenter la mise à jour du statut

branch: operator/05-implement-support

Implémentez la gestion du champ "Supported" dans la section "status" de la ressource. Le principe est simple, vous avez une liste de 100 émoticônes, si le nom de la ressource est inclus dans les 100, vous mettrez la valeur de supported à "true", sinon à "false". Pour cela, modifiez le contrôleur situé dans "operator/controllers/emoji_controller.go" :

- Ajoutez les imports ci-dessous ; "emojivoto" contient les 100 émoticônes extraites du projet exemple de Linkerd :

```
"fmt"
emojivoto "github.com/buoyantio/emojivoto/emojivoto-emoji-svc/emoji"
appv1alpha1 "github.com/easyteam-fr/emojis/operator/api/v1alpha1"
apierror "k8s.io/apimachinery/pkg/api/errors"
```

- Modifiez la méthode "Reconcile" comme ci-dessous :

```
func (r *EmojiReconciler) Reconcile(req ctrl.Request) (ctrl.Result, error) {
    ctx := context.Background()
    log := r.Log.WithValues("emoji", req.NamespacedName)

    // Get the Emoji from the cache, check if its supported and update its status
    var emoji appv1alpha1.Emoji
    if err := r.Get(ctx, req.NamespacedName, &emoji); err != nil {
        if client.ignoreNotFound(err) != nil {
                log.Error(err, "unable to fetch Emoji")
        }
        return ctrl.Result{}, client.ignoreNotFound(err)
    }
    if emoji.Status.Supported == nil {
        supported := false
        e := emojivoto.NewAllEmoji().WithShortcode(
            fmt.Sprintf(":%s:", emoji.Name),
         )
        if e != nil {
                supported = true
        }
        emoji.Status.Supported = &supported
        if err := r.Status().Update(ctx, &emoji); err != nil {
                log.Error(err, "unable to update emoji status")
                return ctrl.Result{}, err
        }
        log.Info("Emoji status.supported updated")
        return ctrl.Result{}, nil
    }

    return ctrl.Result{}, nil
}
```

Tester l'opérateur

Vous pouvez vérifier que le contrôleur fonctionne. Pour cela, relancer le générateur pour lancer le contrôleur :

```
make install
make run
```

Vous pourrez ensuite créer une émoticône "dog" et vérifier que l'attribut "status.supported" est créé et la supprimer comme ci-dessous :

```
kubectl apply -f ./config/samples/app_dog_emoji.yaml
kubectl get emoji dog -o yaml
kubectl delete emoji dog
```

Dans la configuration avec bee, la valeur de "status.supported" doit être "false", car l'émoticône correspondante n'est pas supportée.

Implémenter un "Finalizer" pour les suppressions

branch: operator/06-implement-finalizer

Un "Finalizer" est une annotation qui permet d'intercepter la suppression d'une ressource. Pour faire simple, elle transforme une suppression en une mise à jour. L'utilisation du "Finalizer" consiste à:

- Créer 3 helpers "containsString" qui vérifient qu'une chaîne est dans un slice, "removeString" qui enlève une chaîne d'un slice et "emojiDelete" qui simule la suppression de la ressource:

```
// Helper functions to check and remove string from a slice of strings.
func containsString(slice []string, s string) bool {
    for _, item := range slice {
        if item == s {
            return true
        }
    }
    return false
}

func removeString(slice []string, s string) (result []string) {
    for _, item := range slice {
        if item == s {
            continue
        }
        result = append(result, item)
    }
    return
}
```

```
// Simulate the removal of an emoji
func emojiDelete(emoji *appv1alpha1.Emoji) error {
    return nil
}
```

- Ajoutez ensuite la section suivante dans la fonction "Reconcile" de "emoji_controller.go", avant le test sur emoji.Status.Supported. Cette section enregistre l'annotation qui correspond au "Finalizer" lors de l'insertion d'une ressource. Elle supprime le "Finalizer" de la ressource lorsque cette dernière est supprimée.

```
// name the emoji finalizer
emojiFinalizerName := "emoji.finalizers.app.natives.easyteam.fr"

// examine DeletionTimestamp to determine if object is under deletion
if !emoji.ObjectMeta.DeletionTimestamp.IsZero() {
    // The object is being deleted
    if containsString(emoji.ObjectMeta.Finalizers, emojiFinalizerName) {
        // our finalizer is present, so lets handle any external dependency
        if emoji.Status.Supported != nil && *emoji.Status.Supported == true {
            if err := emojiDelete(&emoji); err != nil {
                return ctrl.Result{}, err
            }
        }
        // Notify the client when the deletion is done by removing
        // the Finalizer for the resource
        emoji.ObjectMeta.Finalizers = removeString(
            emoji.ObjectMeta.Finalizers,
            emojiFinalizerName,
        )
        if err := r.Update(context.Background(), &emoji); err != nil {
            return ctrl.Result{}, err
        }
        log.Info("finalizer removed")
        return ctrl.Result{}, nil
    }
}
// The object is not being deleted, so if it does not have our finalizer,
// then lets add the finalizer and update the object. This is equivalent
// registering our finalizer.
if !containsString(emoji.ObjectMeta.Finalizers, emojiFinalizerName) {
    emoji.ObjectMeta.Finalizers = append(
        emoji.ObjectMeta.Finalizers,
        emojiFinalizerName,
    )
    if err := r.Update(context.Background(), &emoji); err != nil {
        return ctrl.Result{}, err
    }
    log.Info("finalizer registered")
    return ctrl.Result{}, nil
}
```

Vous pouvez tester la création et la suppression de ressources avec "kubectl" et constater, grâce aux logs, que le fonctionnement correspond à celui attendu.

Implémenter la logique d'interactions

branch: operator/07-implement-applogic

Dans cette section, vous allez connecter votre contrôleur à l'application à travers son API. Il faut donc préalablement déployer l'application. Vous utiliserez le namespace par défaut et vous lancerez une commande "curl" pour vérifier qu'elle est effectivement démarrée :

```
cd ../app
export REGISTRY=registry:5000
make docker-build
kustomize edit set image emojis-app=${REGISTRY}/emojis-app:latest
kubectl apply -k .
kubectl port-forward svc/emojis 8080:8080
curl -v 127.0.0.1:8080
```

Le contrôleur est exécuté en dehors de kubernetes. Établissez un tunnel vers l'application avec la commande "kubectl port-forward svc/emojis 8081:8081" et utilisez ce point d'accès en positionnant export EMOJIS_ENDPOINT="http://localhost:8081" préalablement au démarrage du contrôleur.

Vous créerez ensuite un fichier app.go situé dans le répertoire "operator/controllers" qui contient les functions "emojiDelete" et "emojiCreateOrUpdate" pour interagir avec l'API. Supprimez la fonction "emojiDelete" créée dans l'étape précédente :

```
package controllers

import (
  "fmt"
  appv1alpha1 "github.com/easyteam-fr/emojis/operator/api/v1alpha1"
  "net/http"
  "os"
)

var (
  application = "http://emojis:8081"
)

func init() {
  if os.Getenv("EMOJIS_ENDPOINT") != "" {
    application = os.Getenv("EMOJIS_ENDPOINT")
  }
}

func emojiDelete(emoji *appv1alpha1.Emoji) error {
  client := &http.Client{}
  req, err := http.NewRequest(
    "DELETE",
    fmt.Sprintf("%s/emojis/%s", application, emoji.Name),
    nil,
  )
  if err == nil {
    req.Header.Add("Content-Type", "application/json")
    _, err = client.Do(req)
  }
  return err
}
```

```
func emojiCreateOrUpdate(emoji *appv1alpha1.Emoji) error {

  client := &http.Client{}
  req, err := http.NewRequest(
    "PUT",
    fmt.Sprintf("%s/emojis/%s", application, emoji.Name),
    nil,
  )
  if err == nil {
    req.Header.Add("Content-Type", "application/json")
    _, err = client.Do(req)
  }
  return err
}
```

D'autre part, modifiez le fichier "emoji_controller.go" comme suit :

- Ajouter metav1 "k8s.io/apimachinery/pkg/apis/meta/v1" dans les imports
- Ajouter la section ci-dessous en bas de la fonction Reconcile, juste avant le return final

```
if emoji.Status.LastPublishedTime == nil &&
  emoji.Status.Supported != nil &&
  *emoji.Status.Supported == true {
  now := metav1.Now()
  emoji.Status.LastPublishedTime = &now
  emoji.Status.LastPublishedOutput = "Succeeded"
  err := emojiCreateOrUpdate(&emoji)
  if err != nil {
    emoji.Status.LastPublishedOutput = fmt.Sprintf("Error %v", err)
  }
  if err = r.Status().Update(ctx, &emoji); err != nil {
    log.Error(err, "unable to update emoji status")
    return ctrl.Result{}, err
  }
  log.Info("application updated")
  return ctrl.Result{}, err
}
```

Vous pouvez tester le contrôleur comme vous l'avez déjà fait et constater que l'application est mise à jour.

Opérateur Emojis : Déploiement

branch: operator/08-deployment

Pour publier l'opérateur vous supprimerez le fichier controllers/suite_test.go. Ensuite vous ajouterez la variable d'environnement EMOJIS_ENDPOINT en ajoutant la section suivante dans la description du container inclus dans config/manager/manager.yaml:

```
      env:
    - name: EMOJIS_ENDPOINT
      value: http://emojis.default.svc.cluster.local:8081
```

Il vous reste à construire le container qui contient le controller, le sauvegarder, installer la CRD et déployer le contrôleur dans Kubernetes. Pour cela, la variable "IMG" doit référencer votre registry docker comme ci-dessous :

```
make docker-build
IMG=registry:5000/emojis-controller:latest make docker-push
make install
IMG=registry:5000/emojis-controller:latest make deploy
```

Pour superviser les logs du contrôleur, vous pouvez lancer la commande ci-dessous :

```
kubectl logs -f -n operator-system -c manager $(kubectl get pod \
  -l control-plane=controller-manager \
  -ocustom-columns=name:metadata.name \
  --no-headers -n operator-system)
```

Créez des émoticônes à partir des fichiers "sample" et vérifiez que l'application est bien mise à jour.

Conclusion

Cet opérateur est un exemple. Pour aller plus loin, il faut gérer le redémarrage de l'application ou les erreurs dans les appels à l'API d'administration. Comme vous pouvez le constater, sans être un expert du Go, le modèle de réconciliation de Kubernetes est relativement abordable et kubebuilder rend son usage simple. N'hésitez pas à lire la documentation de Kubebuilder et à vous intéresser à Operator-SDK. Dans quelques semaines, vous utiliserez Kubernetes comme personne !

Didier Danse
DevSecOps and Agile Coach | IT Advisor | IT Architect

DevSecOps : les devs et la sécurité, le début d'une belle histoire ?

Partie 2

La sécurité informatique est largement discutée, et ce, depuis les débuts de l'informatique. Les idées reçues au sujet de cette sécurité sont nombreuses et sont bien souvent obsolètes tant les besoins, mais aussi les moyens, évoluent. De ce fait, les objectifs des équipes sécurité ou encore les méthodes appliquées n'ont globalement que peu évolué depuis de nombreuses années.

Parlons outils maintenant

L'Open Source ou des technologies propriétaires ?

Pendant de nombreuses années, l'open source était préconisée pour bien des situations mais assez peu pour les outils de sécurité. Les apports de la communauté *IT Security* et des développeurs ont finalement permis de faire émerger de nombreux projets open source considérés comme fiables, comme le OWASP le propose. Ainsi, la liste des règles de bonnes pratiques est désormais partagée par tous et pour tous. Il est fréquent de voir des fournisseurs de solutions proposer une version open-source de leurs outils qui répondent au besoin premier alors que leur différenciateur est quant à lui payant. On peut ainsi dire que l'open-source permet l'émergence d'outils de qualité qui se voient complémentés par des technologies propriétaires. La bonne nouvelle est que l'open-source couvre l'indispensable. Que ce soit de l'open source ou du propriétaire, l'important restera toujours d'identifier l'outil qui correspond aux technologies et à la manière de fonctionner de l'entreprise. Certains outils sont clairement voués à être utilisés dans un contexte Dev(Sec)Ops, d'autres s'adapteront mieux dans une approche Waterfall.

Les types d'outils

Chaque outil propose des fonctionnalités bien différentes les uns des autres. Ces fonctionnalités peuvent être regroupées en trois catégories décrites ci-dessous :

- **Static Application Security Testing (SAST)** : ce type d'analyse s'effectue sur base du code source directement. Il permet d'identifier des failles potentielles mais aussi d'analyser le respect de bonnes pratiques ou la clarté du code. Ces tests peuvent être effectués à tout moment, durant le développement ou à la compilation de ce code. C'est un type de test apprécié des développeurs qui parlent code et technologie. Malheureusement, le SAST ne peut identifier que ce qui est visible dans le code, ce qui peut rendre ce type d'analyse insuffisant lorsque l'on fait des applicatifs plus complexes. Le SAST sera donc préféré comme premier niveau d'analyse afin de garantir le respect de règles de codage, l'identification de scénarios d'échec tels que l'utilisation de variables non initialisées et d'autres éléments liés au code.
- **Dynamic Application Security Testing (DAST)** : celui-ci permet d'analyser des applications qui sont en cours de fonctionnement, principalement des applications web. Ce type d'analyse permet d'identifier des failles courantes telles que l'injection SQL ou de données malveillantes. De plus, le DAST permet d'identifier des scénarios qui exposent des données ou des risques de sécurité tant lors du transit ou *at rest*. Le DAST fait abstraction des technologies utilisées et tient compte uniquement des protocoles de transfert de données. Ce type de tests est souvent effectué après avoir déployé une version compilée du code.
- **Interactive Application Security Testing (IAST)** : ce dernier type de test injecte du code dans l'application afin d'en comprendre les méandres. Il permet ainsi d'accéder et de tracer l'information au sein de l'application et ainsi détecter les différents scénarios qui s'avèrent difficiles à identifier lorsque l'on se place au niveau du code ou en dehors de l'application comme le SAST et le DAST le font. En utilisant les symboles de débogage, l'IAST permet de faire le lien avec le DAST et ainsi en maîtriser le chemin complet d'une information.

Comme on peut le voir, chacun de ces types de tests permet d'obtenir des résultats bien différents et la combinaison des 3 informations permet d'avoir une vue complète. Alors chacun se spécialise sur un aspect distinct avec des intérêts différents. Le SAST est effectué au plus tôt, ce qui réduit le coût d'un correctif comparé à une situation où le risque est identifié ultérieurement. Le DAST permet, quant à lui, de tester des scénarios d'un point de vue externe, un peu comme un hacker le ferait. L'IAST permet, lui, de faire de l'analyse du chemin sur base d'appels externes.

Ces tests peuvent s'avérer insuffisants dans certaines situations. En cas d'attaque, il s'agit de pouvoir réagir rapidement. C'est pourquoi on peut rajouter les outils utilisés pour connaître ses environnements et les évènements en production :

- **Run-Time Application Security Protection (RASP)** : similaire à l'IAST dans son fonctionnement, le RASP diffère dans sa finalité : l'un a pour but de tester tandis que l'autre a pour but de sécuriser l'application et de réagir à des évènements afin d'empêcher l'accès à des données.
- **Logging and Monitoring** : comme leurs noms l'indiquent, ces outils permettent de garder la trace des activités passées et en cours, ce qui permet d'accélérer l'identification des activités et des causes.

Ainsi, pourquoi ne pas tout miser sur le RASP vu que celui-ci permet une protection active des applications ? Diverses justifications existent. Tout d'abord, il s'agit d'éviter des impacts sur les performances mais aussi une incorrecte perception de sécurité. Moins l'applicatif expose de risques, plus ces outils seront capables de supporter les attaques. Par ailleurs, nous l'avons vu, une faille identifiée en production coûte 100x plus qu'un problème identifié durant les

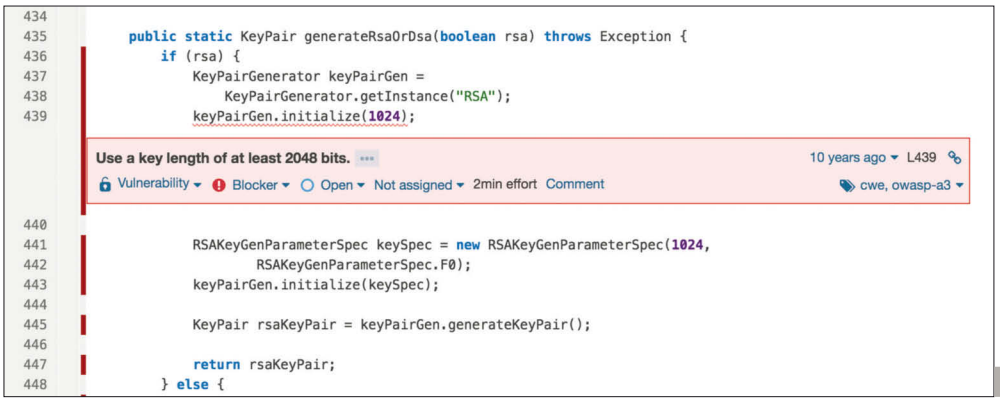
phases de développement. Abordons maintenant quelques outils pour illustrer les différents types de tests. Cette liste est non exhaustive et est purement représentative. Le choix des outils est totalement arbitraire.

Static Application Security Testing (SAST)

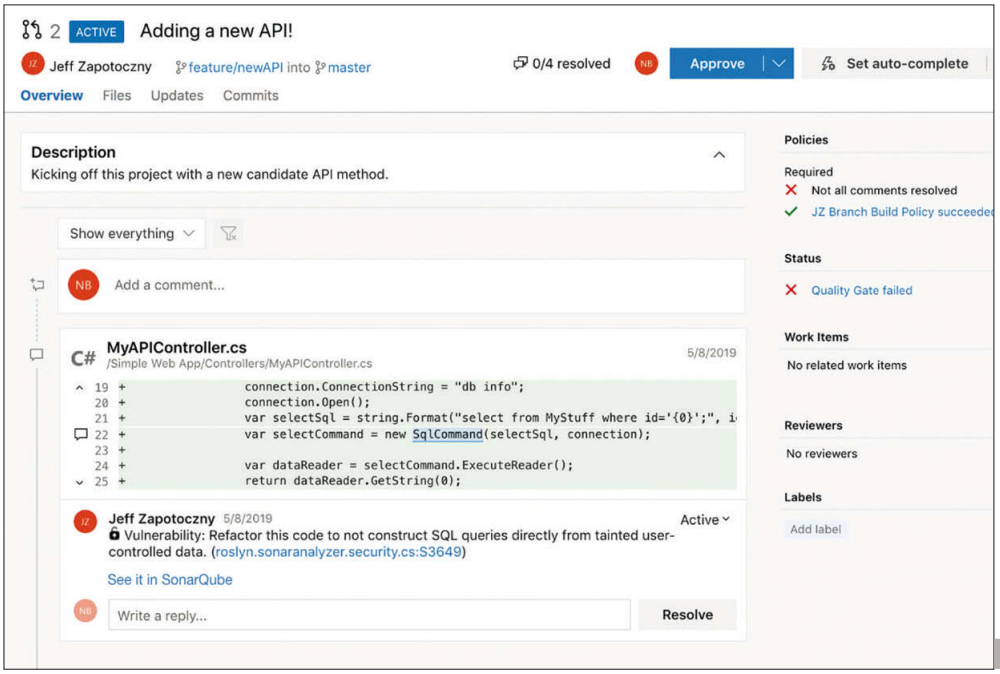
SonarQube

SonarQube supporte de nombreux langages et couvre la grande majorité, si ce n'est pas l'entièreté, des règles de bonnes pratiques concernant le code de qualité en plus des règles concernant la sécurité. L'outil est ainsi capable d'identifier des bugs potentiels, avant même que le code ne soit dans le code repository en l'utilisant depuis l'environnement de développement. La qualité du code ne peut dès lors qu'évoluer. L'approche de l'outil permet soit d'identifier toutes les règles qui ne sont pas respectées sur la globalité du code ou encore sur ce qui est nouveau, afin de ne pas accroître la dette technique. Le développeur est ainsi celui en charge de fournir du code de qualité là où le gatekeeper était nécessaire auparavant. 1

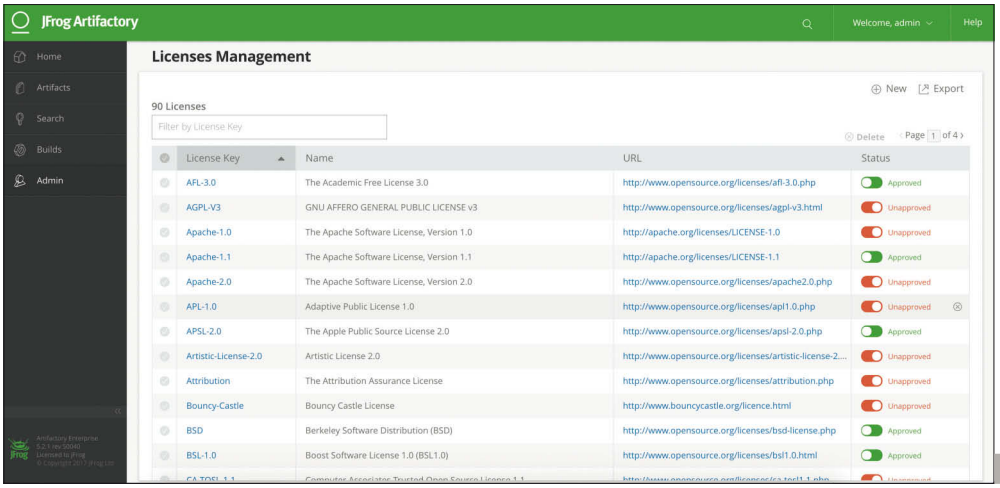
Ce même code peut être vérifié au moment du build et permet de créer des pull requests en interagissant avec Git. 2



1



2



3

Artifactory

Artifactory est un repository d'artifacts. L'intérêt de l'outil, au-delà de cet aspect de repository centralisé, est d'analyser ces artifacts et de fournir du reporting sur les licenses d'utilisation ainsi que les vulnérabilités. L'intérêt est évidemment d'identifier les composants utilisés au sein des artifacts générés mais aussi d'identifier en amont ces informations pour les artifacts provenant d'Internet en agissant comme un proxy. En définissant un point d'entrée (ou de sortie c'est selon), Artifactory devient le point de passage et de contrôle pour l'ensemble des artifacts. 3

WhiteSource

WhiteSource, permet quant à lui de compléter les fonctionnalités d'Artifactory, un peu comme XRay (du même fournisseur de solution qu'Artifactory et qui n'est utilisable qu'avec ce dernier), en permettant une identification continue de l'ensemble des composants, y incluant les dépendances indirectes. En fonctionnant de la source, WhiteSource peut récupérer de nombreuses informations complémentaires. WhiteSource a comme intérêt de pouvoir se connecter à divers sources, ce qui permet d'avoir plusieurs entrepôts, ce qui permet alors de profiter des avantages des uns et des autres.

Pour aller plus loin avec le SAST

D'autres outils à explorer : *[Code]AI*, *Black Duck* (détection des licences et de la composition des logiciels), *CA Veracode* (Code mais aussi licences), *Checkmarx* (sa force est le module de formation en continu qu'il

embarque), *Codacy*, *Fortify*, *Git-Hound* (empêche de publier des données sensibles), *Git-secrets* (encrypte les fichiers contenant des "secrets"), *Kiuwan*, Klocwork, NowSecure, *OWASP Dependency Check* (identification des dépendances et de leur statuts), *OWASP Zed Attack Proxy (ZAP)*, *Parasoft tool suite*, *Retire.js* (détection de librairies javascript vulnérables), *SecureAssist* (XSS, authentification brisée, ...), *Sonatype*, *Synopsys Code Sight*, *truffleHog* (détections de données confidentielles), *TruffleHog*, *gitrob*, *github darks* (détections de données confidentielles), *Whitehat Scout*, *Whitehat Sentinel*.

Dynamic Application Security Testing (DAST)

Gauntlt

Gauntit permet de faire tourner plusieurs outils les plus fréquemment utilisés dans le domaine du test de sécurité. Certains de ces outils se focalisent sur la détection de ports ouverts tandis que d'autres vont bien plus loin avec notamment des tests d'injection.
Sur base de syntaxe *Gherkin* et de l'interpréteur *Cucumber*, il est aisément possible de définir des scénarios et des actions associées, avec ou sans paramètres. Un exemple vaut mieux qu'un long discours :

```
@slow
Feature: Test the app against the ìrobots attackî

Background:
Given "nmap" is installed
And the following profile:
| name      | value            |
| hostname  | www.programmez.com  |

Scenario: read the robots.txt
When I launch an "nmap" attack with:
""" nmap --script http-robots.txt <hostname> """
Then the output should contain:
""" | http-robots.txt: """
```

Ce premier scenario se charge de vérifier, à l'aide de nmap, que l'application n'est pas vulnérable à cause du fichier robots.txt tandis que l'exemple suivant vérifie que l'utilisation de SSL versions 2 et 3 n'est pas possible avec cette application :

```
Feature: Check SSL versions which are enabled

Background:
Given "sslyze" is installed
And the following profile:
| name    | value          |
| hostname | www.programmez.com |
| port    | 443            |

Scenario: Ensure no SSLv2 and SSLv3 are disabled
When I launch an "sslyze" attack with: """ sslyze --sslv2 --sslv3 <hostname>:<port> """
Then the output should not contain: """ Accepted """
```

Ce langage, d'ailleurs disponible en plusieurs langues, est bien connu des développeurs impliqués dans des processus agiles avec l'utilisation des mots clés présents pour définir les critères d'acceptation.
Par défaut *gauntlt* est prêt à fonctionner en quelques clics et lignes de code. Celui-ci vient par défaut avec certains outils bien connus, prêts à l'emploi, grâce à des connecteurs qui font le lien entre l'information interprétée et les outils.

- **arachni**: permet de tester les injections de type XSS. Il est cependant aisé d'étendre la fonctionnalité à d'autres types d'injections ;
- **curl**: récupère des données et vérifie que celui renvoie bien de l'information attendue ;
- **dirb**: identifie des répertoires web sur base d'une liste de mots ;
- **garmr**: lit le header d'une réponse HTTP, notamment pour s'assurer que le nom du serveur web ou sa version ne sont pas présents ;
- **heartbleed**: teste la vulnérabilité OpenSSL du même nom, qui permet notamment de décrypter les données en transit ;
- **nmap**: principalement utilisé pour scanner les ports ;
- **sqlmap**: teste les injections SQL ;
- **sslyze**: teste l'implémentation SSL / TLS ;
- **autres**: il est possible d'étendre les fonctionnalités. Citons par exemple l'intégration du OWASP ZAP dont nous avons parlé précédemment.

Pour aller plus loin avec le DAST

D'autres outils à explorer : *Acunetix*, *BDD Security (*basé sur *Gherkin* qui s'intègre aisément dans un pipeline CI/CD*)*, *CA Veracode*, *Contrast Assess*, *Metasploit*, *NowSecure*, *OWASP OWTF* (tests de pénétration), *Parasoft tool suite*, *Sonatype*, *Whitehat Sentinel*.

Interactive Application Security Testing (IAST)

Acunetix avec AcuSensor

Celui-ci permet le scan d'applications web en fournissant des tests de type DAST mais aussi IAST lorsqu'il est combiné avec AcuSensor. Sans rentrer dans les détails, l'outil annonce faire des tests pour permettre d'identifier 4500 vulnérabilités.

Pour aller plus loin avec l'IAST

D'autres outils à explorer : *Checkmarx*, *Reshift*, *Contrast Assess*.

Un cas un peu à part, le all-in-one: GitLab

GitLab est certainement la toolchain la plus complète qui propose, dans le contexte qui nous intéresse, du SAST, DAST et prochainement du IAST. Chacun se fera sa propre opinion sur la capacité d'avoir un vendor unique qui propose une solution complète, de A à Z mais avoir une approche cohérente globale est indubitablement un avantage certain.

RASP et monitoring

Quelques pistes à explorer : *Logz.io* (analyse de logs), *Dome9 Security* (Cloud), *Immunio*, *RedLock*, *Aqua Security* (Containers).

Un exemple de pipeline complet

Bien que le nombre d'outils ci-dessus s'avère déjà bien chargée, cette liste est non-exhaustive et augmentera dans les prochains mois et années. En effet, de nouveaux outils disposant de nouvelles fonctionnalités, notamment grâce à l'émergence de l'intelligence artificielle, feront leur apparition sur le marché. Maintenant que nous avons vu cette liste, il reste à les assembler pour proposer un pipeline complet. Prenons un exemple. 4

La sécurité durant la phase de développement

Durant la phase de développement, seul le code source est disponible. C'est déjà bien suffisant pour identifier bien des risques et ainsi minimiser les impacts futurs. Ainsi, les outils de SAST s'avèrent bien utiles dans l'environnement de développement. *SonarLint* est le connecteur permettant à un environnement de développement de s'intégrer avec *SonarQube* que nous avons abordé précédemment. Dans ce contexte, c'est directement dans l'environnement que les risques et les failles sont notifiés, durant le développement. Que demander de mieux ?
Le développeur peut également faire tourner l'applicatif sur son poste local. Dans ce cas, il est possible de tester l'application en tant que telle et ses points d'entrée. Certains outils de DAST et d'IAST s'intègrent aisément dans l'environnement de développement.

Quand le code est enregistré dans le dépôt de sources

Au moment de l'enregistrement du code vers le dépôt de code source, la qualité de ce code est vérifiée afin d'éviter que du code présent dans le dépôt ne puisse être de qualité moindre. Mais au-delà de la qualité de code, il est également possible d'analyser ce code afin d'identifier les risques de divulguer des informations confidentielles (mots de passe, configuration, API tokens…). Certaines solutions comme git-secret encrypteront le fichier qui contient ces dits secrets. *Git-hound* quant à lui empêchera d'enregistrer des données sensibles alors que l'on favorisera *truffleHog* pour identifier où se trouvent les données. Les tests sont similaires à ce qui peut être fait dans l'environnement de développement mais le fait d'automatiser cela sur l'ensemble du code entrant rend le processus plus sûr et permet ainsi au code d'être de meilleure qualité.

Durant la phase de compilation et de test

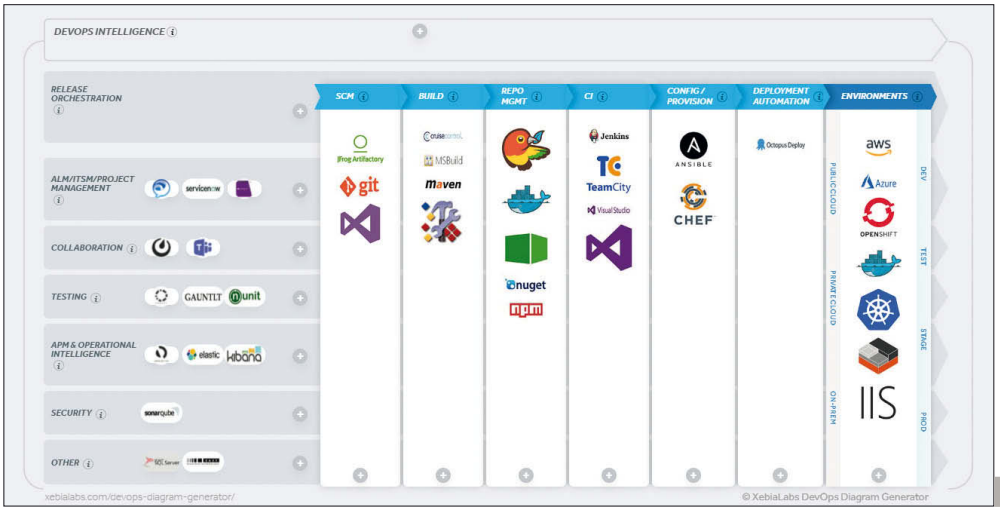
Encore une fois, il est possible d'effectuer les tests précédents sur base du code source qui est utilisé pour compiler l'application, et ce, au moment du build. Nous ne reviendrons pas dessus. Simplement, il s'agit de savoir comment les outils choisis s'intègrent dans le pipeline. Certains se basent sur le code repository tandis que d'autres s'intègrent au moment du build. Dans ce contexte, il est évidemment possible de faire référence à *SonarQube* qui peut d'ailleurs se charger de pousser des pull requests automatiquement lorsque celui-ci détecte des changements à effectuer. D'autres solutions existent mais dans ce cas, seulement les solutions SAST peuvent être utilisées puisque seul le code est disponible.

Le résultat de la compilation peut d'ores et déjà être testé, avant même que les applications ne soient physiquement déployées sur un environnement. Ainsi, au même titre que les tests unitaires, un certain nombre de tests peuvent être effectués sur l'applicatif. Les solutions de DAST sont ainsi les plus adaptées. Il est ainsi recommandé de tester au maximum ces applications à cet instant.

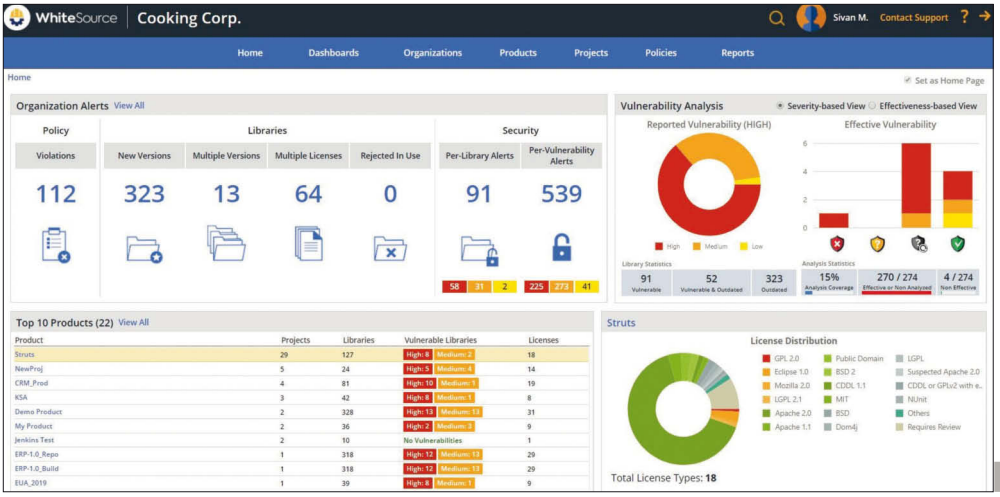
Quand l'artifact est déposé dans un dépôt d'artifacts

A nouveau, il est possible d'effectuer des analyses statiques, cette fois pas sur du code sur les métadonnées liées à un artifact. L'analyse de ceux-ci permet généralement d'identifier un grand nombre d'informations liées aux licences. Certains outils peuvent également réaliser des tests dynamiques puisque le dépôt comporte des solutions d'ores et déjà compilées. Ce type de tests peut être effectué dans un contexte où des tests sont effectués de manière plus approfondies (et donc prennent plus de temps à être conçus). Cela peut notamment être intéressant avant de permettre la promotion d'un artifact vers de la production.

Artifactory, via son composant *XRay* permet de lister toutes les utilisations d'un composant donné et d'y attacher un certain nombre de métadonnées à ces artifacts tels que la licence utilisée mais aussi le risque



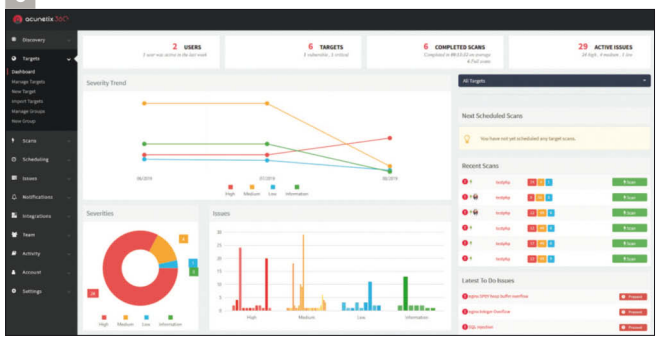
4

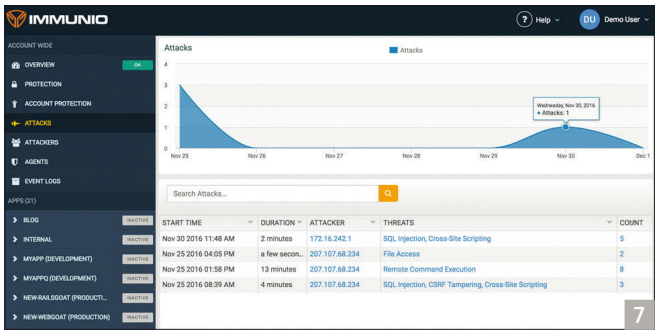


5

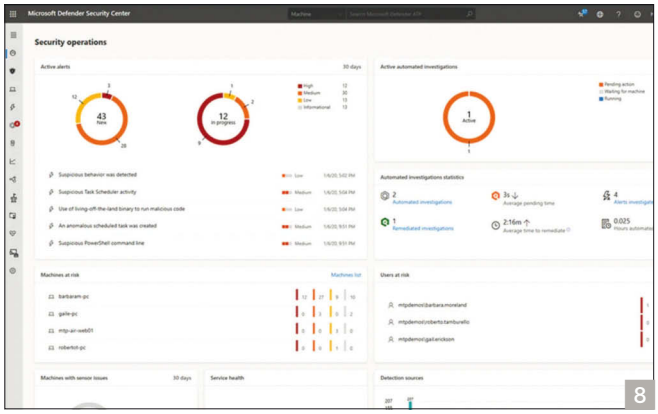
de vulnérabilité. *WhiteSource*, quant à lui, est externe au dépôt et se connecte à n'importe quel dépôt, ce qui permet de profiter des avantages des différents dépôts tout en restant sécurisé. Dans ce contexte, la force d'Artifactory est sa capacité à fonctionner comme un proxy pour télécharger des ressources externes à l'entreprise. 5

6

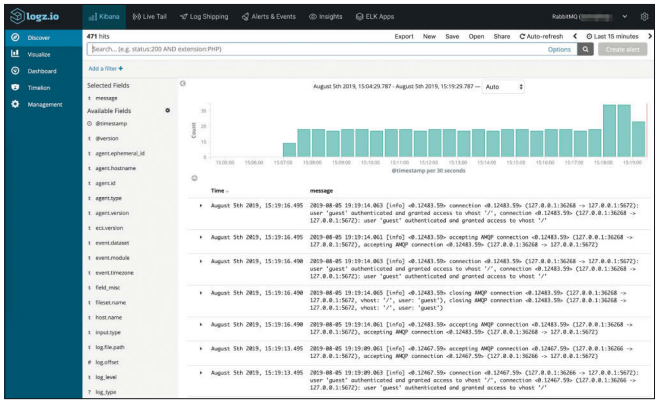




7



8



Delivery & Test

Une fois l'application déployée dans un environnement, il est possible d'en tester différents, notamment grâce à au IAST mais aussi le DAST. Dans le contexte d'un *pipeline CI/CD*, il est important de s'assurer que le test peut être initié par l'outil qui s'occupe du déploiement. Ainsi, *Acunetix* s'intègre parfaitement avec Jenkins. En utilisant *Acunetix* avec *AcuSensor*, il est possible de tester l'applicatif et de reporter les résultats au sein de votre *issue tracker*. Ainsi, ces tests peuvent être effectués préalablement à la mise en production. 6

En production

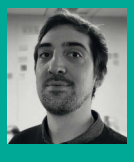
En production, il ne s'agit plus de tester les applications mais d'identifier les tentatives ou les failles de sécurité. Dans ce contexte, le Run-Time Application Security Protection (RASP) s'avère important pour bloquer des activités malicieuses au sein des applications. Ces outils sont bien souvent intégrés avec des outils de monitoring qui s'occupent de récupérer de l'information sur la consommation de ressources physiques et logiques mais aussi avec des outils consolidant les entrées de logs. Des outils tels que *Immunio* 7 s'avèrent alors très utiles pour détecter des comportements suspects et les bloquer mais aussi pour fournir suffisamment d'information afin d'étudier ces comportements au travers des tableaux de bord. 8

Par où commencer ?

Peut-être avez-vous déjà démarré l'intégration de ces outils au sein de votre pipeline CI/CD. Si ce n'est pas le cas, il est temps de s'y mettre, avec ou sans les équipes sécurité. Si ce n'est pas le cas, il est déjà possible de mettre en place un certain nombre de tests afin de minimiser les risques évoqués par l'OWASP. Les équipes sécurité peuvent, en effet, fournir des conseils permettant d'atteindre le meilleur résultat en un minimum de temps. Cependant, dans tous les cas, elles vous remercieront de l'intérêt que vous portez à la sécurité et vous aurez tout leur soutien pour promouvoir de telles initiatives futures. Mais au-delà de se faire des amis, l'intérêt est avant tout de comprendre les fondements de la sécurité, ce qui aura un impact positif sur les livrables fournis aux différents métiers.

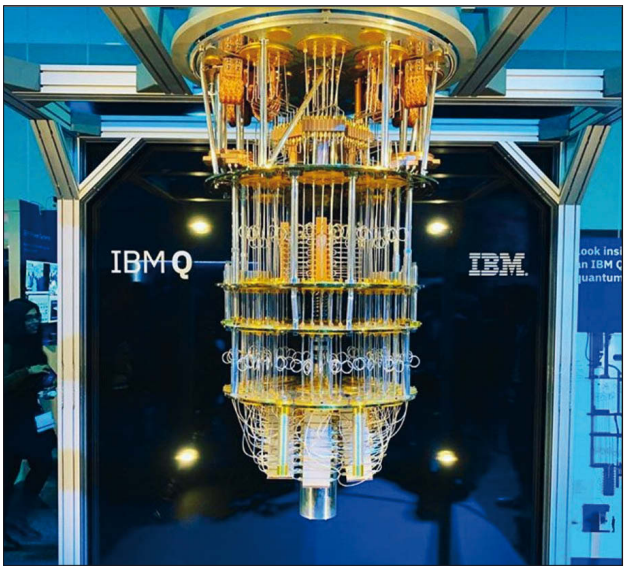
Pour le développeur, quoi de plus motivant que d'éviter de devoir résoudre des bugs connus et qui auraient pu être évités, et de voir comment les choses évoluent ? Pour le visualiser, il s'agit de mettre en place des tableaux de bord avec le score de vulnérabilité, le nombre de vulnérabilités, la distribution des risques mais aussi la durée avant correction. Le simple fait de mettre en place de telles solutions aura des impacts plus que positifs qui seront facilement identifiables et visibles, tant pour vous que pour les équipes dirigeantes.

Avant même de chercher à améliorer la situation, il s'agit de s'assurer de ne pas augmenter la dette technique, notamment en initiant des vérifications via des outils de qualité de code et en réagissant au plus tôt. En parallèle, Il s'agit de démarrer petit et d'activer les tests les uns après les autres. En se focalisant sur certains tests, l'ensemble des intervenants seront à même d'être formés mais aussi de distinguer l'important de l'accessoire. De plus, le fait d'avancer petit à petit permet de créer un esprit de confiance entre les équipes mais aussi vis-à-vis des outils. Une autre raison cruciale pour favoriser cette approche est de s'assurer que les équipes peuvent absorber la charge de travail générée. Cette charge se réduira dans le temps, jusqu'à ce que la dette technique soit estompée.



Clément Breuzet
Passionné de science et d'informatique, je suis actuellement développeur au sein de Avanade France. Cela fait maintenant trois ans que je m'intéresse sérieusement à la physique quantique et plus particulièrement à l'application dans notre domaine. J'ai envie d'être un jour un acteur de ce vaste domaine qu'est l'informatique quantique.

Introduction à l'informatique quantique



Cette magnifique fusion entre la physique quantique et l'informatique est en passe de devenir la révolution technologique de cette décennie. Un nombre incommensurable d'évènements récents le démontrent : la Chine a mis en orbite un premier satellite quantique ; l'algorithme de Shor met en danger l'encryptage de nos clés RSA ; IBM et Google ont déployé leurs ordinateurs quantiques ; Intel met en avant son premier processeur quantique Intel Horse Ridge capable d'orchestrer une flopée de Qubits et annonce 9 ans avant son utilisation industrielle, Microsoft Azure va prochainement proposer la ressource Azure Quantum.

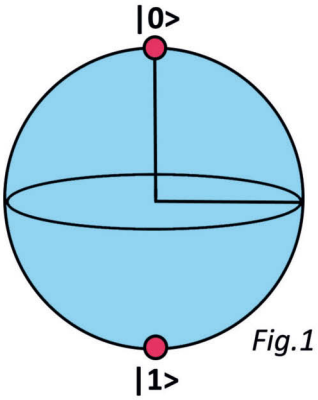


Fig.1

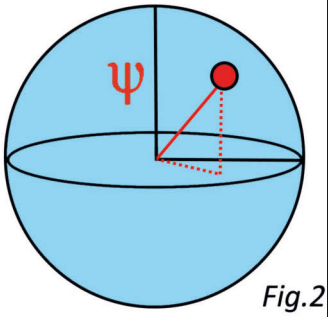


Fig.2

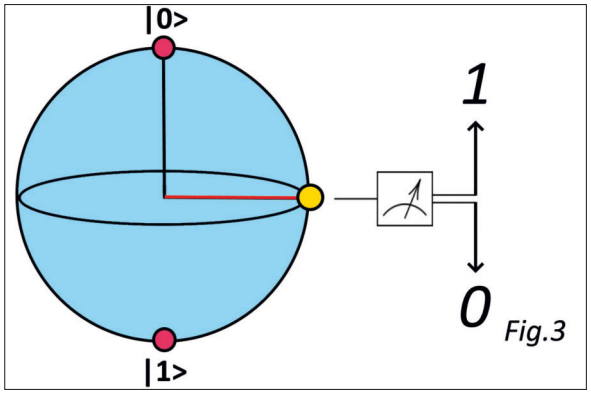
En tant que lecteur de Programmez et en conséquence passionné d'informatique, vous devez vous poser cette question : « Puis-je être un acteur de cette révolution ? ». La réponse est oui, à condition de désapprendre tout ce que vous avez pu apprendre, faire abstraction de toute logique déterministe et accepter d'être dépassé par une théorie qui effrayait et passionnait l'un des plus grand physicien et penseur de notre ère, un certain Albert Einstein. Cet article est orienté sur la programmation quantique, mais avant d'appliquer un exemple concret en Q#, nous allons rapidement faire un tour des concepts. Qu'est-ce qu'un ordinateur quantique ? Un Qubit ? Une porte quantique ? Un circuit quantique ? Nous aborderons ensemble tous ces points que je tenterai de vulgariser au mieux tout en omettant les démonstrations mathématiques complexes afin que cela reste digeste. Gardez en tête que l'informatique quantique ne va pas remplacer notre système binaire actuel, mais va le compléter afin de révéler la vraie nature de l'informatique.

Comment ça marche ?

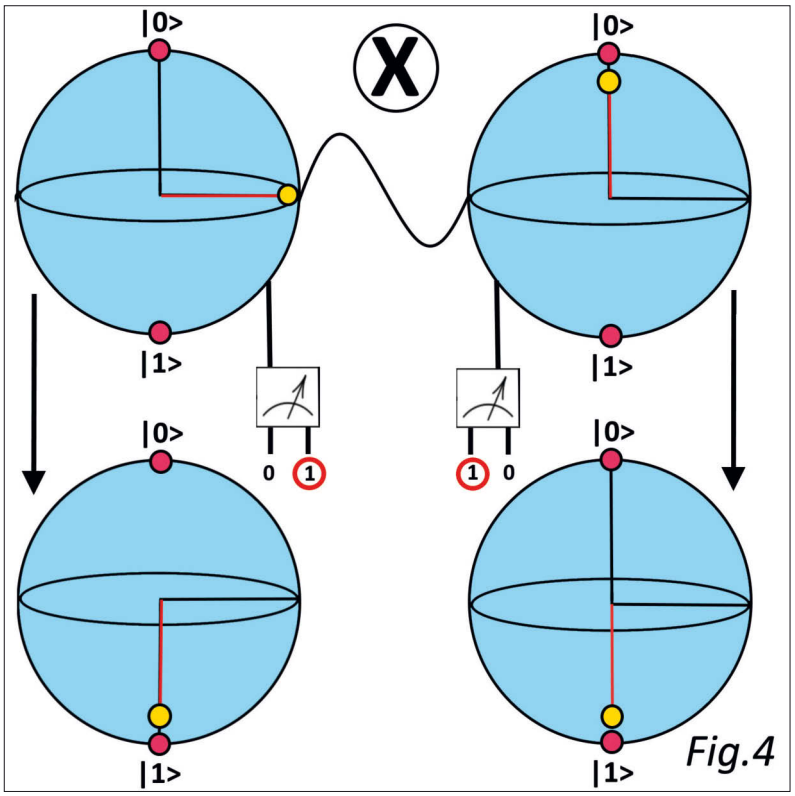
Un ordinateur quantique ressemble de loin à nos ordinateurs classiques, à l'exception prêt que nous ne manipulons pas des bits, mais des bits quantiques. Les structures actuelles sont coûteuses et sont contraintes à des températures extrêmement basses de fonctionnement. En effet, pour se retrouver dans un état fonctionnel, il faut le refroidir à une température proche du zéro absolu ou 0 Kelvin. Cela aura pour conséquence de forcer la résistance électrique des matériaux utilisés à 0, ce que nous appelons des matériaux supraconducteurs. Si vous avez fait un peu d'électronique, cette valeur peut sembler folle, voir inimaginable. Pourtant c'est ce que nous allons chercher à reproduire lors de la construction d'un calculateur quantique. Les premiers ont été imaginés en 1990 et depuis cette date, nous tentons de construire des machines fonctionnelles avec lesquelles nous pourrons faire fonctionner des algorithmes quantiques bien connus. Pour l'utiliser, nous plongeons cette machine complexe dans un bain d'azote liquide et communiquons avec via des ondes radio.

Actuellement, nos systèmes informatiques fonctionnent de manière binaire et déterministe via un bit qui possède deux états 0 ou 1, mais je ne vous apprends rien. Maintenant, fermez les yeux et imaginez une sphère polarisée avec un Nord et un Sud dont l'axe des pôles serait orientable, nous avons alors la possibilité de définir une infinité d'état. C'est le modèle d'un électron ou d'un spin. Un Qubit possède les mêmes propriétés intrinsèques. Il est la plus petite unité de stockage d'information quantique.

Un Qubit est une superposition d'état. J'insiste sur le terme superposition, car nous utiliserons toujours deux vecteurs appelés



ket0 et ket1 que nous pouvons représenter soit par une matrice unitaire $\begin{bmatrix}1\\0\end{bmatrix}$ et $\begin{bmatrix}0\\1\end{bmatrix}$ ou par la notation de Dirac : |0> et |1> (Fig. 1). C'est comme si nous avions un globe terrestre où le nord serait |0> et le Sud |1> et Ψ représente l'orientation de ses pôles, nous l'appelons la forme d'onde quantique. Nous pouvons transmettre des informations ou orienter ces deux kets en les multipliant par un angle. La forme d'onde quantique d'un qubit devient donc Ψ : α |0> + β |1> où α et β sont deux angles, nous avons donc ici deux informations (Fig. 2). Je reviendrai sur cet angle que nous pouvons initialiser grâce à ce que nous appelons une porte quantique. Si nous souhaitons obtenir la valeur de ce bit quantique, nous devons effectuer une mesure. Celle-ci est endémique de la physique quantique. Lorsque nous allons effectuer cette mesure nous ne pourrons pas récupérer la forme d'onde mais uniquement 1 ou 0. Pour exemple, nous allons récupérer 1 si nous mesurons Fig. 2. A présent imaginons que nous manipulions deux Qubits. Nous obtenons alors Ψ = α |00> + β |01> + γ |10> + δ |11>, la forme d'onde quantique est ainsi composée de quatre informations. Si nous utilisons N Qubit nous aurons alors 2^n informations, ce qui vous le comprendrez par la suite, est très efficace pour tout calcul exponentiel et est une grosse évolution par rapport à notre système actuel. Maintenant essayons d'extrapoler ce résultat en binaire, si nous avons 70 Qubit nous aurons 2^{70} informations ou $1{,}8^{e}+21$ bits ou plus simplement 1 Zettabit. Ce volume d'information est plus important que toute l'information stockée par l'humanité. Impressionnant n'est-ce pas ? Mais malheureusement ce n'est pas comme ça que ça marche.



Le bit quantique

Parlons maintenant des propriétés intrinsèques d'un bit quantique et commençons par la plus simple et peut-être la plus connue : la superposition. Qui n'a jamais entendu parler de ce fameux chat de Schrödinger ? Cette pauvre bête enfermée dans une boîte dont nous ne connaîtrions pas l'état avant de l'ouvrir. Lorsque celle-ci est fermée nous pouvons considérer qu'il est à la fois mort et vivant et lorsque nous l'ouvrons, nous avons 50% de chance qu'il soit vivant et 50% de chance qu'il soit mort. C'est l'une des meilleures métaphores de la superposition, parce que c'est exactement ce qu'il se passe lorsque que nous orientons la forme d'onde quantique d'un Qubit dans un état de superposition (Fig. 3)

A gauche nous retrouvons notre Qubit que nous avons mis dans un état de superposition. Ici la forme d'onde quantique est à la fois |0> et |1>. Au centre nous avons le signe qui caractérise une mesure. Il nous sera utile dans la construction d'un circuit quantique car une mesure est une porte quantique. Une ligne (à gauche de la porte) représente un Qubit et les deux lignes (à droite de la porte) représentent des bits. Et si vous me suivez toujours, cette mesure va donc nous renvoyer soit 1, soit 0. C'est ce que nous appelons le « true random » qui contrairement au random que nous utilisons dans nos programmes actuels, n'est pas déterminé. Enchaînons sur un autre état presque diabolique et contre intuitif des propriétés intrinsèque du bit quantique ; l'intrication. Ce phénomène décrit par Albert Einstein comme « Spooky action at distance », traduisez « action fantôme à distance », décrit un enchevêtrement de Qubits fortement liés. Intriqués ou emmêlés par une sorte de force invisible qui serait plus rapide que la lumière. Impossible vous me direz, et pourtant c'est observable et reproductible.

Sur la partie haute de ce schéma (Fig. 4), nous retrouvons deux qubits intriqués, nous parlons ici de système intriqué. Nous pouvons utiliser l'annotation X ou produit tensorielle pour les déclarer intriqués. Pour l'exemple, nous allons mettre le premier qubit dans un état de superposition et le second à 0. Dans un cas nominal où nos qubits ne serait pas intriqués, nous devrions récupérer lors de la mesure du premier 1 ou 0 et du second 0. Mais dans un état d'intrication, ce n'est pas ce qu'il se passe, et ça va à l'encontre de toute logique, accro-

chez-vous ! Si nous mesurons le premier Qubit, alors la valeur de cette mesure sera également la valeur de la mesure du second qubit. Dans notre exemple, la mesure du premier qubit renvoi 1 et la seconde mesure 1. Dans le cas où notre deuxième qubit serait initialisé à 1 et en répétant ce processus, sa mesure renverra toujours l'inverse de la mesure du premier. Il faut avoir en tête que les Qubit peuvent être espacés de plusieurs années lumières et que le même phénomène se reproduira instantanément et donc forcément plus rapidement que la vitesse de la lumière, c'est ce que nous appelons la téléportation quantique. Attention, avec l'intrication nous ne transmettons pas d'informations, donc la légende qui dit que nous pourrons transmettre plus rapidement des données est fausse. Nous influençons un système en effectuant une mesure.

Maintenant que nous avons survolé le concept du Qubit, nous allons découvrir comment jouer avec. En informatique binaire nous connaissons les portes logiques comme le « if » par exemple. En informatique quantique nous allons introduire les portes quantiques. Celles-ci sont des matrices unitaires complexes qui vont nous permettre de manipuler les Qubits et construire des circuits quantiques. Toutes ces portes ont pour action de changer la polarisation d'un bit quantique. Je vais énumérer et expliquer les plus utilisées.

H ou Hadamard (Fig. 5) met un Qubit dans un état de superposition, donc celui-ci sera soit 1 soit 0. X ou PauliX inverse la polarisation d'un Qubit sur l'axe X. On l'appelle un bit-flip. Y ou PauliY inverse la polarisation d'un Qubit sur l'axe Y. Z ou PauliZ inverse la polarisation d'un Qubit sur l'axe Z. Cette porte est souvent utilisée pour les mesures. Des portes beaucoup plus complexes existent comme la porte Swap, contrôlée, de changement de phase et nous pouvons même créer des portes en les assemblant. Grâce à toutes ces portes nous pouvons créer ce que nous appelons des circuits quantiques, ils sont en quelque sorte une forme d'algorithmie. L'idée est d'assembler des Qubits auxquels nous allons appliquer des états afin de récupérer un résultat en sortie. La plupart sont déjà connus comme l'algorithme de Shor ou de Grover, mais l'imagination de chacun pourra contribuer à l'enrichissement des circuits quantiques qui sont de premier abord non intuitif.

Ce circuit (Fig. 6) est vraiment très simple et permet de représenter la découverte de Shrödinger, la superposition. De gauche à droite nous faisons passer un Qubit par la porte de Hadamard puis nous le mesurons. Nous obtenons ainsi de façon indéterminée 1 ou 0. Nous allons maintenant développer ce circuit en Q# et l'utiliser dans un cas applicatif concret. Je vous recommande d'ailleurs « bono.js » qui permet de créer ses propres circuits et de les convertir automatiquement en Q#.

Exemple en Q#

Allez, il est temps de s'amuser et de parler un peu de Q#. Pourquoi ai-je choisi ce langage ? Pour une raison simple. Etant moi-même programmeur en C# et ayant pleine confiance dans les choix pris par Microsoft, quoi de mieux que de pouvoir associer mon langage de prédilection et ce nouveau langage qui permet de manipuler des bits quantiques. Tout d'abord il va falloir mettre en place notre environnement de développement. Premièrement, il faut télécharger Visual Studio 2019 Community, si vous l'avez déjà c'est parfait. Il faut maintenant mettre à jour les extensions de Visual Studio. Allez dans extensions, manage extensions et recherchez « Microsoft Quantum Development Kit », téléchargez-le et redémarrez votre IDE. Vous pouvez maintenant créer une solution en Q# et plus particulièrement un projet Q# application. La spécificité de ce type de projet est la manière dont il est assemblé. La logique d'architecture diffère grandement d'un projet C# classique. Vous allez retrouver un fichier « Driver.cs ». Celui-ci permet de simuler des Qubits en utilisant la classe « QuantumSimulator ». Nous allons donc utiliser une simulation de quantique. Gros avantage de ce type de projet est le fait de pouvoir connecter ce simulateur directement sur un ordinateur quantique, mais pour le moment il va falloir patienter et attendre la sortie prochaine de Azure Quantum qui mettra à disposition des bits quantiques construits par IonQ. Le deuxième fichier est le plus important. « Operation.qs », où l'extension « qs » signifie tout simplement « qsharp ». Ce fichier va contenir ce que nous appelons une opération, ou plusieurs si nous le souhaitons. En effet, en quantique nous n'appelons pas des méthodes mais des opérations qui vont nous permettre de manipuler des Qubits. Le programme que je vais vous présenter maintenant est une application simple du concept de la superposition. Nous allons faire une loterie dites quantique. Je vais vous expliquer comment écrire une opération qui retournera 1 ou 0, ensuite utiliser ce résultat pour construire des entiers qui seront assemblés en octets.

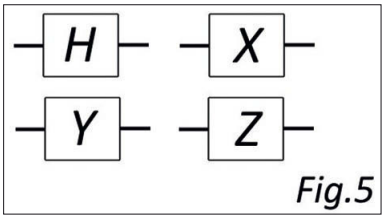


Fig.5

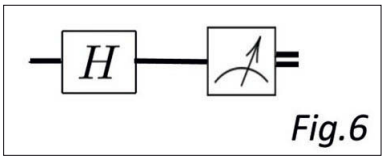


Fig.6

```
operation Superposition () : Result {
    using(q = Qubit()){
        H(q);
        let r = M(q);
        Reset(q);
        return r;
    }
}
```
Fig.7

```
bool[] byteResult = new bool[8];
for (int j = 0; j < byteResult.Length; j++)
{
    var qResult = Superposition.Run(qsim).Result;
    byteResult[j] = QConvert.ResultAsBool.Run(qsim, qResult).Result;
}
```
Fig.8

Cette opération (Fig. 7) que j'ai nommée « Superposition » a pour retour Result. Ce type endémique du Q#, est le résultat d'une mesure projective sur des valeurs propres. Je m'explique. Ici nous n'obtiendrons pas exactement 1 ou 0 mais +1 ou -1 où +1 sera « Zero » et -1 sera « One ». Tout comme le C#, nous pouvons appeler le mot clé « using » sur un Qubit, ensuite nous allons utiliser une opération inclue dans le framework : « H », qui correspond à la porte de Hadamard. Après cette instruction, notre Qubit sera donc dans un état de superposition ; Zero ou One. Maintenant il faut mesurer ce Qubit avec l'instruction « M » et le stocker dans une variable « r ». Le « Reset » sera utilisé pour le remettre dans un état |0>, puis ensuite nous allons

retourner « r ». Nous aurons donc comme retour de cette opération, Zero ou One. Nous allons traiter ce retour dans le fichier « Driver.cs » (Fig. 8).

L'idée est de stocker dans un tableau quatre tableaux de huit bits. Nous allons itérer sur un tableau de bit ou booléen ici et appeler l'opération « Superposition » à chaque tour de boucle. Pour utiliser cette opération, il faut appeler la méthode d'extension « Run » qui prend en paramètres « qsim ». Celui-ci est notre simulateur quantique qui va nous permettre de récupérer un retour « Result ». Une fois ce retour assigné à la variable « qResult », il va falloir le convertir en booléen. Le framework nous fournit la classe « QConvert » qui permet de convertir un « Result » en type souhaité. Nous allons donc utiliser « ResultAsBool », « Run » cette méthode, donner en paramètres le simulateur ainsi que notre « Result » de l'opération « Superposition » : « qResult » et ainsi récupérer ce résultat qui sera stocké dans notre tableau. Une fois la boucle terminée, nous récupérons un tableau de booléen à huit entrées qui va correspondre à notre octet. Si nous reproduisons cette opération quatre fois, nous aurons ainsi quatre tableaux d'octets que nous pourrons convertir en entier. Nous pouvons dire que nous avons effectué ce que nous pourrions qualifier de tirage quantique. Avec cela nous pourrons générer des entiers et ainsi pouvoir faire un tirage de loterie. Vous pouvez retrouver l'intégralité de ce programme ici https://github.com/clementbreuzet/TrueRandom.

Domaines d'applications

Les possibles des domaines d'application de l'informatique sont actuellement réduits, car oui, il faut se le dire, nous n'en sommes qu'au début de cette grande aventure, mais ça va aller vite, très vite. Un des premiers usages que nous verrons se mettre en place est ce que nous appelons la simulation quantique. Imaginez que grâce à notre super ordinateur, qui n'est qu'une représentation de la physique de la matière, nous pourrons la simuler. Ce serait une sorte de machine learning surpuissant pour simuler la matière. Pour illustrer ce concept utilisons comme exemple le procédé d'Haber-Bosch. Celui-ci est utilisé pour produire de l'engrais et représente à lui seul 2% de la consommation d'énergie mondiale. La problématique de ce système, c'est que pour qu'il marche très bien, il faut chauffer l'ammoniac à 500°C. Cela est extrêmement couteux en énergie. Et là, vous posez cette question à l'ordinateur quantique : « Existe-t-il un meilleur catalyseur pour l'ammoniac ? » ce qui permettrait de le chauffer d'une autre manière et ainsi économiser de l'énergie. Même genre de question pour les matériaux supraconducteurs avec lesquels nous pouvons reproduire cette fameuse expérience de lévitation quantique ou construire nos Qubit. Ce phénomène n'étant observable qu'à des températures extrêmement basses, nous pouvons nous poser cette question : « Existe-t-il un matériau supraconducteur fonctionnant à température ambiante ? ». Les questions sont infinies, et libre aux chimistes et physiciens de définir et d'intégrer toutes connaissances de la physique et simuler la matière sans avoir à reproduire l'expérience en laboratoire, ce qui se fait actuellement. Les découvertes seront nombreuses, mais laissons cette partie aux physiciens et regardons à quoi cela va servir dans notre domaine. Deux algorithmes importants vont drastiquement changer le cryptage et le tri. L'algorithme de Shor permet de factoriser de grands nombres premiers ce qui rend vulnérable la clé RSA. En effet cette clé est le produit de deux grands nombres premiers que nous considérons inviolables par un ordinateur classique. Pour l'instant cet algorithme n'est que de la théorie, car nous ne manipulons pas assez de Qubits pour l'exécuter. Mais un jour, lorsqu'il sera possible de le mettre en place, nous pourrons introduire le cryptage quantique BB84 et définitivement enterrer la clé RSA.

Le deuxième algorithme est celui de Grover, qui permet de trier des listes gigantesques. Les circuits quantiques de ces deux algorithmes existent sur internet et vous pouvez librement les consulter.

Lorsque tout cela sera au point et que les calculateurs quantiques pourront effectuer des calculs que nos machines actuelles ne peuvent pas effectuer, nous aurons atteint ce que nous appelons la suprématie quantique et une nouvelle page de l'informatique sera ouverte.

Benoît Prieur *est développeur indépendant pour sa société Soarthec. Il est par ailleurs l'auteur du livre Informatique quantique - De la physique quantique à la programmation quantique en Q# publié aux éditions ENI en février 2019. Il a également assuré en octobre-novembre 2019 des cours pratiques en informatique quantique auprès d'élèves ingénieurs de l'ECE Paris.*

Programmation quantique : comprendre l'essentiel pour écrire et exécuter ses premiers programmes avec IBM QisKit

L'année passée a vu la presse grand public s'intéresser de plus en plus à l'informatique quantique. Le point d'orgue de l'année 2019 ayant probablement été en la matière l'annonce de l'atteinte de la suprématie quantique par l'entreprise Google. Le but du présent article est de rappeler succinctement les grands principes qui régissent la programmation quantique pour ensuite utiliser une des plateformes grand public permettant d'expérimenter la programmation quantique. Nous utiliserons la plateforme IBM QiSKit dont l'une des déclinaisons est utilisable complètement en ligne. QisKit utilise le langage Python mais peut également utiliser Swift et Javascript. Ici nous nous contenterons de travailler avec Python.

Quelques notions de physique quantique

Indifféremment nommé qubit ou qu-bit, il représente l'unité de base en informatique quantique à l'instar du bit en informatique dite classique. Ce bit quantique est associé à une particule qui se trouve être dans un certain état nommé **état quantique**. Cet état quantique se définit grâce à un phénomène propre à la physique quantique appelé la **superposition quantique**. Avant d'expliquer plus avant ce phénomène, précisons que la superposition quantique s'interrompt quand est effectuée la **mesure quantique**. Ainsi un bit quantique est dans un état quantique qui peut être vu comme une combinaison linéaire des différentes valeurs possibles après mesure. Cette combinaison linéaire des différentes valeurs possibles après mesure concrétise ce qu'est la superposition quantique. Une fois la mesure quantique effectuée, une seule valeur de celle qui était possible précédemment, est conservée.

Histoire de rendre tout cela plus concret, reprenons la célèbre expérience de la pensée, le chat de Schrödinger. Celui-ci va nous permettre d'illustrer chaque notion évoquée, puis d'introduire une notation utilisée pour représenter un état quantique, la **notation bra-ket**.

L'expérience du chat de Schrödinger

Erwin Schrödinger énonce les données de son expérience ainsi. Dans une boîte close et opaque se trouvent plusieurs éléments.

- un atome radioactif.
- une fiole de poison mortel.
- un chat.

Si l'atome se désintègre, alors la fiole se retrouve ouverte, le poison se diffuse alors et le chat meurt instantanément. L'atome peut se désintégrer à n'importe quel moment sans qu'à l'extérieur de la boîte on puisse en avoir connaissance. Du point de vue d'un observateur extérieur, il n'y a aucun moyen de savoir si l'atome s'est désintégré et donc de savoir si le chat est mort ou vivant.

D'un point de vue quantique, on peut donc envisager le chat comme étant à la fois *vivant* et à la fois *mort*. On peut en effet établir une combinaison linéaire de l'**état quantique** du chat comme étant une superposition des états finaux après mesure quantique. Ici la **mesure quantique** est ni plus ni moins que l'ouverture de la boîte par un observateur. Une fois ouverte la boîte, le chat est soit mort soit vivant. Les deux états possibles sont en effet *vivant* et *mort*. Tant que la boîte n'est pas ouverte, l'état quantique est donc bien une combinaison des deux **états superposés** *vivant* et *mort*.

En **notation bra-ket**, on écrit un état quantique ψ (psi) du chat de la manière suivante.

$$|\Psi_{chat}\rangle = a_{vivant}.|vivant\rangle + a_{mort}.|mort\rangle$$

Si on considère les deux états superposés du chat (*vivant* et *mort*) comme étant équiprobables alors l'état quantique du chat s'écrit comme ci-dessous.

$$|\Psi_{chat}\rangle = \frac{1}{\sqrt{2}}.|vivant\rangle + \frac{1}{\sqrt{2}}.|mort\rangle$$

Note - *le coefficient multiplicateur devant chaque état dans la précédente équation est égal à la racine carrée de la probabilité associée (ici chaque probabilité est égale à 0.5). En effet:*

$$probabilité(|\text{vivant}\rangle) = probabilité(|\text{mort}\rangle) = (a_i)^2 = \frac{1}{2}$$

Notation bra-ket et mise en évidence de la combinatoire

Prenons un premier exemple à un qubit. Son état quantique peut s'écrire comme une combinaison linéaire des deux états mesurables.

$$|\Psi\rangle = \beta.|0\rangle + \gamma.|1\rangle$$

On peut d'ailleurs écrire chacune des deux valeurs mesurables ainsi.

$$|0\rangle = 1.|0\rangle + 0.|1\rangle$$
$$|1\rangle = 0.|0\rangle + 1.|1\rangle$$

Avec deux qubits, on a quatre valeurs possibles mesurables. Ce sont les valeurs suivantes.

$$|00\rangle$$

$|01\rangle$

$|10\rangle$

$|11\rangle$

On a en effet la combinaison linéaire suivante avec les coefficients α, β, γ et δ.

$$|\Psi\rangle = \alpha.|00\rangle + \beta.|01\rangle + \gamma.|10\rangle + \delta.|11\rangle$$

Comme précédemment on peut exprimer sous forme d'équations chacune des valeurs finalement mesurables.

$$|00\rangle = 1.|00\rangle + 0.|01\rangle + 0.|10\rangle + 0.|11\rangle$$

$$|01\rangle = 0.|00\rangle + 1.|01\rangle + 0.|10\rangle + 0.|11\rangle$$

$$|10\rangle = 0.|00\rangle + 0.|01\rangle + 1.|10\rangle + 0.|11\rangle$$

$$|11\rangle = 0.|00\rangle + 0.|01\rangle + 0.|10\rangle + 1.|11\rangle$$

Si l'on passe à trois qubits on aura huit valeurs possibles. De manière générale, une solution à n qubits verra la possibilité de travailler avec 2^n (2 exposant n). On a donc accès à une combinatoire exponentielle qui permet du moins en théorie de résoudre des problèmes très longs à résoudre en informatique classique. Toutefois, à chaque problème, il faut ou faudra inventer un algorithme quantique *adhoc*. Ce type d'algorithme prend en entrée un ou plusieurs qubits qui seront ensuite soumis à un circuit quantique. Ce circuit quantique est composé de portes quantiques et se termine systématiquement par une mesure quantique. L'informatique classique et son algèbre de Boole manipule des portes logiques. En informatique quantique, on utilise des portes quantiques. Les états quantiques étant finalement des états probabilistes, on soumet en général un circuit quantique plusieurs fois aux mêmes entrées. On analyse ensuite la répartition du résultat des 1000 ou 2000 mesures (une mesure par utilisation du circuit quantique).

Avant de commencer à définir et à utiliser des circuits quantiques avec IBM QiSkit, nous allons en quelques lignes expliquer une autre notion de physique quantique particulièrement utile en informatique quantique. L'intrication (ou l'enchevêtrement) quantique.

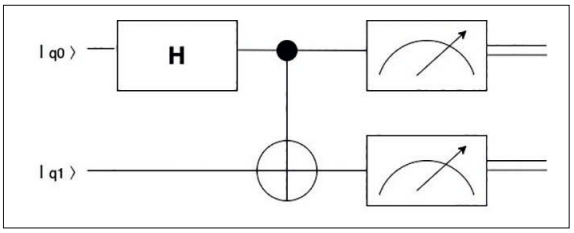
L'intrication quantique

L'intrication quantique est un phénomène physique assez étonnant dont l'existence fut longtemps controversée au sein de la communauté scientifique. Il a été depuis démontré théoriquement et reproduit à plusieurs reprises de façon expérimentale. Il consiste en la mise en évidence d'un lien entre deux particules, pourtant possiblement très éloignées l'une de l'autre. Deux particules A et B qui sont intriquées ont le même état quantique. Si l'une des deux change d'état, l'état de la seconde particule sera le même. Cela implique enfin que le résultat de la mesure respective de deux particules intriquées est le même.

Ce phénomène est fréquemment utilisé en informatique quantique. Pour coder ce qu'on appelle la téléportation quantique ou encore pour simuler les états de Bell qui sera l'objet de notre exemple de code avec IBM QisKit.

Les états de Bell

Un des moyens d'implémenter l'intrication de deux particules, donc de deux qubits, est de recourir aux états de Bell. Ceux-ci permettent de maximiser l'intrication entre les deux qubits. Ainsi, si on sollicite 1000 fois le circuit quantique qui permet cela, la mesure finale nous indiquera 1000 fois que les deux qubits on la même valeur.



Un circuit quantique bien déterminé et bien connu permet d'implémenter les états de Bell. Il implique l'utilisation de deux portes quantiques, la porte de Hadamard et la porte C-NOT.

Le schéma précédent prend deux qubits en entrée (q0 et q1) soumis à une porte de Hadamard puis à une porte C-NOT pour ensuite aborder la mesure des deux sorties. En théorie, les deux valeurs mesurées doivent être identiques.

La plateforme IBM QisKit

La plateforme IBM QisKit permet de s'essayer à l'informatique quantique de plusieurs manières. En premier lieu, QisKit s'installe et s'utilise localement indifféremment sous Linux, Windows ou macOS. L'url suivante détaille la procédure à suivre pour installer QiSKit sur votre machine :

https://qiskit.org/documentation/install.html

La plateforme permet également de travailler entièrement en ligne. C'est la solution que nous utiliserons ici. D'une part il est mis à disposition un éditeur graphique de circuits quantiques nommé *Circuit Composer* (nous en dirons un mot), et, d'autre part, il est possible de coder et d'exécuter tous ses programmes quantiques en Python dans des *notebooks Jupyter* que nous allons largement utiliser ici.

Une fois défini votre circuit quantique (soit avec *Circuit Composer* soit au sein d'un *notebook Jupyter*), on peut exécuter le programme. IBM QisKit permet de soumettre le programme à deux types de solveurs quantiques.

- des simulateurs de machines quantiques conçus avec un ordinateur classique qui sont supposés donner un résultat exact d'un point de vue quantique.
- de vraies machines quantiques. L'exécution se retrouve alors dans une file d'attente. En effet ces machines quantiques sont utilisables par tous les utilisateurs IBM. Cette possibilité est susceptible de donner des résultats partiellement faux, les solveurs quantiques étant particulièrement instables.

Note - *les notebooks Jupyter permettent la programmation interactive faisant alterner du code Python et le résultat graphique ou non de son exécution. L'extension d'un fichier de notebook Jupyter est .ipynb. Un tel fichier peut facilement être partagé y compris sur des sites de gestion de développement comme Github qui savent afficher un fichier .ipynb.*

Pour utiliser QisKit en ligne commencez par vous créer un compte IBM en ligne à l'url suivante :

https://quantum-computing.ibm.com/

Une fois créé votre compte, vous accédez à votre espace muni d'une barre latérale à gauche de l'écran qui vous permet de navi-

guer parmi les différents outils mis à disposition. Ci-dessous une copie d'écran de la barre latérale incluant les liens suivants.

- *Dashboard* qui fait office de page d'accueil.
- *Results* qui permet de consulter les exécutions passées et leurs résultats.
- *Circuit Composer*, qui permet de réaliser un circuit quantique de manière graphique.
- *QisKit Notebooks*, qui permet de travailler avec des *notebooks Jupyter*. 1

Le *Circuit Composer* de QisKit

Cet outil permet de composer son circuit quantique en faisant glisser les différentes portes quantiques de notre programme. Ainsi en quelques clics on réalise le schéma suivant en faisant glisser une porte de Hadamard (H), une porte C-NOT ainsi que deux portes de mesure. 2

Au-dessus du schéma 3 ainsi composé, se trouve un bouton *Run* dans lequel on peut sélectionner un solveur ainsi que le nombre d'utilisations du circuit quantique. Nous choisissons donc de lancer 1024 essais à l'aide d'un des simulateurs quantiques mis à disposition.

Après quelques secondes on obtient un résultat graphique qu'il s'agit d'interpréter. 4

Note – *On n'interprète que les deux derniers digits de chaque résultat. En l'occurrence les trois premiers zéros (000) dans 00011 ne sont pas signifiants ; les deux derniers digits correspondant aux deux mesures de qubits de sortie.*

On voit qu'on obtient 49,121 % des essais (503 essais) qui donnent le résultat suivant dans lequel les deux qubits de sortie sont identiques (à zéro).

$$|00\rangle$$

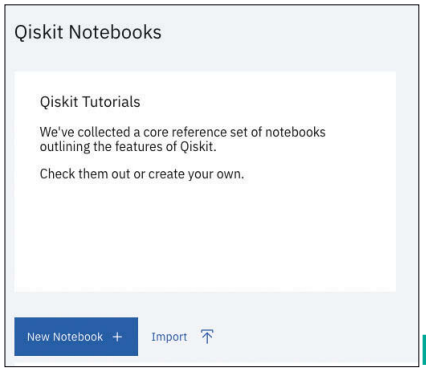
On obtient également 50,879 % des essais (521 essais) qui donnent le résultat suivant dans lequel les deux qubits de sortie sont identiques (à un).

$$|11\rangle$$

C'est dans ce sens que l'on atteint un maximum d'intrications. Les deux mesures sont toujours de valeurs identiques entre elles. En effet, nous n'obtenons jamais les deux situations suivantes.

$$|01\rangle$$

$$|10\rangle$$



5

Les *notebooks Jupyter* dans QisKit

Passons à présent à l'utilisation de *Jupyter*. Nous allons reproduire le même exemple mais cette fois avec du code Python. Nous lancerons une simulation similaire à celle qui précède puis nous exécuterons notre programme quantique sur une vraie machine quantique située à Melbourne. Commençons à créer un nouveau notebook Jupyter en cliquant sur le bouton *New Notebook*. 5

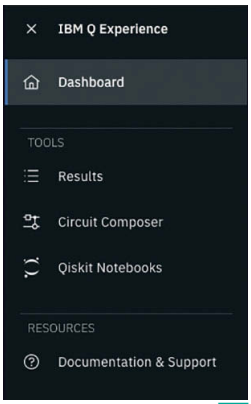
On commence ensuite à coder une première section du *notebook* que nous détaillons tout de suite.

On commence par écrire la clause *%matplotlib inline* nécessaire dès l'instant que l'on travaille avec des *notebooks Jupyter*.

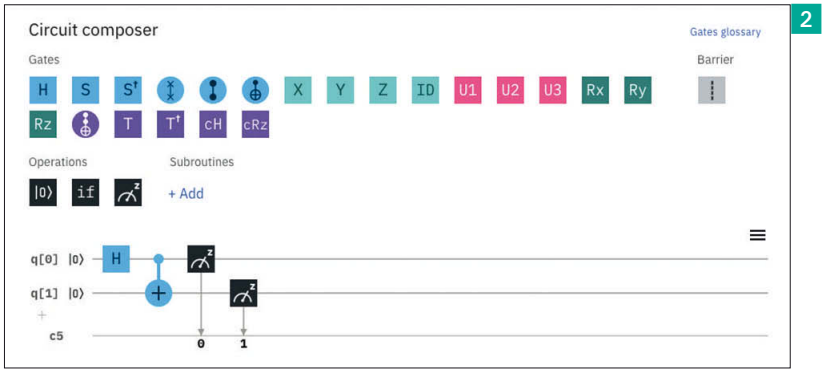
```
%matplotlib inline
```

Puis on déclare les différents modules dont on aura besoin à commencer par qiskit. On importe également numpy et plot_histogram pour avoir un résultat graphique.

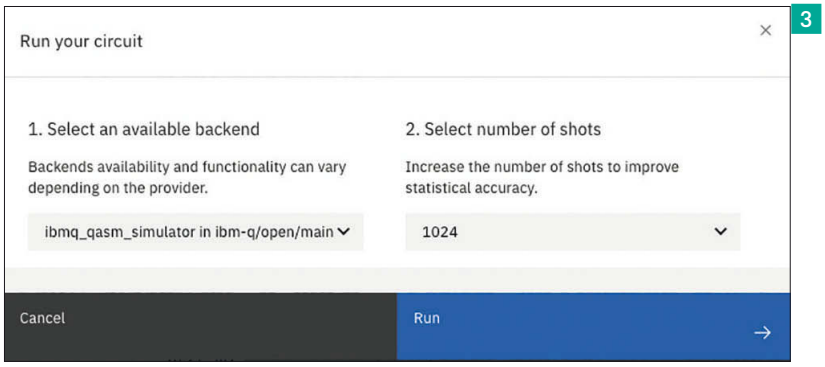
```
import qiskit

from qiskit import(
 IBMQ,
 ClassicalRegister,
 QuantumCircuit,
 QuantumRegister,
```



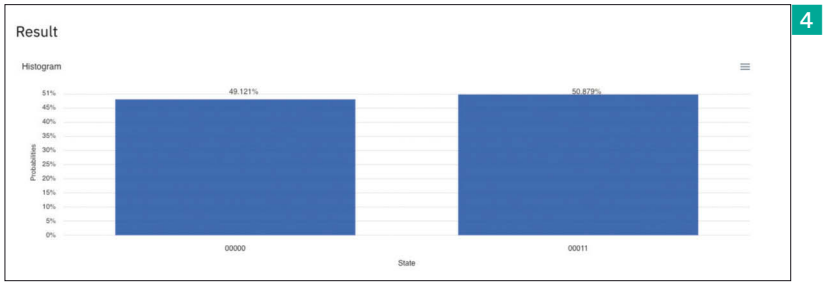
1



2



3



4

PROGRAMMEZ! Le magazine des développeurs

Offres printemps 2020

Nos classiques

1 an 11 numéros	49€*
2 ans 22 numéros	79€*
Etudiant 1 an - 11 numéros	39€*

** Tarifs France métropolitaine*

Abonnement numérique

PDF 1 an - 11 numéros	35€
Option : accès aux archives	15€

Souscription uniquement sur
www.programmez.com

1 an

11 numéros
+ Python et l'analyse forensique (éditions ENI)

50€*

2 ans

22 numéros
+ Python et l'analyse forensique (éditions ENI)

80€*

** Offres limitées à la France Métropolitaine*

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

ABONNEMENT à retourner avec votre règlement à :
Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex.

☐ **Abonnement 1 an :** 49 €
☐ **Abonnement 2 ans :** 79 €
☐ **Abonnement 1 an Etudiant :** 39 €
Photocopie de la carte d'étudiant à joindre

OFFRES PINTEMPS 2020
☐ **Abonnement 1 an :** 50 €
☐ **Abonnement 2 ans :** 80 €

PROG 239
Valable jusqu'au 30 avril 2020

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

email indispensable pour l'envoi d'informations relatives à votre abonnement

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !
☐ Je souhaite régler à réception de facture

*** Tarifs France métropolitaine**

```
QuantumCircuit,
execute,
Aer)

import numpy as np

from qiskit.visualization import plot_histogram
```

Puis on définit notre circuit quantique à l'aide de la fonction *QuantumCircuit* dont la documentation est à cette url :
https://qiskit.org/documentation/api/qiskit.circuit.QuantumCircuit.html
On a besoin de deux bits quantiques et en quelque sorte de deux bits classiques correspondant à l'emplacement de la mesure de chaque bit quantique de sortie.

```
circuit = QuantumCircuit(2, 2)
```

Puis on adjoint une porte de Hadamard sur la première ligne de bit quantique.

```
circuit.h(0)
```

On ajoute ensuite notre porte quantique C-NOT entre la première ligne de bit quantique et la seconde ligne.

```
circuit.cx(0, 1)
```

Puis on ajoute les deux unités de mesure quantique dont on interprétera les résultats.

```
circuit.measure([0,1], [0,1])
```

Pour l'instant on a juste défini un circuit quantique que l'on va dessiner à l'écran avant de l'exécuter.

```
circuit.draw()
print(circuit)
```

6

```
circuit.draw(output='mpl', filename='circuit.png')
```

Ci-dessous la copie d'écran 7 de la première section de notre notebook.
Quand on l'exécute on obtient deux représentations de notre circuit quantique. 7
Nous allons à présent exécuter ce programme quantique. La seconde section du notebook concernera l'exécution du programme sur un simulateur quantique supposé donner un résultat parfait d'un point de vue quantique. La troisième section concernera l'exécution sur une vraie machine quantique.
Voici le code commenté de la deuxième section :
On déclare un simulateur qui sera en charge de l'exécution.

```
simulator = Aer.get_backend('qasm_simulator')
```

On exécute notre circuit quantique pour 1000 essais.

```
job = execute(circuit, simulator, shots=1000)
```

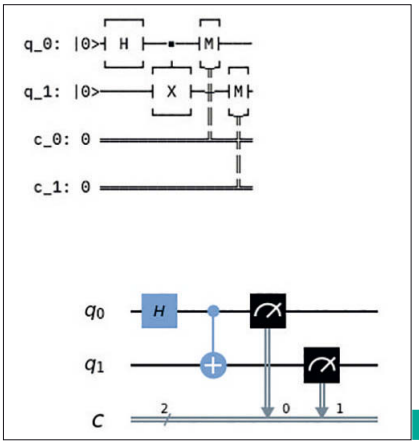
On obtient le résultat de la simulation.

```
result = job.result()
counts = result.get_counts(circuit)
```

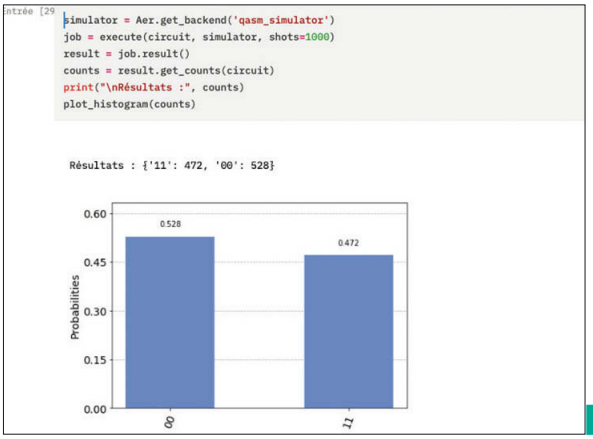
Enfin on affiche un histogramme synthétisant les résultats.

```
%matplotlib inline
import qiskit
from qiskit import(
  IBMQ,
  ClassicalRegister,
  QuantumCircuit,
  QuantumRegister,
  QuantumCircuit,
  execute,
  Aer)
import numpy as np
from qiskit.visualization import plot_histogram

circuit = QuantumCircuit(2, 2)
circuit.h(0)
circuit.cx(0, 1)
circuit.measure([0,1], [0,1])

# Circuit
circuit.draw()
print(circuit)
circuit.draw(output='mpl', filename='circuit.png')
```

6



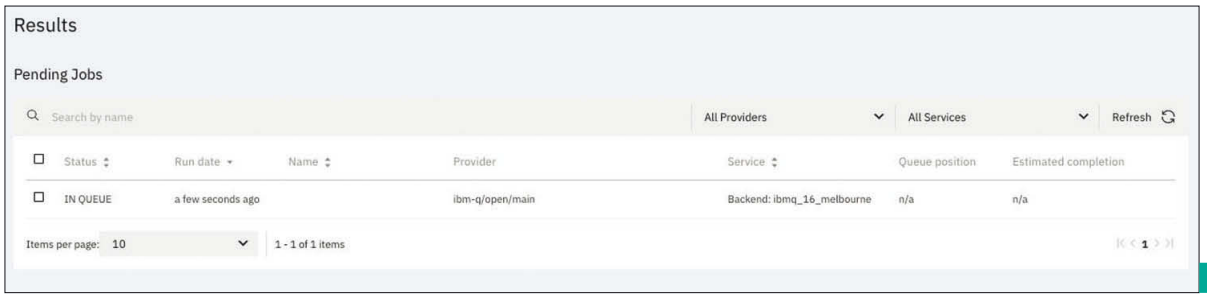
7



8

```
print("\nRésultats :", counts)
plot_histogram(counts)
```

Ci-dessus la copie d'écran de la deuxième section de notre notebook ainsi que son résultat obtenu quasi immédiatement. 8
Les résultats uniquement en 00 ou en 11 confirment que l'on est bien en intrication maximale de manière parfaite (en effet aucun



9

artefact 01 ou 10 n'est présent même de façon minime dans les résultats). Passons à présent à la troisième section qui utilise une vraie machine quantique.
On commence par déclarer cette machine quantique localisée à Melbourne.

```
provider = IBMQ.get_provider(group='open')
device = provider.get_backend('ibmq_16_melbourne')
```

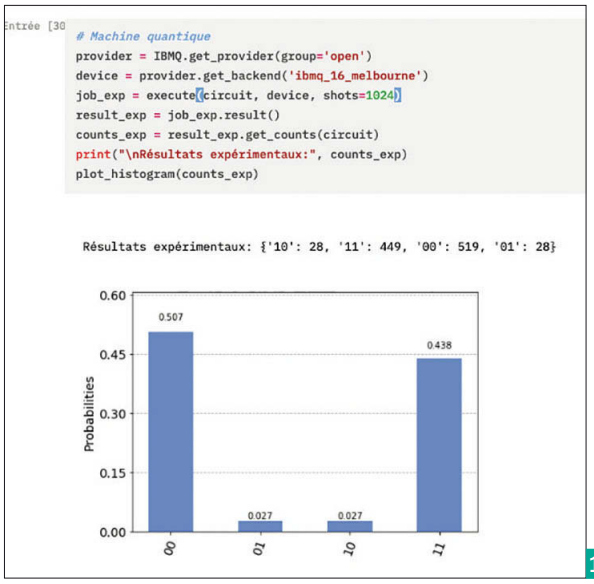
Puis on exécute notre circuit quantique sur cette machine.

```
job_exp = execute(circuit, device, shots=1024)
```

Et comme dans la deuxième section on obtient puis on met en forme les résultats.

```
result_exp = job_exp.result()
counts_exp = result_exp.get_counts(circuit)
print("\nRésultats expérimentaux:", counts_exp)
plot_histogram(counts_exp)
```

Les résultats ne sont toutefois pas immédiats. Ce nouveau calcul est placé dans la file d'attente de la machine – nous ne sommes pas les seuls à la solliciter. On peut voir peu après son lancement, son état et sa présence dans la file d'attente dans l'onglet *Results*. 9
Ci-contre la copie d'écran de la troisième section de notre notebook ainsi que son résultat obtenu après quelques minutes. 10
On voit donc que la machine quantique fonctionne plutôt bien mais a quelques défauts et approximations. Ainsi un peu moins de 6% des tirages ont conduit à obtenir 01 ou 10 ce qui est en contradiction avec le principe même de l'intrication maximale. Le simulateur quantique utilisé précédemment ne produisait pas de tels artefacts. Certes cette exécution montre un défaut (faible) mais elle a été lancée sur une vraie machine quantique ce qui est tout à fait exaltant.



10

A vous de jouer à présent dans ce monde fascinant qu'est celui de l'informatique quantique. Qiskit constituant un environnement particulièrement accessible pour débuter.

Olivier Hess
IBM Q France Leader & IBM Q Ambassador

Jean-Michel Torres
IBM Q France

Xavier Vasques
Director of European IBM Systems Center | Montpellier

Une introduction au calcul quantique

C'est à partir des années 1980, sous l'impulsion du physicien et prix Nobel Richard Feynman que germe l'idée d'ordinateur quantique, ouvrant la voie à des traitements informatiques dont la complexité est hors de portée des ordinateurs actuels.

Voici un tour d'horizon de cette technologie. Quels sont les principes du calcul quantique ? pour quelles applications ? Où en sommes-nous ? Où allons-nous et pour quelles perspectives ?

Aux origines, la physique quantique 1

Au début du XX^{e} siècle la physique ne parvient pas à expliquer un certain nombre d'observations. Elle va devoir évoluer dans un premier temps vers une « nouvelle mécanique » qui deviendra « mécanique ondulatoire » et finalement « mécanique quantique ».

L'enjeu est de mieux décrire la structure fondamentale de la matière et l'évolution dans le temps et dans l'espace des phénomènes de l'infiniment petit.

Max Planck et Albert Einstein furent les premiers à comprendre que les échanges d'énergie lumineuse ne pouvaient se faire que par « paquet » et non pas avec n'importe quelle valeur. Un peu comme un escalier ne permet pas d'atteindre une hauteur intermédiaire entre deux de ses marches. D'ailleurs, Albert Einstein obtient le prix Nobel de physique suite à la publication de sa théorie sur l'aspect quantifié des échanges d'énergie en 1921.

Niels Bohr étendit les postulats quantiques de Planck et d'Einstein de la lumière à la matière, en proposant un modèle reproduisant le spectre de l'atome d'hydrogène. Il obtient le prix Nobel de physique en 1922.

De 1925 à 1927, toute une série de travaux de plusieurs physiciens et mathématiciens donna corps à deux théories générales applicables à ces problèmes :

- La mécanique ondulatoire de Louis de Broglie et Erwin Schrödinger ;
- La mécanique matricielle de Werner Heisenberg, Max Born et Pascual Jordan.

Ces deux mécaniques furent unifiées par Erwin Schrödinger du point de vue physique, et par John von Neumann du point de vue mathématique. Enfin, Paul Dirac formula la synthèse et la généralisation de ces deux mécaniques, que l'on nomme aujourd'hui la mécanique quantique et dont l'équation fondamentale est l'équation de Schrödinger :

$$H(t)\,|\,\psi(t)\rangle = i\hbar\frac{d}{dt}\,|\,\psi(t)\rangle$$

1 *Congrès de Solvay 1927*

De l'informatique à l'informatique quantique

Avant d'aborder le développement d'une "Théorie de l'information Quantique", revenons sur le fonctionnement d'un ordinateur standard, et la « Théorie de l'information classique », celle qui régit le fonctionnement des ordinateurs tels que nous les connaissons aujourd'hui.

Les premiers ordinateurs binaires furent construits dans les années 40 : Colossus (1943) puis ENIAC (IBM - 1945). Colossus a été conçu pour déchiffrer des messages secrets allemands et l'ENIAC pour calculer des trajectoires balistiques. L'ENIAC (Electronic Numerical Integrator And Computer), est le premier ordinateur entièrement électronique construit pour être « Turing-complet » : il peut être reprogrammé pour résoudre, en principe, tous les problèmes calculatoires. L'ENIAC a été programmé par des femmes, dites les « femmes ENIAC ». Les plus célèbres d'entre elles étaient Kay McNulty, Betty Jennings, Betty Holberton, Marlyn Wescoff, Frances Bilas et Ruth Teitelbaum. Ces femmes avaient auparavant effectué des calculs balistiques sur des ordinateurs de bureau mécaniques pour l'armée. L'ENIAC pèse alors 30 tonnes, occupe une surface de 72 m^2 et consomme 140 kilowatts.

Quelle que soit la tâche effectuée par un ordinateur, le processus est toujours le même : une instance de la tâche est décrite par un algorithme qui est traduit en une suite de 0 et de 1, pour donner lieu à l'exécution dans le processeur, la mémoire et les dispositifs d'entrée/sortie de l'ordinateur. Les opérations s'effectuent avec des opérations binaires simples à partir desquelles il est possible de construire des opérations beaucoup plus complexes que les ordinateurs peuvent effectuer des millions de milliards de fois par seconde pour les plus puissants d'entre eux. Tout cela est devenu tellement « naturel » que l'on oublie totalement que chaque transaction sur un serveur informatique, sur un PC, une calculette, un smartphone se décompose en ces « opérations binaires élémentaires » ...

Dans un ordinateur, ces 0 et 1 sont contenus dans des « BInary digiTs » ou « bits » qui représentent la plus petite quantité d'information contenue dans un système informatique. Pour un ordinateur quantique le « qubit (quantum bit) » est l'entité qui repré-

sente la plus petite entité permettant de manipuler de l'information. Il possède deux propriétés fondamentales de la Mécanique Quantique : **Superposition & Intrication.**

Superposition quantique

Un objet quantique (à l'échelle microscopique) peut exister dans une infinité d'états (tant qu'on ne mesure pas cet état). Un qubit peut donc exister dans n'importe quel état entre 0 et 1. Il peut avoir à la fois la valeur 0 et la valeur 1, ou plutôt « une certaine quantité de 0 et une certaine quantité de 1 », comme une combinaison linéaire de deux états notés |0> et |1>, avec les coefficients α et β.

Donc là où un bit classique ne décrit « que » 2 états (0 ou 1), le qubit peut en représenter une « infinité » !! C'est un des avantages potentiels du calcul quantique du point de vue de la théorie de l'information.

La superposition d'états est représentée sous la forme suivante :

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

On peut se faire une idée de la superposition d'états en utilisant l'analogie du ticket de loterie : **un ticket de loterie est soit gagnant, soit perdant**… une fois que l'on connaît le résultat du jeu. Par contre, avant le tirage, ce ticket **n'était ni gagnant, ni perdant**. Il a une certaine probabilité d'être gagnant et une certaine probabilité d'être perdant, il est en quelque sorte gagnant et perdant à la fois (plutôt plus perdant que gagnant en général). Dans le monde quantique, les caractéristiques des particules peuvent être sujettes à cette indétermination : par exemple, la position d'une particule est incertaine. Avant la mesure, la particule n'est ni au point A, ni au point B. Elle a une certaine probabilité d'être au point A et une certaine probabilité d'être au point B. Cependant, après la mesure, l'état de la particule est bien défini : elle est au point A ou au point B.

Intrication quantique

Lorsque l'on considère un système composé de plusieurs qubits, il peut leur arriver de « lier leur destin » c'est-à-dire de ne pas être indépendants l'un de l'autre même s'ils sont séparés dans l'espace (alors que les bits « classiques » sont complètement indépendants les uns des autres). C'est ce que l'on appelle l'intrication quantique. Pour un système de deux qubits intriqués la mesure de l'état d'un de ces deux qubits nous donne une indication immédiate sur le résultat d'une observation sur l'autre qubit.

Pour illustrer naïvement cette propriété on peut là aussi utiliser une analogie : imaginons deux ampoules, chacune dans deux maisons différentes. En les intriquant, il devient possible de connaître l'état d'une ampoule (allumée ou éteinte) en observant simplement la seconde, car les deux seraient liées, intriquées. Et cela, immédiatement et même si les maisons sont très éloignées l'une de l'autre (dans des galaxies différentes peut-être).

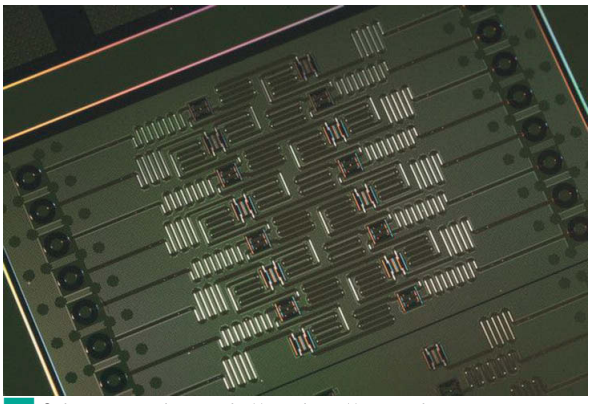
Ce phénomène d'intrication permet de décrire des corrélations entre les qubits. Si on augmente le nombre de qubits, le nombre de ces corrélations augmente exponentiellement : pour n qubits il y a 2^n corrélations. Si n est grand alors 2^n est un nombre gigantesque. A titre d'exemple, avec 300 qubits, on aurait besoin, pour écrire toutes ces corrélations d'autant de chiffres qu'il n'y a d'atomes dans l'univers visible ! C'est cette propriété qui confère à l'ordinateur quantique la possibilité d'effectuer des manipulations sur des quantités gigantesques de valeurs, quantités hors d'atteinte d'un ordinateur classique.

La construction d'un ordinateur quantique : une série de défis technologiques 2

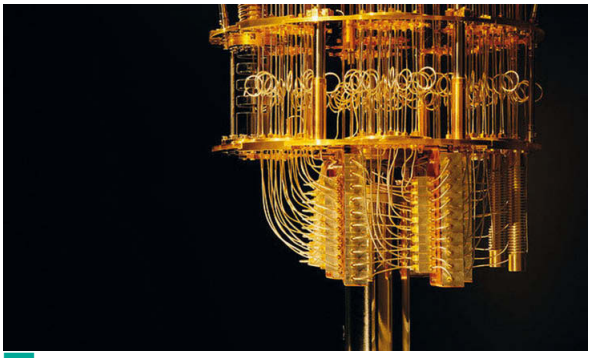
Du point de vue technologique, il existe plusieurs manières de constituer des qubits, ils peuvent être faits d'atomes, de photons, d'électrons, de molécules ou de métaux supraconducteurs.

IBM a choisi d'utiliser des qubits « Supraconducteurs », construits avec des oxydes d'aluminium, dits « qubits transmons ». Pour permettre et garantir les effets quantiques (superposition et intrication) les qubits doivent être refroidis à une température aussi proche que possible du zéro absolu (soit environ -273° C). On parle de 20 milliKelvin ! 3

IBM a démontré la capacité de concevoir un qubit unique en 2007 et en 2016 a annoncé la mise à disposition dans le Cloud d'un premier système physique opérationnel doté de 5 qubits et d'un environnement de développement « QISKit » (Quantum Information Science Kit), permettant de concevoir, tester et optimiser des algorithmes pour des applications commerciales



2 *Qubits supraconducteurs d'IBM (crédit : IBM Research)*



2 *Machine IBM à 53 qubits (crédit : IBM Research)*

et scientifiques. Mise en ligne dès le mois de mai 2016, l'initiative « IBM Quantum Experience » constitue une première dans le monde industriel.

L'offre actuelle d'IBM repose sur des systèmes physiques à 20 qubits et 53 qubits. La volonté d'IBM est de favoriser l'adoption du calcul quantique et d'accompagner ses clients et partenaires dans cette révolution technologique. Ainsi depuis 2017, IBM construit un écosystème d'entreprises, d'universités, d'organismes de recherche et de startups : le « IBM Quantum Network » qui comporte plus de 100 membres.

Cette offre dans le Cloud donne par ailleurs accès gratuitement à un émulateur à 32 qubits ainsi qu'un certain nombre de systèmes à 5 et 14 qubits.

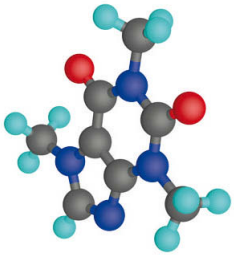
Augmenter le nombre de qubits ? Oui mais cela ne suffit pas

On parle de « volume quantique » comme mesure pertinente de la performance et des progrès technologiques. On définit aussi « l'avantage quantique » comme le point à partir duquel les applications du calcul quantique offriront un avantage pratique significatif qui dépasse les capacités des seuls ordinateurs classiques. Le concept de

volume quantique a été introduit par IBM en 2017. Il commence à se généraliser auprès d'autres constructeurs.

Le volume quantique est déterminé par divers facteurs, dont le nombre de qubits, la connectivité, le temps de cohérence, ainsi que la prise en compte des erreurs sur les portes quantiques et des erreurs de mesure, la connexion entre qubits et l'amélioration des couches logicielles.

Bien évidemment, il faut pouvoir exécuter des tâches sur ces machines, c'est pourquoi IBM a développé une bibliothèque de programmation spécifique appelée QISKit (Quantum Information Science Kit). Il s'agit d'une librairie open-source pour le langage Python, disponible sur qiskit.org. Son développement est très actif, l'ensemble des contributeurs, dont IBM, fait régulièrement évoluer les fonctionnalités de cet environnement de programmation 4



5 *La molécule de caféine*

Le calcul quantique, pour quoi faire ?

Un ordinateur quantique n'a pas pour objectif de rendre les mêmes services qu'un ordinateur classique. Quelles classes de problèmes vont tirer parti de ces futurs ordinateurs ? Commençons par quelques exemples simples :

Pour un ordinateur classique la multiplication est une opération simple : 7 x 3 = 21, 6739 x 892721 = 6016046819, et cela même pour de très grands nombres. Mais le problème inverse est nettement plus complexe. Connaissant un grand nombre (par exemple 7859324261 il est difficile de trouver P et Q tels que : P x Q = 7859324261.

Les techniques courantes de cryptographie reposent sur cette difficulté. Pour un tel problème, on estime qu'un problème qui durerait 1000 jours sur un ordinateur classique, pourrait être résolu en quelques dizaines de secondes sur une machine quantique. On parle alors d'accélération exponentielle.

Un autre domaine dans lequel le calcul quantique pourra apporter les premières contributions significatives est celui de la science des matériaux et de la chimie.

En effet, un ordinateur quantique imite la manière dont la nature « traite » l'information, lui permettant de simuler, de comprendre et améliorer notre compréhension de la nature comme les molécules et leurs interactions ouvrant la possibilité de très grandes découvertes en sciences des matériaux. Des nouveaux types de matériaux seront également possibles comme par exemple des supraconducteurs à température ambiante, ce qui pourrait à l'avenir avoir un impact gigantesque sur les problèmes d'énergie dans le monde. 5

Prenons un autre exemple : aucun ordinateur classique au monde ne peut calculer de façon exacte (c'est-à-dire sans aucune approximation) les états d'énergie de la molécule de caféine, pourtant de taille moyenne avec une quarantaine d'atomes, c'est un problème trop complexe... Aujourd'hui les ordinateurs quantiques sont utilisés pour traiter des problèmes de chimie simples c'est à dire avec un petit nombre d'atomes mais l'objectif est de pouvoir adresser des molécules beaucoup plus complexes.

A un peu plus long terme, c'est l'ensemble des domaines où le calcul fait appel à des algorithmes complexes au sens où le temps de calcul ou la taille mémoire sont tels qu'il n'est pas faisable dans un temps « raisonnable » (par exemple il n'est pas raisonnable, ni utile de passer 10 000 ans pour trouver le code d'une carte de paiement).

Dans ces catégories de problème « éligibles aux ordinateurs quantiques » on trouve beaucoup de cas d'optimisation, dans les domaines logistiques (plus court chemin), de la finance (estimation de risques, évaluation de portefeuilles d'actifs), du marketing (« maxcut », « clique »), de l'industrie et de la conception de systèmes complexes (satisfiabilité, parcours de graphes).

Le domaine de l'intelligence artificielle est également un champ de recherche actif, et des méthodes d'apprentissage pour les réseaux de neurones artificiels commencent à voir le jour. C'est donc l'ensemble des activités humaines concernées par le traitement de l'information qui sont potentiellement concernées par l'avenir du calcul quantique.

De ce fait, le domaine des technologies quantiques et du calcul quantique en particulier est considéré comme un enjeu stratégique. L'Europe, la France et bien d'autres pays soutiennent les efforts de recherche dans ce domaine.

Pour la France, IBM a choisi de créer un centre d'expertise sur le calcul quantique au sein de son site de Montpellier, pour servir de support au développement du calcul quantique en France et à soutenir l'offre d'IBM sur le marché national.

Dans cette logique un projet de collaboration a été mis en place avec le soutien de la région Occitanie avec l'Université de Montpellier : « Projet QuantUM » (https://www-03.ibm.com/press/fr/fr/pressrelease/54572.wss) 6

Conclusion

Il résulte de cette exploration que l'informatique quantique représente le début d'une aventure. Elle connaît actuellement un rythme d'innovation accéléré car les technologies permettent à présent d'expérimenter les théories imaginées depuis le début des années 1970.

Certains domaines comme la communication quantique ou la métrologie quantique font d'ores et déjà l'objet d'industrialisation et de commercialisation. Le calcul quantique proprement dit est en train d'approcher de très près le moment où il aura l'avantage dans certains cas sur les ordinateurs dits classiques. La communauté s'affaire à développer les systèmes physiques, les environnements de programmation, et les algorithmes, avec des résultats spectaculaires semaine après semaine.

Une chose est certaine : cette aventure fait partie des domaines où l'humanité démontre sa motivation et sa capacité à continuer à comprendre les secrets de la nature et à mettre en œuvre des technologies de plus en plus complexes.

Le chemin promet d'être passionnant !

"The effort to understand the universe is one of the very few things which lifts human life a little above the level of farce and gives it some of the grace of tragedy."

(Steven Weinberg)

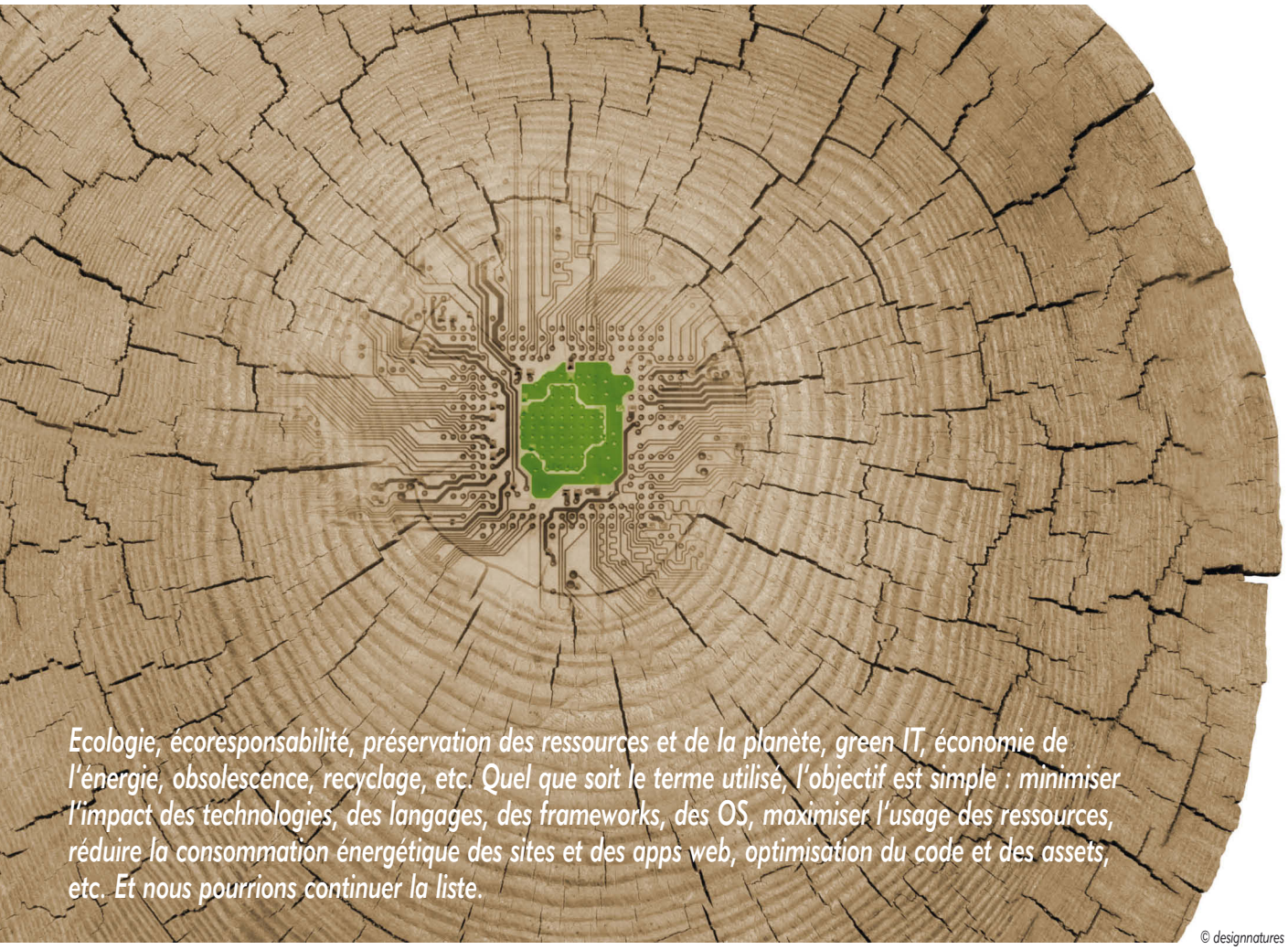
4 *Un programme écrit avec Python et QISKit*

```
%matplotlib inline
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit, execute, Aer
from qiskit.tools.visualization import plot_histogram
backend = Aer.get_backend('qasm_simulator')

# define circuit
qr = QuantumRegister(1)
cr = ClassicalRegister(1)
circ = QuantumCircuit(qr,cr)
circ = QuantumCircuit(qr,cr)
# H X H exemple:
circ.h(qr[0])
circ.x(qr[0])
circ.h(qr[0])
circ.measure(qr,cr)
# see the circuit
circ.draw(output='mpl')
# execution & display
resultat = execute(circ,backend,shots=1024).result()
d = resultat.get_counts(circ)
plot_histogram(resultat.get_counts(circ))
```

6 *IBM Q System One disponible sur le Cloud (© : IBM Research)*

François Tonic

Etre un développeur écoresponsable : oui, c'est possible !



Ecologie, écoresponsabilité, préservation des ressources et de la planète, green IT, économie de l'énergie, obsolescence, recyclage, etc. Quel que soit le terme utilisé, l'objectif est simple : minimiser l'impact des technologies, des langages, des frameworks, des OS, maximiser l'usage des ressources, réduire la consommation énergétique des sites et des apps web, optimisation du code et des assets, etc. Et nous pourrions continuer la liste.

Google incite les développeurs à surveiller le poids de leurs apps. La première motivation va vers les pays où les smartphones ont moins de ressources et des réseaux plus faibles. Mais au-delà, cela ne fera pas mal aux apps de perdre quelques Mo, voir beaucoup plus !

Ecoconception : ISO/TR 14032 :2002

Il est parfois difficile de comprendre le terme « écoconception ». Cette notion va au-delà de la seule informatique. En 2002, une norme a été publiée par l'ISO, organisme de normalisation : ISO/TR 14032 :2002.

" Tout produit, c'est-à-dire tout bien ou service, a des impacts sur l'environnement, ces impacts pouvant se manifester à l'une des étapes ou à toutes les étapes du cycle de vie du produit : acquisition des matières premières, fabrication, distribution, utilisation et élimination. Ces impacts peuvent être légers ou significatifs. Ils peuvent se manifester à court ou à long terme et ils peuvent se produire au niveau local, régional ou mondial (ou toute combinaison de ces aspects)."

Paragraphe 3. 3 : produit

Tout bien ou service.

Note 1 à l'article : il existe quatre catégories génériques de produits :

- les services (par exemple transport);
- les «software» (par exemple logiciel, dictionnaire);

- les [produits] matériels (par exemple pièces mécaniques de moteur);
- les produits issus de processus à caractère continu (par exemple lubrifiant).

Note 2 à l'article : les services comportent des éléments tangibles et des éléments immatériels. La prestation d'un service peut impliquer par exemple les actions suivantes :

- une activité réalisée sur un produit tangible fourni par le client (par exemple réparation d'une voiture);
- une activité réalisée sur un produit immatériel fourni par le client (par exemple une déclaration de revenu nécessaire pour déclencher l'impôt);
- la fourniture d'un produit immatériel (par exemple fourniture d'informations dans le contexte de la transmission de connaissances);
- la création d'une ambiance pour le client (par exemple dans les hôtels et restaurants).

Un «software» se compose d'informations, est généralement immatériel et peut se présenter sous forme de démarches, de transactions ou de procédures.

Un produit matériel est généralement tangible et son volume constitue une caractéristique dénombrable. Les produits issus de processus à caractère continu sont généralement tangibles et leur volume constitue une caractéristique continue. Les produits matériels et issus de processus à caractère continu sont souvent appelés «biens».

Note 3 à l'article :

Adapté de l'ISO 14021:1999, 3.1.11.

Paragraphe 3.3 :

conception et développement

Ensemble de processus qui transforme des exigences en caractéristiques spécifiées ou en spécification d'un produit, d'un processus ou d'un système [SOURCE: ISO 9000: 2000, 3.4.4]

Note 1 à l'article : les termes «conception» et «développement» sont parfois utilisés comme synonymes et parfois utilisés pour définir des étapes différentes du processus global de conception et développement.

Note 2 à l'article : le développement de produit est un processus qui mène de l'idée d'un produit depuis sa planification jusqu'à son lancement sur le marché et la revue du produit, et au cours duquel les stratégies commerciales, les considérations mercatiques, les méthodes de recherche et les aspects de conception sont mises en œuvre pour obtenir un produit utilisable.

Note 3 à l'article : l'intégration des aspects environnementaux dans la conception et le développement de produit peut aussi être appelée «conception pour l'environnement», «écoconception», «partie environnementale de la gestion responsable des produits», etc.

La notion d'écoconception est donc large pour pouvoir couvrir une réalité elle-même large et complexe.

Il faut que chaque acteur prenne conscience du problème : le développeur, l'entreprise, l'utilisateur. Rien que sur le code et les pratiques, il y a de quoi faire.

Chasser les codes inutiles… et tout le reste !

Avez-vous retiré tous les vieux codes qui ne servent pas ou inutiles ? Avez-vous fait le tri dans les librairies et frameworks utilisés ? Avez-vous bien fait le ménage et retiré tout ce qui n'est plus utilisé. Souvent, on laisse, même sans le savoir, quantité de librairies, SDK, API qui ne servent plus… Sur la partie données, même réflexion : ne faut-il pas sérieusement auditer les données et supprimer les données qui ne sont pas utilisées et les archiver ? Plus votre base possède de données, plus vous allez charger les ressources à un moment ou à un autre.

Utilisez-vous les caches pour les données et évitez-vous d'interroger systématiquement les données ? Vous pouvez économiser des requêtes, surtout quand vous êtes en mode front / back office. Vous pouvez aussi utiliser des mécanismes pour mettre à jour les données uniquement par rapport au delta. C'est à dire, on recharge, on récupère, ou alors on n'envoie que les données qui ont changé.

Posons une 1ere réalité

Tableau 1

La comparaison du tableau 1 fait mal. On me dira : c'est l'évolution technologique, les utilisateurs qui en veulent toujours plus. Ah l'utilisateur. Chaque année, il faut présenter de nouveaux modèles, avec toujours plus de choses, une nouvelle version du système, etc. Notre société fonctionne ainsi : pour continuer à vendre, on rajoute toujours quelque chose pour maintenir les cycles (nouveaux acheteurs, le cycle de renouvellement).

On dira que c'est un système stupide, de gaspillage et d'obsolescence programmée. Ce n'est pas entièrement faux mais cette notion d'obsolescence est très large. Il y a l'obsolescence programmée donc volontaire, l'obsolescence technologique et matérielle. Ces deux derniers points sont importants car on ne peut pas interdire les entreprises de vouloir améliorer et innover pour leurs produits. Il faut trouver un sujet milieu. C'est plus un débat de société et même de civilisation.

Le smartphone est un bon exemple de ce problème. Car le rythme des nouveaux modèles provoque le renouvellement même inutile chez de nombreux utilisateurs qui veulent le dernier modèle. Les constructeurs répondent aussi à une cette demande. Un cercle vicieux s'est installé. Le casser n'est pas évident.

Tableau 1

	iPhone 2007	iPhone 2019
CPU	ARM 412 MHz	A13 (ARM maison)
RAM	128 Mo	4 Go
Stockage	4 à 16 Go	64 à 512 Go
Caméra (millions pixels)	2	12
Ecran	Dalle IPS 32x480, 163 PPP	OLED / LCD, 1792x828, 326 PPP
Taille écran	4,7	6,1
Poids	135g	194g
Batterie / autonomie officielle	1400 mAh, 8-10h max.	3110 mAh, 1 journée
Poids photo	Plusieurs centaines de Ko	Plusieurs Mo

Le poids des sites web a explosé

Si on regarde les données de httparchive.org, une page moyenne web dépasse 1 Mo en 2018, aujourd'hui, on explose à +4 Mo sur le desktop, et +5 Mo

sur mobile. En 2015, une page web desktop pesait 500-600 ko ! En 2018, le poids moyen était de 1,6 Mo… voir presque 2,9 Mo pour certains sites. Bref, le site web est devenu obèse.

Cette obésité logicielle s'explique par plusieurs choses : des assets graphiques de plus en plus nombreux, de multiples couches techniques, un manque d'optimisation des pages. Cela se traduit aussi par une explosion du trafic réseau entre les terminaux et les serveurs (= front & back office). Une partie des traitements se font sur le serveur, donc il faut faire des allers-retours entre l'utilisateur et le serveur. Côté utilisateur on économise des ressources et la batterie voire la consommation électrique directe, mais on charge le serveur et on envoie plus de données sur le réseau…

Et nous pourrions multiplier les exemples.

Chaque mail, chaque SMS, chaque vidéo YouTube consomme des ressources, des watts. On estime que les vidéos YouTube rejettent environ 11 millions de tonnes de CO2 par an. Car même si les datacenters géants utilisent massivement les énergies renouvelables pour les alimenter, reste le problème de la chaleur dégagée et de l'eau chaude générée dans les circuits de refroidissement. Et bien entendu, l'impact écologique de la construction du datacenter en lui-même.

Ne soyez pas naïf : toute activité génère du C02 et de la pollution. Depuis que l'Homme existe, c'est ainsi. Il est IMPOSSIBLE d'être à 0 émission ou 0 rejet. Par contre, on peut réduire notre impact et optimiser notre quotidien. Le monde informatique doit prendre sa part que ce soit l'entreprise, les utilisateurs et bien entendu, les développeurs.

Ce n'était pas mieux avant

On dit souvent, à tort, « c'était mieux avant ». Ce n'est souvent pas le cas. Car cela repose souvent sur des idées fausses ou des souvenirs biaisés. Aujourd'hui, les apps sont dématérialisées. Apple, avec l'App Store de l'iPhone, a véritablement révolutionné notre manière de consommer les logiciels et les services d'applications. Les services SaaS proposent aussi un autre modèle de consommation tout comme la virtualisation, le Cloud Computing ou encore les conteneurs. **Tableau 2**

Low Tech vs stack « classique »

Aujourd'hui, un site web, une application web, ressemble à un mille-feuille technologique. En effet depuis 25 ans, les couches s'empilent que ce soit sur le navigateur, intégrant de plus en plus de librairies et d'API, sur le front et le back-end.

Le site web est un cas d'école que l'on prend souvent en exemple. Une des questions à se poser : quels sont mes besoins réels, en anticipant l'avenir ? Si je veux un simple site vitrine, un site institutionnel : faites simple !

Faut-il absolument un CMS ?

- Oui c'est pratique ;
- Oui un CMS mâche le développement.

Mais, un CMS c'est aussi :

- Peut-être 90 % des fonctionnalités que vous n'utilisez jamais ;
- Il nécessite plusieurs Mo de stockage rien que pour son déploiement + Mo pour son site.

Un simple site n'a pas besoin que l'on sorte des plateformes entières. Un site classique HTML / CSS peut suffire surtout quand vous avez très peu de données à faire, à traiter. Les pages dynamiques sont utiles quand vous commencez à avoir beaucoup de contenus ou pour du eCommence.

La partie graphique est un autre élément important que l'on néglige beaucoup. Avez-vous pensé à optimiser vos images ? Sont-elles taillées pour chaque dimension d'affichage ? Utilisez-vous le bon format ? Par exemple, arrêtez d'utiliser le JPEG, utilisez plutôt le format PNG et vous pouvez même user et abuser du format vectoriel SVG. Vous allez pouvoir dégraisser le poids des images et optimiser le temps d'affichage. Surtout, pensez à dimensionner les tailles d'images pour chaque type d'affichage. Il vaut mieux plusieurs images de différentes tailles qu'une seule qui risque de peser plus lourd et de demander plus de ressources réseaux, donc du temps de chargement. Ne multipliez pas les polices de caractères.

Est-ce bien utile de développer un site en JS / framework JS si le couple HTML 5 + CSS suffit ?

Si vous avez des vidéos, faites un travail d'optimisation du poids, en ajustant, par exemple, la qualité de l'image. D'autres, les codecs et les formats jouent un rôle important dans la vidéo. Utilisez des compresseurs vidéo pour améliorer le poids des fichiers.

Je parle ici que de l'aspect technique. Je ne parlerai pas de la partie énergétique. Souvent, on comprend : Low Tech = économie d'énergie, énergie propre, etc. Car sur la partie énergétique, si vous hébergez chez un prestataire, il est impossible de maîtriser la partie serveur. Vous pouvez tout de même trouver des hébergements qui se disent « écoresponsable » comme GreenGeeks. A voir si ces fournisseurs supportent vos technologies et les tarifs proposés.

Rien ne vous empêche de faire votre propre serveur Low Tech avec une carte de type ODROID ou Raspberry Pi, d'y monter un serveur web, de gérer l'IP et d'y héberger votre site web.

Tableau 2

	Logiciel en boîte	Logiciel en App Store
Les +	• Support physique • Logiciels « anciens » disponibles • Installation même sans réseau (en principe)	• Installation en 1 clic (ou presque) • Variété des apps • Pas de matériel supplémentaire • Mise à jour facilitée et dernière version disponible • Et plus souvent, pas d'impression de boîtier, de documentation, pas de coût de transport
Les -	• Impression de boîte et des documents • Gravure du CD / DVD • Usage de papier et de plastique • Transport des boîtes • Coût logistique	• Nécessite réseau / réseau télécom • App store dédié à chaque plateforme (le plus souvent) • Nécessite un usage serveur et consommation de ressource sur son matériel

SQLI DIGITAL EXPERIENCE

Lionel DUPORT
Ingénieur Concepteur Développeur, **SQLI**
16 ans d'expérience dans le développement logiciel, j'ai travaillé dans de multiples secteurs avec des challenges tous différents. Je suis désormais Ingénieur Concepteur Développeur depuis 9 ans sur mobiles (IOS, Android). J'accompagne les clients notamment des grands comptes dans leurs demandes de développement et le suivi de leurs applications. Ma spécialité : la réalité augmentée, l'architecture et le suivi de qualité de code.

L'Écoconception d'une application Android

L'écoconception consistant à intégrer la protection de l'environnement dès la conception de biens ou de services, vise à réduire l'impact environnemental tout au long du cycle de vie du produit. La protection de l'environnement n'étant pas directement le but recherché, on peut effectuer un parallèle avec l'ensemble de règles et d'outils qui permettent d'optimiser la performance d'un site web.

En effet, la performance est un levier direct pour améliorer le taux de conversion d'un site e-commerce, mais aussi son référencement naturel. Selon une étude de KISSMetrics(1), une page qui met plus de 3 secondes à charger fait perdre 40 % des visiteurs. Les grands noms du web l'ont bien compris ; Amazon a calculé qu'une baisse de 1 seconde de la vitesse de chargement de sa plateforme entraînerait une perte de 1,6 milliards de dollars. La performance a aussi un impact réel sur le SEO puisque Google prend en compte cette notion de rapidité dans la prise en compte de son algorithme d'indexation. De nombreux outils existent pour mesurer l'efficacité et la charge d'un site web : Pingdom Tools, Google PageSpeed Insights, WebPageTest, GTMetrix … Et on recense énormément d'articles, de tutoriels ou de plugins pour améliorer les performances et donc le coût énergétique d'une page web.

Si l'optimisation web est bien connue et les outils pour le quantifier nombreux, l'optimisation d'application mobile l'est beaucoup moins, et les outils très rares. Après un état de lieux de la consommation énergétique liée à l'utilisation des applications mobiles dans le monde, je vous donnerai un ensemble de règles à respecter pour réduire l'impact des applications mobiles que vous seriez amenés à développer. Étant Ingénieur concepteur développeur Android depuis 9 ans, c'est tout naturellement sur cette plateforme, et en Kotlin, que sont basés les exemples de code de cet article. **Les stratégies mises en place sont indépendantes du langage.** Il y a peu de différence de performance entre Java et Kotlin, seule la bonne pratique fera la différence.

Qu'en est-il de l'impact environnemental des applications mobiles ?

Aujourd'hui, le numérique représente près de 4 % des gaz à effets de serre(2). On estime que ce chiffre montera de 7 % à 8,5 % en 2025. Dans un contexte où les GES doivent être réduits, le numérique affiche une croissance forte de +9 % par an.

Cette consommation croissante s'explique en partie par le nombre de smartphones utilisés : 2,7 milliards de personnes possédaient un smartphone en 2019, et la prévision pour 2025 est de 5 milliards.

Si on projette la consommation des applications mobiles sur l'ensemble des utilisateurs de smartphone, on obtient une consommation totale de 20,3 TWh, soit un peu moins que l'équivalent de la consommation annuelle en électricité d'un pays comme l'Irlande(3). Cet impact très important peut être réduit en optimisant les applications.

Les stratégies à mettre en place

Nous allons voir comment optimiser les applications suivant deux axes : la réduction de la consommation de la batterie et la réduction des appels réseaux.

Différer les tâches pendant le chargement de la batterie si possible.

Si nous effectuons des tâches pendant que le téléphone est en charge, cela ne comptera pas pour la décharge de la batterie. Sauf si quelque chose doit être fait immédiatement ou rapidement, nous devons le différer jusqu'à ce que le téléphone soit branché. Un bon exemple de cette application différée est la synchronisation de contacts ou de photos avec les différents services web.

Pour ce faire, nous avons besoin de connaître le statut de la batterie du téléphone, voici comment le détecter :

1 La création d'un **BroadcastReceiver** va nous permettre de recevoir les évènements du changement de statut de la batterie. À la réception de l'évènement, nous allons passer par un **Observable** pour notifier les vues.

```
class BatteryStatusReceiver: BroadcastReceiver() {

  override fun onReceive(context: Context?, intent: Intent?) {
    intent?.let {
      // Appel à la méthode pour notifier les observers
      ObservableObject.getInstance().updateValue(it)
    }
  }
}
```

2 Un Observable va nous permettre l'ajout et la suppression d'observers. Nous allons créer ici un **singleton** que l'on va appeler à partir du BroadcastReceiver. La méthode updateValue va notifier les observers.

```
class ObservableObject : Observable() {

  companion object {
    private val instance = ObservableObject()
    fun getInstance(): ObservableObject = instance
```

(1) *https://neilpatel.com/blog/loading-time/*

(2) *https://theshiftproject.org/wp-content/uploads/2018/10/2018-10-04_Rapport_Pour-une-sobri%C3%A9t%C3%A9-num%C3%A9rique_Rapport_The-Shift-Project.pdf*

(3) *https://atos.net/wp-content/uploads/2019/05/atos-atd-benchmark-fr.pdf*

```
    }

    fun updateValue(data: Any) {
        synchronized(this) {
            setChanged()
            // Notifie les observers
            notifyObservers(data)
        }
    }
}
```

3 Pour terminer, il faut écouter les évènements sur la vue, que ça soit une **Activity** ou un **Fragment**, en utilisant l'observer :

```
class MainActivity : AppCompatActivity(), Observer {

    private val batteryStatusReceiver = BatteryStatusReceiver()

    // A appeler lorsque la vue est active (onCreate, OnResume)
    private fun initReceiver() {
        // On va écouter les action de charge connecté et déconnecté
        IntentFilter().let {
            it.addAction(ACTION_POWER_DISCONNECTED)
            it.addAction(ACTION_POWER_CONNECTED)
            registerReceiver(batteryStatusReceiver, it)
        }
        // On ajoute l'activité à la liste des observer
        ObservableObject.getInstance().addObserver(this)
    }

    // A appeler lorsque la vue est inactive (onPause, onDestroy)
    private fun deleteReceiver() {
        unregisterReceiver(batteryStatusReceiver)
        // On supprime l'activité de la liste des observers
        ObservableObject.getInstance().deleteObserver(this)
    }

    override fun update(observable: Observable?, any: Any?) {
        val intent = any as Intent
        when (intent.action) {
            ACTION_POWER_CONNECTED -> // EFFECTUER traitement de synchronisation
            ACTION_POWER_DISCONNECTED -> // TERMINER traitement de synchronisation
        }
    }
}
```

Avec cette mise en place, vous allez pouvoir détecter lorsque le téléphone est en charge ou non, et effectuer des traitements qui peuvent attendre que le téléphone soit en charge !

Concaténation de caractères

Le deuxième conseil que je peux donner, c'est d'utiliser **la classe StringBuilder** pour la concaténation de chaînes de caractères. En effet, elle est beaucoup plus économe en ressources que la simple concaténation. Simple concaténation :

```
var string = "hello"
for (i in 0..9_999) {
    string += " world"
}
```

Concaténation avec StringBuilder :

```
var string = "hello"
for (i in 0..9_999) {
    sb.append(" world")
}
string = sb.toString()
```

La première méthode prend 1,5 secondes sur mon device Android de test, quant à la seconde, elle se termine en 8 ms. En termes de charge CPU, là aussi la première prend un peu moins de 15 %, quant à la seconde elle frôle le 0 % pour le même résultat !

Utilisation du GPS

Il existe deux API qui permettent de récupérer la position de l'utilisateur dans une application mobile Android : l'API Android Location et l'API Google Play Location Service. La deuxième a plusieurs avantages dont ne bénéficie pas la première :

- Choisit automatiquement le provider (GPS, Réseau) en fonction de la précision demandée de l'utilisation de la batterie ;
- Plus rapide ;
- Meilleure précision ;
- Fonctionnalités et paramétrages plus avancés (Geofencing) ;
- Économie de batterie.

L'utilisation de l'API nécessite d'utiliser 3 composants :

- FusedLocationProviderClient : c'est le point d'entrée pour interagir avec l'API ;
- LocationCallback : à utiliser pour recevoir les notifications de l'API lorsque l'emplacement du téléphone change ;
- LocationRequest : contient les paramètres de l'API.

C'est sur ce dernier objet que nous allons configurer les requêtes du GPS et donc optimiser l'API pour économiser la batterie du téléphone.

Définir la période d'expiration de la requête :

```
public LocationRequest setExpirationDuration (long millis)
```

Cette méthode configure un délai d'expiration pour les requêtes de géolocalisation. Le client sera automatiquement stoppé après l'expiration des requêtes.

Gérer la fréquence des requêtes, à adapter au cas d'utilisation :

```
public LocationRequest setInterval (long millis)
```

Définit la fréquence de la mise à jour de la localisation. Attention ! Si d'autres applications ont des mises à jour plus fréquentes, vous les recevrez.

```
public LocationRequest setMaxWaitTime (long millis)
```

Définit la fréquence à laquelle on reçoit les mises à jour.

Mises à jour basées sur le déplacement

```
public LocationRequest setSmallestDisplacement (float
        smallestDisplacementMeters)
```

Cette méthode permet de modifier la distance minimum en mètres à laquelle on va notifier le client. Cette distance est donc à adapter au cas d'utilisation. Dans le cas d'une application pour détecter les restaurants aux alentours, on définira une distance entre 10 et 30 mètres, pour une application de météo, cette distance pourra être plus importante (entre 5 et 10 Km).

Définir la priorité de la requête

```
public LocationRequest setPriority (int priority)
```

La priorité des requêtes à une forte influence sur la consommation de batterie. Quatre priorités sont définies :

- PRIORITY_HIGH_ACCURACY : la précision est fine et la consommation importante ;
- PRIORITY_BALANCED_POWER_ACCURACY : précision limitée à 100 mètres. La consommation batterie est moyenne ;
- PRIORITY_LOW_POWER : précision de 10 km maximum. La consommation batterie est basse ;
- PRIORITY_NO_POWER : l'API est en écoute passive à l'attention d'autres clients. La sollicitation de composant matériel étant inactive, la consommation batterie pour la récupération d'information de localisation est donc nulle.

À partir de l'ensemble des méthodes définies précédemment, nous allons pouvoir définir un exemple d'utilisation de l'API optimisée, pour une application d'affichage de météo qui économisera au mieux la batterie.

```
locationRequest = LocationRequest().apply {
  // Expiration de la requête : 10 minutes
  setExpirationDuration(10 * 1_000L)
  priority = LocationRequest.PRIORITY_LOW_POWER
  // Distance minimum pour notifier le callback : 3 km
  smallestDisplacement = 3_000F
  // 10 minutes
  interval = 10 * 60 * 1_000
  maxWaitTime = interval
  fastestInterval = interval
}
```

L'objet LocationRequest est donc configuré avec une requête d'expiration de 10 minutes, une priorité LOW_POWER, un espace minimum de déplacement de 3 km et un intervalle minimum de notification pour le client de 10 minutes. Cette configuration permet donc de créer une application météo entièrement fonctionnelle tout en économisant la batterie, en sollicitant au minimum le composant GPS et le CPU.

Repository Pattern

Effectuer des appels réseaux a un coût important sur la batterie, mais aussi sur l'ensemble du système dont il est dépendant : routeurs et datacenters. Pour éviter d'effectuer les appels sur des données que nous avons déjà téléchargées, je mets souvent en place dans mes projets le Repository Pattern. Le repository délivre les données du cache si elles sont disponibles, sinon il les récupère depuis l'API. Ce pattern est d'ailleurs très utile dans le cas du développement d'une application avec gestion online/offline. Il possède de multiples avantages :

- Dissocie l'application des sources de données ;
- Fournit des données provenant de plusieurs sources (BDD, API, cache mémoire) sans que le client s'en préoccupe ;
- Isole la couche de données ;
- Accès unique et centralisé des données ;
- Logique métier testable via des tests unitaires ;
- Ajout facile de nouvelles sources. 1

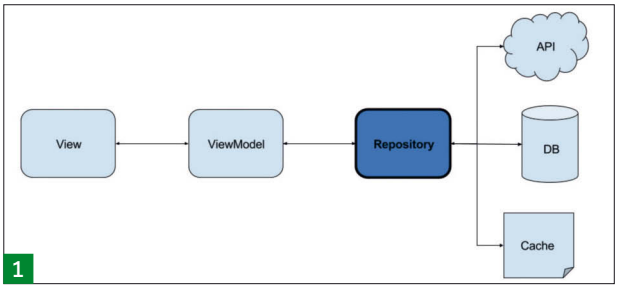
Voyons maintenant comment implémenter ce pattern. Nous allons dans cet exemple utiliser RX, les LiveData et une architecture MVVM (Modèle-Vue/Vue-Modèle) avec des ViewModel dans un contexte où on récupère une liste de Post.

Création d'une interface : en effet l'ensemble des différentes sources de données et le repository, vont partager les mêmes méthodes définies dans l'interface.

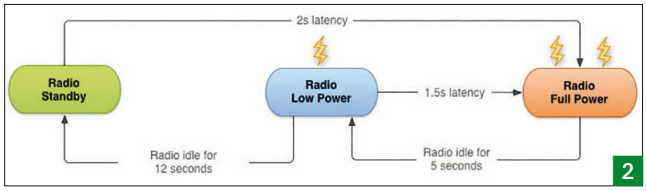
```
interface RepositoryInterface {

  fun getPosts(): Single<List<Post>?>
}
```

Création du Repository : ici le Repository porte l'instance des différentes sources de données (RepositoryAPIDelegate pour les appels API, RepositoryCacheDelegate pour les accès à une base de données). Son rôle est d'arbitrer la source de données à récupérer.

```
private val apiRepo: RepositoryAPIDelegate()
private val cacheRepo: RepositoryCacheDelegate()


override fun getPost(): Single<List<Post>?> = Single.create { emitter ->
    // Dans un premier temps on vérifie si les données sont présente en cache
  cacheRepo.getPost().doOnSuccess { postsCache ->
    if (!postsCache.isNullOrEmpty()) {
        // Il existe des données en cache
        emitter.onSuccess(postsCache)
    }
  }.doAfterSuccess { postsCache ->
    if (postsCache.isNullOrEmpty()) {
        // Si des données ne sont pas présentes en cache nous faisons appel à l'API
apiRepo.getPost().doOnSuccess { postsAPI ->
        if (!postsAPI.isNullOrEmpty()) {
            // Stockage des données dans le cache
            cacheRepo.storeList(postsAPI)
            emitter.onSuccess(postsAPI)
        }
      }.subscribe()
    }
  }.subscribe()
}
```



1



Dans cet exemple, nous créons un Single qui va émettre les données du cache ou de l'API. Dans un premier temps, nous souscrivons à la méthode de récupération des données du cache. En cas de succès et de données non vides, nous retournons les données. Auquel cas, nous faisons appel à l'API qui en cas de succès retournera les données mais aussi fera appel au cache pour y stocker les données.

Stratégies réseau 2

Comme indiqué dans cette machine à état, le composant réseau (3G) comprend 3 différents états. Si des données sont envoyées ou reçues, l'appareil est en haute consommation. Si les données n'ont pas été envoyées et reçues pendant un certain temps, l'appareil commencera à se mettre en veille. L'appareil passera de la haute consommation à un mode basse consommation pour passer en mode veille. Il est très courant d'être dans un modèle où nous demandons un peu de données, attendons un moment, puis demandons plus de données. Le problème avec cette approche est que **nous passons constamment d'un mode veille à un mode réveil et qu'un coût d'énergie est associé** à cela. Chaque appel réseau entraîne le réveil du composant, puis reste éveillé pendant un certain temps pour obtenir une réponse. Dans le pire des cas, on réveille le composant dès qu'il se rendort. La meilleure façon d'éviter ce modèle est de faire la distinction entre ce qui doit arriver maintenant et ce que l'on veut faire plus tard. On souhaitera donc regrouper autant d'opérations réseau que possible. Cela permettra au composant d'entrer une fois à pleine puissance aussi longtemps que nécessaire. Comme il n'est pas possible de prédire ce que l'utilisateur va vouloir faire, il est judicieux de **pré-extraire autant de données que possible afin d'éviter les demandes inutiles**.

Téléchargement volumineux vs plus petits

Il est préférable d'effectuer un téléchargement volumineux de données plutôt que plusieurs téléchargements plus petits. La raison est qu'un téléchargement volumineux permet au composant réseau de passer en mode basse consommation tandis que plusieurs téléchargements plus petits vous maintiendront en pleine puissance ce qui sera plus impactant en matière de consommation de batterie.

Réduction du poids de l'application

La réduction du poids des ressources de l'application (images, fichiers XML) est aussi un facteur de réduction de l'empreinte de celle-ci. Cela peut être effectué de plusieurs manières :

- Utiliser des ressources XML *Drawables* plutôt que des fichiers images si possible ;
- Supprimer les ressources non utilisées ;
- Compresser les images PNG / JPG avec un outil de compression ;
- L'outil *aapt* peut optimiser les images durant le processus de compilation. Par exemple un PNG qui ne requiert pas plus de 256 couleurs pourra être converti en 8 bits. En faisant cela, le résultat visuel sera le même tout en limitant son empreinte de mémoire :
  - L'outil compressera uniquement les images du dossier *res/drawable*,
  - Les fichiers images doivent utiliser moins de 256 couleurs pour être optimisées.
- Compresser les assets en activant *shrinkResources* dans le fichier gradle.

```
android {
  // Other settings

  buildTypes {
    release {
      shrinkResources true
    }
  }
}
```

Mieux cibler les terminaux et les versions d'OS

Publier une application au **format *Android App Bundle*** permet de réduire la taille, de simplifier les releases et de proposer des fonctionnalités à la demande.

Les packages *Android App Bundle* utilisent un nouveau modèle de diffusion d'applications, appelé « *Google Play Dynamic Delivery* » afin de créer et de diffuser des APK optimisés pour chaque configuration d'appareil. Comme il supprime le code et les ressources inutilisées pour les autres appareils, ce modèle de publication permet aux utilisateurs d'installer une application plus petite et plus efficace.

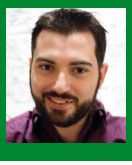
Format texte vs Binaire

Il est extrêmement courant d'utiliser JSON ou XML pour le transfert de données. Même si les données seront compressées à l'aide de GZIP, elles seront toujours beaucoup plus grandes qu'un format binaire sérialisé. En utilisant un **format binaire sérialisé, tel que la librairie FlatBuffers,** vous pouvez transférer la même quantité de données en réduisant la taille de façon drastique. En outre, l'analyse (sérialisation/désérialisation) des formats de texte tels que JSON est plus gourmande en processeur que l'analyse des formats binaires(4).

En passant du côté obscur

L'affichage est une des causes de la consommation de batterie sur un smartphone, mais elle mineure (on estime celle-ci à environ 3%). Dans les écrans OLED/AMOLED, les pixels émettent de la lumière de façon indépendante. Cela signifie que l'utilisation d'un fond sombre peut vous faire économiser de l'énergie : **un pixel noir sera un pixel éteint**. Le LCD quant à lui consomme plus d'énergie car il fait fonctionner le rétro éclairage en permanence. Sur les écrans LCD, l'utilisation de pixels sombre n'entrainera aucune économie. En résumé, il existe plusieurs axes pour optimiser la consommation d'une application mobile Android. Il est bien sûr intéressant en tant qu'architecte, ou simple développeur exécutant, de les connaître. Ne serait-ce que par souci de qualité de code et de produit. Mais les connaître et les appliquer, c'est aussi contribuer à la réduction de la pollution numérique inhérente à la consommation croissante actuelle.

(4) https://blog.mindorks.com/why-consider-flatbuffer-over-json-2e4aa8d4ed07



Jérémy PASTOURET
Jérémy est l'auteur du livre : Phalcon - Développez des applications web complexes et performantes en PHP. Il est aussi rédacteur pour Les Enovateurs - https://les-enovateurs.com/. Il fait partie de l'équipe de Phalcon et contribue à des projets Open Source. Entrepreneur, il a créé Unlock My Data et plus récemment Garwen.

Les 10 commandements de l'écoconception

Aujourd'hui, créer de nouveaux projets pose de nombreuses questions et implique de nombreux choix. L'objectif étant d'obtenir un produit fonctionnel, répondant aux besoins tout en étant efficace et productif. Mais qu'en est-il de l'impact écologique ?

A chaque démarrage de projet, les mêmes questions reviennent : quelles technologies utiliser ? Quels langages, frameworks, outil de gestion des bugs, solutions d'intégration continue choisir ?
Les développeurs font face à toujours plus de possibilités : c'est tentant d'adopter le dernier langage à la mode. Mais quelle est l'empreinte écologique associée à tous ces outils ?
Ces quelques commandements visent à réaliser un produit moins énergivore, tout en réduisant l'empreinte des utilisateurs.

1. Choisir le bon langage pour le bon projet

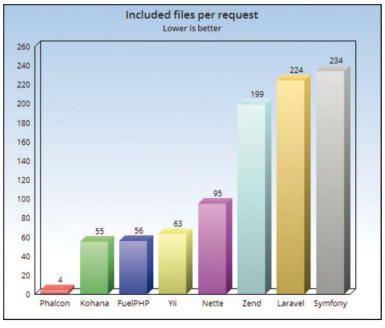
Nombreux sont les paramètres à prendre en compte en fonction de l'équipe, du temps de formation possible.
En se concentrant sur l'aspect écologique, mieux vaut choisir un langage :

- léger (peu de fichiers à installer / déjà utilisé dans le système d'exploitation) ;
- proche du langage machine (comme le C / C++) ;
- sans surcouche supplémentaire (comme la JVM).

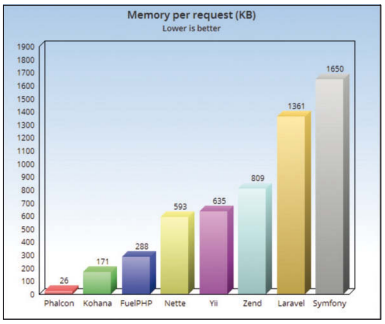
L'idée n'est pas de revenir à du Pascal… mais d'utiliser des langages qui possèdent ces propriétés tout en continuant à évoluer.

2. Choisir le bon framework

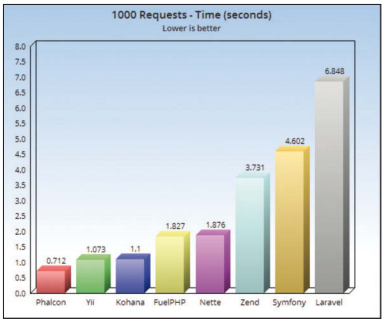
Utiliser un framework est vivement conseillé, car ils apportent de la rigueur, une structure et surtout de la sécurité, même s'ils représentent un coût énergétique supplémentaire. La dépense est à relativiser par rapport aux attaques que peut subir un site web.
Par exemple : un attaquant effectue une injection SQL, lance des boucles infinies et sature le système. Les attaques DDOS (et autres) sont très énergivores. D'autant plus que les hackers ne sont pas près de devenir écolos. Il faut donc se protéger.
Il existe beaucoup de frameworks et nous avons tous nos préférences. Plutôt populaire, flexible, rapide, ou bien celui qui possède le plus grand nombre de composants… Pourtant, leur empreinte est rarement prise en compte.
Pour trouver le moins énergivore, il faut dénicher des benchmarks, ou en réaliser à travers des POC.
Voici un benchmark de frameworks PHP :



On retrouve le nombre de fichiers chargés par framework pour chaque requête reçue.



Cet histogramme indique la mémoire utilisée en Kb par requête pour chaque framework.



Ce dernier schéma indique le temps de traitement de 1000 requêtes pour chacun des frameworks.

On remarque que Phalcon sort du lot ! Pourquoi ses performances sont-elles si bonnes ? C'est un framework très particulier développé comme une extension PHP. Il est compilé en un seul fichier léger, et intégré à PHP.
Lorsque Phalcon est installé sur un serveur, tous les projets peuvent en profiter. Contrairement à Symfony, dont il faut retélécharger tous les fichiers (vendors) qui le composent pour chaque projet.
C'est pour cette raison qu'il est important de comprendre le fonctionnement d'un framework et de vérifier s'il est proche du langage.

3. S'interroger pour chaque plugin / extension / librairie

A chaque problème, une extension ou un plugin. Par exemple, on a besoin d'icônes, alors on ajoute une librairie avec plus 5 000 icônes différentes.
Un jour, il manque une icône particulière. Celle-ci est présente dans une autre librairie, alors on l'ajoute aussi. Avec le temps, le nombre de fichiers à charger par le serveur et par l'utilisateur s'accroît.
La solution ? Se poser les bonnes questions pour chaque extension. Répond-elle à plusieurs besoins ? Si oui, vais-je me servir des autres options ? Si la réponse est non, existe-il une extension moins générale répondant vraiment à mon besoin ?
Si on reprend notre problème de départ, les 5 000 icônes seront-elles utilisées ? Réfléchissez à télécharger des paquets d'icônes, au lieu du catalogue complet.

4. Mettre les fichiers au régime

Peu importe le type du produit, il y a toujours des images. Celles-ci sont téléchargées par les utilisateurs lorsqu'ils récupèrent le programme d'installation, ou une application mobile. Pour les services Web, ces images sont téléchargées et retéléchargées par le navigateur.
Le poids des images augmente la facture énergétique. Ces images sont aussi envoyées dans la bande passante et stockées sur des serveurs répliqués, des backups et sur les postes utilisateurs.
Sauf qu'avec nos différents usages, tablette, smartphone, pas besoin d'image 8K. Les comptables ne saisissent pas leurs informations sur des écrans de 65 pouces.
La solution ? Redimensionner les photos en fonction des usages. Par exemple pour une grande bannière, une résolution de 1920 x 1080 est largement suffisante.
De plus, il existe des sites qui optimisent le poids des images, comme TinyPng.
Pour une optimisation optimale, les fichiers vectorisés SVG sont légers et gèrent toutes les résolutions.

5. Réutiliser proprement du code ou des ressources

Si vous copiez-collez du code au sein d'un même projet… il existe une meilleure façon de procéder. Penser au refactoring ou à la création de fonctions réutilisables. Cela allègera les fichiers de code et la maintenance sera plus facile.
Si vous utilisez des composants externes avec composer, npm… ne versionnez pas ces composants. Ces projets sont déjà versionnés, et vous perdez de l'espace disque inutilement.
L'utilisation du cache est aussi efficace lorsqu'on utilise des composants communs à plusieurs projets. Que ce soit le cache de composer, npm ou même des images Docker, il n'y aura pas de nouveau téléchargement supplémentaire.

6. Ne pas surestimer son service

Avec les infrastructures Cloud actuelles, on a tendance à surdimensionner les machines afin de réduire le temps de réponse.
De cette façon, on reporte l'optimisation de code et on continue d'accroître la puissance des machines au moindre ralentissement. Cela engendre des frais et une consommation d'énergie supplémentaires.
Il faut commencer petit et augmenter les ressources si besoin, en attendant d'optimiser le code. Avec Kubernetes, il est possible de gérer la charge automatiquement en utilisant des conteneurs.
Au lancement d'un projet, il n'est pas utile de dimensionner les serveurs pour accueillir un million d'utilisateurs… qui n'arriveront que dans deux ans.

7. Contrôler la qualité sans trop dépenser d'énergie

Plus un projet est carré, plus les tests se multiplient, plus le projet est stable. Les tests assurent un seuil de qualité du produit attendu par les clients.
De plus, il est tentant d'utiliser GitHub Action, Gitlab CI… Le coût est faible voire gratuit pour les projets Open Source. Pourtant, il faut limiter les tests à leur utilisation réelle. Par conséquent, il ne sert à rien de tester un service avec les différentes versions de PHP, lorsque l'outil n'est utilisé que dans la version 7.2.
Et attention aux évènements qui déclenchent les tests. S'ils sont lancés à chaque fois qu'un développeur envoie du code sur le serveur de version, est-ce vraiment nécessaire de relancer aussi les tests toutes les nuits ?

8. Diviser les fichiers pour mieux coder et regrouper les fichiers pour optimiser le service

Créer différents fichiers en fonction du contexte, c'est bien. Cela permet de maintenir le code plus facilement et de charger uniquement ce qui est nécessaire. Cependant, lancer plusieurs requêtes HTTP pour télécharger différents fichiers consomme plus d'énergie que d'envoyer un seul fichier.
Cela arrive souvent avec les fichiers CSS. L'idéal est de les minifier (en supprimant les commentaires, les espaces et les retours à la ligne inutiles) et de compacter ces fichiers pour réduire le nombre d'appels. L'extension Deflate d'Apache ou PageSpeed permettent de réaliser ce genre d'opération.

9. Réduire les traitements longs

Il est important d'identifier les pages / processus longs d'une application. Ces opérations sont souvent très énergivores. Il faut les analyser, les refactorer, changer d'algorithme ou trouver une autre méthode. Certains algorithmes métier prennent un certain temps de traitement dû à des parcours de tableaux imbriqués. Une des solutions envisageables est d'utiliser la puissance d'un moteur SQL. L'idée : écrire des requêtes SQL ou des fonctions PL / SQL qui réalisent le parcours et obtiennent le résultat attendu plus rapidement que des boucles en PHP. Les vues SQL peuvent aussi être une solution efficace.

10. Passer à des pages statiques

Qu'en est-il des pages web dynamiques générées à chaque appel utilisateur, et modifiées une à deux fois par an ? Elles mettent du temps à se charger et gaspillent de l'énergie inutilement.
En fonction du produit, il y a différentes solutions. Comme mettre en place un système de cache : les pages générées sont stockées et mises à jour si besoin.
Une autre solution consiste à générer directement des fichiers HTML statiques. Ce type d'architecture se nomme JamStack (JAvascript, Api, Markup). Il existe de nombreux outils pour y parvenir : Jekyll, Nuxt, Gatsby. Avec ce procédé, l'utilisateur obtiendra des réponses plus rapidement et le SEO n'en sera qu'amélioré.

Pour plus d'information :

Sur les langages « Green », une équipe de recherche a publié un article sur le coût électrique des langages :
https://greenlab.di.uminho.pt/wp-content/uploads/2017/10/sleFinal.pdf
Sur le benchmark des frameworks :
https://blog.phalcon.io/post/benchmarking-phalcon
Sur Phalcon : https://www.editions-eni.fr/livre/phalcon-3-developpez-des-applications-web-complexes-et-performantes-en-php-9782409022746
Sur Deflate :
https://httpd.apache.org/docs/2.4/fr/mod/mod_deflate.html
Sur PageSpeed :
https://www.modpagespeed.com/
Sur Jamstack : https://jamstack.org/

Benoît SAKOTE
Ex-Technicien support ProxIT by NeoSoft
En reconversion professionnelle (developpeur web) à IFOCOP.
Musicien et Compositeur à ses heures perdues!

InstaPy : un bot pour Instagram

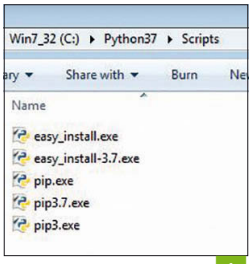
Comment vous vous faites spammer... et comment spammer les autres!

Avertissement : le fait d'utiliser un quelconque système d'automatisation sur un site ou un réseau social peut entraîner des sanctions sur votre compte (ban temporaire ou permanent), voire dans des cas extrêmes des poursuites judiciaires. Nous ne pourrions être en aucun cas tenus responsable de l'utilisation que vous pourriez en faire!

Préambule

Nous allons étudier ici l'installation d'un Bot pour générer de l'activité et gagner des abonnés sur Instagram. Le but ici de cet article n'est pas d'augmenter sa popularité personnelle (quoi que ?!), mais d'apprendre à utiliser un programme simple en Python, et quelques bases sur ce langage fort populaire.

Je ne vous recommande donc pas d'utiliser ce programme à outrance. N'oubliez pas que vous risquez la fermeture irrémédiable de votre compte à tout moment! Je l'ai personnellement utilisé pour lancer la nouvelle page de mon groupe de rock, afin d'arriver facilement à avoir un nombre suffisant de followers (~200) pour vraiment avoir un strict minimum de visibilité dans les recherches.

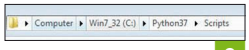


1

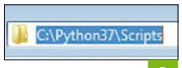
Instagram

On ne présente plus le réseau social leader des smartphones. Acquis par Facebook en 2012, le succès ne dément pas, avec aujourd'hui 1 milliard d'utilisateurs. Si vous n'y avez jamais mis les pieds (ou les pouces), Instagram est un endroit où chacun publie ses photos et ses vidéos, et ou chacun peut "liker" et "commenter" ("partager" est quand à lui peu présent contrairement à Facebook). Le texte est donc un élément moins central par rapport à ses homologues.

Tout compte est rattaché à un e-mail unique, et tout le monde peut suivre un autre compte ou être suivi. Il est impossible de créer un compte à partir d'un autre compte. Même si Instagram dispose d'une version "web", celle-ci est nettement moins permissive que l'application mobile. Sur la version web il est donc impossible de :



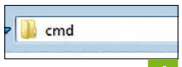
2



3

- Publier un contenu ;
- Publier une "story" ;
- Envoyer des messages privés ;
- Et bien d'autres choses...

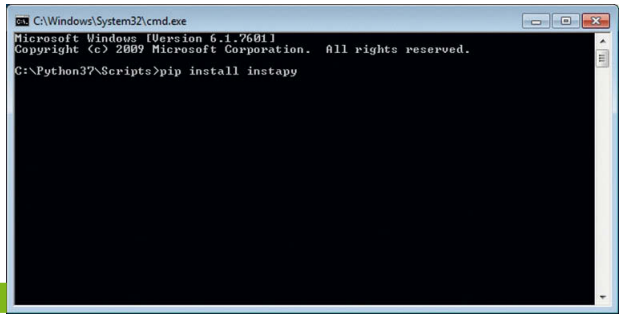
J'insiste donc sur ces points, car vous allez le comprendre très vite, InstaPy a ses limitations.



4

Instapy, les principes

Instapy est un bot qui utilise **Mozilla Firefox** et son contrôle à distance (depuis récemment, car avant, l'application utilisait par défaut **Chromium** via **instapy-chrome**) pour naviguer à votre place sur la plateforme. Vous serez donc limité aux fonctions possibles par Instagram sur la version "web". En interagissant avec les personnes sur le réseau, vous gagnez donc de la visibilité, et des abonnés potentiels. Globalement, cela va se traduire par ces trois actions :



5

- Liker une photo ;
- Mettre un commentaire ;
- S'abonner à un compte.

La troisième méthode est la plus utilisée avec le principe suivant : on s'abonne à une personne, on attend 24h, et on se désabonne !! Ce n'est pas très honnête, un peu faux-cul, mais ça marche plutôt bien ;) ! Maintenant vous comprenez donc pourquoi ce coach fitness ou ce vendeur en drop-shipping s'est abonné à votre compte et pourquoi il a 2500 abonnés!

Pour que le réseau social ne vous repère pas, InstaPy incorpore des possibilités de randomisation. Par exemple : Poster ~10 commentaires choisis parmi une liste de 10 possibles, puis s'arrêter entre 10 et 15 minutes... et recommencer ! Difficilement détectable !

Instapy : Installation

Allez ! On va rentrer dans le concret. Instapy se trouve sur GitHub (https://github.com/timgrossmann/InstaPy), mais on va plutôt utiliser PIP (le gestionnaire de paquets de Python) pour se simplifier la vie.

Commencez donc par installer Python (https://www.python.org/). Pour ma part mon système Windows me limite à la version 3.7.6 (ne me posez pas de questions svp!). Je vous conseille de l'installer directement à la racine (C:\ ), mais surtout pas dans le dossier "appdata" proposé par défaut, sous peine de ne plus vous y retrouver par la suite...

Une fois Python installé, allez dans le dossier C:\Python/Script (1) Petite astuce, tapez directement "cmd" dans la barre d'adresse (2 , 3 et 4) pour avoir votre prompt directement dans le bon dossier! Je parie que vous ne connaissiez pas cette petite astuce... avouez! ;)

Avec votre CMD dans le bon dossier, tapez simplement ceci (5) :

```
pip install instapy
```

Après 1 ou 2 minutes (6) tout devrait être prêt. Instapy est maintenant installé dans ce dossier :

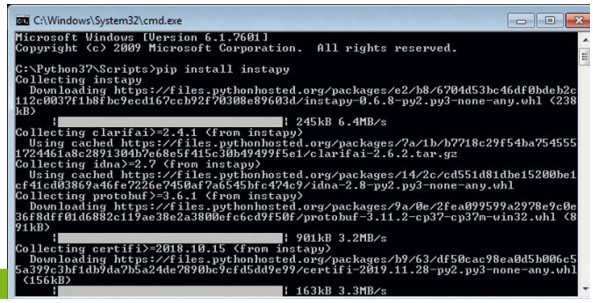
```
C:\Python37\Lib\site-packages\instapy
```

Tout est prêt, passons à la suite!

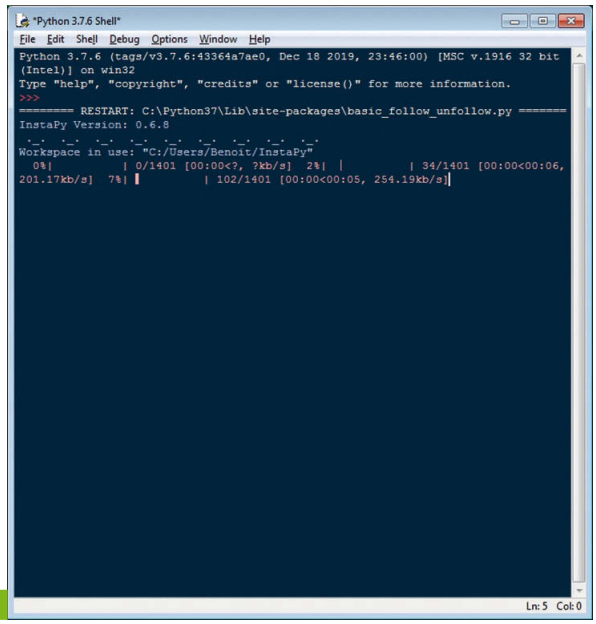
PS : Evidemment, n'oubliez pas de vous créer un compte Instagram !

Instapy : les templates

J'ai une bonne et une mauvaise nouvelle. La mauvaise, c'est que nous n'allons pas vraiment coder… La bonne, c'est que ça va être



6



8

plutôt simple ! On va simplement aller piquer des templates déjà fait sur github : https://github.com/InstaPy/instapy-quickstart . Prenons donc le **basic_follow-unfollow_activity.py** . Copier/coller tout le texte puis mettez-le dans un nouveau fichier *.py ici :

```
C:\Python37\Lib\site-packages\basic_follow_unfollow.py
```

Ouvrez le fichier dans votre éditeur de code favori et modifies en premier lieu le login comme ceci :

```
# login credentials
insta_username = 'sauksie_and_the_frenchees'
insta_password = '123456'
```

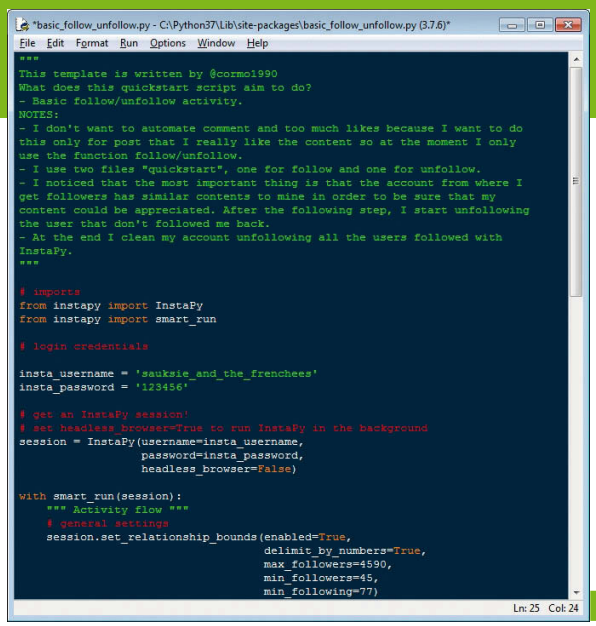
Python, un langage simple mais exigeant : si vous n'êtes pas familier avec Python, sachez simplement que les accolades { et } ne sont pas utilisées. Ce sont les indentations (= les tabulations) qui seront prises en compte.

On peut ensuite configurer les hashtag sur lesquels nous allons faire l'impasse. L'exemple ici concernant un groupe de rock, on va donc bannir certains tags :

```
session.set_dont_like(["pizza", "#store", "#rap", "#rnb", "#jul", "#pnl"])
```

On va ensuite définir des comptes avec des abonnés susceptibles d'apprécier notre contenu :

```
session.follow_user_followers(['gothiquememes', 'theunchained', 'benoit_adam', 'spiral_static_rock'], amount=800,
```



7

Attention, il y a aussi une 2ème ligne similaire à modifier pour le 2ème "rush"!

```
session.follow_user_followers(['user1', 'user2', 'user3'], amount=800
```

On va maintenant modifier les commentaires par défaut par des commentaires plus personnalisés. On peut y mettre des smileys et des mentions @nomdutilisateur

```
photo_comments = ['Super !! @{}',
    'Great @{}',
    'Cool :thumbsup:',
    'Incredible :open_mouth:',
    'Siouxsie And The Banshees is the greatest post-punk band, you agree @{} ?',
    'Love your posts @{}',
    'Crazy !!!!',
    'More please !!',
    ':raised_hands: Yes !',
    'Hey, do you also like Siouxsie And The Banshees @{} ?']
```

Il ne reste donc plus qu'à tester tout ça ! Fermez votre éditeur de code. Il vous faut ouvrir le fichier avec IDLE (7). De là, deux solutions :

- Ouvrez "IDLE (Python 3.7 32 Bits)" puis File => Open et cherchez votre fichier
- Clic-droit sur votre fichier *.py, puis "Edit with IDLE" => "Edit with IDLE 32 Bits"

Il ne vous reste plus qu'à lancer tout ça avec **Run = Run Module** ou tout simplement **F5** ! (8) et (9)

Un problème ? Pas de problèmes !

Pensez à configurer votre firewall, sous Windows, l'alerte se déclenche automatiquement (10).

Il se peut qu'une erreur apparaisse : une faute de syntaxe ou une tabulation qui bloque (je rappelle encore une fois que ce sont les tabulations qui remplacent les accolades { }), si c'est le cas corrigez et re-testez ! Au pire des cas, repartez à 0 sur le template de base!

Si vous avez déjà exécuté le script une première fois et que vous voulez recommencer, soyez certains de bien fermer tous les programmes concernés. Cela comprend Firefox, Python ainsi que le process "Geckodriver.exe" !

Dernière astuce : ne pas afficher le navigateur !

Il n'est pas nécessaire que la fenêtre du navigateur soit visible, modifier simplement cette ligne dans le script :

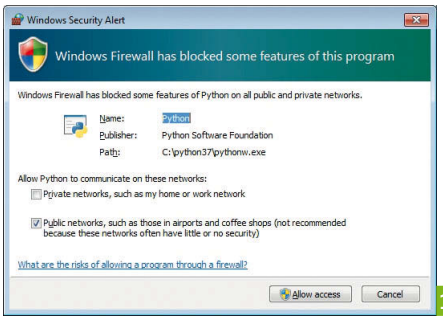
```
headless_browser=False)
```

Remplacez *False* par *True* et relancez le script !

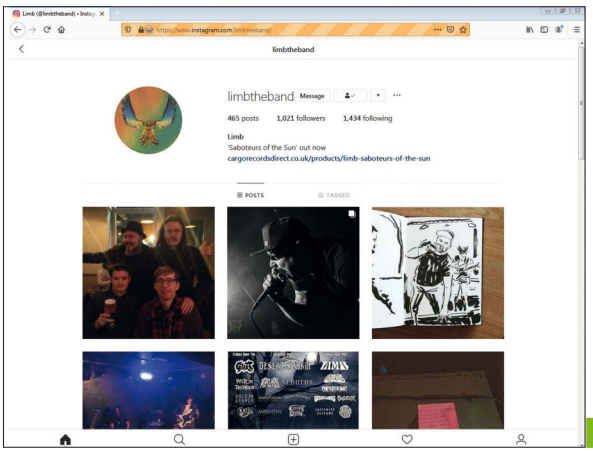
Aller plus loin

InstaPy ne fournit que des fonctions. Vous pouvez donc modifier à loisir les scripts fournis, et même créer vos propres scripts ! Attention toutefois à choisir des valeurs d'attente et de randomisation correctes. La documentation de InstaPy recommande certains seuils minimums pour éviter un ban automatique. Allez-y avec prudence!

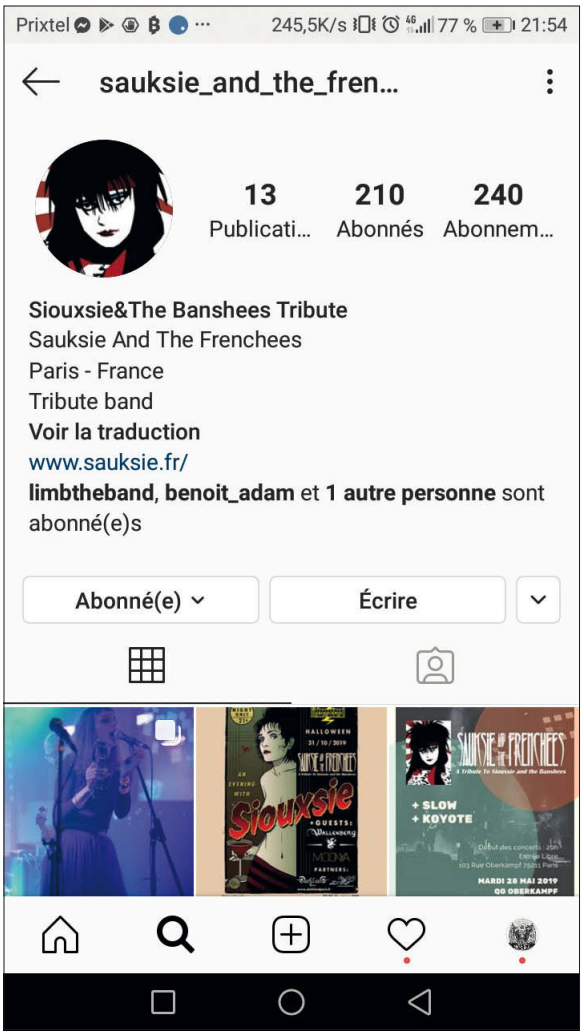
La page principale du projet : https://github.com/InstaPy
Le tutoriel vidéo dont je me suis inspiré : https://www.youtube.com/watch?v=bwB5mR5PpTg
Le tutoriel vidéo officiel : https://www.youtube.com/watch?v=9DkEI2MrFQk
Dédicace à Sylvqin qui m'a fait découvrir InstaPy : https://www.youtube.com/watch?v=Civ6x37ArQo
L'indentation sous Python : https://python.developpez.com/cours/apprendre-python3/?page=page_5#L5-F-2

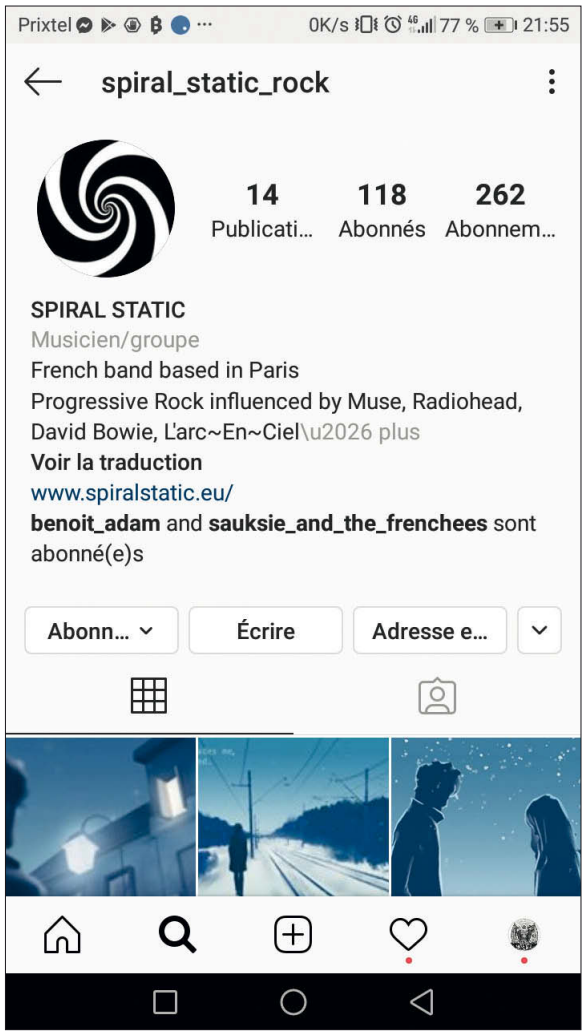


10



9





Benjamin Petetot
Open Source Advocate
chez **Zenika**

zenika
<animés par la passion>

Démarrer dans l'open source : la contribution

Le monde de l'open source est vaste, il est très facile de s'y perdre. Quand on est nouveau dans l'open source, ce monde peut paraître peu accessible et intimidant. Par où commencer ? Y a-t-il des règles et des codes à suivre ? Comment contribuer, même si je ne code pas ? Pas d'inquiétude, dans cet article nous allons te donner tous les codes nécessaires pour bien démarrer, choisir le bon projet et proposer ta première contribution.

Premiers pas

Tout d'abord, il faut déterminer sur quel projet tu souhaites démarrer ta première contribution. Comme il existe des millions de projets open source, il peut être difficile de trouver son chemin.

Une première idée, qui est peut-être la plus simple, est **d'identifier des projets que tu aimes, que tu utilises souvent ou qui t'intéressent.** Il sera alors plus évident de démarrer dans un environnement que tu connais et sur lequel tu garderas de la motivation à plus long terme.

Il est également important de prendre en compte la maturité du projet et sa taille. Il est tentant de démarrer sur de gros frameworks, librairies ou outils comme VS Code, MongoDB, Android, React… Ces types de projets ont de nombreux contributeurs et beaucoup de bugs à corriger ou fonctionnalités à réaliser. Les contributions sur ces projets apportent une certaine satisfaction quand elles sont intégrées. Mais ces projets, avec des millions de lignes de code, ne sont pas les meilleurs choix pour débuter dans l'open source. Ils seront plus difficiles à appréhender par leur taille et les workflows qui viennent avec. Leurs communautés sont déjà très développées et beaucoup de personnes proposent des contributions pour un nombre de mainteneurs limités. Il peut alors se passer un certain temps avant que ta contribution ne soit étudiée, puis éventuellement acceptée.

Démarrer sur des projets plus petits et qui ont du potentiel est un meilleur moyen de débuter. Pour cela, il suffit de surveiller régulièrement ceux qui montent, par exemple sur le trending de GitHub, et de trouver des projets qui te correspondent.

Un autre bon moyen d'identifier un projet sur lequel participer est de **sélectionner ceux qui recherchent activement des contributeurs.** Plusieurs sites et outils permettent d'en trouver :

- Awesome for beginners est une liste de projets à la recherche de contributeurs (débutants ou non) en listant leurs « *Good First Issues* ». De la même manière, le site Up for grabs permet de trouver des projets à la recherche de contributeurs.
- CodeTriage est un outil permettant de s'abonner à des projets open source et de recevoir de nouvelles demandes de contribution tous les jours.
- First contributions est un peu différent, c'est un workshop pour réaliser sa première contribution en moins de cinq minutes. Il présente la base du workflow git pour faire des contributions sur GitHub. En plus de ce tutoriel, le site propose des liens vers les « *Good First Issues* » de gros projets GitHub.
- Les soirées ou évènements communautaires comme le HacktoberFest représentent également un bon moyen de démarrer sur des projets à la recherche de contributeurs. Par exemple, le hack.commit.push propose une journée de développement open source, ouverte à tous, quel que soit votre niveau. En début de journée, des ateliers d'introduction sont présentés pour ensuite enchaîner sur des contributions open source accompagnées de mentor.

Une fois que tu as trouvé le ou les projets de tes rêves, il te faudra commencer par te documenter sur ces projets, par exemple en lisant le README, le code de conduite et le guide de contribution. Ce n'est pas très long, mais nécessaire avant de démarrer une contribution. Ils te donneront les codes, les règles et les procédures à respecter pour la consistance et la maintenance du projet.

Le code de conduite (code of conduct)

La plupart des projets open source écrivent un code de conduite. Il permet de définir des règles et des comportements à adopter ou non sur le projet.

Les points essentiels de ce code définissent les règles à suivre entre les participants, comme savoir respecter les différents points de vue, ou encore écouter et accepter les feedbacks sur ses contributions. Néanmoins il faut s'attendre à de potentiels désaccords. Il ne faut pas les prendre personnellement. Parfois, lors des revues, il est possible que l'on ne soit pas du même avis sur certaines décisions ou approches adoptées. Il est alors très important de rester professionnel et respecter le code de conduite. Tout feedback est bon à prendre et c'est lors de ce type de discussion que l'on apprend le plus.

Il est également très important d'être patient. De nombreux projets open source sont maintenus sur le temps personnel. Il peut donc se passer un certain temps avant que sa contribution ne soit relue ou acceptée. Voici un exemple de code de conduite utilisé par de nombreux projets : https://www.contributor-covenant.org.

Les règles qu'il définit devraient être appliquées autant dans les projets open source que dans le milieu professionnel.

Le guide de contribution

Il permet de définir les workflows et les manières de coder sur le projet afin de comprendre les normes et les attentes de la communauté. Par exemple :

- Comment mettre en place mon environ-

nement de développement ?
- Comment poster une issue ?
- Quelles sont les normes de commit ?
- Quelles sont les attentes sur les tests ?
- …

Si une contribution ne suit pas ces bonnes pratiques, il se peut qu'elle soit rejetée avant même d'avoir été revue. Ces normes servent à garder une cohérence dans le projet mais également à faire gagner du temps aux mainteneurs du projet. Donc à lire avec attention.

Rejoindre la communauté

Un projet ne se limite pas à la documentation et au code source. Il est également important de comprendre et d'interagir avec la communauté du projet. Il y a différents rôles dans un projet open source, les trois principaux étant :

- Les mainteneurs ou core contributors, qui sont à l'origine du projet ou ont intégré ce rôle suite à l'ensemble des contributions qu'ils ont réalisées.
- Les contributeurs, qui participent à la vie du projet en proposant des contributions (documentation, correction de bug…) mais également en répondant à des questions ou des tickets des utilisateurs.
- Et les utilisateurs, qui vont bien sûr utiliser le projet mais surtout donner des feedbacks, poster des bugs, proposer des améliorations ou poser des questions sur des forums.

Les moyens de communication de la communauté varient en fonction des projets. Il peut s'agir d'une mailing list, d'un Slack, d'un Discord ou tout autre outil du même type. Ces derniers permettent de poser des questions, d'aider à définir les roadmaps, ou tout simplement de discuter entre passionnés.

Première contribution

Maintenant que tu as identifié un projet qui t'intéresse et que tu as lu la documentation nécessaire pour bien démarrer, tu es prêt à proposer ta première contribution.

Avant tout, sache que tu n'as pas besoin de savoir coder pour contribuer sur des projets open source :

- Tu aimes écrire ? Corrige ou développe la documentation du projet.
- Tu maîtrises le design ? Crée le logo ou le site du projet.
- Tu aimes communiquer ? Réponds aux questions de la communauté.

Toutes les contributions sont bonnes et servent au projet. Voici une liste non exhaustive des tâches que tu peux réaliser pour aider les projets open source :

Documentation

- Écrire et corriger la documentation du projet ;
- traduire la documentation.

Organisation

- Soumettre des bugs ou nouvelles fonctionnalités ;
- répondre aux issues, les classer et les fermer si nécessaire ;
- répondre aux questions (par exemple sur GitHub, Stack Overflow…).

Communication

- Écrire des articles ou tutoriels ;
- organiser des meetups ou conférences sur le projet ;
- communiquer sur Twitter (nouvelles releases, contributions…) ;
- réaliser un design pour le projet (logo, couleurs…) ;
- développer un site pour le projet.

Coder

- Réaliser des revues de code ;
- développer les tests ;
- automatiser le projet (CI…) ;
- corriger des bugs ;
- réaliser de nouvelles fonctionnalités.

Si tu ne sais pas par où commencer, parcours la liste des issues ouvertes sur le projet. Dans beaucoup de projets, certains tickets sont identifiés comme « *Good First Issues* » ou « *Beginner* ». Ils ont été identifiés par les contributeurs principaux comme des contributions simples et de bons points d'entrée dans le projet. Regarde également les tickets catégorisés « Help needed » qui attendent qu'un contributeur prenne le sujet en main.

Quand tu as identifié une issue ou un ticket sur lequel tu souhaites contribuer, il faut te positionner dessus. Il ne faut pas hésiter à demander des informations aux autres contributeurs. Le dialogue et l'échange sont essentiels pour correctement démarrer une nouvelle contribution. Ils permettront de t'orienter vers les attentes de la communauté, de valider une solution technique ou d'identifier certaines contraintes ou difficultés. Avant de publier ta première contribution, vérifie ton code et n'oublie pas les tests avant de le soumettre. Il ne faut pas hésiter à expliquer le contexte et documenter sa contribution. Tout ce qui pourra aider les autres contributeurs et mainteneurs à relire ta contribution fera qu'elle sera revue et intégrée plus rapidement. Enfin, quand tu auras réalisé ta première contribution, n'oublie pas de communiquer dessus et de la partager au monde entier sur Twitter, Facebook, ou ton réseau social favori .

Licences

Que vous soyez un développeur ou bien un mainteneur de projets Open Source, il est indispensable d'avoir une connaissance des différents types de licences, et de savoir quels sont vos droits et devoirs lorsque vous les utilisez. Plusieurs types de licences s'offrent à vous, et chacune peut présenter des avantages et des inconvénients qu'il convient de mesurer avant d'en choisir une.

Mais il faut rester vigilant. Il peut y avoir des confusions entre les différentes catégories de logiciels « libres » (OSS, freeware, Domaine public…) et il existe des dizaines de licences libres dont certaines sont virales. Cet article a pour but de vous aider à comprendre les différentes caractéristiques des licences et de choisir la plus adaptée à votre projet.

Logiciel libre et open source

La Free Software Foundation (FSF), société américaine à but non lucratif fondée en 1985 par R. Stallman, soutient le système d'exploitation GNU et gère les licences « GPL ».

Cette fondation donne une définition du logiciel libre basée sur quatre libertés irrévocables :

- Liberté d'utiliser le logiciel ;
- Liberté de copier le logiciel (comprend la liberté de vendre des copies) ;
- Liberté d'étudier le logiciel (suppose l'accès au code source) ;
- Liberté de modifier le logiciel et de redistribuer les versions modifiées.

Les œuvres sous licence libre ne sont pas toujours disponibles gratuitement et les services associés (développement, garanties, support, etc.) sont souvent payants. Une œuvre contaminée par une licence libre, qui pouvait au départ n'être disponible que contre paiement, doit être rediffusée librement.

« Think Free as Free speech and not Free beer » – R. Stallman

L'Open Source Initiative (OSI), société américaine à but non lucratif fondée en 1998 par E. Raymond & B. Perens promeut quant à elle le code open source. Selon l'OSI, pour qu'un logiciel soit open source, sa licence doit respecter les dix critères suivants :

- Redistribution libre ;
- Fourniture du code source ;
- Travaux dérivés et modifications autorisés ;
- Intégrité du code source de l'auteur ;
- Aucune discrimination envers des individus ou des communautés ;
- Aucune discrimination envers des domaines d'application ;
- Pas de restriction sur la distribution de la licence (vs NDA) ;
- La licence ne doit pas être spécifique à un produit ;
- La licence ne doit pas restreindre d'autres logiciels ;
- La licence doit être technologiquement neutre.

« Les deux termes décrivent pratiquement la même catégorie de logiciel. Mais ils représentent des vues basées sur des valeurs fondamentalement différentes. »

— R. Stallman (https://twitter.com/OpenSourceOrg/status/1174073039534706688?s=20)

Copyright, copyleft et domaine public

Avant de présenter les différents types de licences disponibles, il est nécessaire de comprendre les différents types de droits appliqués aux œuvres. Voici une explication, en termes simples, des trois grands domaines d'application des droits.

Le **copyright** (ou droit d'auteur) est un droit accordé à l'auteur de l'œuvre originale, y compris le droit d'autoriser ou d'interdire la publication ou la distribution de son œuvre. Il protège les auteurs de la copie ou de la vente non autorisée de leurs œuvres. Les droits d'auteur sont accordés pour une durée limitée, au terme de laquelle l'œuvre entre dans le domaine public.

Le **domaine public** comprend toutes les créations auxquelles ne s'applique aucun droit de propriété intellectuelle exclusif. Ces droits peuvent être expirés, avoir été confisqués, expressément renoncés ou être inapplicables. Les droits d'auteur sont généralement valables jusqu'à cinquante à cent ans après le décès de l'auteur. En termes simples, tout le monde peut utiliser, modifier et vendre ces créations sans l'autorisation de son auteur.

Par exemple, les compositions de Beethoven sont entrées dans le domaine public soixante-dix ans après sa mort en 1827. Ses compositions musicales sont disponibles pour être utilisées et vendues par tous.

Sous **copyleft**, tout le monde peut modifier et distribuer le travail. Cela ne nécessite qu'une condition : la même liberté doit être préservée dans les versions modifiées de l'œuvre originale. Les gens peuvent utiliser, modifier et distribuer le travail comme ils le souhaitent. Toutefois, le copyleft oblige le travail modifié à être distribué sur la base de la même licence. Cependant, il n'est pas nécessaire que le contenu copylefté soit rendu gratuit comme le travail dans le domaine public. La licence publique générale GNU, écrite à l'origine par Richard Stallman, était la première licence copyleft.

Quel type de licence choisir ou utiliser ?

Un logiciel libre a toujours une licence (contrat) d'utilisation associée, il est nécessaire d'analyser les conditions d'utilisation, les droits et les obligations en résultant.

Pour faciliter la compréhension des différents types de licence, nous pouvons les classifier selon le degré de liberté qu'elles accordent à l'utilisateur :

- « Copyleft fort » ;
- « Copyleft faible » ;
- « Permissive ».

Copyleft fort

Une licence avec un copyleft fort a **un caractère fortement contaminant**. Lorsque le composant d'un logiciel est sous une licence très fortement copyleftée, cette même licence s'impose à l'ensemble du logiciel qui contient ce composant lors de sa diffusion. On parle alors de licence « contaminante » ou « virale ». Elle oblige à distribuer le logiciel libre modifié sous la même licence et à rendre disponible le code source associé.

Quelques licences à copyleft fort :

- GPL : General Public License V3 ;
- AGPL : Affero General Public License ;
- CC : Creative Commons déclinée en 6 licences, dont certaines permissives.

Dans le cas de la licence GPL, il existe quelques cas explicites d'utilisation qui ne déclenchent pas le phénomène de contamination, soit par des clauses d'exception inscrites dans la licence (par exemple le GCC Runtime), soit dans le cas de binaires indépendants, c'est-à-dire que votre logiciel utilise un programme ayant la license GPL mais ne le distribue pas. Les threads Stack Overflow sont également sous licence. Lorsque vous copiez-collez un bout de code depuis le forum, il est important de savoir que tous les snippets de code postés sont sous la licence Creative Commons CC.

Copyleft faible

Une licence avec un copyleft faible a **un caractère faiblement contaminant**. Elle oblige à distribuer le logiciel libre modifié sous la même licence et à rendre disponible le code source associé.

Quelques licences à copyleft faible :

- LGPL : Lesser GPL ;
- EPL : Eclipse Public License ;
- MPL : Mozilla Public License.

Permissive

Les licences permissives offrent **la plus grande liberté avec un partage sans condition**. En général, seule la citation des auteurs originaux est demandée. Ainsi elles permettent à tout acteur de changer la licence sous laquelle le logiciel est distribué, sans obligation de diffusion.

Quelques licences permissives :

- BSD : Berkeley Software Distribution ;
- MIT : Massachussetts Institute of Technology ;

Apache.

Il existe également des licences permissives plus ésotériques, comme la Postcard licence dont la seule obligation est d'envoyer une carte postale à l'auteur, ou encore la WTFPL (Do What the Fuck You Want to Public License), dont le titre dit tout.

Conclusion

En tant que développeur, si vous utilisez des librairies ou logiciels tiers, n'oubliez pas de vérifier leur licence car celle-ci peut modifier celle de votre logiciel. En tant que mainteneur de projets open source, choisissez correctement votre licence, que ce soit pour protéger l'œuvre ou son utilisation.

Le site web https://choosealicense.com vous permet de choisir une licence en fonction de vos usages.

CHEAT SHEET : PROMOUVOIR UN PROJET OPEN-SOURCE

Franck Abgrall
Développeur - VuePress Core Team - Speaker - Formateur - Mainteneur Open Source chez **Zenika**

Thomas Betous
Ingénieur d'étude et de développement chez **Zenika** *Nantes*

Ce cheat sheet résume les étapes importantes à suivre pour promouvoir un projet open-source dans les meilleures conditions. Il est possible d'afficher des informations supplémentaires pour un élément de la liste en cliquant dessus.

Langages disponibles :

- English
- Français
- Deutsch
- Español
- 简体中文
- 繁體中文
- Português

Une langue est manquante ? Vous pensez qu'il est possible d'améliorer ce cheat sheet ? https://github.com/zenika-open-source/promote-open-source-project/blob/master/CONTRIBUTING.md !

1. Préparation

- S'assurer que le projet soit assez mature
- Choisir un nom cool pour son projet
- Soigner la présentation du README
- Mettre en avant les points forts du projet
- Mettre une démo du projet à disposition
- L'installation et l'utilisation du projet doivent être les plus simples possibles
- Créer une documentation soignée et structurée.

2. Communiquer le projet

- Mettre en confiance les futurs visiteurs avant de publier sur les réseaux sociaux
- Partager le projet sur les réseaux sociaux et les plateformes spécialisées
- Écrire des articles en mentionnant le projet
- Présenter le projet à des conférences/meetups
- Enregistrer et publier des vidéos de présentation du projet
- Choisir le meilleur moment pour publier sur les réseaux sociaux
- Ne pas spammer les plateformes a la promotion du projet.

3. Garder les utilisateurs

- Mettre régulièrement à jour le projet
- Maintenir le projet et traiter les issues ouvertes
- Inviter les utilisateurs à contribuer
- Récompenser les contributeurs
- Ouvrir un chat pour la communauté du projet
- Demander des retours utilisateurs
- Montrer ce que les autres utilisateurs ont créé avec votre projet.

Lien : https://github.com/zenika-open-source/promote-open-source-project

Benjamin Dupin
Software Crafter chez **Arolla**

Quoi de neuf dans Java 14 ?
Bientôt l'arrivée du pattern matching dans Java !

La 14ème version de Java est sortie le 17 mars. Notez que cette version de Java ne bénéficie pas du LTS (Long Time Support). La prochaine version de Java LTS sera la version 17 qui sortira en septembre 2021.

Il ne s'agit cependant pas d'une petite mise à jour. En effet, le JDK 14 embarque plus d'une dizaine de nouvelles fonctionnalités, concernant à la fois le langage en lui-même, mais aussi des améliorations de la JVM et d'autres outils.
Découvrons ensemble les nouvelles possibilités qu'offre le JDK 14.

Nouveautés du langage

Pattern Matching avec instanceof

On le sait depuis un moment, Java souhaite implémenter le filtrage par motif (pattern matching) et a commencé depuis sa version 12 à enrichir ses switchs.
C'est désormais au tour du instanceof d'être amélioré.
Lorsqu'on utilise l'opérateur instanceof, on est généralement obligé de caster l'objet sur lequel on vérifie l'instance afin de l'utiliser :

```
if (obj instanceof String) {
  String s = (String) obj;
  int length = s.length();
}
```

Maintenant, il n'est plus nécessaire de caster notre objet, on peut directement déclarer une variable après le instanceof, de cette manière :

```
if (obj instanceof String s) {
  int length = s.length();
}
```

Ou encore :

```
if (obj instanceof String s && !s.isBlank()) {.. s.contains(..) ..}
```

NullPointerExcepion améliorée

On a tous déjà eu affaire à la fameuse NullPointerException de Java. Prenons par exemple ce message :

```
Exception in thread "main" java.lang.NullPointerException
  at Prog.main(Prog.java:5)
```

Si à la ligne 5 de la classe Prog on trouve le morceau de code suivant :

```
a.i = 99;
```

Il est facile de déterminer l'origine de l'exception (« a » est null)
Mais si par exemple on retrouve ce morceau de code là :

```
a.b.c.i = 99; // A ne pas reproduire, c'est pour l'exemple ;-)
```

Qu'est ce qui est null ? « a », « b », ou « c » ?
Désormais, les messages des NullPointerException seront bien plus explicites. On trouvera ainsi par exemple :

```
Exception in thread "main" java.lang.NullPointerException:
    Cannot read field "c" because "a.b" is null
  at Prog.main(Prog.java:5)
```

Voici la liste des différents nouveaux messages :
- Cannot load from <element type> array
- Cannot read the array length
- Cannot store to <element type> array
- Cannot throw exception
- Cannot read field "<field name>"
- Cannot invoke "<method>"
- Cannot enter synchronized block
- Cannot exit synchronized block
- Cannot assign field "<field name>"

Bien plus simple pour déboguer n'est-ce pas ?

Les Records

Un record est un nouveau genre de déclaration de type en Java. Tout comme un enum, il s'agit d'une class avec des restrictions.
La déclaration d'un record se fait en lui donnant un nom et un état (ses attributs) :

```
record Point(int x, int y) { }
```

L'équivalent du record ci-dessus en class serait :

```
public final class Point{

  private final int x;
  private final int y;

  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }

  public int getX() {
    return x;
  }

  public int getY() {
    return y;
```

```
    }

    @Override
    public boolean equals(Object o) {
        // Implémentation d'equals en utilisant x et y
    }

    @Override
    public int hashCode() {
        // Implémentation d'hashCode en utilisant x et y
    }

    @Override
    public String toString() {
        return "Point{" +
                "x=" + x +
                ", y=" + y +
                '}';
    }
}
```

Implicitement, un record contient :

- Des champs private final ;
- Un constructeur public ;
- Des getters ;
- Une implémentation des méthodes equals, hashCode et toString qui utilise tous les attributs de celui-ci.

Les records ne peuvent pas hériter d'autres classes, ni contenir des champs supplémentaires que ceux décrivant l'état du record (exceptées des variables static).

Ils sont implicitement *final* et ne peuvent pas être abstract.

Derrière toutes ces contraintes, l'objectif est d'appliquer la politique de l'immutable par défaut, qui est largement préférable aux objets contenant des données.

Il est cependant possible d'explicitement déclarer des membres générés automatiquement, comme par exemple le constructeur :

```
record Range(int lo, int hi) {
  public Range {
    if (lo > hi)
      throw new IllegalArgumentException(String.format("(%d,%d)", lo, hi));
  }
}
```

Les records permettent d'éviter les erreurs ou oublis lorsqu'on rajoute un attribut à un POJO, comme la prise en compte de ce nouvel attribut dans equals, hashCode et toString, et ainsi devenir source de bug. Enfin, et ce n'est pas le moins important, les records vont faire réfléchir à deux fois un développeur qui souhaite modifier le model en rajoutant simplement un nouvel attribut à un objet porteur de données alors qu'il serait peut-être plus judicieux de créer une nouvelle classe.

Amélioration du switch

Initialement sorti en preview avec Java 12, puis légèrement modifié avec Java 13, c'est finalement avec le JDK 14 que les switchs améliorés sont disponibles en version finale. Pour rappel, le switch peut désormais s'écrire de cette manière, sans break :

```
switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> System.out.println(6);
    case TUESDAY                -> System.out.println(7);
    case THURSDAY, SATURDAY     -> System.out.println(8);
    case WEDNESDAY              -> System.out.println(9);
}
```

Mieux encore, le switch devient une expression à part entière :

```
int j = switch (day) {
    case MONDAY  -> 0;
    case TUESDAY -> 1;
    default      -> {
        int k = day.toString().length();
        int result = f(k);
        yield result;
    }
};
```

Notez l'utilisation du mot clef yield pour retourner la valeur dans le cas d'un bloc de code.

Text Blocks

Initialement sorti avec le JDK 13, les Text Blocks permettent l'instanciation de chaînes de caractères sur plusieurs lignes sans concaténation :

```
String query = """
        SELECT `EMP_ID`, `LAST_NAME` FROM `EMPLOYEE_TB`
        WHERE `CITY` = 'INDIANAPOLIS'
        ORDER BY `EMP_ID`, `LAST_NAME`;
        """;
```

Avec Java 14, les Text Blocks sont toujours en preview mais avec quelques nouveautés permettant le contrôle des sauts de ligne et des espaces :

```
String text = """
        Lorem ipsum dolor sit amet, consectetur adipiscing \
        elit, sed do eiusmod tempor incididunt ut labore \
        et dolore magna aliqua.\
        """;

// Equivaut à
String literal = "Lorem ipsum dolor sit amet, consectetur adipiscing " +
        "elit, sed do eiusmod tempor incididunt ut labore " +
        "et dolore magna aliqua.";
```

Et pour spécifier des espaces, on peut utiliser le caractère unicode « \s » :

```
String colors = """
    red  \s
    green\s
    blue \s
    """;
```

Modifications apportées aux Garbage Collectors

Implémentation du NUMA pour G1

Le ramasse-miettes G1 (Garbage First), qui est devenu le garbage collector par défaut depuis Java 9, s'est vu implémenter un nou-

veau système d'allocation mémoire compatible avec les architectures NUMA (pour Non Uniform Memory Architecture, système dans lequel les zones mémoires sont séparées), ce qui devrait largement améliorer les performances des applications Java sur les machines avec plusieurs processeurs. 1

ZGC sur Windows et macOS

Z Garbage Collector est un nouveau ramasse-miettes apparu avec Java 11, toujours expérimental, et désormais compatible avec Windows et macOS. Il se veut scalable et à faible latence (moins de 10ms), pouvant gérer des tas de quelques centaines de mégaoctets à plusieurs téraoctets avec le même temps de pause.

Outils

jpackage, un nouvel outil de packaging

Le besoin derrière cet outil nommé jpackage (qui est toujours en phase d'incubation dans cette version du JDK) est de permettre la création d'un exécutable propre au système d'exploitation au lieu d'un simple JAR. Ainsi pour Windows on peut désormais packager notre application directement en exe ou msi, pkg ou dmg sur macOS et deb ou rpm sur Linux.

Streaming des événements du JDK Flight Recorder

Java Flight Recorder (JFR) est un outil de collecte de données et de diagnostic d'application Java. Il a comme objectif d'utiliser JFR pour faire du monitoring d'application en plus du profilage. Il est d'ailleurs désormais possible de consulter les données émises par le Java Flight Recorder de manière asynchrone :

```
try (var rs = new RecordingStream()) {
 rs.enable("jdk.CPULoad").withPeriod(Duration.ofSeconds(1));
 rs.enable("jdk.JavaMonitorEnter").withThreshold(Duration.ofMillis(10));
 rs.onEvent("jdk.CPULoad", event -> {
  System.out.println(event.getFloat("machineTotal"));
 });
 rs.onEvent("jdk.JavaMonitorEnter", event -> {
  System.out.println(event.getClass("monitorClass"));
 });
 rs.start();
}
```

Dépréciations et suppressions

Il n'y a pas que de nouvelles features dans le JDK 14, il est aussi temps d'un petit ménage de printemps.

Portage vers Solaris et SPARC

Dans le but d'accélérer le développement de nouvelles fonctionnalités, le portage du JDK 14 vers le système d'exploitation Solaris et les architectures de processeur SPARC est déprécié et sera prochainement supprimé. Les développeurs qui souhaitent que ces portages soient toujours maintenus sont invités à exprimer leur souhait auprès d'Oracle.

Suppression du ramasse-miettes Concurrent Mark Sweep (CMS)

Toujours avec comme objectif d'améliorer ou de développer de nouveaux ramasse-miettes pour Java, et notamment G1, le garbage collector Concurrent Mark Sweep (CMS), qui est déprécié depuis Java 9 est désormais supprimé.

Utilisation combinée des ramasse-miettes Parallel Scavenge et Serial Old

Dans de très rares cas, les ramasse-miettes nommés « Parallel Scavenge » et « Serial Old » sont utilisés de manière simultanée sans grande plus-value, comparée au coût de maintenance de cette combinaison de ramasse-miettes, qui est maintenant dépréciée.

Pack200

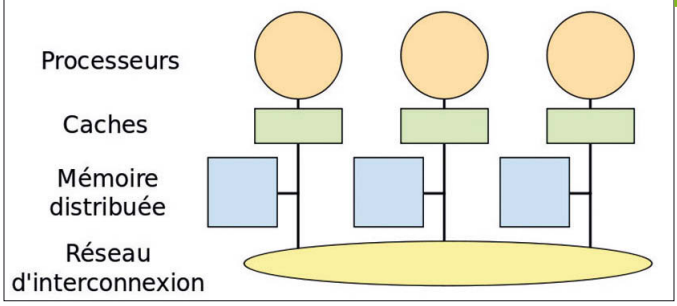
Pack200 est un outil permettant de transformer des fichiers JAR en fichier compressé pack200 permettant de gagner de la bande passante lors de leur téléchargement. Cet outil est déprécié depuis Java 11 et est maintenant supprimé avec Java 14.

Migration vers Java 14

Si vous développez avec Java 11 ou plus : à moins d'utiliser Pack200 ou Concurrent Mark Sweep (qui sont dépréciés depuis Java 11 et supprimés avec le JDK 14), la migration se fera sans aucun problème. Si vous utilisez une version précédente à Java 11, il est plutôt recommandable d'utiliser la version LTS (Java 11 donc), avant d'envisager une migration vers le JDK 14. Pour développer avec le JDK 14, il est recommandé d'utiliser IntelliJ IDEA 2020.1 ou Eclipse IDE 2020-03.

Conclusion

Le développement de Java continue son évolution et son amélioration, en mettant l'accent sur la correction des points faibles du langage, comme par exemple le fait que Java soit trop verbeux.
Le pattern matching devrait arriver très prochainement dans Java, en tout cas avant Java 17 (la prochaine version bénéficiant du support à long terme) prévu pour septembre 2021.
Java (re)devient enfin un sérieux concurrent à NodeJS, ou Python, mais aussi et surtout aux autres langages de la JVM, comme Kotlin et Scala. Serait-ce le début de la fin pour ces langages ?

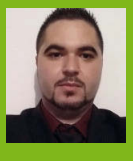


1 *Schéma de principe d'une architecture NUMA*

LES PRINCIPALES NOUVEAUTÉS / AMÉLIORATIONS DEPUIS JAVA 8

- **Java 8 :** apparition des lambdas, Stream API
- **Java 9 :** Collections, nouveautés dans les Streams, Optionals, JShell, HTTPClient, projet Jigsaw
- **Java 10 :** mot clé var, modification dans le ramasse-miette
- **Java 11 :** nouvelles méthodes pour les strings et files, exécution des fichiers sources sans compilation en premier, HttpClient en version finale
- **Java 12 :** support Unicode 11, nouvelle expression switch
- **Java 13 :** Unicode 12.1, Switch Expression (pré-version), Multiline String (pré-version)
- **Java 14 :** voir l'article du mois

La rédaction



Anthony Giretti
MVP, MCSD
Developer, Blogger, Speaker
anthony.giretti@gmail.com
http://anthonygiretti.com

gRPC & ASP.NET Core 3.1 : Comment ça marche ?

Tout d'abord gRPC est un framework RPC (Remote Procedure Call) et non pas un framework Microsoft. gRPC, développé par Google, utilise le protocole HTTP2 pour le transport des données. Il utilise aussi Protocol Buffers comme format de sérialisation qui est également développé par Google. Donc il s'agit d'un descripteur d'interface tout comme le WSDL pour SOAP, si on veut faire une analogie. Avec gRPC, on peut s'authentifier, transmettre des données de manière bidirectionnelle, annuler la communication, paramétrer des délais d'attente.

gRPC n'est pas non plus nouveau, il a été open sourcé par Google en 2015 et supportait déjà Java, C, C++, Node.js , Python et Ruby. Autant dire que Microsoft était en retard, mais le retard est rattrapé. Dans un article précédent nous avions vu comment créer un service gRPC avec ASP.NET Core 3. Dans cet article nous allons voir comment fonctionne plus en détail gRPC en détaillant ses forces et ses faiblesses.

Comment fonctionne gRPC ?

HTTP/2

Le fonctionnement de gRPC tel qu'il a été conçu par Google m'a particulièrement déboussolé au départ. Ensuite, j'ai compris qu'il a été conçu pour offrir le plus de performance possible en tirant partie des avantages du protocole HTTP/2 (ne fonctionne qu'avec HTTP/2). Rappelons les principaux avantages de ce dernier :

- Permet d'avoir une latence réduite,
- Permet le multiplexage (plusieurs requêtes sur la même connexion).

La sérialisation binaire avec gRPC est un avantage indéniable par rapport aux WebAPI, un payload identique sérialisé en binaire comparé à du JSON offre un gain de 25% en termes de performance (sérialisation / désérialisation plus rapide et payload plus léger offrant donc un transport plus rapide).

1

```
syntax = "proto3";
package demo;

service MyOwnService {
  rpc WhoIs(EmptyRequest) returns (WhoIsReply) {}
  rpc IntroduceYourself (IntroduceYourselfRequest) returns (IntroduceYourselfReply) {}
}
```

2

```
syntax = "proto3";

option csharp_namespace = "DemoGrpc.Protobufs";

service CountryService {
  rpc GetAll(EmptyRequest) returns (CountriesReply) {}
  rpc GetById (CountrySearchRequest) returns (CountryReply) {}
  rpc Create (CountryCreateRequest) returns (CountryReply) {}
  rpc Update (CountryRequest) returns (CountryReply) {}
  rpc Delete (CountrySearchRequest) returns (EmptyReply) {}
}

message EmptyRequest {
}

message EmptyReply {
}

message CountrySearchRequest {
    int32 CountryId = 1;
}

message CountryCreateRequest {
    string Name = 1;
    string Description = 2;
}

message CountryRequest {
    int32 Id = 1;
    string Name = 2;
    string Description = 3;
}

message CountryReply {
    int32 Id = 1;
    string Name = 2;
    string Description = 3;
}

message CountriesReply {
    repeated CountryReply Countries = 1;
}
```

Url, verbes, statuts HTTP

Un autre point très important à comprendre est la spécification stricte sur la gestion des urls, des verbes et des statuts http :

- gRPC gère lui-même l'écriture des urls :

https:// {yourdomain}/{packagename}{servicename}/{methodname}

Le package name étant le nom du package défini dans votre fichier proto et servant au versionnage des services. 1

Ce qui donnerait pour une url locale :

https://localhost:5002/demo.MyOwnService/WhoIs

- gRPC utilise toujours le verbe POST.
- Quand gRPC prend en charge une requête http la réponse sera toujours OK 200. En cas d'échec d'authentification sur le serveur, la réponse sera 401 UnAuthorized, ou bien des erreurs 5xx si le serveur présente des difficultés. Si l'application crashe, la réponse sera 200 OK.

On se pose la question comment faire du CRUD dans gRPC avec les bonnes pratiques. Eh bien vous serez surpris, il n'y en a pas (de bonnes pratiques), pourquoi ? Pour deux principales raisons :

- L'héritage n'existe pas dans la version 3 de la syntaxe protobuf (proto 3). Vous allez donc être obligé de réécrire du code souvent similaire, je pense notamment aux « message » autrement dit vos contrats de données. S'ils se ressemblent vous ne pourrez pas créer de « base message » pouvant être réutilisés dans des messages qui en héritent avec des propriétés similaires.

Exemple de redondance : 2

- Deuxième raison, c'est qu'il n'existe pas de validation native des données en entrées. Vous devez faire vos validations « à la main ». Toutefois, Microsoft étudie la possibilité d'ajouter une validation native à son implémentation gRPC.

Exemple :

```
public override async Task<CountryReply> Create(CountryCreateRequest request, ServerCall
Context context)
    {
      if (string.IsNullOrEmpty(request.Name))
      {
        throw new RpcException(new Status(StatusCode.InvalidArgument, "Name is missing"),
"Validation has failed");
      }

      var createCountry = _mapper.Map<Country>(request);
      var country = await _countryService.AddAsync(createCountry);
      return _mapper.Map<CountryReply>(country);
    }
```

Pour palier ce manquement temporaire, j'ai créé une librairie basée sur **FluentValidation** qui permet de faire de la validation automatique. Le principe de fonctionnement est quasiment identique à celui qui serait utilisé sur une WebApi.
Voici le lien du projet contenant un tutoriel et le lien vers les packages Nuget : https://github.com/AnthonyGiretti/grpc-aspnetcore-validator

Gestion des erreurs

gRPC s'appuie sur les «Trailers » qui sont une sorte de headers. Ils sont cependant différents de ces derniers car ils permettent de transporter des métadonnées (ils sont envoyées à la fin d'une réponse HTTP) ainsi que le « grpc-status » que l'on retrouve dans les headers de la réponse avec une étrange spécificité : quand la réponse contient un statut gRPC en erreur, ce statut se trouve dans les headers, si la réponse renvoie un statut gRPC à 0 c'est-à-dire « OK » celui-ci se trouve dans les trailers. Enfin avec gRPC-web que nous verrons dans un prochain article, le grpc-status (quand il est OK) se trouve dans le body de la réponse.
Vous trouverez dans ce lien la réponse originale à ces questions que j'ai posées auprès de James Newton-King : https://github.com/grpc/grpc-dotnet/issues/761
Voici la liste des différents statuts que gRPC permet de gérer :

Code	Numéro
OK	0
CANCELLED	1
UNKNOWN	2
INVALID_ARGUMENT	3
DEADLINE_EXCEEDED	4
NOT_FOUND	5
ALREADY_EXISTS	6
PERMISSION_DENIED	7
RESOURCE_EXHAUSTED	8
FAILED_PRECONDITION	9
ABORTED	10
OUT_OF_RANGE	11
UNIMPLEMENTED	12
INTERNAL	13
UNAVAILABLE	14
DATA_LOSS	15
UNAUTHENTICATED	16

Comme vous pouvez le voir, il existe également un statut gRPC pour gérer l'échec d'authentification, même si l'authentification peut être gérée en amont avant que la requête soit prise en charge par gRPC. Ce dernier offre la possibilité de gérer l'authentification une fois la requête prise en charge.
Enfin, pour en savoir plus sur les spécifications de gRPC, vous pouvez vous rendre à cette adresse ici : https://github.com/grpc/grpc/blob/master/doc/PROTOCOL-HTTP2.md
Grâce à ces précieuses clarifications j'ai pu créer des règles de « retry » avec la librairie Polly qui permet de rendre plus résilients les appels http avec un HttpClient. Exemple :

```
public static class StatusManager
{
  public static StatusCode GetStatusCode(HttpResponseMessage response)
  {
    var headers = response.Headers;
    if (!headers.Contains("grpc-status"))
      return StatusCode.OK;

    return (StatusCode)int.Parse(headers.GetValues("grpc-status").First());
  }
}


class Program
{
  static async Task Main(string[] args)
  {
    // DI
    var services = new ServiceCollection();

    Func<HttpRequestMessage, IAsyncPolicy<HttpResponseMessage>> retryFunc = (request) =>
    {
      return Policy.HandleResult<HttpResponseMessage>(r => {
        var grpcStatus = StatusManager.GetStatusCode(r);
        return r.StatusCode != HttpStatusCode.OK && grpcStatus != StatusCode.OK && grpc
Status != StatusCode.InvalidArgument;
      })
      .WaitAndRetryAsync(3, (input) => TimeSpan.FromSeconds(3 + input), (result, timeSpan,
retryCount, context) =>
                {
                  var grpcStatus = StatusManager.GetStatusCode(result.Result);
                  Console.WriteLine($"Request failed with {grpcStatus}. Retry");
                });
    };

    services.AddGrpcClient<CountryServiceClient>(o =>
    {
      o.Address = new Uri("https://localhost:5001");
    }).AddPolicyHandler(retryFunc);

    var provider = services.BuildServiceProvider();

    var client = provider.GetRequiredService<CountryServiceClient>();
```

```
        var countries = (await client.GetAllAsync(new EmptyRequest())).Countries;

    }
}
```

Gestion globale des erreurs coté serveur

Autre petit désappointement ici, l'absence totale de gestion globale des erreurs telle qu'un filtre d'action dans ASP.NET MVC ou WebApi, ou encore telle qu'un middleware ASP.NET Core dans lequel on peut contrôler la réponse renvoyée au client. C'est dommage. Toutefois, il existe une alternative toute simple qui est, malheureusement, le copier-coller !

Exemple : vous devrez pour chacun de vos services, l'encapsuler dans un block try / catch et renvoyer une exception de type « RpcException » avec le statut grpc adéquat :

```
public class CountryGrpcService : CountryService.CountryServiceBase
{
    private readonly ICountryService _countryService;
    private readonly IMapper _mapper;

    public CountryGrpcService(ICountryService countryService, IMapper mapper)
    {
        _countryService = countryService;
        _mapper = mapper;
    }

    public override async Task<CountriesReply> GetAll(EmptyRequest request, ServerCall
Context context)
    {
        try
        {
            var countries = await _countryService.GetAsync();
            return _mapper.Map<CountriesReply>(countries);
        }
        catch (Exception e)
        {
            throw new RpcException(Status.DefaultCancelled, e.Message);
        }
    }

    public override async Task<CountryReply> GetById(CountrySearchRequest request, Server
CallContext context)
    {
        try
        {
            var country = await _countryService.GetByIdAsync(request.CountryId);
            return _mapper.Map<CountryReply>(country);
        }
        catch (Exception e)
        {
            throw new RpcException(Status.DefaultCancelled, e.Message);
        }
    }

    public override async Task<CountryReply> Create(CountryCreateRequest request, Server
CallContext context)
    {
        try
        {
            var createCountry = _mapper.Map<Country>(request);
            var country = await _countryService.AddAsync(createCountry);
            return _mapper.Map<CountryReply>(country);
        }
        catch (Exception e)
        {
            throw new RpcException(Status.DefaultCancelled, e.Message);
        }
    }

    public override async Task<CountryReply> Update(CountryRequest request, ServerCall
Context context)
    {
        try
        {
            var updateCountry = _mapper.Map<Country>(request);
            var country = await _countryService.UpdateAsync(updateCountry);
            return _mapper.Map<CountryReply>(country);
        }
        catch (Exception e)
        {
            throw new RpcException(Status.DefaultCancelled, e.Message);
        }
    }

    public override async Task<EmptyReply> Delete(CountrySearchRequest request, ServerCall
Context context)
    {
        try
        {
            await _countryService.DeleteAsync(request.CountryId);
            return new EmptyReply();
        }
        catch (Exception e)
        {
            throw new RpcException(Status.DefaultCancelled, e.Message);
        }
    }
}
```

Comme je l'ai dit auparavant, le serveur va toujours renvoyer un statut OK 200 même en cas d'erreur. Ceci va compliquer le monitoring d'application, tout particulièrement avec les outils actuels tels qu'Application Insights. Avec ce dernier vous observerez toujours les logs suivants, ce qui n'est donc pas d'une grande aide : 3

3

Rassurez-vous, Microsoft travaille actuellement sur un outil de monitoring dédié à gRPC.

Audit et interception des requêtes

Il existe une fonctionnalité intéressante dans gRPC, la possibilité d'intercepter facilement une requête et de faire de l'audit. Pour ce faire gRPC introduit les intercepteurs.
Les intercepteurs sont des services qui fonctionnent un peu comme filtres d'actions dans une WebApi.
Ils dérivent d'une classe de base nommée « Interceptor », ils supportent l'injection de dépendance et vous devez surcharger le handler adéquat, ici « UnaryServerHandler » puisque, lors de la réalisation de cet exemple, j'ai utilisé un service unaire.
Exemple d'implémentation d'un intercepteur faisant du logging :

```
public class LoggerInterceptor : Interceptor
 {
    private readonly ILogger<LoggerInterceptor> _logger;

    public LoggerInterceptor(ILogger<LoggerInterceptor> logger)
    {
       _logger = logger;
    }

    public override Task<TResponse> UnaryServerHandler<TRequest, TResponse>(
       TRequest request,
       ServerCallContext context,
       UnaryServerMethod<TRequest, TResponse> continuation)
    {
       LogCall(context);
       return continuation(request, context);
    }

    private void LogCall(ServerCallContext context)
    {
       var httpContext = context.GetHttpContext();
       _logger.LogDebug($"Starting call. Request: {httpContext.Request.Path}");
    }
 }
```

Exemple d'enregistrement d'un intercepteur :

```
public void ConfigureServices(IServiceCollection services)
{
  services.AddGrpc(options =>
  {
     options.Interceptors.Add<LoggerInterceptor>();
  });

  // Services à enregistrer
}
```

A noter que les intercepteurs ne permettent pas de gérer globalement une erreur dans l'application. Cependant, ils peuvent être utilisés pour contrôler la réponse au client si vous souhaitez mettre fin à une requête. Par exemple, une erreur de validation d'un processus que vous aurez mis en place. Le validateur que j'ai mis au point se basant sur FluentValidation interrompt la requête courante et envoie un statut grpc au client « InvalidArgument » :

```
internal class ValidationInterceptor : Interceptor
 {
    private readonly IValidatorLocator _locator;
    private readonly IValidatorErrorMessageHandler _handler;

    public ValidationInterceptor(IValidatorLocator locator, IValidatorErrorMessageHandler
handler)
    {
       _locator = locator;
       _handler = handler;
    }

    public override async Task<TResponse> UnaryServerHandler<TRequest, TResponse>(
       TRequest request,
       ServerCallContext context,
       UnaryServerMethod<TRequest, TResponse> continuation)
    {
       if (_locator.TryGetValidator<TRequest>(out var validator))
       {
          var results = await validator.ValidateAsync(request);

          if (!results.IsValid && results.Errors.Any())
          {
             var message = await _handler.HandleAsync(results.Errors);
             var validationMetadata = results.Errors.ToValidationMetadata();
             throw new RpcException(new Status(StatusCode.InvalidArgument, message),
validationMetadata);
          }
       }
       return await continuation(request, context);
    }
 }
```

gRPC et Azure App Service & IIS

ASP.NET Core gRPC n'est actuellement pas pris en charge sur Azure App Service ou IIS. L'implémentation HTTP / 2 de Http.Sys ne prend pas en charge les en-têtes de fin de réponse HTTP sur lesquels gRPC s'appuie.

Conclusion

Nous avons vu les forces et les faiblesses de gRPC dans ASP.NET Core 3.1. Bien que gRPC ne soit pas encore complètement mature dans l'écosystème Microsoft, il offre d'étonnantes possibilités. Notamment d'améliorer la performance des appels machine à machine tels que deux serveurs, ou bien des objets entre eux dans le monde de l'Internet des objets.
Il existe une déclinaison compatible avec les navigateurs de gRPC nommée gRPC-web qui est supportée par Azure App Service. Nous verrons dans un prochain article comment implémenter un service gRPC compatible avec un navigateur web en utilisant sa déclinaison gRPC-web et un SPA type Angular, le tout hébergé dans un App Service Azure.

Nicolas Bouteillier
Artisan développeur
Agile - server admin - fullstack
(php/symfony/jquery/bootstrap)
http://www.kaizendo.fr - https://fr.linkedin.com/in/kaizendo
https://twitter.com/NicolasKaizen

Découvrons la puissance de Vagrant !

Le jour où j'ai eu l'opportunité d'accueillir mes premiers stagiaires, je me suis demandé comment faciliter leur intégration dans l'entreprise. Comment faire en sorte que la mise en place soit rapide, sans heurts, qu'ils puissent se mettre à développer sans passer des heures à courir après telle configuration, tel module ou se casser les dents sur une incompatibilité et des soucis de versions ?

Du point de vue de l'entreprise, plus vite un développeur se met à développer et plus vite il crée de la valeur ! Encore faut-il travailler dans de bonnes conditions.
Je voulais aussi éviter les soucis de « je comprends pas, ça marche sur mon pc... », très frustrant, je pense qu'on est tous passés par là à un moment ou un autre. Côté stack, mon projet est basé sur le framework Symfony. C'est donc majoritairement du PHP.
Pour mes serveurs, j'utilise Debian. Mes environnements de pré-production et production sont identiques (même OS, même stack, mêmes versions). Mon environnement de dév lui ne l'était pas, ma machine de travail est sous Kubuntu, ce qui m'a posé problème à plusieurs reprises, avec notamment les mises à jour de php qui diffèrent entre Debian et Ubuntu. Je me retrouvais avec une version plus récente en local.
Une des solutions pour que tout le monde ait le même environnent est d'acheter une machine par personne, toutes identiques, installer la même distribution, faire les même configurations, verrouiller les paquets (ou faire les mises à jour en même temps) et imposer l'ensemble à tout le monde. Bref, avoir un parc homogène au niveau matériel.
À part le coût important évident du matériel (même si le comptable ou le chef d'entreprise, peut mettre en avant les amortissements de matériel, la récupération de la TVA), je n'aime pas l'idée d'imposer les outils de travail à mes collaborateurs, mon but est que le travail soit bien fait, et lorsqu'on passe beaucoup de temps sur un outil, c'est toujours mieux d'être à l'aise.
De plus, en cas de panne, la mise en route d'un autre poste de dév peut être longue, acheter, installer, configurer… On se retrouve avec un parc hétérogène, à moins de prévoir un poste de rechange, mais le parc devient vite obsolète…

La virtualisation est l'autre solution.

Revenons à mes stagiaires qui allaient venir avec leurs pc. Comment faire pour qu'ils soient prêts à bosser, sans encombre, mais aussi pour leur apporter quelque chose d'un peu sympa niveau techno ? Un de mes objectifs dans le cas des stagiaires étant aussi de leur apporter du contenu, des bonnes pratiques, des technos intéressantes.
C'est là que Vagrant entre en jeu pour la VM et Ansible pour le provisionnement.

1) quoikes ?

Pour faire simple Vagrant va automatiser la mise en place de la machine virtuelle nécessaire au cloisonnement de l'environnement de dév en s'appuyant sur un seul fichier de configuration.
Très souple, il se charge de faire le nécessaire en arrière-plan, pour ce qui est notamment du montage des répertoires partagés, le nombres de cœurs, la ram, l'accès ssh, la configuration du fichier hosts etc.
Il va aussi se charger de gérer les images des OS, les télécharger, les mettre à jour.
Avec la commande Vagrant on peut gérer tout ce qui touche aux VM, par exemple les connexions en ssh, powershell, rdp, les snapshots, les plugins ...

```
vagrant --help
Usage: vagrant [options] <command> [<args>]

    -v, --version            Print the version and exit.
    -h, --help               Print this help.
```

Common commands:

box	manages boxes: installation, removal, etc.
cloud	manages everything related to Vagrant Cloud
destroy	stops and deletes all traces of the vagrant machine
global-status	outputs status Vagrant environments for this user
halt	tops the vagrant machine
help	shows the help for a subcommand
hostmanager	plugin: vagrant-hostmanager: manages the /etc/hosts file within a multi-machine environment
init	initializes a new Vagrant environment by creating a Vagrantfile
login	
package	packages a running vagrant environment into a box
plugin	manages plugins: install, uninstall, update, etc.
port	displays information about guest port mappings
powershell	connects to machine via powershell remoting
provision	provisions the vagrant machine
push	deploys code in this environment to a configured destination
rdp	connects to machine via RDP
reload	restarts vagrant machine, loads new Vagrantfile configuration
resume	resume a suspended vagrant machine
snapshot	manages snapshots: saving, restoring, etc.
ssh	connects to machine via SSH
ssh-config	outputs OpenSSH valid configuration to connect to the machine
status	outputs status of the vagrant machine
suspend	suspends the machine
up	starts and provisions the vagrant environment
upload	upload to machine via communicator

```
validate       validates the Vagrantfile
version        prints current and latest Vagrant version
winrm          executes commands on a machine via WinRM
winrm-config       outputs WinRM configuration to connect to the machine

For help on any individual command run `vagrant COMMAND -h`

Additional subcommands are available, but are either more advanced
or not commonly used. To see all subcommands, run the command
`vagrant list-commands`.
```

Cet article est une introduction pour vous faire découvrir le combo vagrant + ansible, vous pouvez faire infiniment plus avec. Comme toujours : RTFM

2) mise en place

Il faut évidemment avoir Vagrant installé sur la machine hôte, dans mon cas, une commande :

« sudo apt install vagrant » fait l'affaire.

Voici un exemple pour lancer une Debian X64 avec virtualbox : l'exemple ci-après vous montre 3 choses intéressantes (code ci-dessous). Vous y verrez trois choses intéressantes, d'abord j'ai partagé les fichiers avec nfs(*), c'est bien plus rapide, je vous le conseille. On peut voir un plugin Vagrant nommé «hostmanager» qui permets d'avoir des noms d'hôtes localement, pratique pour avoir par exemple un sous domaine pour Mailhog ** (logiciel qui permet de simuler l'envoi et capter tous les emails du projets): mail.monbeausite.localdev .
Pour finir, j'utilise Ansible pour provisionner la VM, on le voit en fin de fichier.
On y revient un peu plus tard dans l'article.
Vagrant peut utiliser plusieurs systèmes de virtualisation, les «providers», je ne suis pas forcément fan de virtualbox, mais lorsque je l'ai mis en place pour la première fois, c'était l'option la plus simple au niveau multiplateforme. Si je devais reconfigurer aujourd'hui, je testerai avec Docker.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
 def fail_with_message(msg)
  fail Vagrant::Errors::VagrantError.new, msg
 end
 # The most common configuration options are documented and commented below.
 # For a complete reference, please see the online documentation at
 # https://docs.vagrantup.com.

 # Every Vagrant development environment requires a box. You can search for
 # boxes at https://vagrantcloud.com/search.
  config.vm.box = "debian/contrib-stretch64"

 # Disable automatic box update checking. If you disable this, then
 # boxes will only be checked for updates when the user runs
 # `vagrant box outdated`. This is not recommended.
 # config.vm.box_check_update = false

 # Create a forwarded port mapping which allows access to a specific port
 # within the machine from a port on the host machine. In the example below,
 # accessing "localhost:8080" will access port 80 on the guest machine.
 # config.vm.network "forwarded_port", guest: 80, host: 8080

 # Create a private network, which allows host-only access to the machine
 # using a specific IP.
 config.vm.network "private_network", ip: "192.168.50.10"
 config.vm.hostname = "monbeausite.localdev"

 if Vagrant.has_plugin?('vagrant-hostmanager')
  config.hostmanager.enabled = true
  config.hostmanager.manage_host = true
  config.hostmanager.aliases = %w(monbeausite.localdev.com tools.monbeausite.localdev
mail.monbeausite.localdev db.monbeausite.localdev)

 else
  fail_with_message "vagrant-hostmanager missing, please install the plugin with this
command:\nvagrant plugin install vagrant-hostmanager\n\n"
 end

 # Create a public network, which generally matched to bridged network.
 # Bridged networks make the machine appear as another physical device on
 # your network.
 # config.vm.network "public_network"+

 # Share an additional folder to the guest VM. The first argument is
 # the path on the host to the actual folder. The second argument is
 # the path on the guest to mount the folder. And the optional third
 # argument is a set of non-required options.
 # config.vm.synced_folder "../data", "/vagrant_data"
 config.vm.synced_folder "./", "/project", type: 'nfs', map_uid: 0, map_gid: 0

 # Provider-specific configuration so you can fine-tune various
 # backing providers for Vagrant. These expose provider-specific options.
 # Example for VirtualBox:
 #
 # config.vm.provider "virtualbox" do |vb|
 #  # Display the VirtualBox GUI when booting the machine
 #  vb.gui = true
 #
 #  # Customize the amount of memory on the VM:
 #  vb.memory = "1024"
 # end
 config.vm.provider "virtualbox" do |v|
  #v.customize ["modifyvm", :id, "--cableconnected1", "on"]
```

(*) Pensez à vérifier la compatibilité avec NFS, notamment pour les utilisateurs windows. Un partage Samba peut être utilisé à la place.
(**) https://github.com/mailhog/MailHog

```
  v.name = "monbeausite_dev_env"
  v.memory = "8192"
  v.cpus = 6
  #vb.customize ["modifyhd", "disk id", "--resize", "size in megabytes"]
 end
 #
 # View the documentation for the provider you are using for more
 # information on available options.

 # Enable provisioning with a shell script. Additional provisioners such as
 # Puppet, Chef, Ansible, Salt, and Docker are also available. Please see the
 # documentation for more information about their specific syntax and use.
 # config.vm.provision "shell", inline: <<-SHELL
 #   apt-get update
 #   apt-get install -y apache2
 # SHELL
  config.vm.provision "ansible_local" do |ansible|
  ansible.galaxy_role_file = '/project/ansible/requirements.yml'
  ansible.playbook = "/project/ansible/site.yml"
  ansible.verbose = "v"
 end
end
```

3) Push sur le repository

L'idée est de partager ça avec toute l'équipe de dév, pour que n'importe quel nouveau dév n'ait plus qu'à cloner le repository, et lancer la VM avec la commande : vagrant up

Dans mon projet, j'ai donc mon fichier Vagrantfile à la racine.

4) Ansible

Comme je l'évoquais plus haut, j'utilise Ansible pour provisionner ma VM.

Si j'ai Vagrant qui lance une VM, c'est bien, mais s'il n'y a que l'OS de base, c'est moins bien…

J'ai besoin de faire des choses sur ma VM, dans mon cas, installer ma stack (Apache, Mariadb, Php), créer mes Vhosts, créer mes bases de données, installer mes outils (mailhog, git …).

Ansible permet d'automatiser beaucoup de choses, c'est un outil très puissant, open source, porté par Red Hat.

Il a entre autres comme énorme avantage de ne nécessiter aucune installation côté client, tout se fait depuis l'hôte, via ssh.

Dans ce cas précis, on utilise ansible_local qui sera installé dans la VM, toujours dans l'optique d'avoir le moins de prérequis possible sur l'hôte.

La configuration se fait via des fichiers YAML, c'est clair, plutôt simple et la documentation est très bien faite.

Il existe un site, https://galaxy.ansible.com/ qui répertorie des « Roles » qui sont des ensembles d'actions d'installation et de configuration.

Ils peuvent être facilement intégrés à vos «Playbook» qui sont eux comme un ensemble de rôles et de tâches. Il n'est pas obligatoire d'en utiliser, mais ils permettent d'encapsuler, garder, partager des suites de configuration et d'installation, et dans mon cas, ça me permet d'avoir un «Playbook» qui contient tous les rôles nécessaires pour provisionner ma VM de développement. Ça prendrait trop de place de mettre ici tout mon playbook, et ça n'aurait que peu de sens.

Vous pouvez pour commencer simplement chercher une stack qui vous convient sur Ansible Galaxy et l'utiliser.

En attendant, pour que vous ayez une idée de ce que c'est, voici une simple « task » qui installe des éléments communs du système :

```
---
# tasks file for common
- name: Install basic packages
  apt:
    name: [
      'rkhunter',
      'htop',
      'nano',
      'ntp',
      'ntpdate',
      'git',
      'unzip',
      'apt-transport-https',
      'python-pip',
      'letsencrypt',
      'logrotate',
      'curl',
      'lftp',
      'rsyslog',
      'util-linux-locales',
      'locales',
      'debconf-i18n',
      'fail2ban'
    ]
```

Vous pouvez voir la simplicité, il y a le nom du module : Apt. Ensuite ça boucle sur les « items » et pour chacun d'eux, ça les installe.

Ansible peut être utilisé dans beaucoup de scénarios, je m'en sers par exemple pour déployer mon code en production.

Conclusion

Vous avez découvert les bases de Vagrant, avec un fichier de configuration assez basique mais qui utilise des mécanismes très pratiques.

Ce n'est pas un tutoriel pas à pas, mais ça vous permet d'avoir une idée de ce que vous pouvez faire,

Dans un premier temps faites une simple VM, amusez-vous avec, vous pourrez ensuite découvrir et expérimenter avec Ansible.

Une fois mise en place la VM avec Vagrant et la provision avec Ansible, vous pourrez intégrer un nouveau membre dans votre équipe de développement en peu de temps.

Il y a encore beaucoup à dire sur le sujet, n'hésitez pas à nous écrire à redaction@programmez.com si vous voulez un article plus précis sur Vagrant ou Ansible !

Enjoy !

Aurélie Vache
Cloud Dev(Ops) chez Continental
Duchess France & DevFest Pitchouns Leader
Toulouse Data Science core-team & Marraine Elles bougent
@aurelievache

Tester un service gRPC en Go avec le Table Driven Tests

Partie 2

Tout le monde sait qu'une couverture de code à 100% n'existe pas et n'apporte aucune valeur ajoutée. En fait, au quotidien, ce que nous voulons vraiment c'est tester notre business logic, l'intelligence de notre application. Dans cet article nous allons partir d'une petite application CLI codée en Go qui ne possède pas encore de tests unitaires, puis nous allons réaliser quelques tests unitaires de services gRPC. Suite et fin.

Passons à la pratique

Concrètement pour rajouter notre cas de test on va :

- créer un fichier *service_test.go* à côté de son homologue service.go que nous voulons tester
- créer une fonction *TestSayHello(t *testing.T)*
- définir nos cas de tests
- boucler dans tous les cas de test, appeler notre service et tester l'erreur si attendue et le résultat si compliant avec celui défini

Ce qui donne ceci :

```
package greeter_test

import (
  "context"
  "testing"

  . "github.com/onsi/gomega"

  "github.com/scraly/hello-world/internal/services/pkg/v1/greeter"
  helloworldv1 "github.com/scraly/hello-world/pkg/protocol/helloworld/v1"
)

func TestSayHello(t *testing.T) {
  testCases := []struct {
    name      string
    req       *helloworldv1.HelloRequest
    message   string
    expectedErr bool
  }{
    {
      name:     "req ok",
      req:      &helloworldv1.HelloRequest{Name: "me"},
      message:  "hello me",
      expectedErr: false,
    },
    {
      name:     "req with empty name",
      req:      &helloworldv1.HelloRequest{},
      expectedErr: true,
    },
    {
      name:     "nil request",
      req:      nil,
      expectedErr: true,
    },
  }

  for _, tc := range testCases {
    testCase := tc
    t.Run(testCase.name, func(t *testing.T) {
      t.Parallel()
      g := NewGomegaWithT(t)

      ctx := context.Background()

      // call
      greeterSvc := greeter.New()
      response, err := greeterSvc.SayHello(ctx, testCase.req)

      t.Log("Got : ", response)

      // assert results expectations
      if testCase.expectedErr {
        g.Expect(response).ToNot(BeNil(), "Result should be nil")
        g.Expect(err).ToNot(BeNil(), "Result should be nil")
      } else {
        g.Expect(response.Message).To(Equal(testCase.message))
      }
    })
  }
}
```

Aurélie, il est bien joli ton code, mais pourquoi tu crées une nouvelle variable **testCase** qui prend comme valeur tc alors que tu aurais pu utiliser **tc** directement ?

La réponse est dans cet article : https://gist.github.com/posener/92a55c4cd441fc5e5e85f27bca008721

En résumé, sans cette ligne il y a un bug (avec le t.Parallel() bien connu des Gophers - on utilise une closure qui est dans une go routine), et au lieu d'exécuter 3 cas de test : "req ok", "req with empty name" et "nil request", il y aurait bien trois exécutions de tests, mais toujours avec les valeurs du premier cas de test :-(.

Et gomega, c'est quoi ?

Gomega (https://onsi.github.io/gomega/) est une librairie en Go qui permet de faire des assertions. Dans notre exemple nous vérifions si ce que nous avons obtenu est null, non null ou bien égal à une valeur exacte, mais la bibliothèque gomega est bien plus riche que cela. Pour exécuter vos tests unitaires fraîchement créés, si vous utilisez VisualStudio Code, vous pouvez directement les exécuter dans votre IDE, c'est très pratique :

• Ouvrez le fichier *service_test.go*

Puis cliquez sur le lien "run package tests" :

```
run package tests | run file tests
1   package greeter_test
```

* Ouvrez le fichier service.go: 1

1

```
import (
    "context"

    apiv1 "github.com/scraly/hello-world/internal/services/pkg/v1"
    helloworldv1 "github.com/scraly/hello-world/pkg/protocol/helloworld/v1"
    "go.zenithar.org/pkg/log"
    "golang.org/x/xerrors"
)

type service struct {
}

// New services instance
func New() apiv1.Greeter {
    return &service{}
}

// ---------------------------------------------------------------------

func (s *service) SayHello(ctx context.Context, req *helloworldv1.HelloRequest) (*helloworldv1.HelloReply, error) {
    res := &helloworldv1.HelloReply{}

    // Check request
    if req == nil {
        log.Bg().Error("request must not be nil")
        return res, xerrors.Errorf("request must not be nil")
    }

    if req.Name == "" {
        log.Bg().Error("name but not be empty in the request")
        return res, xerrors.Errorf("name but not be empty in the request")
    }

    res.Message = "hello " + req.Name

    return res, nil
}
```

Le code surligné en vert est le code qui est couvert par les tests, super, pas une ligne de rouge !

Sinon, on peut exécuter tous les tests unitaires de notre projet en une ligne de commande, toujours grâce à notre merveilleux magefile :

```
$ go run mage.go go:test go:analyzecoverage
## Running unit tests
∅ cli/hello-world
∅ cli/hello-world/cmd
∅ cli/hello-world/config
∅ cli/hello-world/dispatchers/grpc
∅ internal/services/pkg/v1
√ internal/services/pkg/v1/greeter (1.089s)
∅ internal/version
∅ pkg/protocol/helloworld/v1

DONE 4 tests in 2.973s
## Analyze tests coverage
2019/08/12 14:10:56 Analyzing file test-results/cover.out
2019/08/12 14:10:56 Business Logic file
2019/08/12 14:10:56 Minimum coverage threshold percentage 90.000000 %
2019/08/12 14:10:56 Nb Statements: 10 Coverage percentage: 100.000000 %
```

Super, 100% de couverture de test sur notre business logic !

Conclusion

Si vous avez l'habitude de copier-coller lorsque vous écrivez vos cas de tests, je pense que vous devrez sérieusement jeter un œil aux Table Driven Tests, c'est vraiment une bonne pratique à suivre lors de l'écriture de tests unitaires et comme nous l'avons vu, écrire des tests unitaires qui couvrent notre code devient un vrai jeu d'enfant.

Duy Anh PHAM
Ingénieur logiciel Java/JEE chez Clevermind, je développe des applications Web d'entreprises depuis plus de 10 ans et je me passionne à tout ce qui permet d'obtenir une application maintenable et performante du code source jusqu'au déploiement en particulier la concurrence/programmation concurrente.

L'essentiel des threads pour comprendre les phénomènes de concurrence

Partie 3

Avec la généralisation des processeurs multicœurs dans les appareils tels que les ordinateurs, smartphones, tablettes et autres dispositifs avec système d'exploitation, développer une application multithreadée devient un enjeu majeur pour répondre aux besoins toujours croissants et nombreux des utilisateurs. Suite et fin de notre dossier sur les threads.

Les risques liés à l'utilisation des threads

L'emploi des threads permet de dynamiser tout ou partie d'un système pour une meilleur réactivité. Cependant une mauvaise synchronisation voire une mauvaise conception générale peut entraîner les phénomènes comme un deadlock, liveness, starvation ou ressources épuisées expliqués ci-dessous.

Deadlock

Le deadlock est une situation où au moins 2 threads sont bloqués (état BLOCKED) parce que chacun attend une ressource occupée par un autre. Lorsque cela se produit, l'application est figée et seul un arrêt du processus peut mettre fin au blocage.
Pour illustrer un deadlock, l'exemple de transferts de fonds entre comptes bancaires sera utilisé. Typiquement, nous voulons nous assurer que chaque transaction est correcte en synchronisant les comptes émetteur et destinataire.

```
public class DeadlockMain {

    private static final int COMPUTE_HARD = 1000;

    public static void main(String[] args) {

        final Runner runner = new Runner();

        Runnable r1 = () -> {
            try {
                runner.firstThread();
                System.out.println("Thread 1 is finished");
            } catch (InterruptedException e) {
            }
        };

        Runnable r2 = () -> {
            try {
                runner.secondThread();
                System.out.println("Thread 2 is finished");
            } catch (InterruptedException e) {
            }
        };

        Thread t1 = new Thread(r1, "Thread-1");
        Thread t2 = new Thread(r2, "Thread-2");

        t1.start();
        t2.start();

        try {
            t1.join();
            t2.join();
            runner.finished();
        } catch (InterruptedException e) {
        }
    }

    private static class Runner {

        private Account account1 = new Account();

        private Account account2 = new Account();

        private final Object lock1 = new Object();

        private final Object lock2 = new Object();

        public void firstThread() throws InterruptedException {

            Random random = new Random();

            for (int i = 0; i < COMPUTE_HARD; i++) {

                synchronized (lock1) {
                    synchronized (lock2) {
                        try {
                            // transfer from 1 to 2
                            int amount = random.nextInt(100);
                            Account.transfer(account1, account2, amount);
                            System.out.println(Thread.currentThread().getName()
                                    + " completes the transaction from account1 to account2 of
" + amount + " EUR");
                            TimeUnit.MILLISECONDS.sleep(100);
                        } finally {
                        }
                    }
```

```
            }
        }

    }

    public void secondThread() throws InterruptedException {

        Random random = new Random();

        for (int i = 0; i < COMPUTE_HARD; i++) {
            transferCredits(random);
        }
    }

    private void transferCredits(Random random) throws InterruptedException {
        // locks must be held and released in same order that in others threads
        // otherwise deadlock happens
        synchronized (lock2) {
            synchronized (lock1) {
                try {
                    // transfer from 2 to 1
                    int amount = random.nextInt(100);
                    Account.transfer(account2, account1, amount);
                    System.out.println(Thread.currentThread().getName()
                            + " completes the transaction from account2 to account1 of " +
amount + " EUR");
                    TimeUnit.MILLISECONDS.sleep(100);
                } finally {
                }
            }
        }
    }

    public void finished() {
        System.out.println("Account 1 balance : " + account1.getBalance());
        System.out.println("Account 2 balance : " + account2.getBalance());
        System.out.println("Total of accounts balances : " + (account1.getBalance() +
account2.getBalance()));
    }
}

private static class Account {
    private int balance = 10000;

    public void deposit(int amount) {
        balance += amount;
    }

    public void withDraw(int amount) {
        balance -= amount;
    }

    public int getBalance() {
        return balance;
    }

    public static void transfer(Account acc1, Account acc2, int amount) {
        acc1.withDraw(amount);
        acc2.deposit(amount);
    }
}

}
```

Le lancement du programme Java montre qu'il se bloque et ne se termine jamais.

```
Thread-1 completes the transaction from account1 to account2 of 43 EUR
Thread-2 completes the transaction from account2 to account1 of 55 EUR
Thread-1 completes the transaction from account1 to account2 of 91 EUR
Thread-2 completes the transaction from account2 to account1 of 27 EUR
Thread-1 completes the transaction from account1 to account2 of 75 EUR
Thread-2 completes the transaction from account2 to account1 of 4 EUR
Thread-1 completes the transaction from account1 to account2 of 14 EUR
Thread-2 completes the transaction from account2 to account1 of 78 EUR
Thread-1 completes the transaction from account1 to account2 of 6 EUR
Thread-2 completes the transaction from account2 to account1 of 75 EUR
Thread-1 completes the transaction from account1 to account2 of 92 EUR
...
```

Le profiler Java VisualVM fourni dans le JDK permet d'étudier l'état des threads et détecte les deadlocks comme illustrés ci-dessous. 1 2 3

Le résumé du thread dump montre clairement le deadlock avec les threads incriminés :

- Thread-1 a pris le verrou 0x00000000ec689668 et attend le verrou 0x00000000ec689678 obtenu par Thread-2. Ce dernier attend le verrou 0x00000000e c689668 pris par Thread-1 ;
- Les 2 threads sont à l'état BLOCKED.

La solution pour éviter le deadlock est de toujours prendre les verrous dans le même ordre. La méthode **transfertCredits()** doit prendre les verrous dans le même ordre que firstThread() c'est-à-dire 1 puis 2 afin d'éliminer le deadlock et que le programme se termine correctement avec le résumé final des comptes.

```
Thread-1 completes the transaction from account1 to account2 of 61 EUR
Thread-2 completes the transaction from account2 to account1 of 70 EUR
Thread-1 completes the transaction from account1 to account2 of 86 EUR
...
Thread-1 completes the transaction from account1 to account2 of 4 EUR
Thread 1 is finished
Thread-2 completes the transaction from account2 to account1 of 43 EUR
Thread-2 completes the transaction from account2 to account1 of 10 EUR
Thread-2 completes the transaction from account2 to account1 of 3 EUR
...
Thread-2 completes the transaction from account2 to account1 of 92 EUR
Thread-2 completes the transaction from account2 to account1 of 88 EUR
Thread 2 is finished
Account 1 balance : 9870
Account 2 balance : 10130
Total of accounts balances : 20000
```

Livelock

Le livelock est une situation où au moins un thread dans l'état RUNNABLE exécute des instructions puis tombe continuellement en erreur. Il finit par ne plus progresser dans la tâche à réaliser. Le

phénomène observé ressemble à une boucle infinie ce qui rend l'application inutilisable.

Quelques exemples illustrent ce phénomène :

1 Un programme, pour son bon fonctionnement, possède un module en arrière-plan qui vérifie l'espace occupé par le disque et procède automatiquement à un nettoyage de fichiers temporaires du système pour garantir un disque jamais plein. Il offre également la possibilité de se mettre à jour régulièrement en téléchargeant des modules principaux. Lorsque le disque dur est plein, le téléchargement s'interrompt et tombe en erreur. A ce moment le thread de nettoyage automatique détecte la situation et se lance. La procédure de téléchargement tente à nouveau de télécharger à nouveau, mais faute d'espace libre suffisant, il tombe encore en erreur. Le nettoyage s'opère, le téléchargement reprend sans succès et un cycle sans fin démarre.

```
public class LivelockDiskSpaceMain {

    public static void main(String[] args) {

        DiskManager diskManager = new DiskManager();

        Thread t1 = new Thread(() -> {

            while (true) {
                if (diskManager.isTooLow()) {
                    diskManager.reclaimSpace();
                    sleep(500);
                }
                sleep(100);
            }
        }, "diskCleanerTask-t1");

        Thread t2 = new Thread(() -> {

            String name = Thread.currentThread().getName();
            int applicationSize = 60;
            int packSize = 10;
            int maxPart = applicationSize / packSize;
            int progressBar = 0;
            int currentSize = 0;
            while (currentSize < applicationSize) {

                sleep(500);
                try {
                    int amountDone = diskManager.useSpace(packSize);
                    currentSize = currentSize + amountDone;
                    progressBar++;
                    System.out.println(name + " progressing " + progressBar + "/" + maxPart);
                } catch (RuntimeException e) {
                    System.out.println(name + " insufficient space. Download aborted. Retry
in a few moments");
                    progressBar = 0;
                    currentSize = 0;
                    sleep(500);
                }
            }
            System.out.println("End download");
        }, "downloader-t2");

        t1.start();
        t2.start();

        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    private static void sleep(long timeout) {
        try {
            TimeUnit.MILLISECONDS.sleep(timeout);
        } catch (InterruptedException e) {
        }
    }

    private static class DiskManager {
        private static final int MAX_THRESHOLD = 80;
        private static final int MAX_SPACE = 100;
        private static final int AMOUNT_TO_FREE = 4;
        private int usedSpace = 50;

        private synchronized void reclaimSpace() {
            usedSpace = usedSpace - AMOUNT_TO_FREE;
            System.out.println(Thread.currentThread().getName() + " space after clean : " + usedSpace);
        }

        private synchronized int useSpace(int amount) {
            if (isTooLow()) {
                System.out.println(Thread.currentThread().getName() + " no more available space");
                throw new RuntimeException("Space disk almost full");
            } else if (usedSpace + amount <= MAX_SPACE) {
                usedSpace = usedSpace + amount;
                System.out.println(Thread.currentThread().getName() + " remaining space : " + usedSpace);
            }
            return amount;
        }

        private boolean isTooLow() {
            return usedSpace >= MAX_THRESHOLD;
        }
    }
}
```

Un résultat possible d'exécution

```
downloader-t2 remaining space : 60
downloader-t2 progressing 1/6
downloader-t2 remaining space : 70
downloader-t2 progressing 2/6
downloader-t2 remaining space : 80
downloader-t2 progressing 3/6
diskCleanerTask-t1 space after clean : 76
```

```
downloader-t2 remaining space : 86
downloader-t2 progressing 4/6
diskCleanerTask-t1 space after clean : 82
downloader-t2 no more available space
downloader-t2 insufficient space. Download aborted. Retry in a few moments
diskCleanerTask-t1 space after clean : 78
downloader-t2 remaining space : 88
downloader-t2 progressing 1/6
diskCleanerTask-t1 space after clean : 84
downloader-t2 no more available space
downloader-t2 insufficient space. Download aborted. Retry in a few moments
diskCleanerTask-t1 space after clean : 80
diskCleanerTask-t1 space after clean : 76
downloader-t2 remaining space : 86
downloader-t2 progressing 1/6
diskCleanerTask-t1 space after clean : 82
downloader-t2 no more available space
downloader-t2 insufficient space. Download aborted. Retry in a few moments
diskCleanerTask-t1 space after clean : 78
...
```

Chronologie des évènements :

- Le thread downloader-t2 effectue un téléchargement par parquet de 10 octets jusqu'à la 6è itération normalement ;
- Le thread diskCleanerTask-t1 se déclenche lorsqu'il voit que le seuil de 80% est atteint et libère une petite quantité d'espace ;
- Le thread downloader-t2 annule tout ce qu'il a fait et reprend depuis le début ;
- Le thread diskCleanerTask-t1 se déclenche…

2 Une application intègre des données venant d'un partenaire extérieur via un système de messagerie transactionnelle. Si l'intégration d'un message se passe bien le message est alors effectivement retiré de la file et passe au message suivant. Dans le cas où le traitement d'intégration échoue, l'infrastructure de messagerie annule la transaction et remet le message en tête de file. A chaque fois que le traitement échoue pour ce type de message et que la politique de remise dans la file se fait de la même manière, alors le système ne peut plus intégrer et se « bloque ».

```
public class LivelockQueueWithConsumerMain {

    public static void main(String[] args) {

        Deque<Integer> queue = new LinkedList<>();

        int maxElements = 5;

        for (int i = 0; i < maxElements; i++) {
            queue.add(i);
        }

        Runnable consumer = () -> {
            Integer id = 0;
            int typeMessage = new Random().nextInt(maxElements);
            while ((id = queue.poll()) != null) {
                System.out.println("Processing value : " + id + " ...");
                if (id == typeMessage) {
                    System.out
                            .println("Simulate error for value : " + id + ". Rollback transaction and
putting back to the queue");
                    queue.addFirst(id);

                } else {
                    System.out.println("Processing value : " + id + " complete");
                }
            }
        };

        ExecutorService executor = Executors.newCachedThreadPool();
        executor.execute(consumer);

        executor.shutdown();
    }
}
```

Résultat possible d'exécution :

```
Processing value : 1 ...
Processing value : 1 complete
Processing value : 2 ...
Simulate error for value : 2. Rollback transaction and putting back to the queue
Processing value : 2 ...
Simulate error for value : 2. Rollback transaction and putting back to the queue
Processing value : 2 ...
Simulate error for value : 2. Rollback transaction and putting back to the queue
Processing value : 2 ...
Simulate error for value : 2. Rollback transaction and putting back to the queue
...
```

Le programme tourne indéfiniment. Nous simulons un traitement en erreur sur un certain type de message (en l'occurrence le type 2) et remettons le message en tête de file. Le programme continue de dépiler à nouveau le message posant problème. Une boucle infinie s'installe.

Starvation

Le phénomène de starvation se produit lorsqu'un thread ne peut accéder aux ressources pour continuer sa progression. Cela peut venir de :

- La gestion des priorités. Un thread de basse priorité aura peu de chance d'être choisi par le scheduler s'il est en concurrence avec d'autres threads de priorité plus élevée. Le comportement du scheduler dépend de l'OS. La gestion des priorités n'est pas recommandée car elle n'est pas fiable si on souhaite qu'un programme se comporte correctement dans le système.
- Un thread ayant attendu très longtemps un verrou ne parvient pas à l'obtenir lorsqu'un thread qu'il attend invoque notify() de la classe Object. Si plusieurs threads sont en attente de ce même verrou, en fait c'est un thread pris au hasard qui sera réveillé. La javadoc ne précise pas qu'il s'agit nécessairement de celui qui a attendu le plus longtemps. Il n'y a pas respect d'équité. Pour y remédier, il est préférable de remplacer la synchronisation wait/notify par l'utilisation de l'interface Lock et de l'implémentation ReentrantLock avec le constructeur en passant le paramètre fair à true :

```
Lock lock = new ReentrantLock(true);
```

Cette configuration s'accompagne d'une baisse en performance lors de contention avec beaucoup de threads, mais elle assure en contrepartie un respect de l'ordre d'attente des threads.

Performance et épuisement des ressources

Une application doit permettre de répondre aux requêtes dans un temps limité et acceptable sans être surchargée au point de ralentir et faire tomber le système.

Si chaque requête était traitée par un nouveau thread et que le débit de requêtes arrivant est plus important que la vitesse de traitement, alors le système créerait un nombre croissant de threads engendrant potentiellement un dépassement d'allocation mémoire (out of memory) au niveau applicatif et une dégradation des temps de réponse au niveau système.

Il convient d'employer un ou plusieurs pools de threads (threads réutilisables) bornés avec une file d'attente pour absorber les nombreuses requêtes avec une politique de rejet en cas d'excès. Typiquement une application Web va renvoyer une page invitant l'utilisateur à se reconnecter ultérieurement. Ou alors une application d'intégration peut persister la requête dans une autre file pour être traitée en asynchrone. C'est le niveau de service (SLA) entre fournisseur et client qui déterminera la politique à adopter.

Dans tous les cas, seule une surveillance de l'application avec des métriques permet de dimensionner au mieux le pool de threads qui doit permettre de gérer autant de requêtes que l'application et le matériel peuvent convenablement en supporter. En d'autres termes, nous ne voulons pas qu'une requête attende alors que nous avons encore la capacité de la gérer. Mais nous ne voulons pas non plus nous lancer dans le traitement d'une requête pour laquelle nous n'avons plus de capacité.

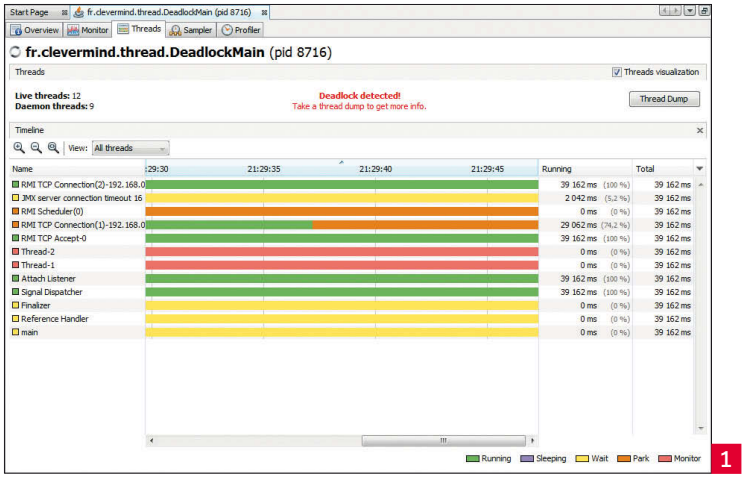
Conclusion

Cet article passe en revue les concepts de base pour comprendre le fonctionnement des threads ainsi que les moyens permettant d'écrire un code thread-safe grâce au mécanisme de synchronisation. Le développeur dispose d'une API bas niveau pour manipuler directement des threads et les faire communiquer.

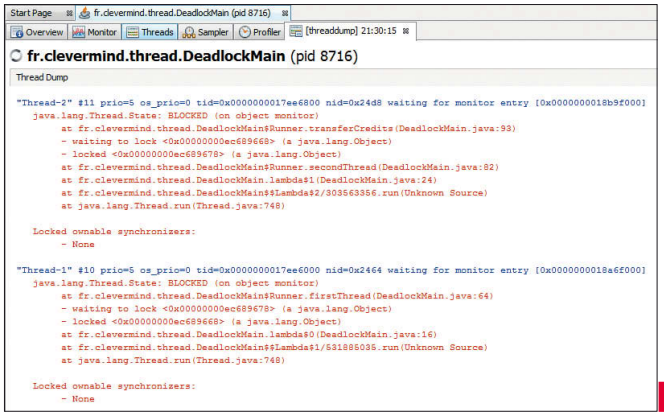
L'avènement de Java 5 constitue une avancée majeure en proposant une API de haut niveau dans le package java.util.concurrent facilitant le développement d'applications concurrentes avec le fameux framework Executor pour exécuter des tâches dans un pool de threads. Le développeur peut également manipuler des collections concurrentes et performantes, utiliser des outils de synchronisation de verrou très flexibles ou encore les variables atomiques.

Ecrire du code thread-safe est une tâche ardue qui demande rigueur et discipline. Cette complexité s'ajoute naturellement au code de production. Le développeur doit d'une part protéger les variables en obtenant les verrous sans tomber dans les deadlocks, d'autre part éliminer les phénomènes de starvation/livelock en gérant correctement les erreurs notamment la reprise sur erreur. Cela peut même nécessiter une reconception du système pour aboutir à un code sûr. Le défi est de se prémunir des problèmes de concurrence qui ne doivent pas être résolus par l'affirmation que cela passera la prochaine fois. Il s'agit de véritables dysfonctionnements dont l'analyse demande un redoublement d'effort pour identifier les causes puisque les bugs sont souvent non reproductibles sur un environnement de développement.

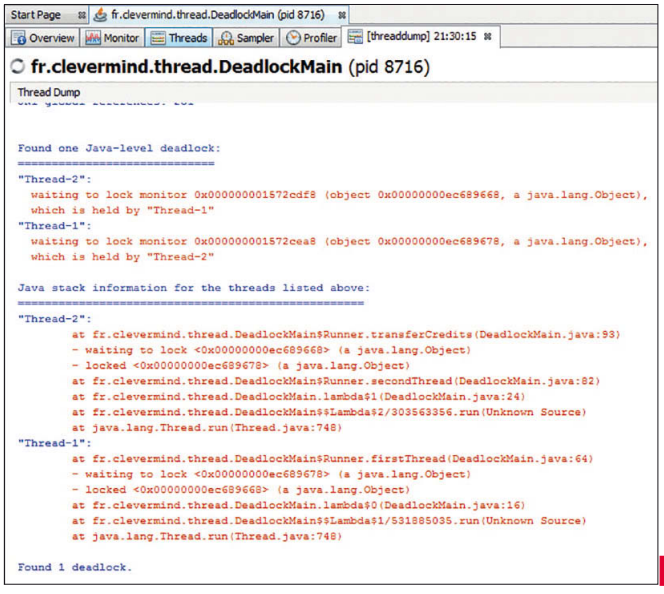
Enfin pour des questions de performances, le code synchronisé doit être le plus court possible tout en protégeant les variables partagées. La surveillance et mesure des performances par un outil dédié doit aider à optimiser le code et régler les différents paramètres liés aux pools de threads de l'application dans son environnement de production pour répondre à un maximum de requêtes sans surcharger l'application et le système.



1

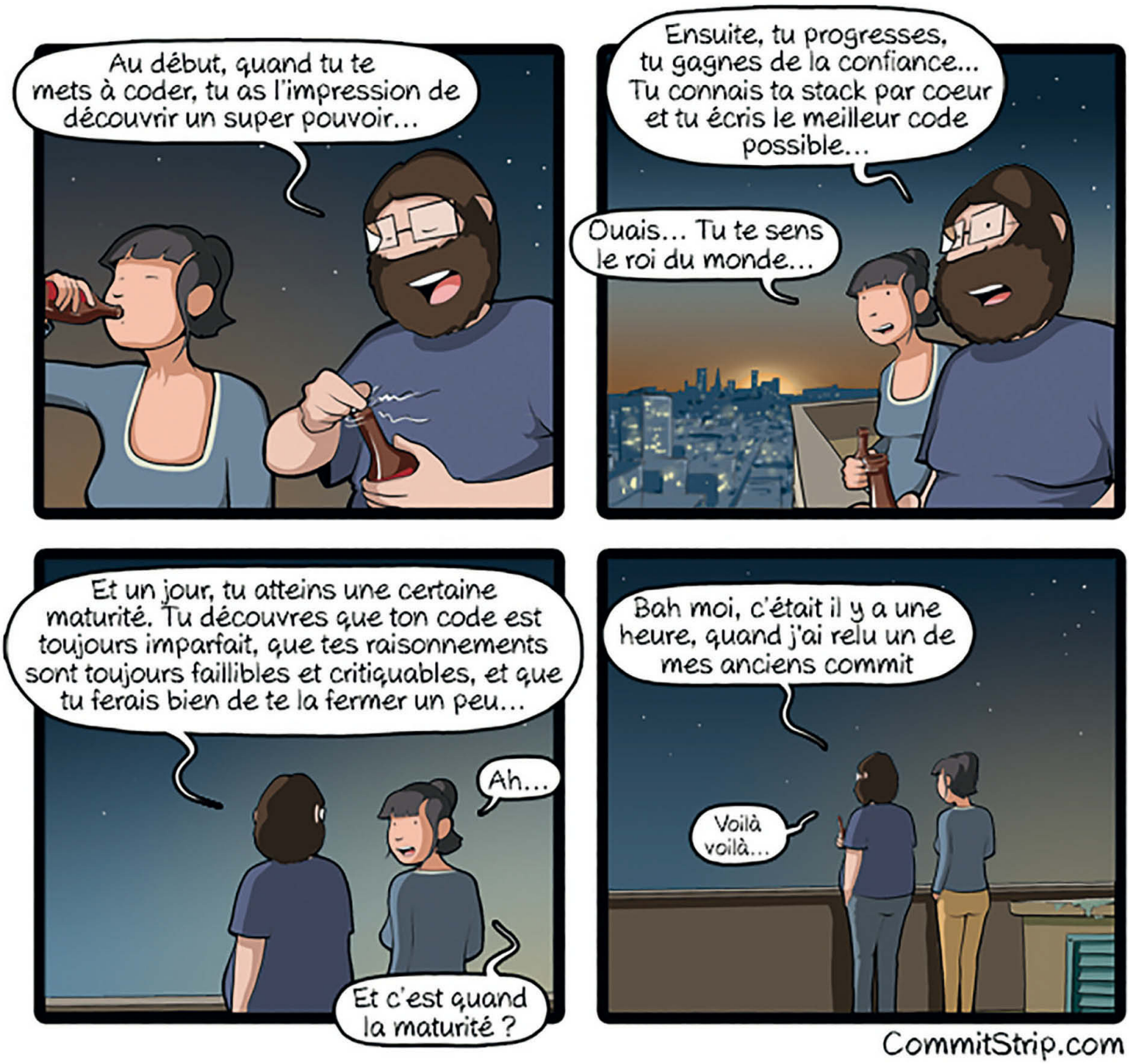


2



3

Le code de la maturité



Une publication Nefer-IT, 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08 - Directeur de la publication : François Tonic
Rédacteurs en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Ont collaboré à ce numéro : la rédaction de ZDnet
Nos experts techniques : D. Anh Pham, A. Vache, N. Bouteillier, A. Giretti, B. Dupin, B. Petetot, B. Sakote, J. Pastouret, L. Duport, O. Hess, J-M Torres, X. Vasques, B. Prieur, C. Breuzet, D. Danse, G. Guillou, P. Charrière, O. Bouzereau, C. Pichaud, F. Larivé,
Couverture : © D.R., F. Iarvé - source image arbre : designnatures - **Maquette** : Pierre Sandré.
Publicité : François Tonic / Nefer-IT - Tél. : 09 86 73 61 08 - ftonic@programmez.com.
Imprimeur : SIB Imprimerie
Marketing et promotion des ventes : Agence BOCONSEIL - Analyse Media Etude - Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD Téléphone : 09 67 32 09 34

Contacts : Rédacteur en chef : ftonic@programmez.com - Rédaction : redaction@programmez.com - Webmaster : webmaster@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908 - © NEFER-IT / Programmez, mars 2020

Abonnement :
Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex. - Tél. : 01 55 56 70 55 - *abonnements.programmez @groupe-gli.com*
Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30.

Tarifs
Abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 € - Autres pays : nous consulter.

PDF
35 € (monde entier) souscription sur www.programmez.com

INFORMER
pour transformer l'entreprise

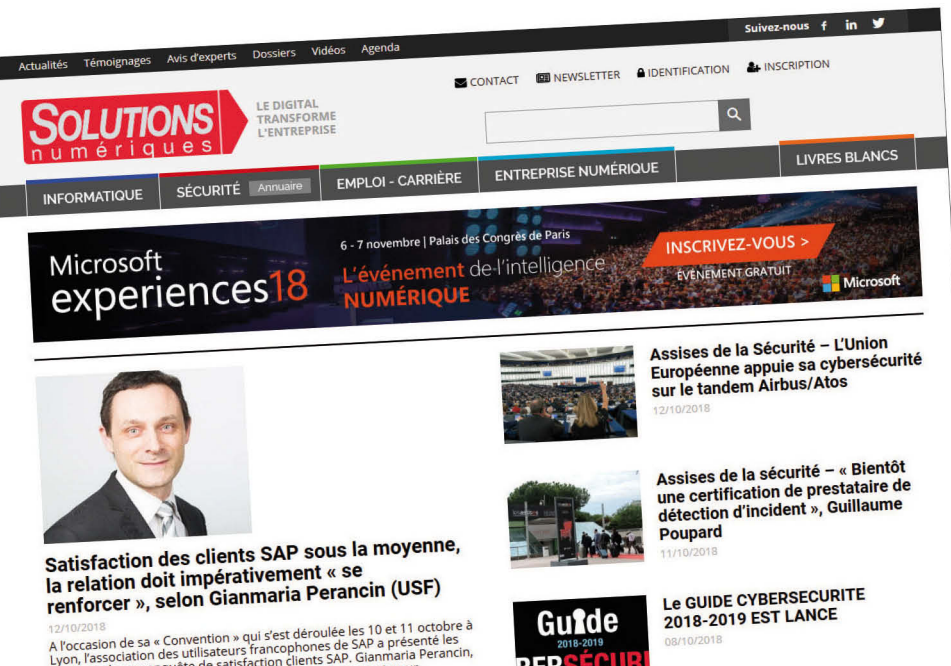
La dématérialisation, le Cloud, les communications unifiées, les nécessités de la cybersécurité transforment le travail et toute l'entreprise, les services publics. Le magazine, le site, ses newsletters vous informent sur cette actualité mouvante et vous aident à décoder les tendances

Abonnez-vous

www.solutions-numeriques.com/abonnement/

NOUVEAU : le site du TELETRAVAIL !

Pour répondre à vos besoins d'information



- Vous êtes responsable informatique ou bien dirigeant ou cadre d'entreprise ? *2 sites répondent à votre profil*
- La cybersécurité vous concerne ? *Cliquez sur l'onglet. Vous trouverez les infos, l'annuaire, le lexique, etc*
- L'emploi, les salaires, les formations, les offres vous intéressent ? *Le site sur l'Emploi dans le numérique est à votre disposition*

www.solutions-numeriques.com

A DÉCOUVRIR D'URGENCE

116 pages - Format magazine A4

Disponible sur www.programmez.com