CONCOURS : RELEVEZ LE DÉFI DE MOBIOOS. 2250 € À GAGNER !

programmez.com

Programmez!

Le magazine des développeurs

230 JUIN 2019

CLOUD COMPUTING

L'impact sur ton code

Au-delà du buzz

L'approche Serverless

Moderniser une app

© Maxiphoto

CLI en GO

PYTHON

PHP

REDIS

.Net Core 3.0

M 04319 - 230 - F: 6,50 € - RD

LE SEUL MAGAZINE ÉCRIT PAR ET POUR LES DEVELOPPEURS

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

Olymp

La formation nouvelle génération

Le logiciel indispensable aux centres de formation pour profiter des avantages de la formation présentielle, et de la formation en ligne.

Edité par

IdeaStudio

EDITO

Le code de Schrödinger

« En physique quantique, nous avons le chat, le développeur a le code : tant qu'on n'utilise pas le code, on ne sait pas s'il fonctionne ou non. »

Durant 3 semaines, les développeurs ne dorment plus, sont surexcités car c'est la saison des grandes conférences techniques : BUILD pour les passionnés de Windows, I/O pour les accros au moteur de recherche et WWDC pour les adorateurs des pommes croquées. Et, hasard du calendrier (ah le bug du calendrier qu'est-ce qu'il est taquin), BUILD et I/O se sont affrontés à distance. La foule durant les keynotes I/O était impressionnante et profitait du soleil.

Parmi les grosses annonces : Android Q en bêta, le projet Mainline pour améliorer les mises à jour au niveau système (14 modules sont concernés), Android Studio 3.5 (préversion avec le projet Marble qui doit améliorer le résultat final du développement). L'annonce la plus intéressante concerne Flutter que nous avions évoqué dans le n°229. Google a dévoilé trois versions supplémentaires : pour le web, pour le desktop et pour les matériels embarqués (encore très embryonnaire) ! L'idée est simple : coder en Flutter et déployer partout. Tiens cela me rappelle quelque chose (note de lecture : regarde vers Java).

Flutter sera alors votre framework de développement unique, mais les contraintes entre le mobile, le web et le desktop ne sont pas les mêmes. Google devra adapter les couches à chaque plateforme pour assurer la portabilité et minimiser les modifications / adaptations. Le défi est donc énorme. Les nouvelles éditions en sont encore au stade des développements et rien ne dit que ces portages seront efficaces, ni un jour disponibles. Parfois, Google est très joueur avec le développeur. On ne sait jamais si le projet est vivant ou mort, comme le chat de Schrödinger.

Côté BUILD, Microsoft a continué à doper Windows à Linux. En 2016, le sous-système Windows pour Linux (WSL) apparaissait dans Windows 10 (à condition d'activer le mode dev). Il était temps de passer à la v2. Et pour bien faire les choses, un vrai noyau Linux sera déployé et non plus une simple émulation du noyau et des librairies système… En quelque sorte, Windows sera bi-OS : Windows ET Linux. Le noyau utilisera la branche 4.19 (dite LTS). Un important travail est réalisé pour optimiser le noyau et réduire sa taille pour ne pas peser sur Windows. Et toute la partie émulation sera supprimée. Ce travail sera open source. Cette couche arrivera courant de l'été. A cela on peut rajouter la nouvelle version de Edge s'appuyant sur Chromium, tu sais, le projet open source servant à builder Chrome !

Eh oui, depuis la chute d'Internet Explorer par les assauts de Firefox et le coup de grâce donné par Chrome, la stratégie navigateur de Microsoft allait nulle part.

SOMMAIRE

Dans le prochain numéro !

Programmez! #231, dès le 28 juin 2019

SPÉCIAL ÉTÉ

Minecraft / guide des cartes IoT / Python + MicroPython / Gameboy / Angular / Serverless / .Net Core 3.0 / Amazon Alexa

Un numéro à ne pas rater !

Et pourtant, l'éditeur a tout fait pour nous faire oublier la purge qu'était IE6 et tous ces webmasters en dépriment à cause de lui.

IE7, etc., n'a finalement été qu'un échec et Edge, avec toutes les promesses faites à sa présentation, n'a pas fait mieux. Pendant ce temps-là, deux navigateurs enfonçaient les défenses de Port Réal : Safari (sur mobile surtout) et Chrome. Et le triomphe a été total pour WebKit même s'il a été forké par Google. A partir de là, si Microsoft voulait continuer à exister sur le navigateur, le choix était rapide à faire : utiliser un projet open source de référence. Le choix était limité à WebKit et Chromium…

Encore un effort et Office sera un LibreOffice repackagé…/Troll

Enfin, une annonce que l'on n'attendait pas forcément : Windows Terminal. « Nous vous avons écouté » dixit Microsoft, même s'il a fallu beaucoup beaucoup beaucoup de temps pour avoir un vrai Terminal (on met moins de temps à traverser la Moria, même avec un Balrog). On va pouvoir s'amuser un peu. Si vous êtes Linux et macOS, vous pouvez troller, le Terminal ça vous connaît !

Dans les trolls niveau boss de fin de niveau, nous avons le «sans rire, c'est une vraie annonce » ?. Le ministère de l'économie et des finances a sorti fin avril : « il faut que nous ayons un cloud stratégique qui permette de stocker des données stratégiques et de les protéger… qui garantisse l'indépendance et la souveraineté sur un certain nombre de données stratégiques ». Les dossiers sont à déposer avant fin 2019.

On ne nous aurait pas fait le coup il y a quelques années avec le cloud souverain à la française ? Il faudrait que je fasse un peu de ménage dans ma mémoire. Encore un bug du garbage collector.

N'OUBLIEZ PAS !

- Les sources du numéro sur https://github.com/francoistonic
- Nos podcasts sur programmez.com
- Nos vidéos sur youtube

Ouf, l'édito du mois est terminé.

// bonne lecture !
//
// ToDo List :
// meetup informatique quantique : 25 juin / // Paris
// DevCon #8 : octobre !
//
// 2020 : une nouvelle série à Westeros
// 2021 : le Seigneur des Anneaux version // Amazon

François Tonic
ftonic@programmez.com

Aux Pays-Bas,

les mises à jour c'est la liberté ?

Une mise à jour a rendu inutilisable l'ensemble des bracelets électroniques utilisés par la police néerlandaise afin de suivre les condamnés. Pendant plusieurs jours au début du mois de mai, les bracelets ne parvenaient plus à transmettre les données de géolocalisation aux salles de contrôles de la justice néerlandaise. En attendant le correctif (finalement arrivé jeudi 9 mai), les policiers ont été contraints de se charger de la surveillance des détenus utilisant ces dispositifs, soit via des rendez-vous au commissariat, soit en renvoyant temporairement certains détenus en cellule. Cher payé pour une mise à jour ratée.

DockerHub

piraté, les données de 190 000 utilisateurs exposées

Docker a informé une partie de ses utilisateurs qu'un piratage de son service Dockerhub avait pu exposer leurs données. L'incident de sécurité a eu lieu le 25 avril et a exposé les données de 190 000 utilisateurs, dont les identifiants et les tokens de connexions Github ou Gitlab ont pu être volés. Docker a prévenu par mail les utilisateurs affectés par l'incident de sécurité et les tokens potentiellement exposés ont été révoqués. Dans le doute, la société appelle néanmoins ses utilisateurs à la vigilance.

Azure et Github

Microsoft intègre ses protégés

A l'occasion de la conférence Build, Microsoft a détaillé ses plans de rapprochement entre Azure et Github. Microsoft concentrera donc ses premiers efforts sur l'unification de l'authentification : il sera maintenant possible pour les développeurs disposant d'identifiants Github de les utiliser pour s'authentifier sur les portails Azure et Azure Devops. En plus de cela, Microsoft annonce avoir intégré Azure Active Directory au sein de Github Enterprise. Il sera donc possible d'utiliser les règles définies via Active Directory pour définir les groupes et autorisations d'accès sur Github entreprise.

Les extensions Firefox

mises hors service par un certificat

Les utilisateurs de Firefox ont eu une mauvaise surprise le week end du 5 mai : l'ensemble des extensions du navigateur était inutilisable, sans correctif disponible, à cause d'un problème de certificat non renouvelé chez Mozilla. Une situation de crise qui a forcé les développeurs de Firefox à publier en urgence une mise à jour de leur navigateur dès lundi afin de corriger le problème (et donc de renouveler le certificat fautif.) Depuis, les équipes de Mozilla ont fait leur mea culpa et promettent dans plusieurs posts de blog que tout sera fait pour que cela ne se reproduise plus.

Hertz et Accenture

soldent en justice leur différend à 32 millions $

32 millions $ pour un site web, cela peut paraître cher payé : c'est en tout cas l'avis de la société américaine Hertz, qui traîne en justice la société de service Accenture suite au fiasco d'un projet de refonte de site web. Le projet débute en 2016 s'embourbe rapidement : la faute au cahier des charges mal défini selon Accenture et la faute à une gestion de projet hasardeuse de la part des développeurs selon Hertz. Après de multiples changement d'équipes et frais supplémentaires, la note s'élève à 32 millions $ pour Hertz et le site est loin d'être opérationnel. Hertz a donc décidé d'arrêter les frais et de régler tout cela devant le tribunal de New York. Au vu de l'ambiance, on imagine que les juges doivent trépigner d'impatience à l'idée de se plonger dans les détails de trois longues années d'incompréhension entre les deux géants.

App Store

la justice ébrèche le vase clos d'Apple

Pour la Cour Suprême des États Unis, le fonctionnement de l'App Store relève d'une « pratique monopolistique. » Cette décision rendue lundi 13 mai ouvre la voie à l'ouverture d'un procès antitrust visant les pratiques d'Apple, attaqué en justice par un groupe de citoyens américains qui contestaient les pratiques tarifaires d'Apple en la matière. En l'absence d'alternative à l'App Store, la commission de 30 % que s'arroge la société de Cupertino apparaît abusive aux yeux de certains et la Cour Suprême estime qu'il y a bien là matière à procès. Une décision dont Apple se serait certainement passé : la société s'appuyait en effet sur son écosystème clos et mieux sécurisé pour se démarquer de son grand concurrent, Google et son Play Store.

amazon alexa

European Alexa Perks Program

Tu rêves de développer des Skills et d'explorer les multiples possibilités des assistants vocaux ? Viens relever le défi Amazon Alexa !

DE NOMBREUX LOTS SONT À GAGNER !

Amazon Alexa est utilisé sur +100 millions de terminaux dans le monde.

Pour rejoindre ce défi de programmation, c'est très simple :
https://developer.amazon.com/fr/en-gb/alexa-skills-kit/alexa-developer-skill-promotion

1 tu t'inscris

2 tu codes tes Skills

3 tu les publies, avant le 30 novembre

A toi de jouer !

Informations, FAQ, inscription sur : https://developer.amazon.com/fr/en-gb/alexa-skills-kit/alexa-developer-skill-promotion

RETROUVEZ LA RÉDACTION DE PROGRAMMEZ! SUR BEST OF WEB, DEVFEST LILLE & JHIPSTER

Juin

4 : Paris Container Day/Paris

Le 4 juin prochain aura lieu la quatrième édition du Paris Container Day. Le Paris Container Day est la conférence pionnière en France dédiée à l'écosystème des conteneurs et de ses bonnes pratiques. Pour cette nouvelle édition, les participants pourront assister à une vingtaine de conférences techniques, retours d'expérience et des fast tracks. Le thème de cette année est "Standards & Craftsmanship". Organisée par Xebia.
https://paris-container-day.fr

7 : Best of Web/Paris

Une conférence qui réunit le Best Of des présentations données pendant l'année, sur une journée à Paris ! Les organisateurs de meetups se réunissent pour proposer une journée avec les meilleurs talks qui ont été donnés dans leurs meetups respectifs. Contrairement à ce que le nom pourrait laisser entendre, on n'y parle pas exclusivement de Web. On y parle de tous les sujets présents parmi les meetups fédérés, dont Human Talks, Duchess France, Scala UG, Software Crafters, ... qui ne sont pas des communautés spécialisées dans le Web.

Venez échanger avec les communautés ! +500 personnes attendues.

Pour accéder aux formations du jeudi, il faut avoir un billet pour la conférence du vendredi et s'inscrire pour une somme symbolique de 10€ afin de réserver sa place. Il y a au maximum une quinzaine de participants pour que le formateur puisse travailler dans les meilleures conditions.

Les places sont vendus pour la conférence : 50€ le billet regular.
Site : http://bestofweb.paris

11 au 14 juin : INFORSIF/Paris

Depuis 1982 le congrès INFORSID (INFormatique des ORganisations et Systèmes d'Information et de Décision) rassemble chaque année des chercheurs et des professionnels afin d'échanger sur les problématiques d'ingénierie et de gouvernance des systèmes d'information.
Site : http://inforsid.fr/Paris2019/

11-13 juin : OW2 Con'19/Paris

La conférence annuelle OW2 est une rencontre internationale d'experts, d'architectes, de développeurs et de chefs de projets open source. **Communauté open source internationale et indépendante,** OW2 est dédiée au développement de logiciels d'infrastructure de qualité industrielle. OW2 regroupe des entreprises, des collectivités et des organismes de recherche de premier plan dont Orange, l'Inria, la Mairie de Paris, Cap Gemini et Airbus Défense.

Le thème central de OW2con cette année est : "**Open Source : vers la maturité industrielle**". Alors que l'open source se généralise, les développeurs informatiques, les fournisseurs, les utilisateurs et même les organisations open source telles qu'OW2 doivent s'adapter. Les fusions à coups de milliards de dollars annoncées en 2018 démontrent que l'open source atteint sa maturité. Les projets open source deviennent de plus en plus concurrentiels et de plus en plus critiques et tous les développeurs, fournisseurs et utilisateurs informatiques doivent avoir une stratégie open source.

Plusieurs thèmes technologiques seront couverts lors de la conférence : *les applications d'entreprise, la sécurité, l'accessibilité, l'open cloud, l'IoT, la blockchain, l'intelligence artificielle, la qualité et les tests logiciels, les modèles économiques et la gouvernance des logiciels libres, open source dans les grandes villes.*

OW2con est ouvert à tous, l'évènement est gratuit et les conférences ont lieu en anglais.
Site : https://www.ow2con.org/view/2019/

14 : DevFest Lille

Le DevFest Lille Saison 3 est la 3e édition d'un évènement complet de conférences sur le Web, le mobile, le cloud et les objets connectés. Plus de 700 participants attendus cette année au

Kinépolis de Lomme, le plus grand cinéma d'Europe. Une nouvelle saison pleine de surprises pour ce Devfest Lille sous le thème des Séries TV qui se déroulera le 14 juin prochain.
Site : https://devfest.gdglille.org

18 : Google Cloud Summit Paris

Venez découvrir la plateforme cloud de Google le 18 juin prochain à la Porte de Versailles. L'éditeur fera les choses en grand : +30 sessions sur tous les sujets chauds du moment, 4 salles en parallèles ! Tous les experts Google et les meilleurs partenaires seront présents sur scène. Bien entendu, de nombreuses entreprises viendront parler de leur projet Google Cloud Platform !

24 au 28 juin : COMPAS 2019/Bayonne

Compas est la Conférence francophone en informatique autour des thématiques du parallélisme, de l'architecture et des systèmes. L'édition 2019 aura lieu du 24 au 28 juin à l'IUT de Bayonne.
Site : https://2019.compas-conference.fr/

25 juin : l'informatique quantique vers l'infini et au-delà / Paris

Programmez ! vous propose son meetup de fin de saison sur l'informatique quantique ! A partir de 19h, au cœur de Paris, dans les locaux d'Arolla.

Informations et inscription :
https://www.programmez.com/

MEETUP PROGRAMMEZ! #6

SPÉCIAL

programmation quantique

Tout savoir ou presque sur l'informatique quantique

25 juin à partir de 19h

Où

Arolla
21, rue du bouloi
75001 Paris
Batiment D au fond à gauche

Métros / RER :
Métro 1 : station Louvre-Rivoli
Métro 4 / RER A, B & D : Châtelet - Les Halles

Inscription et informations sur **www.programmez.com**

Meetups organisés par [Programmez! Le magazine des développeurs]

25 juin : Coding_the_universe / Paris

Développeur(se) Sécurité ou Devops, viens découvrir Zenika lors d'une soirée interstellaire avec nos Z ! Inscris toi seul ou avec ta propre équipe pour déjouer notre Serious Game ! Au programme : du challenge, de l'adrénaline, une course contre la montre et surtout beaucoup de bonne humeur et de convivialité, et sans nul doute de belles rencontres.
Site : https://seriousgamezenika.eventbrite.fr

27 juin : JHipster Conf' / Paris La Défense

Ne ratez pas la 2ème édition de la JHipster Conf le 27 Juin à Paris ! Cette année, la JHipster Conf possèdera deux tracks (français / anglais) ! Les plus grands contributeurs et passionnés de la communauté JHipster seront présents :) Les speakers : Julien Dubois, Deepu K Sasidharan, Pia Mancini, Matt Raible, Josh Long...
Site : https://jhipster-conf.github.io/

27 & 28 : Sunny Tech/Montpellier

Plusieurs conférences et ateliers en parallèle sur 2 journées, sujets transverses allant des outils, méthodologies de conception et développement logiciel jusqu'à la gestion de produit et notamment l'expérience utilisateur.
Site : https://sunny-tech.io

Septembre

16-17 : Agile en Seine 2019/Paris

Agile en Seine se veut une conférence multipratique, ouverte à tous, dont l'objectif est de mettre en avant toute la diversité de ce que l'on nomme communément « Agilité » : du Scrum à l'agilité à l'échelle, en passant par le Design Thinking, le Kanban, le Lean Startup, le Lean, l'entreprise libérée, l'Holacracy … Pour cette édition 2019, nous vous proposons un nouveau format, sur 1,5 jour, le 16 septembre après-midi et le 17 septembre 2019.
La première demi-journée sera consacrée à l'animation d'ateliers de mise en œuvre des pratiques Agile et la seconde journée à des conférences de partage et des retours d'expérience.
Site : https://www.agileenseine.com

Octobre

3 : DevFest Toulouse 2019/Toulouse

Le DevFest Toulouse aura lieu, pour sa 4e édition, le 03 octobre 2019 au Centre des Congrès Pierre Baudis de Toulouse et réunira 900 développeurs, personnes travaillant dans les métiers techniques de l'informatique et étudiants. En attendant le jour J, le CFP (Call For Papers / Appel à Orateurs) du DevFest Toulouse est désormais ouvert ! Cette année, vous pouvez soumettre du 25 mars jusqu'au 25 mai (2 mois). À mi-CFP, une présélection de nos "early speakers" sera effectuée, ne tardez donc pas à soumettre vos propositions ! ;) La délibération finale se fera ensuite pendant le mois de juin.
Site : https://devfesttoulouse.fr/fr/

Novembre

15 : BDX I/O/Bordeaux

Site : https://www.bdx.io/

CONFÉRENCES ORGANISÉES PAR XEBIA

27 juin : 2e édition du DataXDay => https://dataxday.fr/
DataXDay est l'unique conférence Data qui réunit DataScientists, Data Engineers et Data Architects autour d'une quinzaine de conférences techniques. Venez échanger autour de la Data Science du PoC à la mise en production, des architectures Microservices et Serverless, de Craftsmanship, de sécurité, de Cloud, etc. Rendez-vous le 27 juin prochain.

7-8 Octobre : 4e édition de la FrenchKit
La FrenchKit sera de retour pour une quatrième édition les 7 et 8 octobre prochains. La FrenchKit est la première conférence iOS et macOS en France. Durant ces 2 journées, certains des développeurs les plus reconnus de la communauté internationale traiteront un large éventail de sujets, des API Cocoa à Swift.

28 novembre : XebiCon, Build The Future => https://xebicon.fr/
À travers une soixantaine de conférences, la Xebicon vous donnera les clés pour tirer le meilleur des dernières technologies. Conférences techniques, retours d'experiences clients, hands-on, venez partager et échanger sur les nouveautés technologiques autour du Cloud, de la Data, des nouveaux standards d'Architecture, des nouveaux langages Front-End, de l'IoT, de la culture DevOps, des transformations agiles à l'échelle ou encore de la mobilité.

Girls Can Code !

Les inscriptions pour la 6e édition des stages Girls Can Code! sont ouvertes. Ces stages d'une semaine d'initiation à la programmation pour les collégiennes et lycéennes visent à promouvoir l'informatique auprès des filles dans un cadre agréable et créatif. Les stages Girls Can Code! sont gratuits, sauf pour les éventuels frais de transport et d'hébergement jusqu'au lieu du stage. Encadrés par des étudiants, le niveau est adapté à chacune des participantes et aucun prérequis n'est nécessaire.

Pour l'édition 2019, 3 stages sont prévus :
- EPITA Paris, du 8 au 13 juillet 2019,
- EPITA Paris, du 26 au 31 août 2019,
- ENS Lyon, du 26 au 31 août 2019.

Vous pouvez en apprendre davantage sur Girls Can Code! ainsi que les éditions précédentes sur cette page : https://gcc.prologin.org

Merci à Aurélie Vache pour la liste 2019, consultable sur son GitHub :
https://github.com/scraly/developers-conferences-agenda/blob/master/README.md

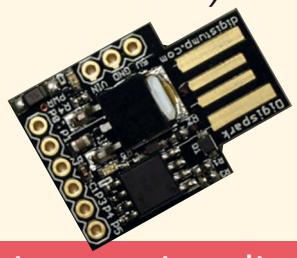
Tous les numéros de Programmez! sur une clé USB (depuis le n°100)

34,99 €*

Clé USB. Photo non contractuelle. Testé sur Linux, macOS, Windows. Les magazines sont au format PDF.

* tarif pour l'Europe uniquement.
Pour les autres pays, voir la boutique en ligne

+



1 carte de prototypage Attiny85, compatible Arduino, offerte !

Commandez-la directement sur notre site internet : www.programmez.com

Complétez votre collection

Prix unitaire : ***6,50€***

Lot complet

1 CADEAU À OFFRIR !

39,99 €

(au lieu 45,50 €)

☐ 218	: __ ex	☐ 225	: __ ex	☐ Lot complet : 218 - 219 - 220 - 221 225 - 228 - 229 39,99 €	Commande à envoyer à : **Programmez!** 57 rue de Gisors - 95300 Pontoise
☐ 219	: __ ex	☐ 228	: __ ex		
☐ 220	: __ ex	☐ 229	: __ ex		
☐ 221	: __ ex				

soit __ exemplaires x 6,50 € = ____ € soit au **TOTAL** = _____ €

Prix unitaire : 6,50 € (Frais postaux inclus)

☐ M. ☐ Mme ☐ Mlle Entreprise : _____ Fonction : _____

Prénom : _____ Nom : _____

Adresse : _____

Code postal : _____ Ville : _____

E-mail : _____ @ _____

Règlement par chèque à l'ordre de Programmez ! I Disponible sur www.programmez.com

Offres pouvant s'arrêter à tout moment, sans préavis **PROG 230**

Roadmap 2019 des langages & des IDE

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

DÉJÀ DISPONIBLE

React 16.8 **SWIFT 5** **Java 12** **Kotlin 1.3.30** **Rust 1.34.0**

1ER SEMESTRE 2019

Angular 8

Date de sortie : mai.
Les principales nouveautés attendues / annoncées : les versions se succèdent à une cadence élevée, tous les 6 mois environ. La v8, qui arrive maintenant, introduit plusieurs améliorations :

- Sur la CLI : possibilité d'utiliser ES5 et ES2015+ avec une amélioration sur le chargement et le build du code JS ;
- Pré-version d'Ivy ;
- Web Worker : amélioration sur les performances et la parallélisation des applications ;
- Mise à jour des dépendances, notamment outils.

Sur la compatibilité, attention : seuls les codes provenant d'Angular 7 sont pris en compte (dans la note officielle).
Et aussi : retrait de @angular/http, ngcc -make disponible, diverses corrections de bugs. Les équipes préviennent aussi d'un changement provocant une non-compatibilité de code : rajouter impératif de @blazel/typescript dans le package.json

Au-delà : la v9 est prévue pour octobre – novembre.

SWIFT 5.1

Date de sortie : printemps.
Les principales nouveautés attendues / annoncées : ce sera la première mise à jour de SWIFT 5. Cette version doit apporter des corrections de bugs et terminer la stabilité des librairies et des modules. Bien entendu, la 5.1 est retrocompatible avec la 5.0. On doit s'attendre à quelques changements dans ce que les équipes appellent la « module stability » comme l'apparition du .swiftinterface pour le fichier d'interface à la place de .swiftmodule. Quelques éléments ici : https://swift.org/blog/5-1-release-process/

Attention : la stabilité ABI n'est pas encore totalement disponible sur les plateformes non-macOS.

Au-delà : la prochaine version majeure devrait être la v6.

Rust 1.35

La version 1.35 est attendue pour fin mai (quand sortira Programmez!). La 1.36 pour début juillet et le développement est en cours.
Pour suivre la roadmap : https://forge.rust-lang.org

Ruby on Rails 6

Date de sortie : mai ou juin
Les principales nouveautés attendues / annoncées : parallèlement au langage, Rails continue à évoluer de son côté. Les grosses nouveautés attendues concernent les actions sur les mailbox, les évolutions sur les fonctions de tests, particulièrement sur l'Action Cable testing et les tests parallèles. Pour la migration, il faudra partir d'une version 5.2 pour faire la transition. La RC1 a été publié fin avril.

2E SEMESTRE 2019

React 16.9

Date de sortie : sans doute dans l'été.
Les principales nouveautés attendues / annoncées : en attendant la prochaine version « majeure », React 16.8 est régulièrement mis à jour. La version 16.8.6 a été déployée fin mars avec différents bug fix.

Pour suivre les sorties :
https://github.com/facebook/react/releases

Java 13

Date de sortie : septembre - octobre
Les principales nouveautés attendues / annoncées : cette version apportera plusieurs nouveautés et améliorations. Parmi les annonces faites :

- Shenandoah : ramasse-miettes à faible temps de pause ;
- API de constantes JVM ;
- AArch64 : sert à supprimer toutes les sources liées aux arm64port. Le but est de faciliter le portage ARM ;
- archives CDS par défaut ;
- amélioration sur la GC G1.

Kotlin 1.3.40

Date de sortie : été ?
Les principales nouveautés attendues / annoncées : le développement de cette nouvelle version est en cours. Nous savons d'ores et déjà que les appels trimIndent et trimMargin seront traités par le compilateur et non le runtime. Plus de détails dans les semaines à venir.

Goland 1.13

Date de sortie : août/septembre.
Les principales nouveautés attendues / annoncées : cette version doit activer le mode module par défaut (le mode par défaut d'auto à activer) tout en déconseillant le mode GOPATH. On aura aussi des nouveautés sur les génériques, la gestion des erreurs.
Au-delà : Go 2 sera LA grosse évolution du langage Go, même si les équipes ne veulent pas faire une version de rupture, mais des évolutions au fil des versions. Un des changements sera la manière de définir les évolutions et comment la gouvernance fonctionne. L'idée est que la communauté soit plus impliquée. La compatibilité avec Go 1.x sera un des aspects cruciaux. Pour le moment, le planning de Go2 reste à préciser.

C# 8.0

Date de sortie : automne.
Les principales nouveautés attendues / annoncées : cette nouvelle évolution du langage phare .Net doit arriver avec .Net Core 3.0. Parmi les nouveautés attendues :

- Les types de références nullables doivent en finir avec les exceptions null. Pour cela, elles vous empêchent de mettre null dans des types de référence ordinaire comme string. Par défaut, ces types seront non nullables ! Cela pourrait avoir un impact sur les codes existants ;
- Les flux asynchrones ;

- Types range et index ;
- Expressions Switch.

Au-delà : il faut s'attendre à des itérations périodiques comme pour C# 7. Pas de détails pour le moment.

Python 3.8

Date de sortie : octobre
Les principales nouveautés attendues / annoncées : plusieurs nouveautés devraient plaire aux développeurs. Tout d'abord, on bénéficie du nouveau paramètre pythonpycacheprefix. Il permet d'utiliser le cache bytecode dans une branche séparée du système de fichiers parallèle. Il sera à préférer au __pycache__ présent dans les sous-répertoires des dossiers sources. On disposera aussi de la méthode as_integer_ratio(), dans le type int. Le contrôle-C, SIGINT, sera modifié pour éviter les problèmes liés à l'exception KeyboardInterrupt. Plusieurs modules auront droit à des améliorations comme asyncio avec ProactorEventLoop ou encore dans gc avec de nouveaux paramètres dans le get_objects().
Pour plus de détails : https://docs.python.org/dev/whatsnew/3.8.html .
Au-delà : en toute logique, la prochaine version majeure sera la 3.9. Pour le moment, pas grand-chose n'a été communiqué. Cette version ne devrait pas arriver avant 2021.

PHP 7.4

Date de sortie : décembre (?).
Les principales nouveautés attendues / annoncées : la 7.4 doit apporter le préchargement (preloading), au cœur de PHP pour pouvoir améliorer les performances. Ce mécanisme permet de charger l'ensemble des fichiers PHP dès le démarrage et ainsi améliorer les accès et assurer une disponibilité constante de ceux-ci. Par contre, en cas de changement des fichiers, il faudra redémarrer le serveur. On notera aussi la disponibilité des propriétés typées, l'extension FFI (Foreign Function Interface) pour appeler du code C, le nouvel opérateur Null Coalescing, l'apparition de nouvelles méthodes (__serialize & __unesrialize).
On notera aussi le retrait de PEAR ou la dépréciation de ext/wwdx.
Attention : la 7.4 n'est pas encore totalement figée. Des changements peuvent intervenir.

Au-delà : après la 7.4, difficile d'y voir clair. Il était question d'une v8. Pour le moment, rien de très précis.

Ruby 2.7

Date de sortie : décembre (?).
Les principales nouveautés attendues / annoncées : le langage Ruby continue à évoluer. La 2.6 est sortie fin 2018, notamment avec un nouveau compilateur JIT. On bénéficie aussi d'un nouveau module RubyVM ::AbstractSyntaxTree. Il analyse une chaîne de caractères et retourne les nœuds d'arbre syntaxique. Un travail d'optimisation a été réalisé.

COURANT 2020

C++20

Date de sortie : 2020.
Les principales nouveautés attendues / annoncées : les spécifications de C++20 sont désormais figées, et un premier document complet sera disponible cet été. Cette future version du langage mettra 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 conservée. 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.

LTS

On parle souvent de LTS et de non-LTS. LTS signifie support long terme. C'est-à-dire que cette version est supportée officiellement durant x années et recevra les mises à jour et les patchs de sécurité nécessaires. Une version non-LTS a une durée de vie très courte, quelques mois. Seules les versions issues d'une forte communauté, d'une fondation, d'un éditeur sont LTS. Chacun fait un peu ce qu'il veut sur le support. Exemple : Angular est sur un cycle de 6 mois pour les versions majeures. Le support se fait sur une version.

Versionning

On parle de versions majeures et de versions mineures. Ces dernières sont souvent des versions de bug fix, de sécurité, introduisant peu ou pas de nouveautés.
React explique aussi la structure des versions avec x.y.z :
X = une version majeure introduisant une rupture ;
Y = nouvelle fonction, version mineure (ex. : 15.6) ;
Z = bug fix. On corrige les bugs, les problèmes importants.
N'oubliez pas de lire les notes de versions (release notes). Elles indiquent les changements, les modifications, les nouveautés, la migration entre les versions.

CÔTÉ IDE

NetBeans 11 : disponible depuis avril. Cette version supporte mieux JDK 11, diverses améliorations sur la partie PHP 7.x, JUnit 5.3.1

Eclipse 2019-06 sera la prochaine version d'Eclipse IDE. Elle sera libérée en juin.

Côté **JetBrains**, on nous annonce :
- 23 mai : TeamCity 2019.1
- 26 juin : UpSource 2019.1
- 22 juillet : WebStorm 2019.2
- 24 juillet : CLion 2019.1

Android Studio : la version 3.5 est actuellement en bêta. Parmi les améliorations : Project Marble à jour, plug-in Gradle, évolution de la gestion mémoire, rapport de l'usage mémoire généré si besoin. Cette version s'appuie sur IntelliJ IDEA 2019.1.

Spyder : IDE pour Python. Les versions 3.3.x remontent maintenant à août 2018. Pas de nouvelles versions pour le moment.

Qt Creator : la v4.9.0 est disponible depuis avril dernier. IDE pour le C++ et la librairie Qt. Cette version ajoute le support ECMAScript 7, En même temps, l'éditeur a sorti QT Design Studio 1.1.1.

RAD Studio : actuellement, nous en sommes aux sous versions de la 10.3 de Rio disponible depuis quelques mois. La 10.3.1 prend en charge iOS 12, utilisation de Ext JS, amélioration des supports Android SDK / NDK, compilation Win32 Clang C++17. Une amélioration de macOS 64 est attendue dans les versions 10.3.x ainsi que diverses améliorations sur Windows et C++. La version 10.4.0 est attendue pour cet été / automne.

Abonnez-vous à Programmez! Abonnez-vous à Programmez! Abonne

[Programmez!] Le magazine des développeurs

Nos classiques

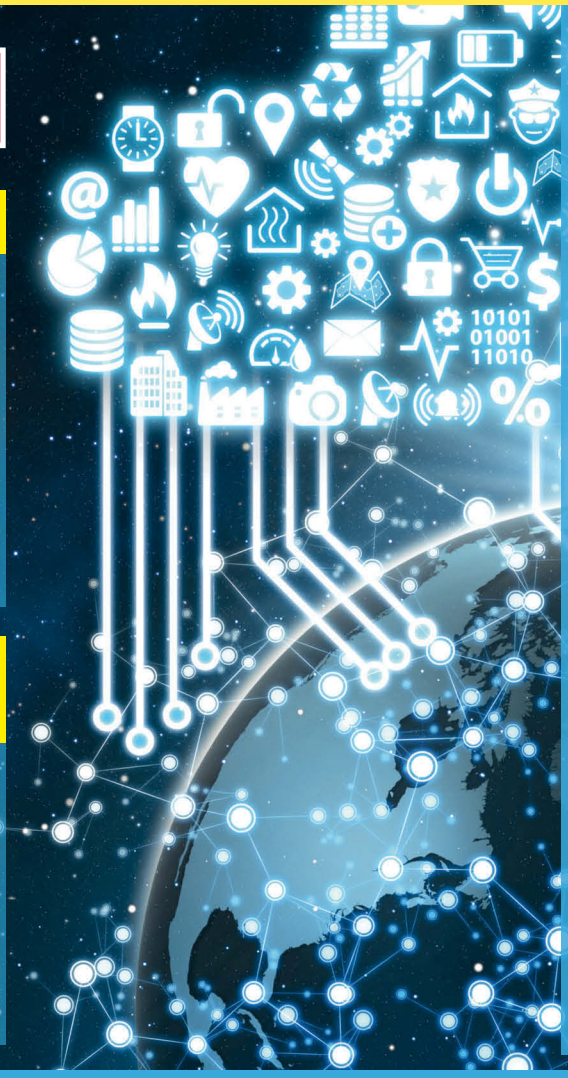
1 an 11 numéros	49€*
2 ans 22 numéros	79€*
Etudiant 1 an - 11 numéros	39€*

** Tarifs France métropolitaine*

Abonnement numérique

PDF 1 an - 11 numéros	35€
Option : accès aux archives	10€

Souscription uniquement sur
www.programmez.com



Offres 2019

1 an ..... **59€**
11 numéros
+ 1 vidéo ENI au choix :

- Symfony 3*
  Développer des applications web robustes
  (valeur : 29,99 €)
- Raspberry Pi*
  Apprenez à réaliser et piloter une lumière d'ambiance
  (valeur : 29,99 €)

2 ans ..... **89€**
22 numéros
+ 1 vidéo ENI au choix :

- Symfony 3*
  Développer des applications web robustes
  (valeur : 29,99 €)
- Raspberry Pi*
  Apprenez à réaliser et piloter une lumière d'ambiance
  (valeur : 29,99 €)

** Offre limitée 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

- ☐ **Abonnement 1 an :** 59 €
  11 numéros + 1 vidéo ENI au choix :
- ☐ **Abonnement 2 ans :** 89 €
  22 numéros + 1 vidéo ENI au choix :

  ☐ Vidéo : Symfony 3
  ☐ Vidéo : Raspberry Pi

PROG 230
Valable jusqu'au 28 juin 2019

☐ 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**

A DÉCOUVRIR D'URGENCE

NOUVEAU !

LE CADEAU GEEK IDÉAL

Découvrez l'âge d'or des micro-ordinateurs de 1973 à 2007

9,99 €
(+ 3 € de frais postaux)

116 pages - Format magazine A4

☐ **Découvrez l'âge d'or des micro-ordinateurs de 1973 à 2007** :
9,99 € (+3 € de frais postaux) = 12,99 €

Commande à envoyer à :
Programmez!
57 rue de Gisors - 95300 Pontoise

PROG 230

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !

commandez sur www.programmez.com

François Tonic

La localisation : un enjeu pour le développeur

Le sujet est revenu au premier plan avec la disponibilité de Fluent, en v1, de Mozilla, mi-avril dernier.

« Firefox est localisé dans près de 100 langues dans le monde entier. Un projet d'une telle envergure est confronté à de nombreux défis de localisation qu'il est difficile de surmonter en utilisant des solutions de localisation traditionnelles. La localisation de logiciels a été dominée par un paradigme obsolète : les traductions correspondent à la langue source, qui est souvent l'anglais. Cependant, il existe de nombreux aspects grammaticaux et stylistiques qui ne correspondent pas entre les langues. Dans les langues de genre grammatical, par exemple, les adjectifs et les participes passés doivent s'accorder avec le genre du nom. Par exemple en français pour dire "connected" on peut dire connecté, connectée, connectés et connectées. » précise l'éditeur dans son annonce.

La localisation d'une app n'est pas forcément simple et plus elle contient de textes, de fenêtres, plus l'opération sera longue et coûteuse en temps et en budget. Il faut traduire, implémenter la traduction, tester et déployer. Manque de chance pour le développeur, il faut ajuster les objets d'information pour ne pas couper les caractères. Et vérifier en mode portrait et paysage, pour voir si tout passe bien… Sans oublier les écritures qui n'ont pas le même encodage que les langues dites latines.

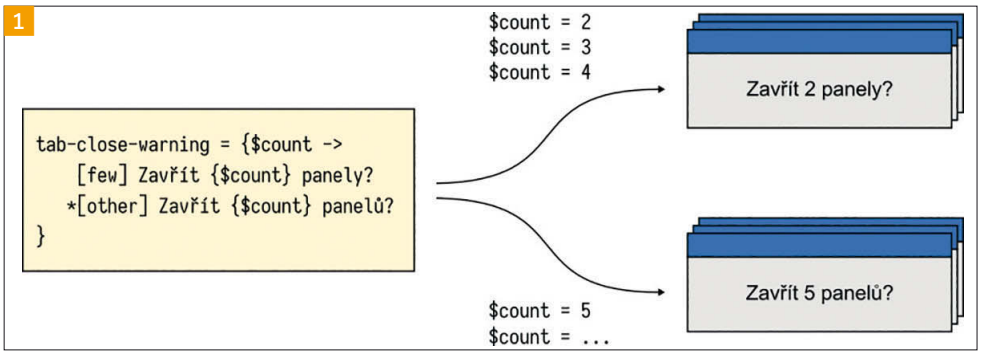
Fluent, donc issu de Mozilla, est un ensemble de pratiques de localisation. Il s'agit d'un format de fichier contenant les éléments localisés. Une des idées est d'avoir les éléments dans un format et un fichier. Ainsi, le développeur n'a pas à maintenir, dans le projet proprement dit la localisation. Il s'agit aussi de gérer les particularités des langues sur les formes grammaticales, les différents pluriels, etc.

Mozilla n'est pas parti de 0 pour créer le format Fluent. Une partie du travail s'inspire de XLIFF et MessageFormat.

Le post de l'équipe donne l'exemple du terme « tab » (onglet ») dont l'orthographe varie selon les valeurs de count : 1

En tchèque, le pluriel varie selon le nombre.

1



Par exemple, pour les valeurs 2, 3 et 4, il faut utiliser une forme plurielle particulière du mot… Et on peut multiplier les exemples. Imaginez de devoir faire la différence entre AM et PM ou bonjour, bonsoir (selon l'heure), etc. Mais au final, cela ne doit pas avoir d'impact pour le développeur, ou alors en limitant celui-ci.

Pour exprimer cette variabilité, Mozilla utilise le concept de localisation asymétrique.

Un fichier Fluent va posséder des nombreux messages pour chaque traduction et langue, avec les asymétries évoquées plus haut. Il est possible qu'un message fasse référence à un autre message dans le même fichier ou dans un autre fichier. Fluent est capable d'agréger l'ensemble des fichiers en un bundle.

Mozilla précise que si la localisation est simple, le fichier sera simple et veut éviter toute complexité inutile.

```
sync-signedout-caption = Take Your Web With You
sync-signedout-caption = Il tuo Web, sempre con te
sync-signedout-caption = Zabierz swoją sieć ze sobą
sync-signedout-caption = So haben Sie das Web überall dabei.
```

La version 1.0 du format supporte JavaScript, Python et Rust.

A voir si des éditeurs et des communautés vont adopter Fluent, en complément à leur mécanisme de localisation ou à la place de ce qui existe déjà. A voir comment Fluent va évoluer et si cette initiative pourra convaincre en dehors de Mozilla.

Page officielle : https://projectfluent.org
GitHub : https://github.com/projectfluent/fluent/wiki

Localisation ou internationalisation : ne pas confondre

Souvent, on utilise les deux termes, sans réellement les différencier. Si nous voulons être rigoureux, il faudrait en effet les différencier :

- Localisation : localiser une app, un document, un produit à la langue, aux usages du pays, d'une région donnée ;
- Internationalisation : designer et concevoir une production, un document de manière globale en incluant l'ensemble des localisations.

Bref, internationaliser chapote l'ensemble des localisations. Dans la notion d'internationalisation, nous allons séparer les ressources localisées du code, du contenu de son application. Eviter d'injecter une localisation directement dans le code. Plus vous rendez indépendantes les ressources de votre internationalisation, plus vous serez souple pour adapter et injecter de nouvelles localisations. Dans la localisation, vous allez pouvoir définir les formats monétaires, date / heure, le format numérique, le clavier, les symboles, etc.

Attention : localisation est ≠ de traduction. La traduction traduit littéralement les termes sans se soucier du contexte local. Eviter Google Translate pour les localisations complexes ou subtiles. La localisation des formats date/heure/monnaie/nombre est parfois un casse-tête. N'oubliez pas le sens de lecture de certaines langues ou les problèmes d'encodages. Eh oui, tout le monde n'utilise pas une langue latine ou le klingeon (ghay cha). Structurez bien vos fichiers de localisation, évitez de tout mélanger.

Si vous utilisez des icônes, des idéogrammes, là encore : attention aux contextes locaux. Car le sens peut changer…

Rebecca Fitzhugh,
Principal Technologist de Rubrik

“Aujourd’hui, le cloud a plus que jamais besoin de l’Open Source”

Rebecca Fitzhugh est Principal Technologist de Rubrik. Rebecca a fait ses armes dans l’armée américaine puis en tant que consultante indépendante. Elle a à cette occasion développé une solide connaissance du monde du cloud mais aussi autour des API. Chez Rubrik, elle dirige le développement technologique des solutions ainsi que la stratégie Open Source de l’éditeur.

Programmez!. Peux-tu me décrire ton parcours en quelques mots ? Comment es-tu tombée dans l’open source, l’API ?

RF. J’ai passé beaucoup de temps à travailler sur les solutions utilisateurs dans l’administration américaine, notamment au ministère de la Défense, d’abord en interne, puis en tant que consultante. Dans le monde de la défense, nous travaillions souvent avec de vieux systèmes, très fermés et pas du tout API friendly, ce qui était très frustrant pour moi. En parallèle de cette activité, je n’ai donc cessé de me former sur l’Open Source et les API pour enrichir mes compétences et trouver une certaine complétude.

Programmez!. Quel est aujourd’hui ton rôle chez Rubrik ? Es-tu comme Gandalf qui guide et soutient la Communauté ou plus comme Darth Vader qui tient fermement ses troupes ?

RF. Je suis aujourd’hui Principal Technologist chez Rubrik. À ce poste je gère le développement technologique des solutions, mais je veille aussi sur la mise en application de la stratégie Open Source. Dans la façon dont je tiens ce rôle, j’aime à penser que je suis plus Gandalf que Darth Vador. Je fais en sorte de toujours être disponible pour aider mes équipes et les soutenir sur les divers projets que nous engageons.

Programmez!. Pourquoi parles-tu d’une approche « API first » quand tu évoques Rubrik Build ?

RF. Les solutions étant pensées pour fonctionner comme une plateforme, elles sont toutes construites, et reliées via des API. Étant donné que nous avons toujours souhaité faciliter la prise en main de nos solutions de Data Management par nos clients - et leur donner la possibilité de les intégrer directement à leurs processus - il était logique pour nous de leur fournir des API. Toutefois, ces derniers temps, le nombre d’API et de langages à utiliser a considérablement augmenté, c’est pourquoi nous avons lancé Rubrik Build. En s’appuyant sur notre programme, les développeurs peuvent accéder à nos API et mettre en place de nouvelles intégrations, voire détourner le fonctionnement de nos solutions pour répondre à différents besoins que nous n’eussions pas forcément anticipés. Par exemple, nous en avons vu certains construire des portails PHP entier à l’aide de nos API, et d’autres utiliser notre solution Radar, à la base prévue pour détecter les ransomwares, pour traquer toutes sortes d’événements sur leur infrastructure. Tout ceci ne serait pas possible sans une vraie approche « API first ».

Programmez!. On parle beaucoup de la proximité de l’Open Source et du Cloud. Finalement, en quoi consiste-t-elle ?

RF. Aujourd’hui, nous connaissons tous les avantages du cloud, notamment public : sa flexibilité, son agilité, la possibilité de provisionner très rapidement des ressources, etc. Toutefois, les utilisateurs demandent aujourd’hui plus d’intégration et surtout une automatisation accrue du cloud public. C’est là qu’intervient l’Open Source. Et pour répondre aux attentes des développeurs et des utilisateurs, le cloud en a plus que jamais besoin.

Programmez!. Le cloud est-il forcément open source ou l’open source va-t-il se diluer dans le cloud ? Car aujourd’hui, le cloud est une nouvelle forme de dépendance technique non ?

RF. Pour moi, ce n’est aucun des deux. Cloud et Open Source peuvent très bien fonctionner ensemble ou séparément. Après, il est vrai que les acteurs du cloud ont eu tendance à racheter beaucoup de projets Open Source, justement pour renforcer l’ouverture de leurs plateformes. Tout dépend finalement des stratégies qu’ils adoptent, mais ils ont aujourd’hui clairement besoin de l’Open Source. Ceci, d’autant plus que les utilisateurs exigent toujours plus de flexibilité et d’ouverture de la part des acteurs du cloud. On ne peut pas espérer construire efficacement des outils pour le cloud sans un minimum d’ouverture, et donc d’Open Source.

Programmez!. On parle aussi beaucoup d’Infrastructure as-a-Code, de microservices, de serverless/stateless, etc. Qu’en penses-tu ? N’est-il pas dangereux de trop croiser les flux.

RF. Tout va dépendre des cas d’usage des utilisateurs, de ce qu’ils veulent adopter et de leur maturité. Il est clair qu’aujourd’hui, nous pouvons faire beaucoup de choses. Par exemple, puisque nous parlions plus tôt d’automatisation. Aujourd’hui, on peut presque tout automatiser au niveau de certains workflows pour se retrouver avec des infrastructures complètement serverless et stateless. Les utilisateurs peuvent intégrer de très nombreuses briques à leurs systèmes pour adopter ces technologies, mais attention, tous ne peuvent pas y arriver et encore moins espérer le faire d’un seul coup. •

RÉPONSES FLASH

- *Python ou JavaScript ?*

Python

- *Mr robot ou Halt and catch fire*

Halt and Catch Fire

- *GCC ou LLVM*

GCC

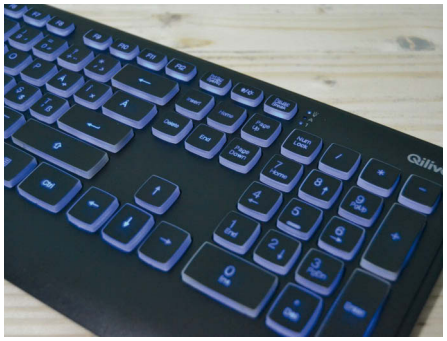
- *Quantique ou Code Quantum*

Quantique

François Tonic

Un clavier idéal ?

Voilà une question qui peut sembler anodine mais qui est cependant intéressante. On distingue généralement trois types de clavier : mécanique, à chiclets et à membranes. Et ensuite, c'est l'ergonomie qui peut jouer. Finalement, le clavier c'est aussi un choix personnel.



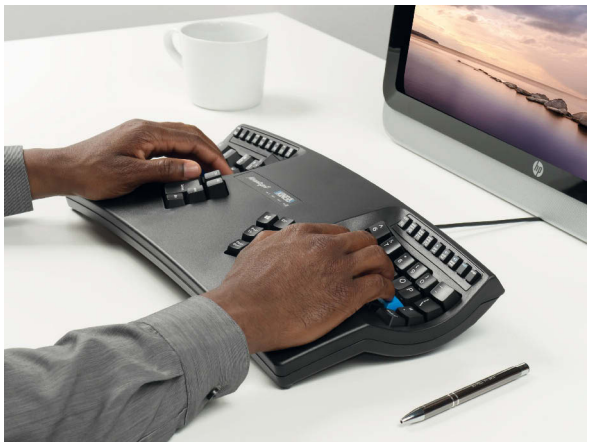
Nous avons connu à peu près tous les types de claviers depuis que nous sommes accroc à l'ordinateur, soit 36 ans. Nous avons utilisé les affreux claviers à membranes du premiers ZX, les touches caoutchouc plus ou moins rigide et les claviers mécaniques. Depuis presque 3 ans ce sont les claviers des MacBook qui montrent les limites de la finesse.

Clavier type chiclet

Ce clavier est reconnaissable par ses touches plates qui ressortent assez peu de la caisse. Elles possèdent des espaces entre chacune d'elles. Ce sont des claviers plutôt silencieux. Nous sommes un peu mitigés sur ces claviers :

- La course des touches est parfois très courte ce qui surprend (et pas dans le bon sens)
- Selon le mécanisme de chiclet, on aura une frappe molle ou franche et donc plus agréable.
- Faire attention aux impuretés qui peuvent s'y infiltrer

Sur les claviers desktops, on notera de nettes différences entre les générations. Nous l'avons expérimenté avec plusieurs claviers desktop Apple : le modèle 2018 n'est pas très agréable à la frappe (idem pour le MacBook Air 2018), alors que le modèle plus ancien a une frappe plus franche. Et attention à la réactivité de la frappe.



Clavier type membrane

Un grand classique. Pour nous, plus d'inconvénient que d'avantages : vitesse de la frappe faible, longévité du clavier en question (la membrane peut s'abimer assez rapidement). Si vous l'utilisez intensivement, le clavier membrane n'est peut-être pas l'idéal.

Clavier mécanique

C'est notre préféré : la frappe est franche et rapide, une bonne réactivité, robustesse (en principe). Pour un usage intensif ou brusque (type gamer), le clavier mécanique reste le must. Les tarifs sont souvent plus élevés.

Connectivité

Aujourd'hui, le clavier sans fil et sur batterie / pile est partout. Heureusement, vous pouvez opter pour un clavier filaire. Le sans fil est pratique en déplacement ou pour limiter les fils. Nous avons opté pour toutes nos machines pour du filaire. On ne perd pas la connexion (eh oui cela arrive), pas de batteries à recharger (c'est bête mais c'est un défaut !). Là chacun/e aura sa préférence.

Ergonomie

Il existe une multitude de claviers : très plats, avec des supports, touches plus larges, avec ou sans pavé numérique, touches spéciales, rétroéclairés (avec ou plus moins de couleurs).

Il existe aussi des claviers dit ergonomiques. Le clavier présentera plusieurs « pavés » de touches ou avec la possibilité de séparation du clavier en deux morceaux. L'avantage de ces claviers est de pouvoir ajuster l'angle des différentes sections et mieux les orienter par rapport à vos mains. Très pratique pour éviter une trop grande fatigue des mains ou des doigts ou si vous êtes sensible aux douleurs articulaires. Inconvénients : l'encombrement, le prix.

Amokrane : *"au travail j'ai un clavier Advantage2 de Kinesis et chez moi, le Freestyle2 du même constructeur. Les deux sont top et ont fait disparaître mes douleurs aux poignets. Attention, il y a une courbe d'apprentissage pour ces claviers. Autre avantage : ils sont programmables…"*

Les prix sont à la hauteur des ambitions de Kinesis : 350-400 € pour l'Advantage2. Attention : pas de pavé numérique. •

LA RÉVOLUTION DES CLAVIERS : DE L'OLED À L'E-INK

Il y a 14 ans, Art Lebedev voulait révolutionner le clavier avec un clavier où chaque touche était un écran OLED et permettait donc d'être entièrement personnalisable. L'idée était intéressante, mais le résultat fut très décevant : mauvaise frappe avec de la lourdeur des OLED, presque 3 ans entre l'annonce et la disponibilité réelle et un prix de +1250 € ! Ce fut un flop total.

L'idée du clavier 100 % personnalisable n'est pas morte. Au lieu d'écran OLED, plusieurs projets utilisent le e-ink, ou encre numérique. L'avantage de cette technologie est la basse consommation et la visibilité des affichages, tout comme la faible épaisseur. Un projet français avait été présenté au CES 2019 : Nemeio. Dans la même idée, l'Australien Sonder avait dévoilé des prototypes début 2018. Le projet est plus avancé que Nemeio avec l'ouverture des précommandes à 199 $. Des « templates » claviers seront disponibles.

Hybride Cloud, DaaS, IoT et Edge Computing : Nutanix brouille encore plus les limites de l'infrastructure

Luke Skywalker lui-même, en la présence de son interprète, Mark Hamill, présent sur scène, s'est rendu avec 6500 autres personnes à la conférence .Next annuelle de Nutanix. Et la présence d'un adepte de la Force n'avait rien d'anodin tant cette conférence en a été une démonstration de force. Depuis sa création, il y a tout juste 10 ans, Nutanix s'échine à construire une offre qui permettrait de rendre l'infrastructure transparente. Les annonces faites sur scène montrent que l'éditeur qui se présente comme un des leaders du cloud computing d'entreprise est en passe d'y arriver. La plus marquante d'entre elle, Xi Cluster va permettre d'abattre définitivement la barrière, déjà ténue, qui existe entre cloud public et privé en permettant de porter indifféremment des applications entre une infrastructure Nutanix et AWS.

Concrètement, les détenteurs d'un compte AWS vont pouvoir déployer en Bare-metal sur des infrastructures EC2 des clusters Nutanix AOS avec toute la stack de Nutanix, y compris l'hyperviseur AHV. Il ne sera alors pas nécessaire de modifier les applications pour les porter sur le cloud public qui deviendra une simple extension via Virtual Private Cloud, Direct Connect ou un VPN. Interrogé sur la similarité de sa solution avec ce que propose déjà VMware sur le sujet, Nutanix a reconnu que l'idée était bonne mais que la mise en application par son concurrent n'était pas optimale, rappelant qu'avec cette dernière, il était nécessaire de reconstruire les applications pour les porter sur le cloud public. En outre, Xi Cluster ne devrait pas rester exclusif à AWS puisqu'à en croire Sunil Potti, vice-président de Nutanix, la solution sera portée sur les autres plateformes, notamment GCP.

Et si Nutanix porte certaines de ses solutions cloud privé sur le cloud public, pour d'autres, la firme fait l'inverse. La solution de DaaS, Xi Frame qui jusqu'à présent permettait de déployer des bureaux virtuels depuis les clouds publics Azure et AWS sera prochainement disponible sur les solutions on premise de l'éditeur. Avec la dernière mise à jour Xi Frame sur AHV, les clients peuvent désormais délivrer des postes de travail virtuels depuis un cloud privé ou public, et même combiner les deux environnements depuis une console unique.

Dans la droite lignée de ce qu'il avait présenté lors du NEXT On Tour Paris le mois dernier sur l'IoT et le Edge Computing, Nutanix en a également dit plus sur ses solutions en présentant son programme U2, comprenez "Vous aussi" (Voir encadrés) associé à Xi IoT. Ce dernier prévoit ainsi la mise en place d'une politique de partenariat qui va permettre à des éditeurs et développeurs tiers d'ajouter à sa plateforme dédiée au développement et au déploiement de solutions IoT et Edge des fonctions déjà pré-paramétrées pour accélérer la mise en route de solutions dédiées. Elles seront disponibles à travers la plateforme IoT Cloud Library. Xi IoT va également s'armer d'environnement de test et des "virtual edge" pour faciliter le test and dev. À noter que la plateforme est accessible dans une version d'essai pour permettre à tous de s'essayer au développement de solutions IoT.

DEEPOMATIC PARTENAIRE PRIVILÉGIÉ DE NUTANIX AUTOUR DE L'IOT ET DU EDGE COMPUTING

Fort d'un projet mené conjointement pour la mise en place de caisses intelligentes avec Compass, Nutanix et Deepomatic, une startup française spécialisée dans la reconnaissance d'image, ils ont annoncé lors de la conférence .Next la mise en place d'un partenariat autour de la plateforme Xi IoT. Les algorithmes développés par Deepomatic vont en effet rejoindre la librairie de fonctions intégrée à la plateforme de Nutanix dédiée au développement, au déploiement et la gestion du cycle de vie des applications Edge et IoT.

Ainsi les entreprises et les développeurs pourront rapidement mettre en place des cas d'usages exploitant les solutions de reconnaissance d'image de Deepomatic sans passer par de longues phases d'apprentissage et d'entraînement liées aux projets IA. Deepomatic vise ainsi à faciliter le développement de projets d'IA dans des cycles courts. En organisant et en centralisant les données visuelles sur une plateforme unifiée, il est possible d'entraîner et d'évaluer les systèmes de reconnaissance vidéo de manière simple et rapide. Grâce à l'infrastructure Cloud d'entraînement de Deepomatic, les entreprises ont accès à des algorithmes de Deep Learning de pointe sans codage et sans se soucier des problèmes d'évolutivité. Les débutants formeront leur premier modèle d'IA en quelques minutes tandis que les utilisateurs expérimentés auront accès à des options avancées pour développer leur expertise.

QUESTIONS À CHRISTOPHE JAUFFRET,

Solution Architect Specialist de Nutanix

Programmez : Quel est l'objectif derrière le projet U2, dédié à l'IoT, annoncé lors de .Next à Anaheim par Nutanix ?

CJ - Le sous-entendu est clair. Il s'agit de permettre à tout le monde de développer rapidement des IoT et IA, ainsi que leurs déploiements à l'échelle, aussi bien sur le cloud que dans un datacenter et sur le Edge, sans avoir besoin de compétences très poussées. Il va fonctionner de concert avec notre plateforme Xi IoT qui va être accessible gratuitement pour permettre à tous de tester des solutions. À cet effet, nous allons permettre aux utilisateurs de déployer des "virtual edge" sur lesquels il sera possible de tester leurs solutions. Dans la même veine, ils pourront s'appuyer sur l'application Xi IoT Sensor pour utiliser leur smartphone comme un objet connecté et tester différents workloads sur différents types de flux.

P : Quelles sont les fonctions IA et Deep Learning qui seront disponibles via la plateforme Xi IoT ?

CJ - Nous mettons en place une librairie de fonctions qui sera ouverte à la contribution de partenaires. Pour l'instant, nous pouvons déjà citer Deepomatic, une startup française spécialisée dans les algorithmes de reconnaissance d'image et le deep learning. D'autres arriveront bientôt. On peut espérer voir arriver des solutions pour l'analyse des flux, pour la surveillance de sites industriels, des réseaux de neurones prêts à l'emploi, etc. Nous invitons aujourd'hui tous les acteurs à nous rejoindre autour de la solution Xi IoT.

P : Qu'en est-il des technologies supportées par la plateforme pour développer ou travailler sur les fonctions existantes ?

CJ - On peut aujourd'hui intégrer différents frameworks directement à Xi IoT, comme TensorFlow par exemple. Nous apportons par ailleurs un support multi-langage avec notamment Python, Javascript et Golang. Mais l'idée est aussi, et je me répète, de permettre à des personnes non spécialistes de tester, de développer et de déployer rapidement des solutions mêlant IA, machine learning, IoT et Edge Computing.

Cloud & développeur : Impossible de se passer du cloud

Partie 1

Ah le cloud computing ! A la rédaction de Programmez ! nous en parlons depuis 11 ans ! Aujourd'hui, c'est une plateforme référence : infrastructure, stockage, transformation IT, nouveaux projets, plateforme IoT, plateforme de développement et d'exécution, services back-end, etc. Bref, il sert à tout et il est partout.

Le cloud fait partie des outils et des plateformes standard du développeur. De nombreux services back-end sont en réalité des services cloud. Dans le tout public, le cloud s'est imposé, visible ou non : gmail, les services bureautiques, le stockage en ligne, etc.

Il existe de multiples tendances ou questions cloud qu'il faut garder en tête :

- Portabilité : c'est un des enjeux à ne pas négliger. Comment s'assurer que mes actifs (= mes services, mes données, mes codes) peuvent fonctionner sur un cloud B à la place du cloud A ? On peut aussi appeler cela le multi-cloud… Bref : le but est de limiter l'adhérence / dépendance technique. C'est un enjeu très important à ne surtout pas négliger.
- Réversibilité : un point sensible depuis plusieurs années. On parle plutôt données et comment s'assurer que mes données stockées sur des services clouds seront récupérées à tout moment et sans être coincées par une dépendance technique… Mais finalement cela peut aussi concerner les codes, les workloads, les codes d'infrastructures, etc.
- PaaS / CaaS : la notion de plateforme est désormais au rendez-vous. Le PaaS avait été promis dès le départ mais il n'a pas su trouver son marché car les entreprises voulaient migrer l'infrastructure d'où l'écrasante domination du IaaS. Tout naturellement, avec l'émergence des conteneurs et des orchestrateurs, le CaaS pourrait mettre tout le monde d'accord entremêlant IaaS et PaaS. Google avait eu cette vision il y a 3 ans en misant sur Kubernetes. Désormais, c'est une réalité. Le conteneur ne va pas faire disparaître les machines virtuelles (VM) et le IaaS, mais il est là pour dominer l'infrastructure et il promet d'être plus flexible et de permettre de se concentrer sur les piles techniques et non l'OS virtualisé.
- Pérennité des acteurs : c'est bien là un des problèmes du cloud. Quand un fournisseur ferme, quid de mes données, de mes applications ? Même les grands fournisseurs peuvent fermer des services, des API. Prévoyez toujours un plan B (qui fonctionne !) en surveillant attentivement les dépendances techniques / aux services.
- Résilience : oui, bien entendu, la résilience est une notion cruciale. C'est la capacité de l'infrastructure de continuer à exécuter les workloads, à maintenir les services malgré les pannes ou les attaques. Une application critique en panne à cause de serveurs ou tout « simplement » les services de paie ou de gestion des commandes et c'est la paralysie de l'entreprise.

Un énorme marché qui va croître encore et toujours : la revanche du PaaS ?

Pour 2019, les analystes IDC prévoient (marché mondial) :

39 milliards $ pour le marché mondial IaaS

19 milliards $ pour le PaaS

95 milliards $ pour le SaaS

Cela n'empêche pas les fournisseurs de matériels d'infrastructures de continuer à vendre du serveur. Cependant, il faut aller au-delà des chiffres pour comprendre le marché :

1 - oui les serveurs se vendent et une partie servent à héberger les services cloud en entreprise ou chez les fournisseurs cloud ;

2 - il ne faut surtout pas réduire le cloud à la partie IaaS pour juger du dynamisme du marché serveur / infrastructure car aujourd'hui, et surtout demain, le PaaS, le CaaS prendront de plus en plus d'importance, et le IaaS continuera à être important sur le marché Cloud mais sa part devrait logiquement décroître ;

3 - une partie du marché infrastructure n'existe plus car les entreprises utilisent des services SaaS pour les apps / usages logiciels en lien et utilisent des serveurs pour faire la même chose en interne, voire, suppriment ces serveurs.

A croire que le PaaS est (re)devenu un service incontournable comme l'apprend Gartner mi-mai dernier :

- +360 fournisseurs forment le marché du PaaS (bon disons qu'une poignée est réellement visible) ;
- Le marché devrait doubler d'ici 2022.

En réalité, Gartner y met un peu de tout : plateforme low code, tous les services d'intégration, le serverless, etc.

Il existe de nombreux autres fournisseurs. Nous n'avons mis que les principaux. Cette hégémonie ne devrait pas s'arrêter, bien au contraire, il est probable que les géants des technologies vont continuer à accentuer leur poids dans les marchés du cloud qui sont désormais leurs sources de revenus. En 10 ans, nous sommes passés d'une informatique matérielle avec des disquettes, des CD, des licences à une informatique immatérielle. Nous avons eu exactement la même chose avec les logiciels et notre manière de la consommer

Les principaux fournisseurs

Plateforme	Fournisseur	CA en milliards $ pour 2018	Début 2019
Azure	Microsoft	+10(1)	+73% sur la croissance d'Azure par rapport à début 2018
AWS	Amazon	29-30	+ 41 % annoncés. AWS pèse 13 % du CA d'Amazon et 50 % des bénéfices !
IBM Cloud	IBM	12-13	Surveiller l'intégration de Red Hat
Cloud Platform	Google	+4	GCP doit continuer à grossir pour pouvoir prétendre concurrencer Azure et AWS
Alibaba Cloud	Alibaba	-4	Le défi est de croître en dehors de l'Asie
Oracle Cloud	Oracle	+26(2)	?

(1) Pas de montant compatible spécifique à Azure. Estimation

(2) Montant global incluant les services cloud, le support, les licences

(merci à l'App Store d'Apple qui a été le déclencheur). Bien entendu, nous ne sommes pas à 100 % de services cloud mais le marché est désormais sur le cloud : que l'on soit d'accord ou pas.

Vive le cloud hybride !

Le cloud hybride n'est pas une nouveauté. Sous ce terme, nous trouvons différentes réalités :

- Débordement de ressources vers des services cloud : pour étendre les capacités de ses infrastructures, par exemple pour du calcul sur quelques heures ;
- Avoir une architecture à la fois sur site et sur le cloud ;
- Dans une application, avoir plusieurs types de ressources.

Aujourd'hui, quand on évoque le cloud hybride, on parlera souvent : de plateformes cloud que l'on déploie sur site pour héberger des applications / infrastructures cloudisées. Ces plateformes sont identiques (ou plus ou moins) aux clouds publics. Par exemple, cloud foundry fonctionne à la fois sur un cloud public, privé, chez un hébergeur ou chez soi.

Microsoft a lancé Azure Stack qui propose une partie d'Azure sur des serveurs (souvent packagées pour Azure Stack). AWS propose, pour le moment en bêta, de son côté Outposts. On disposera de RDS, ECS, etc. Google a dévoilé Anthos. Cette solution est officiellement disponible depuis avril seulement. Anthos repose sur une architecture conteneur pour faire tourner les services Google Cloud Platform.

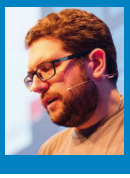
Dans ces trois solutions provenant des géants du cloud public, le matériel n'est pas un simple serveur que l'on réinstalle. Les serveurs / blaies / blades / appliances doivent être compatibles et certifiés, garantissant le (bon ?) fonctionnement du cloud.

Et attention aussi au modèle économique :

- Azure Stack : on paie les ressources utilisées à l'heure ou à la ressource consommée ;
- Anthos : 10 000 $ / mois pour 100 vCPU (contrat d'un an) ;
- Outposts : on attend toujours les tarifs.

Dans ce dossier, nous allons vous parler de nombreux sujets : stratégiques et techniques. Si vous avez des questions, des précisions, n'hésitez pas à les poser, nos experts pourront sans doute vous aider ou vous donner des pistes : ftonic@programmez.com

Ludovic Piot (@lpiot)
resp. offre DevOps & Cloud
@ SOAT



Adrien Blind (@adrienblind)
dataOps evangelist
@ Saagie

Le choix du Cloud, passée la hype…

Voilà un peu plus de 10 ans, nos 2 géants du Web Amazon et Google mettaient sur le marché respectivement Simple Storage Service, premier service de stockage objet massif en ligne en 2007 et App Engine, une plateforme managée pour faire tourner votre applicatif Web écrit en Python ou en Java en 2008. Depuis, la hype, les espoirs fous, les désillusions cuisantes ont vécu. Il est temps de faire l'état des lieux de ce qu'est le Cloud, de ce qu'il sait réellement faire, dans quel cas choisir ses services et sur quels points il faut rester tout particulièrement vigilant.

niveau 100

Le cycle de la hype : des fantasmes irrationnels à la tendance de fond du marché

D'après Gartner, l'adoption de toute nouvelle technologie suit un *cycle de la hype*(1), parcourant tour à tour 5 états irrationnels mus par les émotions des technophiles.

Vu le nombre de fantasmes croisant dans son sillage encore maintenant, plus de 10 ans après son émergence, vous imaginez bien que le *Cloud* ne fait pas exception ! 1

Il y a le **pic des attentes exagérées** : performances, économies, sécurité accrue, résilience renforcée, maintenance réduite quasi *auto-magique*, le *Cloud* comme réponse à tous les besoins !

En miroir, des craintes tout aussi fantasmatiques : violation des données privées, espionnage industriel de la part des *Clouds providers*, gouvernance territoriale des données, boîte noire concernant la sécurité, *lock-in* du vendeur, disruption des périmètres de responsabilité entre *dev* et *ops*, voire… disparition des *ops*.

Un périple plein d'embûches où soufflent le chaud et le froid.

Mais après 10 ans de pratique, la **pente de l'illumination** apparaît enfin. L'adoption du *Cloud* par les services *IT* est une tendance de fond, qui connaît une croissance explosive, faisant la part belle aux services managés(2), alors que la part de marché du *IaaS*(3) commence déjà à se tasser.

Et depuis 2 ans maintenant, les services *Cloud* sont utilisés majoritairement en production par les entreprises. Le stade du POC est dépassé, les *early adopters* ont montré la voie. Place au **plateau de la productivité**.

Lancement de la technologie

Fin des années 2000, les usages *Internet* explosent. De nouveaux champions bouleversent la façon d'appréhender l'*IT*. Pensez ! GMail offre une quantité de stockage 10 à 50 fois supérieure à ma messagerie d'entreprise, une vélocité sans précédent malgré des millions d'utilisateurs autour du globe. Et pendant des années, sa version *beta* n'a connu aucune interruption de service, asseyant pour longtemps la réputation d'un Google indestructible.

A côté, une petite société fait la nique à la Redoute et aux 3 Suisses, avec son site de e-commerce ultra-réactif, avec un moteur de recommandation jamais vu !

Et non seulement, ces géants du *Web* affichent-ils une qualité opérationnelle inimaginable à l'époque, mais en plus ils partagent leur savoir-faire dans des publications offertes à la communauté. Et à partir de 2007, ils ouvrent l'usage de leurs plateformes opérationnelles au public, qui, de communauté, devient clientèle.

SoLoMo

Et ça tombe à pic, parce qu'au même moment, l'iPhone, puis Android arrivent sur le marché, et avec eux, une nouvelle ère de consommation des services Internet, qu'on peut résumer par cet acronyme : **SoLoMo**.

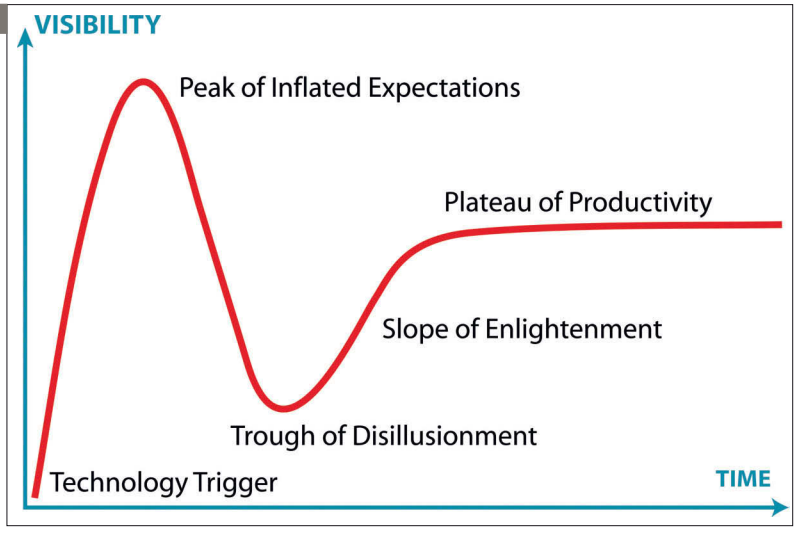
Social, d'abord, parce que les usages se font avec un réseau de relations qui se tisse de manière exponentielle. Et avec qui on partage ses états d'âme, ses photos, ses trouvailles. On passe d'un utilisateur consommateur d'Internet, à un utilisateur produisant et partageant des données. Avec ce que cela implique comme enjeu de modélisation, de stockage, d'indexation et de restitution de volumes de données jamais atteints. **Local**, ensuite, car avec son *GPS*, le smartphone fait exploser les usages liés aux données géolocalisées avec des communautés transcontinentales et des adaptations aux règles de chaque zone géographique.

(1) https://fr.wikipedia.org/wiki/Cycle_du_hype

(2) *Services managés = xPaaS, CaaS, FaaS*

(3) *IaaS = Infrastructure as a Service (autrement dit, VMs et réseau dans le Cloud)*

1



Mobile, enfin, parce que ce petit terminal utilisé en mobilité implique de l'instantanéité, des applications qui répondent en temps réel malgré un réseau de télécommunication peu fiable.

Les principales caractéristiques du Cloud

Une élasticité à toute épreuve, activable à la demande, voire auto-adaptative selon les sollicitations, avec un coût resserré, qui colle au plus près à l'usage réel. Une plateforme opérée dans les règles de l'art, performante, résiliente, accessible de partout, et que l'on peut adapter aux évolutions rapides de l'application, qui suit avec agilité les usages de ses utilisateurs. Voilà de quoi ont besoin les développeurs d'applications mobiles.

Attendez ! *Elastic*, *on-demand*, *pay-as-you-go*, *agile*, *available everywhere*. Ce sont exactement les promesses du *Cloud* public dans sa définition la plus stricte, d'ailleurs normalisée par le **NIST**(4) dès 2011.

Ainsi, moins de 2 ans après l'arrivée de ses premiers services sur le marché, le *Cloud* public devient la plateforme *back-end* naturelle du *smartphone*.

Venir bousculer le statu quo imposé par les services d'IT internes

Dans les entreprises aux besoins plus classiques, le *Cloud* public, c'est surtout un *sparing partner* inespéré pour casser enfin les genoux du *statu quo* qui dure depuis les années 80 entre les études et la production. Ainsi donc il faut 3 mois pour obtenir un nouveau serveur ? Qu'à cela ne tienne ! L'inconnu d'en face (un certain *Werner Vogel*(5)) m'en fournit un en 5 minutes, montre en main. Et pour quelques € par mois. Que ce soit dans son efficience opérationnelle, économique, dans ses choix technologiques et dans le catalogue de services qu'il propose. Le service de production est terriblement challengé par des initiatives d'émancipation des projets qui se montent en *green field* : en dehors de tout respect des normes, standards (et contraintes) imposés par les *Ops*. En cette même année 2009 qui voit naître le *BYOD*, *Chris Anderson*(6) signe une tribune prophétique dans Wired intitulée "*the Black Wire and the White Wire*". Ou comment les services *IT* des entreprises ont désormais comme principal concurrent… un simple câble de connexion à Internet !

Les premières attentes

Pourtant, alors que le *Cloud* public semble (sur le papier) en parfaite adéquation avec les besoins et les usages de son époque, les premiers services *Cloud* démarrent timidement.

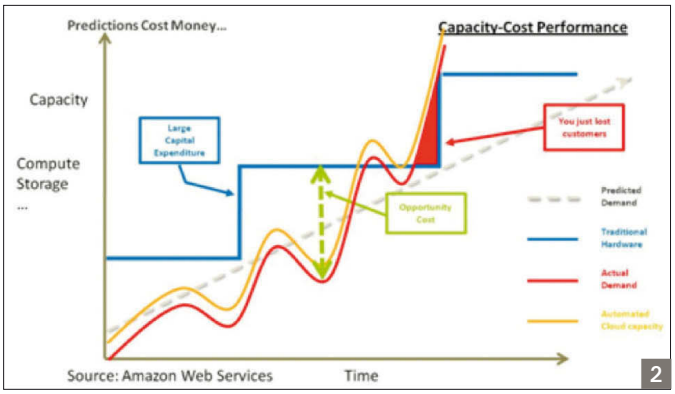
Certes, `AWS S3` affiche des records de résilience avec onze "9" de durabilité de la donnée(7). Mais des opérations aussi simples que la lecture et l'écriture d'un fichier nécessitent une adaptation sérieuse du code applicatif passant d'un `fopen()` à la consommation d'une *API Web*.

Côté *compute*, c'est encore pire ! `Google App Engine` est un concentré du *cycle de la hype* à lui tout seul. Véritable pionnier du *serverless*, il offre une plateforme de *run* applicatif *Web* dans laquelle l'infrastructure est complètement masquée et managée. Elle s'adapte à la charge, gère elle-même les déploiements, les sauvegardes et ajuste les coûts. Mais le catalogue de *runtimes* disponibles est très limité (`python`, `java`), les bases de données disponibles sont des technologies propriétaires : `BigTable` est l'ancêtre de `HBase`, autant dire qu'on est bien loin des traditionnels *SGBD-R* et que ça fleure bon le *vendor lock-in* !

Quant au prix, le *pay-as-you-go* éveille immédiatement la méfiance des services *IT*, incapables de construire un budget sur un modèle de tarification bien trop avancé. En 2009, bien peu d'*IT* ont un *monitoring* assez mature pour savoir faire des projections de nombre de requêtes par seconde, de durée de *compute* d'une fonction applicative ou de bande passante sortante par objet requêté.

Le pic des attentes exagérées

Le *Cloud* public démarre réellement avec `AWS EC2` : des instances de machines virtuelles volatiles, du disque *NAS*, un *load-balancer* basique. La proposition d'AWS est infiniment moins ambitieuse que celle de Google. Mais bien plus facile à comprendre et à appréhender pour le commun des mortels : on est comme à la maison.



Seule différence (mais de taille !), une logistique d'excellence(8) avec des *VM*s mises à disposition en quelques dizaines de secondes.

Et un discours *marketing* qui met l'accent sur les avantages apportés par ce *provisionning* quasi-instantané et automatisable en réaction de certains évènements détectés par la plateforme : l'*auto-scaling*. Couplé au *pay-as-you-go*, on a là la plateforme idéale : auto-adaptative, constamment optimisée par *auto-magie* pour offrir la meilleure performance à l'utilisateur, tout en limitant les coûts et les efforts d'administration au strict nécessaire. Tout cela en se reposant sur le savoir-faire des géants du Web à nous fournir une plateforme incassable.

Oui mais…

Le gouffre des désillusions

Les qualités des services de *Cloud* public sont bien là, effectivement. Mais pour en obtenir les bénéfices, encore faut-il comprendre comment tirer parti du potentiel de ces caractéristiques nouvelles.

Prenons le *pay-as-you-go*, par exemple. 2

Bien sûr, on ne paye que ce que l'on consomme, c'est à dire ce que l'on **provisionne**. Mais justement : il peut y avoir un écart énorme entre ce que l'on provisionne et ce dont on a réellement besoin.

Bien sûr, on peut tirer des économies substantielles des environnements de test

(4) https://csrc.nist.gov/publications/detail/sp/800-145/final

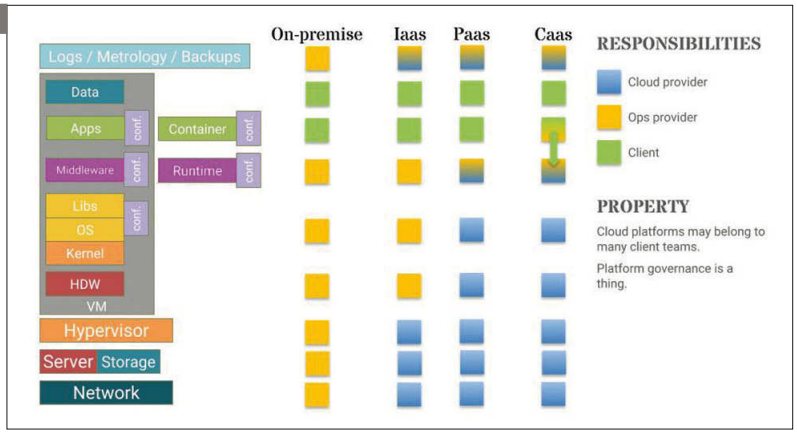
(5) *Le CTO d'AWS.*

(6) *Il est l'auteur bien connu du livre Free, dont est issue la célèbre phrase : "Si c'est gratuit, c'est toi le produit".*

(7) https://aws.amazon.com/fr/s3/faqs/

(8) *finalement n'est-ce pas là l'essentiel du savoir-faire d'Amazon ?*

3



utilisés seulement quelques heures par *sprint*. Encore faut-il mettre en place les mécanismes de destruction automatique des ressources lorsque l'environnement n'est plus utilisé, et automatiser sa reconstruction *ex nihilo* lorsqu'il est à nouveau nécessaire.

Autrement dit, si on applique au *Cloud* les mêmes règles d'usage qui prévalaient dans le monde *on-premise*, alors les économies seront difficiles à trouver, loin s'en faut !

De même, l'*auto-scaling* n'a rien d'*auto-magique*, car si la plomberie est grandement masquée et facilitée par la plateforme, il reste l'essentiel à produire, à savoir les scripts d'automatisation qui vont régir les adaptations que la plateforme déclenche pour réagir à tel ou tel évènement survenu.

Dans son enthousiasme irrationnel du *pic de la hype*, notre enthousiaste du *Cloud* a retenu que la plateforme offrait la meilleure résilience possible. Et compris que c'était la plateforme qui apportait cette perspective fabuleuse. Voilà un fâcheux contresens !

Car dans les faits, les *SLA*s des services d'*IaaS* sont inférieurs à ceux du premier hébergeur traditionnel[9].

La résilience, c'est à l'utilisateur des ressources *IaaS* de l'apporter dans le *design* de l'architecture applicative en y incluant des *patterns* d'architecture *software* distribuée et redondée : la fameuse application *cloud-native*. La *hype* de l'*auto-magie* a bien vécu ! Longue vie au *Cloud*, le vrai ! Car une fois les fantasmes évanouis, il reste encore à adresser les nouvelles problématiques apparues avec ces nouveaux paradigmes du *Cloud* public.

(9) *AWS n'offre aucun SLA concernant ses instances EC2, ni au niveau unitaire, ni même sur un cluster dans une Availability Zone.*

Une matrice des responsabilités à redéfinir

En tout premier lieu, il y a à établir une nouvelle répartition des responsabilités, avec un *Cloud* dont les niveaux de *management* diffèrent selon les services fournis, mais aussi selon l'usage qu'en fait l'utilisateur. Le cas du *patch management* illustre parfaitement la complexité et la subtilité de ce sujet.

En toute logique, sur du *IaaS*, il est du ressort de l'administrateur système, c'est à dire de l'utilisateur du service *Cloud*. Alors que sur du *PaaS*, plateforme managée, la responsabilité devrait incomber au *Cloud provider*. Dans les faits, cette règle logique connaît de subtiles variations.

Ainsi, chez AWS, le *patch management* incombe à AWS dès lors que vous consommez du *IaaS* ou du *PaaS* sur base d'AMIs fournies par Amazon, et que vous vous placez dans un *design* d'*immutable infrastructure*[10]. Dans les autres cas, c'est à vous qu'il incombe. Même si on parle de *PaaS* et de services managés. 3

Une activité qui passe du ticket au pilotage d'APIs

Dans les rangs du client, ça bouge aussi. On l'a vu, le *Cloud* requiert non seulement des efforts d'implémentation et de mise en œuvre, mais tout cela suivant des paradigmes nouveaux et souvent contre-intuitifs par rapport à de l'administration traditionnelle : on ne résout plus des tickets sollicités par ses équipes de *dev*, on crée des automates qui pilotent les *API*s du *Cloud*. Et ces automates ont justement vocation à rendre les équipes autonomes dans leur activité sur le *Cloud*. Si vous vous demandez pourquoi le marché connaît une telle demande de profils *DevOps*, ne cherchez plus la raison. Elle est toute entière là.

(10) https://www.oreilly.com/ideas/an-introduction-to-immutable-infrastructure

Dompter le cheval sauvage

Parce qu'au-delà de revoir les périmètres de responsabilité, l'autonomie des équipes et la fluidité de collaboration entre équipes, nos utilisateurs de services *Cloud* vont devoir apprendre à maîtriser les innombrables services de la plateforme : comment chacun d'eux se comporte, quelles sont leurs limitations et comment articuler les uns avec les autres des dizaines de services différents.

Or ces services sont souvent propriétaires, sans code source accessible, sans accès à l'équipe de développement.

La maîtrise va donc passer par une démarche de *try, fail and learn*. Autrement dit, d'expérimentation empirique à l'aune de laquelle va se forger l'expérience.

Dans un *tweet* de novembre 2018 en marge d'une conférence, un *Cloud Architect AWS* indiquait qu'on ne devait pas estimer le *sizing* optimal d'une `lambda`. Mais qu'il convenait plutôt de monter un banc de tests pour *benchmarker* par l'essai l'ensemble des tailles de `lambda` et ainsi déterminer la taille optimale. Plutôt lourdingue comme approche, quand on sait à quel point `AWS Lambda` se veut être la plateforme de *run* sans friction pour les développeurs pressés et pas intéressés par le *run* !

Design piloté par les coûts

Et cette démarche d'expérimentation, de *refactoring*, de *design* par les tests empiriques, elle s'étend jusqu'à la gouvernance économique. Parce que là encore, les outils d'accompagnement pour comparer efficacement les coûts de deux *designs* possibles pour répondre à un *use-case*, ces outils-là, les *Cloud providers* n'en fournissent guère.

Et pourtant, Google reconnaissait lui-même, lors d'une keynote d'ouverture de sa conférence Google Next que 60 % de l'effort passé à opérer des services dans le *Cloud* portait sur l'optimisation des coûts d'utilisation. Et pas dans de petites proportions : en mai 2018, Tristar Medical Group indiquait que son effort *DevFinOps* avait permis de réduire sa facture AWS de

60 % (soit une économie de $240k par an). *Pay-as-you-go*, oui. Mais les économies promises par l'OPEX vs. CAPEX, elles n'arrivent qu'au terme d'efforts importants à repenser l'architecture technique, voire l'usage de la plateforme en fonction des coûts. Parce qu'à usage égal, un service *IaaS* va coûter entre 2 et 3 fois plus cher que le même service chez un hébergeur traditionnel comme OVH[11].

A tel point que des prestataires de service en ont fait leur business principal : le *cloud brokering*[12]. Sur ce sujet spécifique, je ne peux que vous conseiller l'excellent *talk* de l'ami Quentin Adam[13].

Vendor lock-in et réversibilité

Dernier sujet de désenchantement, c'est le sujet de la **réversibilité**. Et c'est plutôt logique : la promesse du *Cloud provider*, c'est de fournir des services qui, par leur nature, leur efficience, leur pertinence, vont être de formidables accélérateurs pour vos équipes *IT*. S'il fait bien son travail, l'effort pour ressortir de ses services et passer chez un concurrent, ou revenir *on-premise* sera si douloureux, si coûteux, si hasardeux, qu'il en deviendra inenvisageable : c'est le *vendor lock-in*.

La question n'est même plus que le service utilisé est propriétaire ou peu interopérable. Les *Cloud providers* utilisent d'autres leviers aujourd'hui.

Ainsi, ils déploient des efforts considérables pour faciliter au maximum l'import massif des données dans ses services de stockage. AWS fournit même un service de migration physique de vos données depuis vos *datacenters* vers les siens, par l'affrètement de semi-remorques bardés de baies *NAS*[14]. Des *exabytes* de vos données vers AWS. Euh… le service pour ressortir ces données d'AWS pour les rapatrier on-premise ? Il n'existe pas. Merci, au revoir.

C'est ce que l'on appelle le **poids gravitationnel de la donnée** : la donnée coûte cher à transférer, en temps, en contrôle d'intégrité, en coût de bande passante, en complexité à en conserver la cohérence. Dès lors que votre volume de données atteint une masse critique chez un de vos fournisseurs, il devient terriblement improbable que vous envisagiez de quitter ce fournisseur. Quand bien même son service de stockage est hautement interopérable.

Une autre forme de *vendor lock-in*, c'est de proposer un service d'un usage très simple, mais d'une telle complexité sous-jacente, qu'il est impossible de trouver un équivalent sur le marché, ou du moins à benchmarker ses *challengers*. Il est facile de mettre en compétition des VMs AWS EC2 vs. des instances Google Compute Engine. Mais allez comparer des services d'intelligence artificielle ! Comment comparer IBM Watson à… mais à quoi au fait ? Ou bien comparer Spanner à… eh bien… rien.

Enfin, le dernier mode de *vendor lock-in*, passe justement par l'interaction très intriquée de dizaines de services les uns avec les autres. Dès lors, envisager une migration en réversibilité devient un vrai casse-tête et un chantier de grande ampleur, alors même que l'intégration de ces différents services avait paru si simple.

Le plateau de la productivité

Je sens que j'ai plombé l'ambiance dans le gouffre des désillusions.

Voilà 10 ans, l'architecture applicative a influencé la nature des services *Cloud* qui ont émergé. Les architectures micro-services, les architectures distribuées, le *design for failure* ont permis d'envisager l'infrastructure comme une commodité, ce qu'est le *Cloud IaaS*. Et les géants du *Web* nous ont montré le chemin en exposant leurs propres expériences d'architecture technique. Mais ces "grands frères" ne sont plus juste des *mentors*, ils sont devenus nos *dealers*, qui ont des intérêts financiers à ce que l'on reste leurs *junkies* : désormais, ils ont des plateformes à nous vendre.

Ainsi, désormais, le *design* des services *Cloud* influence la manière dont on conçoit les applications : stockage objet, *cloud-native app*, ou encore *serverless* ou *Functions as a Service*.

Il convient de reprendre le contrôle, de préserver son autonomie, de réduire les coûts opérationnels à l'échelle. En favorisant les standards interopérables et les technologies *open source*.

En étudiant l'impact, aussi, de tout choix de se créer une adhérence forte avec un service. Et ce choix peut avoir sa pertinence à un moment donné. Pour accélérer son *time-to-market*, pour limiter son effort de management, pour s'épargner la plomberie pénible et sans intérêt, pour pallier à certains manques de compétences (dans l'emploi d'un service cognitif, par exemple). Et en assumant l'impact business d'une réversibilité compliquée à mettre en œuvre[15].

Envisager de ne pas forcément picorer à tous les râteliers, en restreignant l'usage aux seuls services nécessaires.

Se rendre compte, enfin, que les *Cloud providers* vous conseillent de faire exactement l'inverse de ce qu'ils font eux-même : "*concentrez-vous sur votre cœur de business et délestez-vous de vos problématiques d'infrastructure*". Mais, n'ont-ils pas eux-mêmes une démarche *full-stack* ? Ne font-ils pas eux-mêmes le choix de faire de leurs *assets* technologiques des différenciateurs majeurs dans la façon dont ils opèrent leur business ?

10 ans ont passé, et certains services du *Cloud* n'ont plus autant de pertinence qu'à l'époque. Les technologies ont évolué, ainsi que la maturité globale en architecture technique, et appréciation des pratiques *DevOps*, en adhésion aux méthodes agiles.

Ainsi, aujourd'hui, des bases de données massivement distribuées, des technologies de stockage objet élastique existent et sont relativement simples à mettre en œuvre.

De même, avec l'émergence des orchestrateurs de conteneurs, l'*Infra as Code* et l'orchestration de plateforme ne sont plus les prés carrés des seuls services *IaaS* ou *PaaS*, mais s'opèrent tout aussi bien *on-premise*.

Finalement, avec la maturité, le *Cloud* public est devenu un fournisseur technologique comme un autre. Ni incontournable, ni illégitime.

En un mot, abordez l'utilisation du *Cloud* en ingénieur et pas en *fashionista*.

My2Cents.

---

(11) Retour d'expérience vécu, quoique datant d'il y a quelques années maintenant.

(12) https://lalettreducloud.com/2016/08/17/86-des-entreprises-comptent-sengager-dans-le-cloud-brokering/

(13) Les comptables ont fucked-up la gestion de mon I.T. https://youtu.be/0ip1FoBsLB4

(14) AWS SnowMobile https://aws.amazon.com/fr/snowmobile/

(15) Souvenez-vous de la gueule de bois provoquée par l'augmentation des tarifs Google Maps de 14 000 % en mai 2018

Christopher Maneu
Azure Engineer & Cloud Developer Advocate
chez **Microsoft R&D**
@cmaneu
www.maneu.fr

Le cloud n'a pas d'impact sur mon code !

« Que mon projet soit hébergé sur un cloud ou sur un hébergement classique, cela ne change rien à mon code !». Peut-être avez-vous déjà entendu cette phrase… Voire, peut-être l'avez-vous même déjà prononcée? Pas d'inquiétude ! C'est arrivé aux meilleurs la première fois. Nous allons voir pourquoi cette affirmation est fausse, et surtout ce qu'il faut savoir avant d'envoyer votre code dans les nuages.

De plus en plus de sociétés, d'entreprises, d'administrations et même d'associations déploient des services sur des plateformes cloud. Mais au-delà d'appréhender les SDKs autour de services spécifiques – tels que le stockage de fichiers Azure Blob Storage ou Amazon S3 – il est rare de penser que d'autres adaptations de notre code soit nécessaire. Si c'est vrai pour les applications « Cloud Native » - conçues depuis l'origine pour profiter des avantages du cloud – c'est encore plus vrai pour toutes les applications existantes qui sont migrées. Petit tour d'horizon des considérations de ce mode d'hébergement pour nous, développeurs.

Bien qu'Azure sera pris en exemple ici et là pour illustrer mes propos, les concepts sont valables pour tous les types de clouds – publics ou privés – voire sur d'autres types d'hébergements.

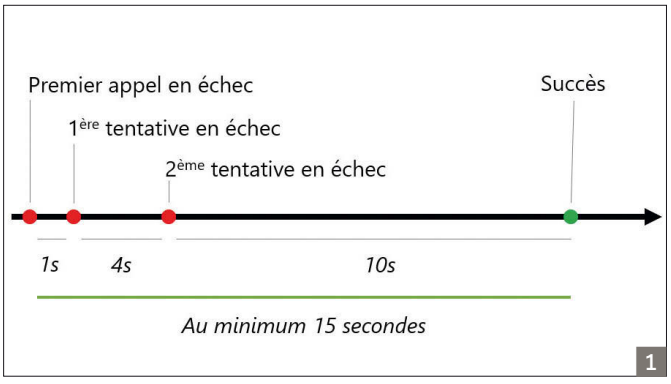
Le cloud n'est pas si « stable » qu'on peut le penser

Commençons par corriger une fausse idée : beaucoup sont persuadés que les services cloud présentent beaucoup moins de coupures et interruptions que les hébergements « classiques » - qu'ils soient à demeure, ou la location de serveur dédiés -. Ce n'est pas exact : bien que la qualité de service – souvent traduite par un Service Level Agreement (SLA) – soit généralement meilleure dans un cloud, c'est surtout le type d'interruptions qui change. Dans le monde cloud, il est rare d'avoir une interruption totale de service pendant une période de temps étendue. Il y a pourtant une chance plus importante d'être confronté à des microcoupures, de l'ordre de la dizaine de millisecondes, avec une certaine fréquence. Cela est dû notamment au fait que de nombreuses ressources sont partagées. Si nous prenons par exemple le service Azure App Service. Il vous permet d'héberger un site web ou une API REST juste en poussant votre code sur un repository Git. Ceci sans vous soucier de la configuration ou maintenance du serveur, le serveur hébergeant votre code étant en réalité une machine virtuelle qui vous est dédiée. Cependant, le load-balancer qui est devant, ou bien le stockage réseau qui héberge votre code, sont deux composants mutualisés. Si un autre client qui partage ces ressources subit un pic de charge, il se peut pendant un court instant que vos ressources soient affectées. L'infrastructure Azure effectuera automatiquement – sur la plupart des services - les actions nécessaires pour résoudre ce problème, mais il se peut que quelques requêtes à votre base de données soient en erreur pendant un court laps de temps. On appelle cela **les erreurs transientes**.

Pour adresser cette problématique, il est nécessaire de modifier votre code, et d'appliquer le pattern Retry dès que vous communiquez avec un service externe (https://aka.ms/programmez229-retry). Le principe est simple : si l'appel à ce service est en échec, et que cet échec peut se résoudre par lui-même, alors on retente l'appel après un petit délai. Si vous avez affaire à une erreur transiente, votre utilisateur ne verra même pas qu'il y a eu une erreur. Pour vous aider à implémenter cela dans vos projets, il y a de nombreuses solutions. Tout d'abord, un bon nombre de SDK Azure intègrent ce pattern (vous avez le détail dans la documentation https://aka.ms/programmez229-retryazure). Ensuite, il existe de nombreuses librairies en fonction de votre langage qui vous apportent les fonctionnalités nécessaires pour l'implémenter ; on peut citer Tenacity en Python ou Polly dans le monde .Net.

Ne vous jetez pas sur votre code immédiatement pour autant! Vous pourriez vous faire plus de mal que de bien ! Si l'erreur n'est que temporaire, et à l'échelle de quelques microsecondes, cela devrait fonctionner. Cependant, si l'erreur est plus longue, ou bien que le nombre de clients soit important, vous devez mettre en place quelques garde-fous. Tout d'abord, vous devez définir un nombre maximum de tentatives, ainsi que l'intervalle entre deux tentatives. Cela est nécessaire pour la santé de vos serveurs – vous ne voulez pas créer une armée de processus qui ne font que tenter une opération impossible – mais aussi pour l'expérience utilisateur – a un moment donné, si cela ne fonctionne pas, il faudra proposer une expérience adéquate. Les règles sont donc à adapter au contexte : un service en tâche de fond pourra avoir plus de tentatives avec des délais importants alors qu'un bouton sur une page de paiement n'aura qu'un nombre de tentatives limitées dans un laps de temps court. 1

Il est également conseillé de ne pas avoir de délai entre les répétitions uniformes, souvent en utilisant un *exponential backoff* ; non seulement le délai entre deux répétitions sera différent – de plus en plus important avec le nombre de répétitions – mais sa valeur sera également différente pour une répétition donnée entre deux processus. Si



1

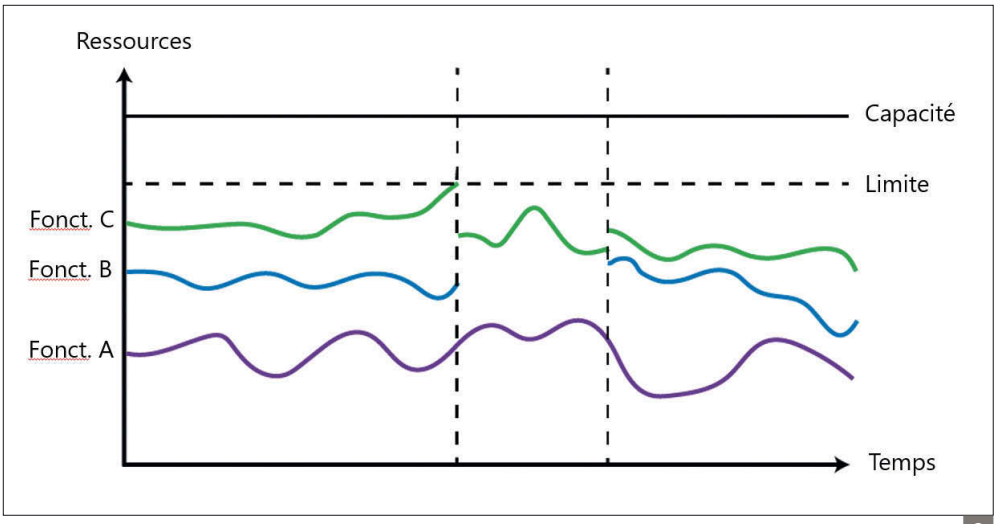
votre base de données est indisponible pendant 30 secondes, un certain nombre de processus clients vont commencer à réessayer de jouer vos requêtes. Lorsque votre serveur est de nouveau opérationnel, il pourrait recevoir un afflux massif de requêtes qui pourrait entraîner un incident. Ajouter cette dose d'aléatoire permet d'éviter la congestion des services et du réseau.

Le cloud peut vous couper l'herbe sous le pied, et ceci pour votre bien

Dans un environnement cloud, il se peut que le serveur vous retourne une erreur alors qu'il a les ressources pour traiter votre requête. C'est souvent le résultat de l'implémentation d'une limitation volontaire appelée *throttling* (https://aka.ms/programmez229-throttling). Pour protéger le système – et donc les autres utilisateurs de la plateforme, ou parfois votre porte-monnaie, le serveur va simplement refuser d'exécuter votre requête si vous avez dépassé votre capacité. Le code http ***429 Too many requests*** est souvent utilisé pour vous indiquer cela, associé avec le header http *Retry-After*, indiquant le nombre de secondes avant que vous puissiez réessayer votre requête.

Cela est bien évidemment important en tant qu'utilisateur d'un service cloud. Par exemple, si vous avez du code qui utilise le stockage de fichier Blob d'Azure, il peut être intéressant d'aller dans la documentation à la recherche des *scalability targets* pour ce service (https://aka.ms/programmez229-scalability-target) : ce sont l'ensemble des limites pour ce service. On y apprend notamment que le nombre maximum de requêtes par compte de stockage est de 20 000 requêtes par seconde. D'autres services ont un nombre de requêtes qui est comptabilisé à la minute. Ces informations peuvent donc vous aider à adapter le code de *retry* en fonction du service concerné.

Si ce pattern est intéressant pour votre code appelant d'autres services, il peut l'être tout autant pour protéger les services que vous développez vous-même. Admettons que vous développiez une API REST à destination d'un autre service, de partenaires ou même de clients. Si vous connaissez la charge maximale supportée par votre infrastructure, vous êtes alors capable de mettre en place cette limite logicielle pour *couper* vos utilisateurs avant d'arriver à ce point.



2

Il existe de nombreuses manières d'implémenter ce *throttling*. Tout comme pour le pattern *retry* , il existe peut-être une librairie qui fait cela pour vous dans votre code : le bundle RateLimiteBundle pour le framework PHP Symfony, ou bien encore de nombreux middlewares pour ASP.Net MVC. Il n'y a cependant pas de miracles : faire cela bien nécessite d'y passer un certain temps, de bien comprendre les règles de limitations que vous souhaitez implémenter et de tester votre implémentation. 2

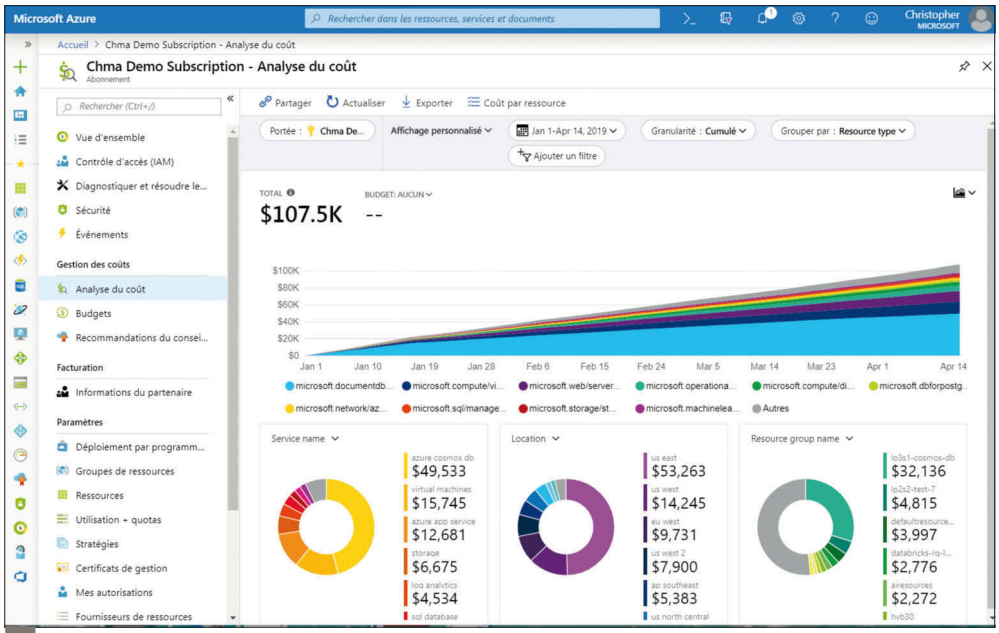
Utiliser ces librairies peut vous permettre d'implémenter des scénarios très complexes : règles différentes par clients, changement de valeurs dynamiques par rapport à la mise à l'échelle de l'infrastructure, etc. Cependant cela nécessitera au moins deux ingrédients : un investissement certain, notamment en termes de code, afin d'implémenter et surtout de tester comme il se doit la mise en place. Ensuite, vous aurez probablement besoin d'un stockage partagé d'information afin d'appliquer cette limite selon vos critères : si vous avez 3 serveurs web et que vous souhaitez limiter le nombre de requêtes par adresse IP d'origine, il faudra un moyen pour ces 3 serveurs de stocker cette information. Ce sont généralement des services tels que MemCache ou Redis qui sont choisis. Cela fait donc des services en plus à gérer.

Dans un certain nombre de cas, et notamment pour les APIs, il existe un autre moyen d'implémenter ces limitations avec moins d'effort : utiliser un service d'API Gateway. Disponible sous la forme de services prêts à l'emploi comme Azure API Management (https://aka.ms/programmez229-apimanagement) ou de projets open source tel qu'Ocelot, ils vous apportent de nombreux services pour gérer l'exposition d'APIs, et notamment l'implémentation de *throttling*, voire l'intégration de *retry policies*.

Votre code vaut/coûte de l'Or !

En tant que développeur, on se soucie rarement du coût d'exécution de notre code. Après tout, tant que les serveurs ont encore un peu de RAM pour le faire tourner, tout ira bien, non ? Or, sur un environnement cloud tout se paye ! Vous avez besoin d'accéder trois fois au même fichier ? Pas de soucis, cela fera trois appels à l'API Blob facturés. **Il est donc de la responsabilité du développeur de cultiver une sensibilité aux coûts d'exécution de ce qu'il écrit**. Vous intégrez le SDK *Custom Vision* (permettant la reconnaissance d'objets sans rien connaître à l'IA) ? Allez donc sur la page des tarifs du service pour comprendre comment celui-ci est facturé : à l'appel, à la capacité pré-réservée, à la minute ou à l'heure, … En fonction du modèle de tarification, vous saurez alors ce que vous devez surveiller, notamment en phase de développement et de recette, et ainsi évaluer si cela a du sens dans votre scénario métier : *combien d'appels à l'API d'un service cloud sont nécessaires pour générer la facture d'une commande sur mon site e-commerce ?*

Cette étape d'analyse de la *consommation cloud* est généralement une première phase avant la réarchitecture d'une application afin de profiter des avantages d'une plateforme Cloud. Certains services rendent



3

possible de mettre en place des architectures qui non seulement vont vous permettre de passer à l'échelle, mais également de réduire vos coûts. L'un des meilleurs exemples publics sur ce sujet est probablement https://haveibeenpwned.com/ du MVP Troy Hunt. Ce site vous permet de savoir non seulement si votre adresse email fait partie d'une des plus importantes brèches de données personnelles, mais également si un mot de passe spécifique est déjà apparu dans une de ces brèches. Au moment où j'écris ces lignes, il y a 7 840 611 051 comptes dans la base, 551 509 767 mots de passes uniques et plus de 100 Millions d'utilisations par mois. Prenez quelques instants pour visiter le site – et observer sa rapidité d'exécution – et essayez de deviner un coût d'exécution par mois. 1000 € ? 10000 € ? Vous êtes loin du compte : moins de 100€ / mois ! Cela est rendu possible, notamment par une grande attention aux choix d'architectures (et de code) avec le biais d'optimiser pour le coût. Ainsi, dans cette architecture, il n'y a quasiment pas de bases de données relationnelles, mais l'usage des blobs et table storage (un stockage de type *clé-valeur*). S'il y a bien de l'hébergement de sites web dans Azure App Service, Troy utilise également massivement des Azure Functions – un des services *serverless* d'Azure – et enfin bien évidemment un système de cache devant ces services.

Evaluer ces opportunités d'optimisation sont donc en partie de la responsabilité du développeur. Bien évidemment sur une équipe de développement importante, d'autres rôles seront concernés : architectes, développeurs data, etc. Mais les développeurs ont entre leurs mains les outils pour réaliser les plus grandes avancées sur ce sujet.

Ce travail ne doit pas s'arrêter aux phases de développement ou de recettes. Analyser ces données de consommation y compris sur les environnements de production est un point important. S'il y a probablement quelqu'un dans votre organisation qui va regarder le montant affiché en bas à droite de votre facture cloud, de précieuses informations peuvent se cacher dans une analyse plus fine du détail de la facture. Par exemple sur Azure, vous pouvez utiliser la fonctionnalité ***Analyse de coût*** pouvant vous donner accès aux données de facturation y compris sans avoir accès aux ressources de production. 3

L'étude de ces données peut vous permettre, par exemple, d'observer un pic de facturation sur un service. Il faudra alors comparer cela aux données de télémétrie que vous avez, afin de voir si cette hausse est légitime, ou si un problème sous-jacent ne serait pas passé inaperçu. Alors en tant que développeur, quand dois-je regarder ces données ? Cela dépend de votre contexte, du nombre de fois qu'un déploiement en production arrive, et de la structure de votre équipe. A vous de voir quel rythme vous convient le mieux. Tous ces outils donnent le pouvoir aux développeurs d'agir sur la consommation, mais il vient également avec une responsabilité : participer à l'effort collectif de maîtrise des coûts. L'accès à ces informations permet donc de sensibiliser les développeurs à cette problématique, et leur donne les moyens de comprendre. Ils seront donc plus autonomes dans leurs actions de réduction des coûts. La fonctionnalité de **budgets** permet également de déclencher des alertes très fines – au niveau d'un groupe de ressources ou d'une ressource spécifique – et ainsi de réagir rapidement.

Il n'y a pas que votre code dans votre code

Il y a fort à parier que votre code se situe dans un repository – probablement dans Git. Là où avant il y avait simplement votre code, voire de la documentation et quelques scripts de bases de données, il y a désormais un nouveau venu : **votre infrastructure**. Il existe en effet de plus en plus d'outils et de services vous permettant de traiter votre infrastructure comme du code et ainsi, d'instancier, mettre à jour ou refactorer les ressources nécessaires à votre service. Ansible, TerraForm ou bien simplement des modèles Azure ARM : tous finissent sous la forme de code dans vos repositories. Tout comme votre code, ils peuvent avoir des tests avec des outils comme Pester. Et enfin, ils peuvent être utilisés dans une chaîne de déploiement continu pour automatiser leur utilisation.

Alors, si c'est de l'infrastructure, à qui *appartient* réellement ce code ? Si la réponse à cette question dépend du contexte et de l'organisation de votre équipe, vous avez probablement une contribution à apporter en tant que développeur. Il n'y a pas de tests ? Ecrivez-en ! Il y a du code en doublon ? Faites-donc du refactoring !

Le cloud impacte le cycle de vie de vos développements

Il y a deux types d'impacts principaux dans le cycle de développement d'applications hébergées sur le cloud : d'une part le cloud vous permet d'accélérer vos développements, de l'autre il peut vous contraindre à aller plus vite.

La disponibilité quasi-immédiate et à la

demande de ressources dans le cloud vous ouvre les portes de nombreux scénarios. La disponibilité de nombreuses offres managées vous permet plus de choix concernant les bases de données ou d'autres services – sachant que vous n'aurez qu'à utiliser et non pas à gérer ces éléments. C'est donc un facteur de vélocité pour votre équipe de développement. Cela s'ajoute bien évidemment aux outils de gestion de cycle de vie qui sont eux aussi hébergés sur un cloud. Si vous avez déjà eu à configurer un serveur de build, combien de temps cela vous a-t-il pris d'avoir accès au serveur puis de le configurer ? Dans Azure DevOps (https://aka.ms/ programmez229-devops), il vous faudra probablement moins de 5 minutes, et avec une build sous Mac, Linux et Windows. L'implémentation de certaines bonnes pratiques est donc bien plus accessible dans ces conditions.

A l'inverse, les clouds évoluent très rapidement, ce qui peut vous imposer de suivre ces évolutions et de mettre à jour votre code plus régulièrement que ce que vous aviez envisagé. Là où les services de base sont conservés pendant une durée longue, de nombreux services en mode PaaS – prêts à l'emploi pour les développeurs – vous laissent parfois seulement 6 mois pour effectuer les changements dans votre code. Pour certaines équipes ce n'est pas un problème, pour d'autres projets, il se peut que l'équipe de développement n'existe plus et ainsi que l'effort de maintenance soit plus important.

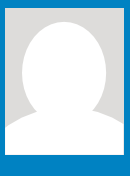
La sécurité de votre service repose (encore plus) sur votre code

Dans un environnement traditionnel, la sécurité est souvent *périmétrique*. Nous allons nous assurer que la base de données est isolée d'un point de vue réseau des frontaux web. Certains services seront peut-être déployés dans une DMZ, et les déploiements effectués via un VPN.

S'il est possible de répliquer ce type d'organisation dans le cloud, le plus généralement cette sécurité périmétrique est plus orientée vers une sécurité par identité. Dans ce cas, une plus forte vigilance sera à apporter à votre code afin, d'une part, d'implémenter la bonne gestion d'identités, mais également, d'autre part, de vous assurer que l'ensemble de vos services -notamment internes- utilisent correctement l'identité de l'utilisateur.

Avec l'explosion du nombre de services auxquels un bout de code peut avoir besoin de communiquer va l'explosion des *secrets* et chaînes de connexion. Cela peut vous imposer l'utilisation d'un coffre-fort de secrets – tel qu'Azure Key Vault – afin de gérer de manière sécurisée leur utilisation, mais également leur renouvellement à intervalle régulièr (cette bonne pratique est appelée *rotation des clés*, et c'est pour cela que plusieurs services dans Azure vous donnent deux clés d'accès par défaut !).

Cet article ne fait qu'aborder quelques-uns des points d'attention que devrait avoir tout développeur qui verra son code être exécuté dans un environnement cloud. Si vous souhaitez aller plus loin sur ce sujet, je vous invite à regarder le guide de l'architecture des applications Azure – bien que de nombreux concepts soient applicables à tous les clouds – sur https://aka.ms/programmez229-architecture.



Alexandre Vasseur
Pivotal

Le cloud pour moderniser les applications ? Est-ce si simple?

Les développeurs d'entreprise sont confrontés à de nombreux choix lorsqu'ils écrivent des applications et logiciels. Quelle est la bonne architecture pour le long terme ? Sur quelles technologies dois-je parier ? Une des décisions les plus difficiles qui s'ajoute avec la réalité de l'adoption du cloud : comment équilibrer la flexibilité de déployer les applications sur n'importe quel cloud public avec les avantages d'utiliser les services à valeur ajoutée du cloud - tous assez différents ?

Vous pouvez tirer d'importants avantages fonctionnels d'un tel service cloud. Par exemple facturation à l'usage, grande capacité, disponibilité, conception multi projet multi utilisateur, maturité des services réseaux, gabarit de taille de machines virtuelles, services de bases de données et NoSQL managées ou d'API machine learning évoluées, hébergement multi site mondial, etc. Toutefois, tous ces choix peuvent rendre difficile le changement d'un fournisseur de cloud à un autre, limitant ainsi la flexibilité. 1 Idéalement, les équipes de développement auraient le choix sans faire de compromis. Les développeurs pourraient alors utiliser n'importe quel cloud service sur leur infrastructure cloud préférée. Et il n'y aurait aucun coût ni complexité pour passer d'un fournisseur de cloud a un autre.

Infrastructure-as-code, Containers, Plateformes Applicatives ou Serverless ?

En utilisant l'Infrastructure-As-A-Code avec par exemple des scripts et outillages bas niveaux tels que Terraform, on a un driver pour chaque cloud, mais aussi de nombreux fichiers yaml ou json tous différents pour chaque cloud.

Exemple de Terraform pour Google Cloud - le fichier n'est pas portable vers un autre cloud et on retrouve un id de VM et de zone de cloud qui fragilise la démarche "as code" et multi cloud…

```
// Static IP address for forwarding rule
resource "google_compute_address" "cf-pks-demo" {
 name = "${var.env_name}-cf-pks-demo"
}
// TCP target pool
resource "google_compute_target_pool" "cf-pks-xyzdemo" {
 name = "${var.env_name}-cf-pks-demo"
 instances = [ "europe-west1-b/vm-5885308e-7e86-4006-4a90-1e52e07b739a" ]
}
```

```
// TCP forwarding rule
resource "google_compute_forwarding_rule" "cf-pks-xyzdemo" {
 name       = "${var.env_name}-cf-pks-demo"
 target     = "${google_compute_target_pool.cf-pks-demo.self_link}"
 port_range = "8443"
 ip_protocol = "TCP"
 ip_address = "${google_compute_address.cf-pks-demo.address}"
}

resource "google_dns_record_set" "wildcard-pks-dns-demo" {
 name = "demo.${var.env_name_pks}.${var.dns_suffix}"
 type = "A"
 ttl  = 300
 managed_zone = "${var.dns_zone_pks}"
 rrdatas = ["${google_compute_address.cf-pks-demo.address}"]
}
```

Un équivalent pour le cloud Amazon AWS aurait une syntaxe complètement différente pour le même cas fonctionnel (règle de TCP load balancer avec inscription DNS) - par exemple le dernier bloc serait :

```
resource "aws_route53_record" "wildcard-pks-dns-demo" {
 zone_id = "${var.dns_zone_pks}"
 name    = "demo.${var.env_name_pks}.${var.dns_suffix}"
 type    = "A"
 ttl     = "300"
 records = ["${aws_eip.cf-pks-demo.public_ip}"]
}
```

En utilisant les **containers**, on arrive à une plus grande portabilité pour un container via le mécanisme de démon exécutant le container combiné au mécanisme de packaging de format de container, et aussi à l'aspect programmatique de la Dockerfile (voir exemple

1

ci-après). Cependant, très vite, l'architecture devient spécifique pour les aspect réseau, persistance, monitoring et accès aux services du cloud.

Le rythme d'innovation des fournisseurs de cloud dans le **PaaS** (Platform As A Service), le **CaaS** (Container As A Service) et le SaaS ou FaaS sont remarquables. Les nouvelles technologies dites *de plateformes cloud applicatives et containers* telles que **Kubernetes**, **Cloud Foundry**, et Heroku, GoogleAppEngine ou Azure Service Fabric representent les changements de génération en informatique d'entreprise.

Mais alors comment concevoir son application pour en tirer parti tout en gardant contrôle et flexibilité ? 2

Moderniser les applications avec la flexibilité du cloud - Choisissez la bonne approche

Avant de commencer l'effort de modernisation de vos applications, réfléchissez a la manière d'équilibrer les gains pour les déploiements et opérations ("jour 2") et les gains en efficacité de développement et organisation des developpeurs (productivité, qualité, communauté très large, robustesse de l'architecture). La figure 3 présente une série d'options disponibles pour moderniser votre application - et la perception va dépendre du point de vue.

En général, les options à l'extrême droite du tableau par l'introduction d'une plateforme offrent aux développeurs les plus grands gains d'efficacité et sont parfaites et simples pour la scalabilité dynamique et le déploiement continu (CI/CD). Cependant, elles nécessitent également des modifications de code, des processus de livraisons, ou d'architecture plus invasives. Les options sur la gauche sont moins invasives, mais offrent moins d'avantages aux développeurs car elles portent encore beaucoup de complexité ou d'étapes manuelles notamment sur les dimensions réseau, clustering, placement multi zone et automatisation à l'échelle.

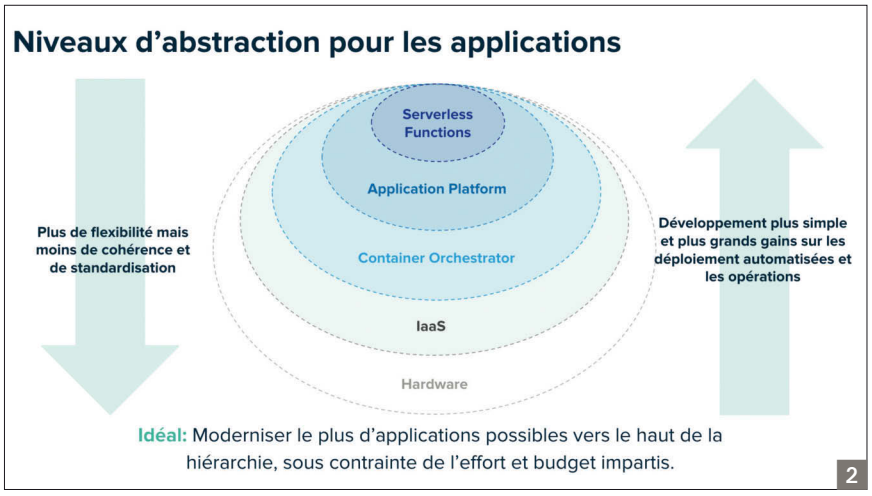
Dans la plupart des portefeuilles d'applications d'entreprise, la plupart sinon la totalite des niveaux va être nécessaire car il n'y a pas qu'une seule application. Il est souvent recommandé de faire une analyse de portefeuille d'applications. Certaines de ces applications ne sont pas stateless, ne sont pas des API ou microservices, ne scalerait pas horizontalement, ne sont pas du code développé mais parfois du progiciel configuré. Le périmètre des applications modernes ou aussi appelé "cloud native" (s'exécutant idéalement dans tout cloud) s'oppose alors au périmètre plus classique des applications "existantes" de l'entreprise. 4

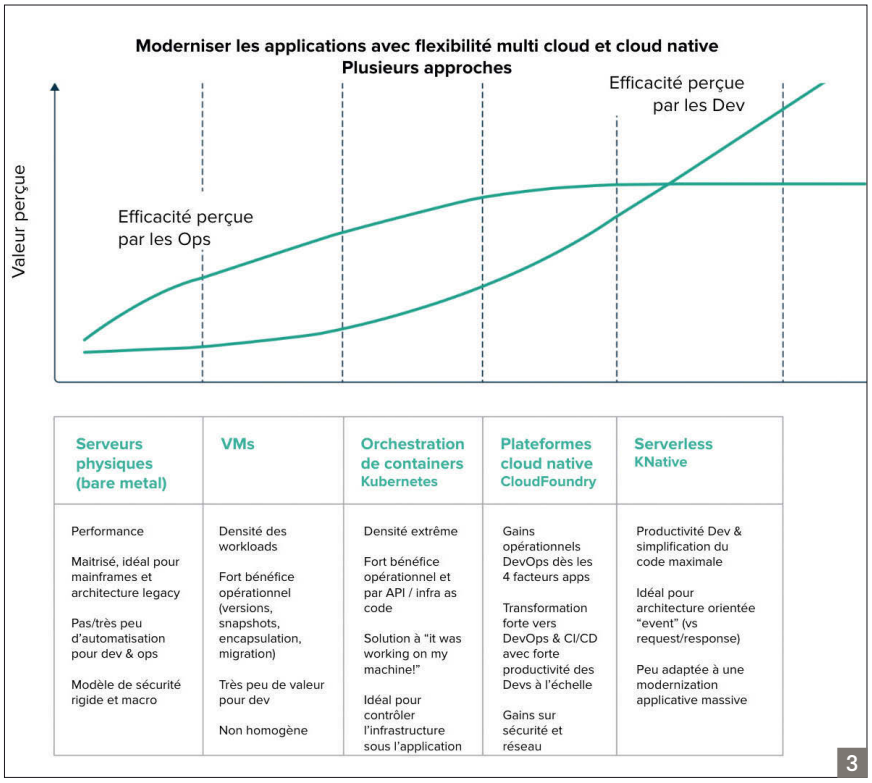
Les choix les plus courants sont désormais les orchestrateurs de conteneurs et les plateformes d'applications («plateformes PaaS»). Certains développeurs optent pour un degrè de modernisation encore plus élève avec des architectures dites «à 12 facteurs» ou même serverless. Cette méthode est généralement appelée conception d'applications natives dans le cloud. 5

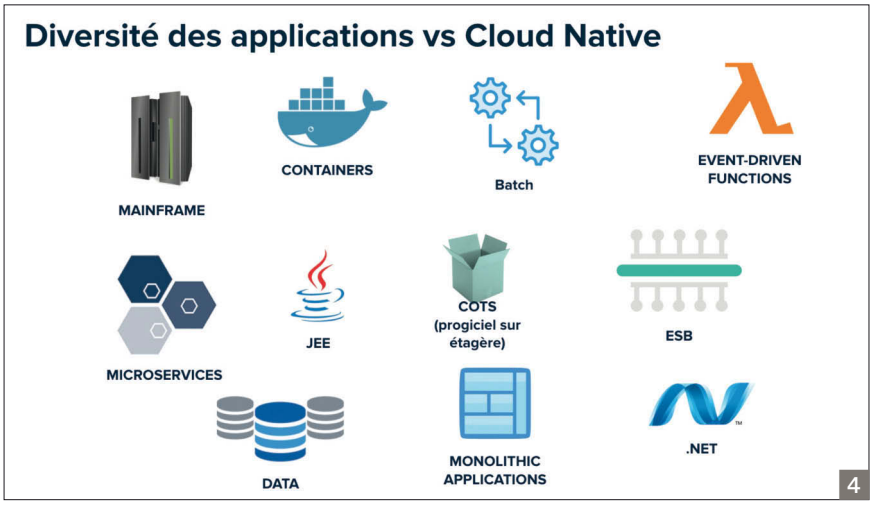
Conteneurs : une abstraction superieure a IaaS

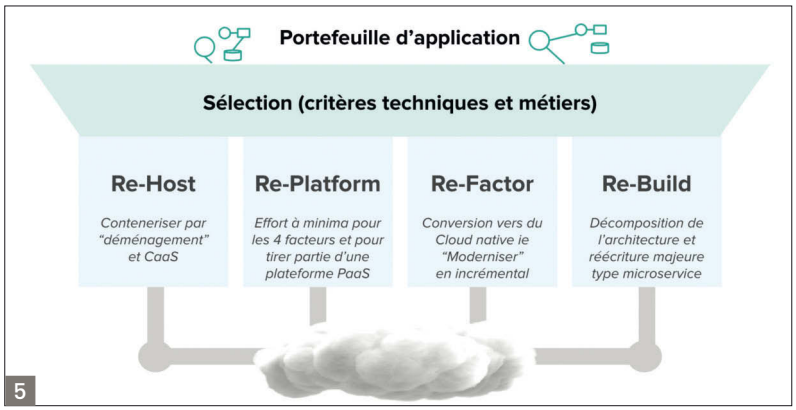
Les conteneurs sont extrêmement populaires. Et il est facile de voir pourquoi :

• La technologie résout élégamment le problème des «travaux sur









5

ma machine» en conditionnant le logiciel dans des formats standards, par exemple Docker ou OCI.
- Cette normalisation rend les conteneurs très portables à travers les nuages.
- Les conteneurs supportent une densité de charge de travail élevée et simplifient certaines fonctions ops.

Mais les développeurs sont modérement plus productifs avec cette abstraction. Ils sont maintenant responsables de la pile complète dans un conteneur. Cela ajoute une surcharge pour les développeurs. Lorsqu'une image de conteneur est prête à être partagée avec le monde, elle est poussée dans un conteneur orchestrateur. Kubernetes est l'option la plus populaire, mais encore faut-il savoir faire tourner et maintenir Kubernetes !

Exemple de Dockerfile pour application Java Tomcat - vite fragile, complexe ou obscure par héritage d'images Docker non contrôlées

```
# Dockerfile
FROM demo/maven:3.3-jdk-8
RUN apt-get update && \
  apt-get install -yq --no-install-recommends wget pwgen ca-certificates && \
  apt-get clean && \
  rm -rf /var/lib/apt/lists/*
ENV TOMCAT_MAJOR_VERSION 8
ENV TOMCAT_MINOR_VERSION 8.0.11
ENV CATALINA_HOME /tomcat
RUN wget -q https://archive.apache.org/dist/tomcat/.../apache-tomcat-${TOMCAT_MINOR_
VERSION}.tar.gz && \
  wget -qO- https://archive.apache.org/dist/tomcat/.../apache-tomcat-${TOMCAT_MINOR_
VERSION}.tar.gz.md5 |
  md5sum -c - && \
  tar zxf apache-tomcat-*.tar.gz && \
  rm apache-tomcat-*.tar.gz && \
  mv apache-tomcat* tomcat

ADD create_tomcat_admin_user.sh /create_tomcat_admin_user.sh
RUN mkdir /etc/service/tomcat
ADD run.sh /etc/service/tomcat/run
RUN chmod +x /*.sh
RUN chmod +x /etc/service/tomcat/run
EXPOSE 8080
  # etc - pour y déposer l'application ...
```

Plateformes et application à quatre facteurs

Le concept d' **"application à 12 facteurs"** est au cœur du mouvement cloud-native. Ces facteurs incarnent une approche de développement rapide, évolutive qui favorise le maintien en production avec par exemple de l'observabilité. Ces principes sont alignés avec la livraison continue et la productivite élevée du développeur en mode DevOps qui sont recherchés aujourd'hui.

Mais avoir 12 des 12 facteurs (quand on modernise le code et la configuration) pour tout un portefeuille d'applications vers le cloud est difficile. Heureusement, il existe un sous-ensemble de ces 12 facteurs qui constituent la «barre minimale» pour pouvoir tirer parti d'un environnement cloud. Gardez ces quatre facteurs à l'esprit lors de la modernisation des applications existantes et vos applications seront conteneurisables facilement voir déployables dans un PaaS :
- Un processus, un port ;
- Pas d'utilisation du systeme de fichiers local ;
- Toute la journalisation (log) va vers la sortie standard pour être streamée ensuite, et non pas sur le système de fichier ;
- Script de démarrage sans cérémonie ou ordre particulier.

Le principe de l'application a 12 facteurs a été introduit pour la première fois par Heroku et vous pouvez en apprendre plus sur https://12factor.net/ .

Une fois qu'une application est refactorisée pour prendre en charge ces contraintes simples, les bénéfices sont immédiats lorsque vous l'exécutez sur une plateforme moderne. Vous y gagnez :
- Planification et mise a l'echelle automatisées des containers ;
- Surveillance de la santé ;
- Quatre couches de haute disponibilité, avec redémarrage au niveau de l'instance d'application, du processus, de la machine virtuelle et de la disponibilite niveaux de zone ;
- L'application peut également fonctionner dans plusieurs environnements distincts : développement, tests, création de fonctions ou déploiements de production séparés au niveau régional ;
- Certaines plateformes telles Cloud Foundry sont également capables de conteneuriser pour vous via le concept de **Buildpack** (https://buildpacks.io) qui permet de normaliser la création des containers (Java, PHP, Python, .Net etc) et de faire porter la conteneurisation directement par la plateforme et non par les équipes DevOps, renforçant ainsi la gouvernance et la sécurité.

Buildpacks.io

La plateforme Cloud Foundry est connue et adaptée pour les applications 4 à 12 facteurs ; elle est complémentaire de Docker et de l'orchestration de container Kubernetes. En particulier vous pouvez combiner applications Java/Spring, buildpacks, 4 à 12 facteurs et également des containers *stateful* tels que MongoDB, ou une stack ELK dans Kubernetes. Vous pouvez l'essayer via https://run.pivotal.io
6

Exemple de déploiement avec Cloud Foundry buildpack, cf push et manifest applicatif

```
cf push -f manifest.yml myApp
---
```

```
applications:
- name: attendees
 random-route: true
 memory: 1G
 instances: 2
 # SpringBoot app, 4 factors app
 path: target/pcf-ers-demo-1.0.jar
 buildpack: java_buildpack
 services:
  - elk_on_kubernetes
```

Le futur - Frameworks microservices, et KNative

En plus des aspects modernisation et niveaux d'abstraction et 4 à 12 facteurs, plusieurs frameworks émergent pour simplifier encore la mise en place d'applications tirant nativement parti des plateformes d'abstraction telles que Kubernetes et Cloud Foundry. On peut notamment citer **Spring Cloud** avec Spring Cloud Netflix OSS et Spring Cloud Kubernetes.

Spring Cloud fournit aux développeurs des primitives permettant de créer rapidement certains modèles courants dans les systèmes distribués microservices aussi bien avec Cloud Foundry qu'avec Kubernetes. Grâce à Spring Cloud, les développeurs peuvent rapidement mettre en oeuvre des services et des applications qui implémentent ces modèles et qui fonctionnent bien dans n'importe quel cloud - l'ordinateur portable du développeur, les centres de données et les plateformes telles que Cloud Foundry. Certains patterns vont au-delà des 12 facteurs pour les microservices distribués:

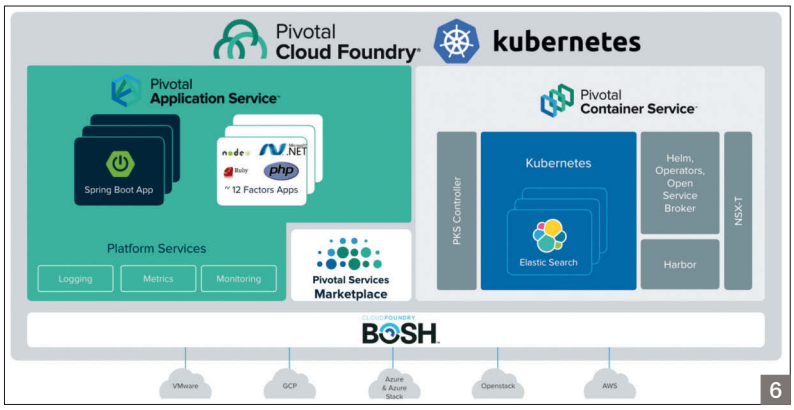
- Configuration distribuée et versionnée ;
- Enregistrement de service et découverte ;
- Routage et service mesh ;
- Appels de service à service ;
- Equilibrage de charge ;
- Disjoncteurs entre microservice ;
- Verrous globaux ;
- Élection et état du cluster ;
- Messagerie distribuée.

Vous pouvez accéder aux exemples de code via https://spring.io/projects/spring-cloud

Le projet **KNative**, porté notamment par Google et Pivotal (https://pivotal.io/knative), est apparu en 2018 et émerge comme une voie intéressante pour avoir un socle Kubernetes plus adapté au déploiement d'applications containers mais aussi 4 et 12 facteurs et serverless.

Exemple de serverless avec KNative

```
@SpringBootApplication
public class UppercaseApplication {
```




```
@Bean
public Function<String, String> uppercase() {
  return s -> s.toUpperCase();
}

public static void main(String[] args) {
  SpringApplication.run(UppercaseApplication.class, args);
}
}

pfs function create uppercase \
 --git-repo https://github.com/projectriff-samples/java-boot-uppercase.git \
 --image $REGISTRY/$REGISTRY_USER/uppercase \
 --verbose

// pfs service invoke uppercase --text -- -w '\n' -d 'welcome to pfs'

curl 35.239.12.146/ -H 'Host: uppercase.default.example.com' -H 'Content-Type: text/plain' -w '\n'
-d 'welcome to pfs'

WELCOME TO PFS
```

KNative embarque notamment le service mesh Istio permettant la mise en réseau des microservices, et permet aussi l'usage des buildpacks de Cloud Foundry.

A date, il est encore loin d'être mature pour la production et n'est pas encore adopté en entreprise, à l'inverse de Cloud Foundry, mais nul doute que la communauté va rapidement l'enrichir et que des solutions supportées vont exister. Pivotal a d'ailleurs déjà lancé le "Pivotal Function Service" basé sur Kubernetes et KNative.

Avec le cloud, le développeur est de plus en plus confronté aux principes de modernité et de cloud natif et microservices pour son code. Ceci sur la base des "12 factors" et de la conteneurisation. Il est aussi confronté aux enjeux de l'architecture et du "jour 2" propre à l'adoption et au déploiement via les plateformes containers, cloud et de type PaaS. A cela il faut ajouter le délicat équilibre d'être parfois sur des projets nouveaux (code à partir de zéro) ou sur des projet de modernisation applicative (changement de code et changement d'architecture) ; ceci nécessitant de fortes compétences en plus de l'attitude à adopter le changement rapide des piles technologiques. •

Cédric Michel
Développeur .Net et Scrum Master, est un collaborateur de la société Satellit. Celle-ci est une entreprise de 70 consultants spécialisée dans le développement Microsoft ainsi que la transformation business, digitale notamment à travers l'AI. Pour en savoir davantage, n'hésitez pas à aller consulter le site : **www.satellit.be**

Créer votre propre build step pour Azure DevOps Services.

Partie 2

niveau 100

À travers cet article, nous allons premièrement apprendre comment réaliser une extension pour Azure DevOps. Cette extension contiendra un build step qui sera capable de lancer un exécutable (ici, une application console écrite en .NET) et de lui passer des paramètres en argument. Ensuite, nous allons voir comment récupérer le retour de l'exécution afin de visualiser les logs dans votre Azure Pipelines.

Écriture du script PowerShell.

Le script PowerShell se charge de récupérer les paramètres depuis le build definition et de les envoyer comme paramètres à notre application console. C'est également ce script qui se charge de récupérer l'ouput de l'application console et de l'afficher dans la console Azure Pipelines (également utilisé comme log).

```
"inputs": [
        {
      "name": "parameter",
      "type": "string",
      "label": "Parameter to launch the Synchronisation MongoDb tool",
      "defaultValue": "-q -f",
      "required": true,
      "helpMarkDown": "Parameter send to tool"
    }
```

Pour ce faire, nous allons récupérer la valeur du champ « parameter » créé dans le task.json. Cela est possible avec la méthode Get-VstsInput du SDK. Cet élément sera envoyé comme paramètre à notre application console. Pour pouvoir lancer l'application en PowerShell, nous allons instancier l'object System.Diagnostics.Process du framework .NET. Grâce à celui-ci, nous allons pouvoir lancer l'exécution via la méthode StartInfo. Il faut au préalable setter la propriété StartInfo avec une nouvelle instance de System.Diagnostics.ProcessStartInfo. C'est avec ce dernier que vous pourrez lui donner le chemin vers l'exe ainsi que les paramètres (arguments) à passer à votre application.
L'appel Write-Host permet de renvoyer l'output du script dans le programme appelant. Et donc, il sera visible lors de l'exécution de votre build step et les logs. Voici le contenu du script : tool.ps1

```
[CmdletBinding()]
param()
Trace-VstsEnteringInvocation $MyInvocation

try
{
    Write-Host "... Powershell Starting ..."

    $parameter = Get-VstsInput -Name parameter -Require
    Write-Host "Parameter : $($parameter)"

    $exeFileName = "tool/MongoSync.exe"
    $exeFilePath  = Join-Path $PSScriptRoot $exeFileName

    $pinfo = New-Object System.Diagnostics.ProcessStartInfo
    $pinfo.FileName = $exeFilePath
    $pinfo.Arguments = $parameter
    $pinfo.UseShellExecute = $false
    $pinfo.CreateNoWindow = $true
    $pinfo.RedirectStandardOutput = $true
    $pinfo.RedirectStandardError = $true

    $process= New-Object System.Diagnostics.Process
    $process.StartInfo = $pinfo
    $process.Start() | Out-Null
    $process.WaitForExit()


    $stdout = $process.StandardOutput.ReadToEnd();
    Write-Host $stdout
    Write-Host "... Powershell Step finished ..."
}finally{
    Trace-VstsLeavingInvocation $MyInvocation
}
```

Génération automatique de l'extension.

Pour faciliter la génération du package de l'extension (.vsix), Nous utilisons quelques commandes en post build. 8

```
copy /Y "$(TargetPath)" "$(ProjectDir)Vsts-Extension\home\buildAndReleaseTask\tool"
copy /Y "$(TargetDir)*.dll" "$(ProjectDir)Vsts-Extension\home\buildAndReleaseTask\tool"

del /q $(ProjectDir)Vsts-Extension\home\*.vsix
cd $(ProjectDir)Vsts-Extension\home\
call $(ProjectDir)Vsts-Extension\home\package.bat

copy /Y "$(ProjectDir)Vsts-Extension\home\*.vsix" "$(TargetDir)"

del /q $(ProjectDir)Vsts-Extension\home\buildAndReleaseTask\tool
del /q $(ProjectDir)Vsts-Extension\home\*.vsix
```

Ces commandes permettent de copier l'exécutable et l'ensemble des librairies utiles dans le répertoire tool. Il sera alors possible de venir générer la nouvelle version de l'extension grâce au fichier package.bat créé précédemment. Ensuite, nous plaçons l'extension dans le répertoire bin. Finalement, nous supprimons les fichiers générés dans l'arborescence de base.

Déployer et rendre l'extension publique sur le store.

Maintenant que votre profil est validé et l'extension générée, vous pouvez l'uploader sur le store. 9 10

À ce niveau-ci l'extension ne sera visible que par vous-même, ou éventuellement aux personnes à qui vous auriez explicitement donné l'accès. 11

Pour rendre votre extension publique, il vous suffit d'ajouter la section suivante dans le fichier vss-extension.json

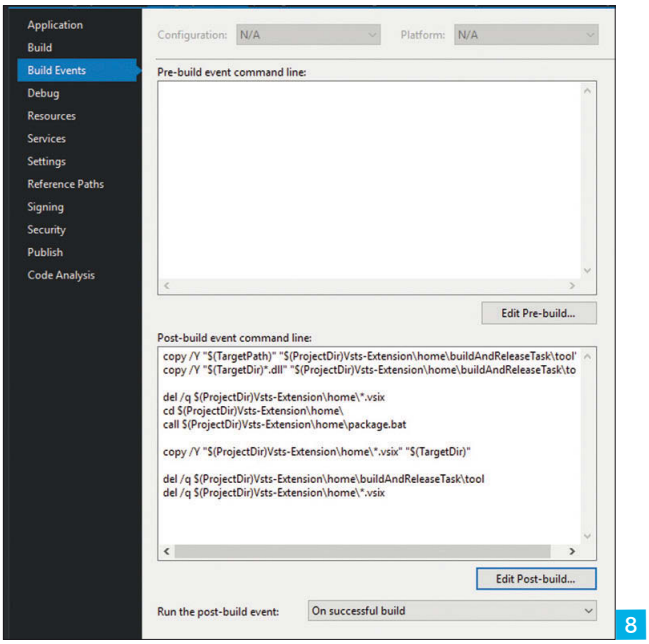
```
"galleryFlags":[
  "Public"
]
```

Ensuite, via le bouton update, envoyez votre dernière version.

Attention entre chaque déploiement vous devez incrémenter le numéro de version de l'extension dans le fichier : vss-extension.json

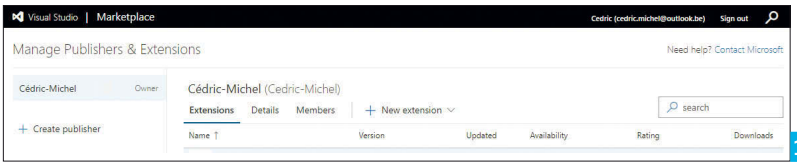
Si vous le désirez, vous pouvez également incrémenter la version de la task en elle-même dans le fichier task.json

Voilà, on y est ! Il ne vous reste plus désormais qu'à utiliser votre extension ! 12 13 •



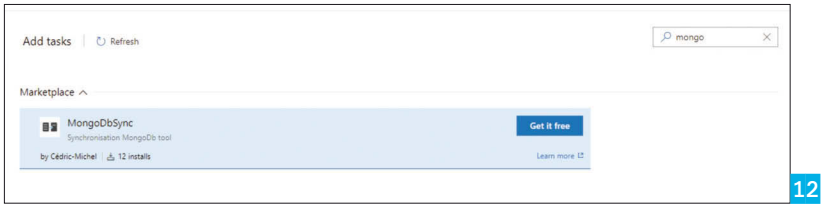
8

9

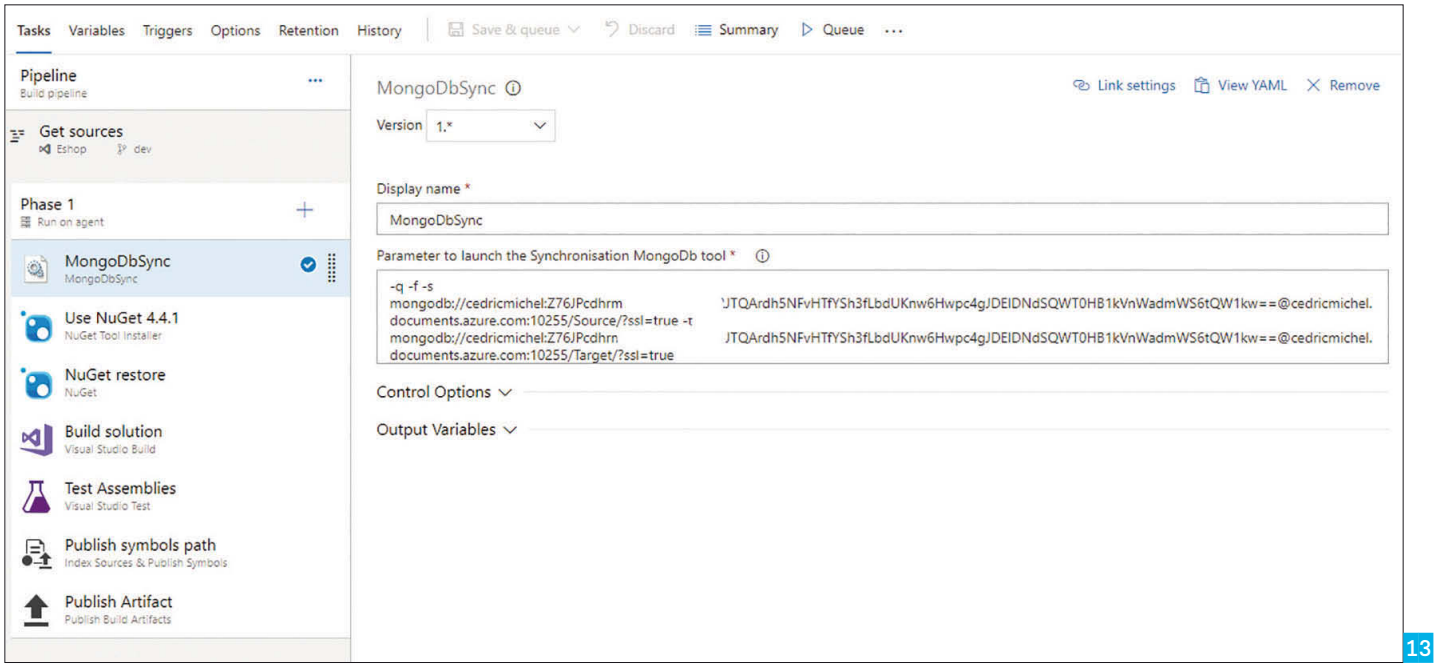


10

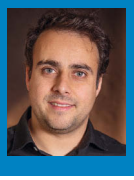
11



12



13



Steve Houël
Architecte Cloud & DevOps chez Ippon Technologies, évangéliste Serverless et contributeur sur des projets OpenSource comme JHipster et daSWAG. Il est fier d'être un #GeekEnthusiast.

Architecture Serverless
De la Théorie à la pratique

niveau 200

Partie 1

Nous le savons, le monde informatique est en constant changement. Que ce soient les évolutions matérielles, l'avènement de l'IoT dans les objets de tous les jours ou encore les services proposés par les Cloud Providers. Le monde du développement logiciel n'échappe pas à cette tendance. Outre les nouveaux frameworks Web qui sortent plus vite que notre courbe d'apprentissage, les architectures applicatives elles-aussi se voient repensées, remaniées. Il y a encore peu de temps, nous pensions tous qu'un bon vieux monolithe était "LA" solution simple, efficace et économique.

Or suite à la naissance de la conteneurisation et du DevOps, un nouveau panel d'architectures a vu le jour. Nous avons ainsi vu l'émergence des architectures Microservices. Simples, scalables et rapides à développer lorsque l'on se base sur des générateurs tels que JHipster (https://www.jhipster.tech/), elles ont ouvert de nouvelles voies dans le développement d'applications Web. Mais comme toute architecture, celle-ci vient avec son lot de contraintes. L'une d'entre elles est la gestion de l'infrastructure. Même si le DevOps, la conteneurisation et les orchestrateurs tels que Kubernetes ont apporté beaucoup dans cette problématique, ils ne l'ont pas résolue pour autant et seules quelques entreprises avec un certain degré de maturité dans ces domaines et cette culture se sont vues le droit d'exploiter tout le potentiel de ce type d'architecture.

Aujourd'hui, de nouveaux services font parler d'eux dans le monde de l'IT et principalement lorsque l'on parle de fournisseur de services Cloud, c'est le Serverless.

Qu'est ce que le Serverless ?

Les architectures Serverless se réfèrent à des applications qui dépendent de manière significative de services tiers (connues sous le nom de Backend as a Service ou «BaaS») ou sur un code personnalisé exécuté dans des conteneurs éphémères, entièrement gérés (Function as a Service ou «FaaS»). Le fournisseur de ce type de service le plus connu et utilisé est actuellement AWS avec ses fonctions *Lambda* ou Google avec ces Cloud functions. De nos jours la migration de nombreuses fonctionnalités côté FrontEnd nous a permis de supprimer nos besoins de serveurs "Always On". Selon les circonstances, de tels systèmes peuvent réduire considérablement le coût et la complexité opérationnelles et ainsi se résumer à payer uniquement les frais d'utilisation (bande passante, volume de stockage, temps d'exécution). Ainsi on ne paie que ce que l'on consomme (connu aussi comme le ***pay-as-you-go)***.

Comme de nombreuses tendances dans le logiciel, il n'y a aucune vision claire de ce qu'est '**Serverless**', et cela n'a pas été aidé par le fait qu'il s'agisse vraiment de deux domaines différents, mais qui se chevauchent :

- Le terme "Serverless" a d'abord été utilisé pour décrire des applications qui dépendent de manière significative ou totale des applications / services tiers ('dans le cloud') pour gérer la logique et l'état du serveur. Ce sont généralement des applications "client complète" (pensez à des applications Web en une seule page ou à des applications mobiles) qui utilisent le vaste écosystème de bases de données accessibles sur le cloud (comme Parse, Firebase, AWS DynamoDB, ...), les services d'authentification (Auth0, AWS Cognito), etc. Ces types de services ont été précédemment décrits comme *Backend as a service* (j'utiliserai le terme **BaaS** comme abréviation dans le reste de cet article).
- Serverless peut également symboliser des applications dont une certaine quantité de logique serveur est toujours écrite par le développeur, mais contrairement aux architectures traditionnelles, elle est exécutée dans des conteneurs stateless qui sont déclenchés par le biais d'événements, éphémères (uniquement une invocation) et entièrement gérés par une tierce partie. Ceci correspond au concept du *Function as a service* ou **Faas**. *AWS Lambda* est l'une des implémentations les plus populaires de FaaS à l'heure actuelle, mais il y en a d'autres sur d'autres fournisseurs de services Cloud tels que Google Cloud Platform avec ses Google Functions ou Microsoft Azure avec ses Azure Functions. Nous avons aussi la possibilité d'utiliser ce type de technologies directement sur votre orchestrateur Kubernetes via du OpenFaaS par exemple. Nous reviendrons sur cet écosystème ultérieurement.

Nous parlerons principalement dans cet article de la deuxième définition car elle est plus récente et possède le plus de différences avec la vision que l'on a d'une architecture technique traditionnelle (elle est tout simplement plus hype !!).

Serverless ≠ sans serveurs ?

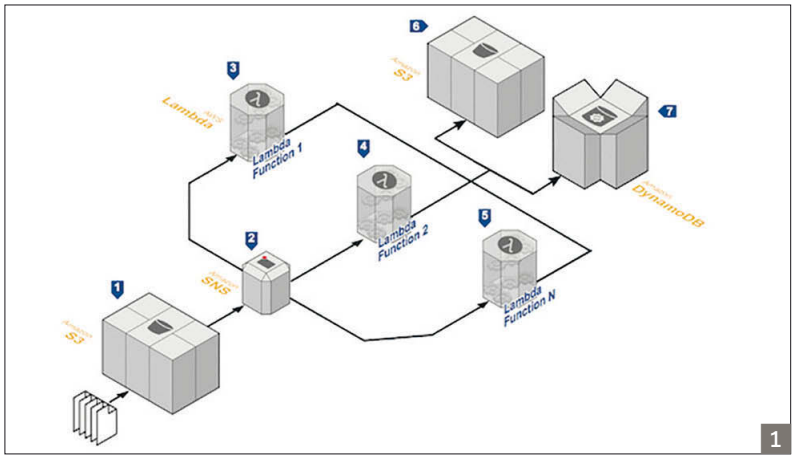
Le terme Serverless est source de confusion car, dans de telles applications, il y a à la fois des notions de matériel et de systèmes. La différence avec les approches traditionnelles est qu'une entreprise misant sur le Serverless ne s'occupe généralement pas de ces deux aspects, elle externalise la responsabilité à un fournisseur (AWS, Google, Azure, ...) et ainsi se concentrent uniquement sur la partie fonctionnelle de son application.

Quelques cas d'utilisations

Afin d'illustrer au mieux ce type de technologies, étudions maintenant des cas concrets d'utilisation du Serverless.

Traitement de fichiers temps réel

Cette première architecture concerne le traitement des fichiers. Elle s'applique très bien lorsque l'on a besoin d'avoir différents traitements sur la donnée source et de générer plusieurs formats en sortie voire d'effectuer des actions simples de transformations. Dès lors, nous pouvons utiliser une concaténation de plusieurs services et effectuer l'intégralité de notre processus de traitement de données. Cette architecture s'applique très bien dans le domaine de l'analyse et du traitement de données, cependant il faut rester sur des cas d'usages simples et en aucun cas nous pourrons atteindre la puissance d'un ETL. 1



1

(source http://www.allthingsdistributed.com/images/serverless-fileprocessing.png)

Application Web

En combinant du FaaS avec d'autres services managés, les développeurs peuvent créer de puissantes applications Web évoluant automatiquement dans une configuration hautement disponible sur plusieurs zones, sans aucun effort supplémentaire pour l'évolutivité, la mise à l'échelle, les sauvegardes ou la redondance des données.

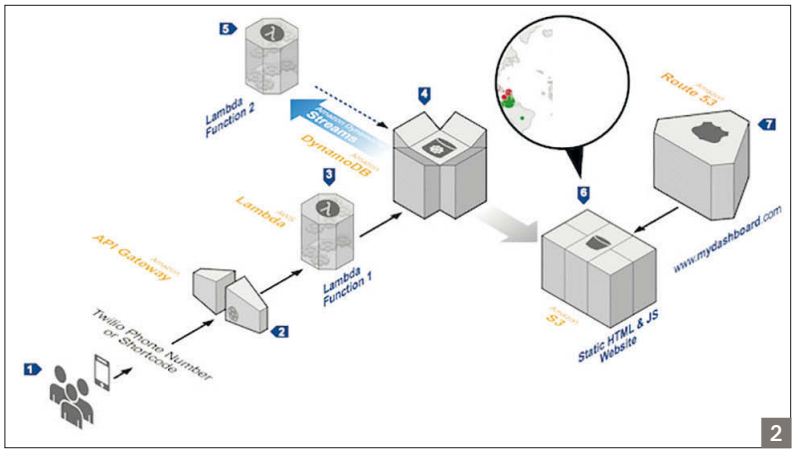
Cette architecture est un exemple concret d'une application Web de vote dynamique, qui reçoit les informations via SMS (au travers d'un service BaaS comme **Twilio**), agrège les totaux dans une base de données et utilise un gestionnaire de fichiers pour mettre à disposition les résultats en temps réel.

Cette architecture peut être générée rapidement au travers de technologies **InfraAsCode** (Serverless Application Model, Terraform ou le framework Serverless). 2

Bien sûr, nous pouvons aussi imaginer une application SaaS intégralement basée sur ces technologies Serverless.

Backend IoT

Ce type de technologie se prête particulièrement bien au domaine de l'IoT avec des flottes d'objets connectés pouvant se multiplier de façon exponentielle. Cet exemple d'architecture de référence pour le domaine de l'IoT illustre comment il est possible de récupérer et traiter les données issues de



2

http://www.allthingsdistributed.com/images/serverless-webapp.png)

capteurs. En tirant parti de ces services, vous pouvez créer des applications rentables, capables de répondre et de s'adapter aux demandes massives générées par les objets connectés. 3

Traitement d'un flux de données temps réel

Vous pouvez utiliser des fonctions et des services de gestion de flux de données pour traiter de nombreux cas d'utilisation comme le suivi des activités, l'analyse des clics sur une application Web ou même l'analyse des médias sociaux. 4

Ce qui n'est pas Serverless

Jusqu'à présent, j'ai défini le terme "Serverless" pour signifier l'union de 2 principes - «**Backend as a Service**» et «**Functions as a Service**».

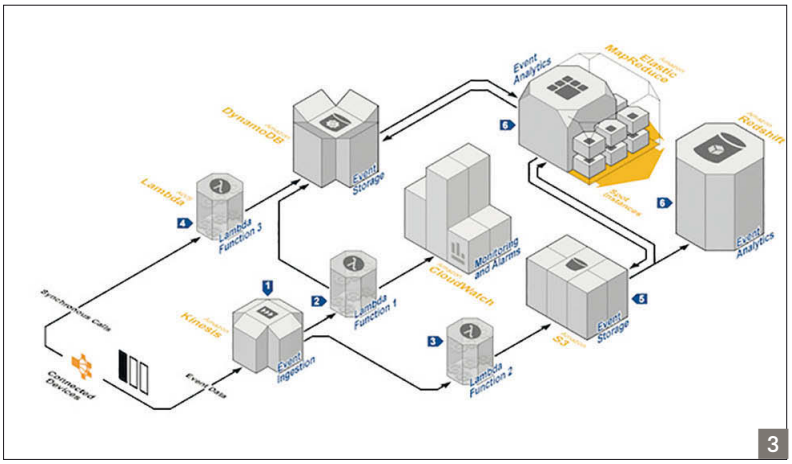
Avant de commencer à examiner la promesse que peut offrir ce type d'architecture, j'aimerais d'abord expliquer ce qui n'est pas Serverless. Actuellement lorsque l'on recherche ce terme, nous pouvons trouver de nombreuses définitions ou assimilations erronées.

PaaS (Platform as a Service)

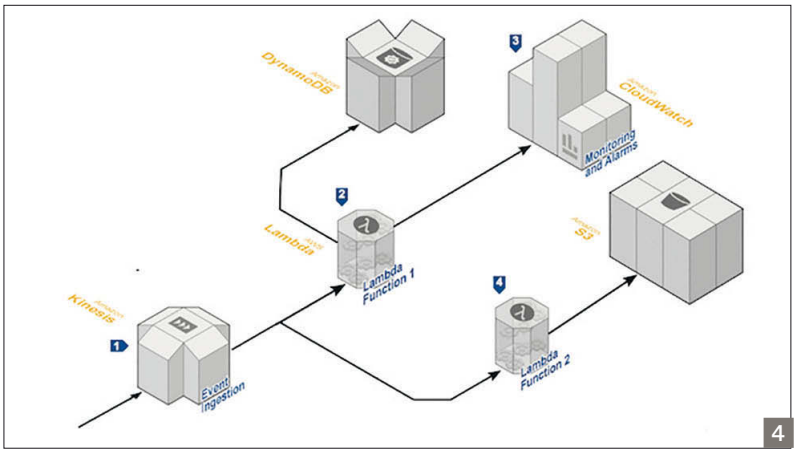
On peut en effet trouver des similitudes entre certains services PaaS comme Heroku qui propose des déploiements à la volée de notre application sur simple exécution d'une commande. Cependant ceux-ci ne procurent pas encore le niveau d'abstraction nécessaire pour être considéré comme Serverless. Nous avons toujours le besoin d'allouer une quantité de ressources pour garantir l'exécution de notre application ou notre système et l'adapter à la demande des utilisateurs.

Conteneur

La conteneurisation jouit d'une popularité croissante de nos jours, surtout depuis l'arrivée de Docker et la montée en puissance des orchestrateurs tels que Kubernetes. Nous pouvons en effet trouver certaines



(source http://www.allthingsdistributed.com/images/serverless-iot.png)



(source http://www.allthingsdistributed.com/images/serverless-streamprocessing.png)

similarités entre FaaS et la conteneurisation. Mais rappelons-le, FaaS offre une couche d'abstraction telle, que nous n'avons plus la notion de processus système au contraire de Docker qui est basé sur la notion de processus unique.

Parmi ces similitudes, nous retrouvons l'argument de la mise à l'échelle automatique. Fonctionnalité disponible niveau conteneur grâce à des orchestrateurs par le biais d'outils spécifiques comme le contrôleur *Horizontal Pod Autoscaler*. Dans ce cas nous pouvons nous poser la question du pourquoi faire du FaaS alors que nous pouvons faire du conteneur ?

Il n'y a pas de réponse miracle à ce jour. Ce qu'il faut bien prendre en compte avant tout c'est le cas d'utilisation et pourquoi on veut faire du FaaS. À ce jour les orchestrateurs peuvent fournir un niveau de scalabilité équivalent surtout s'ils sont préconfigurés et managés par un fournisseur de services (exemple de EKS sur AWS, GKE sur Google ou AKS sur Azure). La différence se fera sur la granularité voulue lors du développement de votre solution et des avantages du FaaS par rapport aux conteneurs.

#NoOps

Il ne faut pas confondre Serverless et NoOps. Si on prend le mot Ops (Opérations) cela ne signifie pas uniquement des opérations d'administration systèmes. Cela signifie également au moins le suivi, le déploiement, la sécurité, la connectivité, le monitoring, la supervision et aussi souvent une certaine quantité de débogage de production et de mise à l'échelle du système. Ces problèmes existent toujours avec des applications Serverless, je dirais même qu'elles sont plus compliquées étant donné la jeunesse de la technologie et les nouvelles fonctions et paramètres à prendre en compte. Certaines seront simplement déportées et gérées directement par les développeurs et des architectes. La culture DevOps prendra alors tout son sens.

LE SERVERLESS : LICORNE OU DÉSILLUSION

Les technologies Serverless sont souvent comparées au monde des licornes tant les promesses qu'elles offrent font rêver. Étudions cela de plus près et voyons ensemble les avantages et inconvénients de cette technologie.

Avantages

Coût opérationnel réduit

Serverless est par nature une solution simple d'externalisation. Il vous permet de payer quelqu'un pour gérer les serveurs, les bases de données et même la logique des applications. Etant donné que votre service fait partie d'un ensemble de services similaires, la notion d'économie d'échelle va alors s'appliquer - vous réduisez vos coûts de gestion étant donné que le même service est utilisé par de nombreux autres.

Les coûts réduits apparaissent comme le total de trois notions :

- le coût d'infrastructure,
- le coût des employés (opérations / développement),
- le coût de maintien des compétences.

Alors que certains des gains de coûts peuvent venir uniquement de l'infrastructure de partage (matériel, réseau) avec d'autres utilisateurs, l'attente derrière peut aussi se traduire dans une réduction des coûts liés à l'utilisation du personnel d'exploitation du fait de l'utilisation de technologies managées. L'aspect compétence ne doit pas être oublié vu la rapidité d'évolution des technologies et le coût que cela représente d'effectuer cette veille technologique ou même de recruter une personne compétente. Ce dernier point d'autant plus, du fait que le marché du travail dans ce secteur est à flux tendu.

Cet avantage, cependant, n'est pas trop différent de ce que vous obtiendrez en utilisant des technologies de type Container as a Service (CaaS) ou Platform as a Service (PaaS).

Coût de développement réduit

Afin d'illustrer ce point, prenons en exemple le cas de l'authentification. De nombreuses applications disposent de leur propre service d'authentification et de gestion des utilisateurs, implémentant par la même

occasion leur propre niveau de sécurité. Parmi les fonctionnalités implémentées, nous pouvons retrouver :

- L'enregistrement et la validation d'un utilisateur (Enregistrement Suppression) ;
- La récupération d'un mot de passe ;
- La connexion et l'accès aux services ;
- Le machine to machine ;
- Le Multi Factor Authentication (MFA).
- ...

Dans l'ensemble nous retrouvons à peu de choses près ces fonctionnalités dans la plupart des applications actuelles. Même si des solutions de type générateurs d'applications comme JHipster existent et permettent de générer rapidement plusieurs types d'authentification, il n'en reste pas moins à la charge de l'entreprise de maintenir ce code et de les faire évoluer. A ce jour, nous voyons l'émergence de services tels que Auth0 qui fournissent des fonctionnalités d'authentification "prêtes à l'emploi". L'application peut ainsi se décharger de ces fonctionnalités et laisser le fournisseur du service être responsable de leur maintien et de la partie sécurité.

Un autre exemple qui se prête bien au jeu est l'utilisation de services de base de données tels que DynamoDB (NoSQL), Cloud Datastore (NoSQL) ou Aurora Serverless (Relationnelle). On retrouve principalement ces cas d'utilisations au sein des architectures mobiles qui préfèrent créer une communication directe entre le client (mobile), la base de données et ainsi supprimer tous les tiers (administration de la base de données, son optimisation, ...). Ce système apporte aussi une nouvelle couche de sécurité et permet une gestion plus fine des tables et des données accessibles en fonction des profils utilisateur.

Mise à l'échelle automatique

Pour ma part l'un des avantages les plus importants du Serverless est la mise à l'échelle horizontale automatique et l'optimisation de la consommation que peut proposer un fournisseur de services. Cela peut se traduire par plusieurs avantages, principalement au niveau infrastructure, mais surtout cela permet d'avoir une facturation très fine et de ne payer que la charge dont vous avez besoin, que ce soit en temps de calcul utilisé (à partir de 100 ms pour AWS Lambda) ou en quantité de données récupérées ou analysées. Selon votre architecture et vos cas d'utilisations cela peut engendrer une économie certaine surtout lors des phases de non consommation des services, exemple d'une application fonctionnant uniquement pendant les heures ouvrées d'une entreprise. Ce type de développement peut certes se faire sur les autres systèmes mais nécessitera un développement spécifique alors que cela est nativement disponible sur les solutions type Serverless.

Un autre cas d'exemple d'économie est l'utilisation ponctuelle d'une fonction. Par exemple, disons que vous exécutez une application serveur qui ne traite que 1 demande chaque minute, qu'il faut 50 ms pour traiter chaque requête et que votre utilisation moyenne de CPU pendant une heure est de 0,1%. D'un point de vue charge de travail serveur, cela est extrêmement inefficace.

La technologie FaaS capte cette inefficacité et vous permet ainsi de ne payer que ce que vous consommez, c'est-à-dire 100ms (valeur minimum) de calcul par minute, soit moins de 0,2% du temps global.

L'optimisation, la clé de votre facture

Même si cette nouvelle architecture propose des nouvelles fonctionnalités telles que la mise à l'échelle, elle n'en subit pas moins les contraintes inhérentes au développement d'applications. Ainsi la phase d'optimisation des fonctions prend encore plus de valeur vu qu'elle permettra, en plus d'améliorer le temps de réponse aux utilisateurs, d'économiser de l'argent sur la facturation.

Par exemple pour une opération qui initialement prend 1 seconde et qui après optimisation prend 200ms, nous aurons une réduction immédiate de notre facture de 80% du coût de calculs.

Illustrons ce cas par un exemple concret sur la base du service AWS Lambda :

Si vous avez attribué 512 Mo de mémoire à votre fonction et que vous l'avez exécutée 3 millions de fois en un mois (pour une durée d'une seconde par exécution), vos frais sont calculés comme suit :

Frais de calcul mensuels

Le tarif de calcul mensuel revient à 0,00001667 USD par Go-s et le niveau gratuit offre 400 000 Go-s.

Taux de calcul total (en secondes) =
3 M * (1 s) = 3 000 000 secondes

Taux de calcul total (en Go-s) =
3 000 000 * 512Mo/1 024 =
1 500 000 Go-s

Taux de calcul total – Taux de calcul offert =
Taux de calcul facturable par mois (en Go-s)
1 500 000 Go-s – 400 000 Go-s offerts =
1 100 000 Go-s

Frais de calcul mensuels =
1 100 000 * 0,00001667 USD =
18,34 USD

Frais de requêtes mensuels

Le tarif de requête mensuel est de 0,20 USD par million de requêtes et le niveau gratuit offre un million de requêtes par mois.

Nombre total de requêtes – Nombre de requêtes offertes = Nombre de requêtes facturables par mois

3 M de requêtes – 1 M de requêtes offertes = 2 M de requêtes facturables par mois

Frais de requêtes mensuels = 2 M * 0,2 USD/M = 0,40 USD

Frais mensuels totaux

Frais totaux = Frais de calcul + Frais de requêtes = 18,34 USD + 0,40 USD = **18,74 USD par mois**

Frais mensuels totaux après optimisation décrite précédemment :

Taux de calcul total optimisé (en secondes) =
3 M * (0,2 s) = 600 000 secondes

Taux de calcul total (en Go-s) = 600 000 *
512Mo/1 024 = 300 000 Go-s

Taux de calcul total – Taux de calcul offert =
Taux de calcul facturable par mois (en Go-s)
300 000 Go-s – 400 000 Go-s offerts
= -100 000 Go-s donc 0 Go-s

Frais de calcul mensuels optimisés
= 0 * 0,00001667 USD = 0 USD

Frais totaux après optimisation =
Frais de calcul optimisé + Frais de requêtes =
0 USD + 0,40 USD = **0,40 USD par mois**

Une informatique plus "verte"

Etant donné que l'écologie est l'un des sujets du moment, j'ai décidé de faire un paragraphe dédié à cela.

De nos jours, nous avons vu le nombre de datacenters augmenter exponentiellement au fil des années. Leur consommation en énergie est énorme et de plus en plus de Cloud providers se sensibilisent à l'écologie et aux énergies renouvelables. Ainsi Google, Apple et d'autres parlent de construire et d'héberger certains de leurs datacenters dans des zones à fort potentiel en énergies renouvelables ou dans des zones froides afin de réduire l'impact sur l'environnement de ces sites (consommation en énergie pour le refroidissement de ces sites ou réutilisation de la chaleur générée pour du chauffage). L'une des causes de cette hausse du nombre de serveurs est due au maintien et à la consommation des serveurs dits "inactifs" face à une demande toujours croissante des entreprises.

Cela représente un fonctionnement extrêmement inefficace et surtout un impact environnemental non négligeable.

Ces charges non utilisées viennent principalement de décisions faites par les entreprises sur les capacités nécessaires au fonctionnement d'une application et des "marges de sécurité" faites afin de pallier les fluctuations. Avec une approche Serverless, nous ne prenons plus de décision sur la capacité nécessaire à l'exécution d'une fonctionnalité, cette charge revient désormais aux fournisseurs du service. Il se doit de fournir une capacité de calcul suffisante pour nos besoins en temps réel. Il pourra ainsi avoir une vision globale des capacités nécessaires pour l'ensemble de ces clients. Ils pourront par ce biais optimiser la gestion des ressources et permettre une réduction du nombre de serveurs et en conséquence l'impact environnemental des datacenters.

Les inconvénients

Eh oui, nous ne sommes pas là que pour vanter les mérites de cette nouvelle technologie et dire qu'elle peut résoudre tous nos problèmes. Comme toute technologie, elle vient aussi avec son lot d'inconvénients qui ne sont pas à prendre à la légère, car ceux-ci pourraient devenir votre pire cauchemar selon vos cas d'utilisations.

- Il faudra toutefois séparer ces inconvénients en 2 catégories :
  - ceux inhérents à ces nouvelles architectures et cette technologie,

ceux qui ont pour origine sa jeunesse et son manque d'outillages et de solutions.

Verrouillage des Cloud Providers

Le premier et pas des moindres pour de nombreuses entreprises est la dépendance forte que l'on crée avec le fournisseur de service. À ce jour aucune standardisation n'est sortie afin d'adopter un langage commun de déploiement entre les fournisseurs. Même si certains frameworks (Serverless.com) essaient de briser ces limitations et proposent une approche agnostique lors de vos développements, lors de la conception de votre solution et du choix des fonctionnalités, vous devrez choisir un fournisseur unique afin de garantir une certaine homogénéité de communication entre les différentes couches et pour pallier le verrouillage que les fournisseurs font de leurs services. Par exemple pour le stockage de vos données au sein d'une base de données qui nécessitera généralement un développement ou des librairies spécifiques (type ORM), vous pourrez toujours trouver des passerelles afin d'utiliser plusieurs fournisseurs de services différents (Authentification via Auth0, BDD via Cloud Datastore et API Gateway + Lambda via AWS). Toutefois, dans ce cas, cela compliquera fortement l'administration et la facturation de votre solution et même son développement et sa gouvernance. Après, la question qu'il faut se poser est : allons-nous réellement changer de fournisseur de services vu le coût et le risque que la migration va engendrer ?

Optimisations des serveurs

À partir du moment où vous décidez d'utiliser des technologies Serverless, vous abandonnez par ce fait le contrôle de certains systèmes tiers et de leur configuration. Même si cette gestion par le fournisseur vous conviendra dans 99% des cas, il reste 1% des cas où votre solution nécessitera d'avoir une configuration spécifique du service afin d'améliorer ses performances ou sa qualité de services (exemple des bases de données).

Sécurité

Je ne pouvais pas écrire un article sur le Serverless sans parler de la sécurité de cette technologie. De nombreuses entreprises se sensibilisent de plus en plus aujourd'hui à la sécurité de leurs applications et à l'accès à leurs données. Les services Serverless ne vont pas échapper à la règle et vont apporter leur lot de questions. Je vais tenter d'en expliquer 2, mais de nombreuses autres sont à considérer.

- En sécurité, on parle souvent de périmètre ou de surface d'action d'une solution. Cela correspond à l'empreinte que celle-ci a sur Internet. Plus l'empreinte est grande, plus la surface d'attaque est importante. Or l'utilisation de plusieurs services Serverless va augmenter votre empreinte et créer une certaine hétérogénéité dans vos politiques de sécurité. Par ce fait vous augmenterez votre probabilité d'intention malveillante à l'encontre de votre solution et la probabilité qu'une de ces attaques soit réussie.
- Si vous utilisez le service de base de données de type BaaS et permettez l'accès direct à vos données via une API cliente, vous perdrez la barrière de protection qu'une application serveur traditionnelle peut fournir de par sa configuration réseau ou ses restrictions d'accès au serveur. Cependant, le point positif est que c'est désormais votre fournisseur de service qui se chargera au travers de ces services IAM de vous fournir toute la sécurité nécessaire pour votre utilisation.

Problème à l'utilisation

Passons maintenant aux inconvénients inhérents aux solutions actuellement disponibles. Ceux-ci pourront en effet être corrigés avec l'évolution de la technologie et de l'écosystème qui l'entourent.

Durée d'exécution

Un problème actuel concerne la limitation faite sur la durée d'exécution des fonctions. Actuellement nous avons une durée limite par exemple 15 minutes pour AWS Lambda, 9 min pour Google Cloud Function et 15 min pour Azure Function. Cette contrainte restreint le périmètre d'actions des fonctions et empêche leur utilisation pour un grand nombre de cas d'utilisation comme le traitement de vidéo et certaines transformations de vos données.

Latence de démarrage

Le temps de réponse de votre fonction FaaS à une demande dépend d'un grand nombre de facteurs et peut aller de 10 ms à la minute. Soyons un peu plus précis en utilisant *AWS Lambda* comme exemple.

Si votre fonction est implémentée en NodeJS ou Python qui sont des langages interprétés avec un contenu simple (moins d'un millier de lignes de code), la durée d'initialisation devrait se situer entre 10 et 100 ms. Si en revanche, votre fonction AWS Lambda est exécutée dans une JVM (Java, Scala, …), vous pourrez avoir un temps de démarrage drastiquement supérieur dû à la "chauffe" de la JVM et ainsi atteindre la minute selon la quantité de code. Il y a aussi un effort à faire côté Cloud provider pour proposer une compatibilité avec des technologies comme GraalVM our Quarkus.

D'autres facteurs sont aussi à prendre en compte comme le fait de lancer votre fonction au sein d'un réseau (Virtual Private Cloud) qui nécessitera la création et le positionnement d'une interface (ENI : Elastic Network Interface) et ajoutera directement au moins 10 secondes de démarrage.

Ces phases de chargement et d'initialisation que l'on nomme "Cold Start" se produisent principalement dans certains scénarios :

- Votre fonction traite rarement des événements (plus de 15 min environ entre les invocations) ;
- Vous avez des pics très soudains dans le trafic et cela provoque des appels concurrents à votre fonction.

Ces problèmes sont-ils préoccupants ?

Cela dépend du style et de la forme de votre trafic pattern, de votre fonction et des différentes conditions d'exécution de celle-ci (réseaux, langage, …). Le FaaS n'est pas la solution ultime pour l'exécution de votre application et de vos cas d'utilisations, et parfois une solution PaaS peut apporter des atouts non négligeables.

Tests

Vu que l'on parle beaucoup de développement, il en est de même pour les tests. Même si certains pensent que "Tester c'est douter", nous nous devons de traiter ce point, d'autant plus qu'il s'agit d'un lourd inconvénient lors de l'utilisation de ce type de services dans un contexte professionnel.

Certains peuvent penser qu'en raison de l'isolation de chacune des fonctions, il peut être relativement facile de les tester. Ceux-là ont raison pour le périmètre des tests unitaires vu qu'il s'agit simplement d'un bout de code indépendant et que de nombreux appels tiers pourront faire l'office d'un Mock, cependant lorsque l'on aborde le sujet des tests d'intégration c'est une autre paire de manches. De nouvelles notions et questions vont alors apparaître venant principalement du fait que vous dépendez de services externes (base de données, authentification). Nous pouvons nous interroger sur leur périmètre et sur la pertinence d'effectuer ces tests de bout en bout. Si tel est le cas, est-ce que ces services sont compatibles avec vos scénarios de tests comme la gestion des états avant et après ? De plus, est-ce qu'une grille de coûts spécifique est prévue par les fournisseurs lors de tests de charge par exemple ?

Si votre volonté est au contraire de vous soustraire à ces services temporairement, il vous faudra alors des systèmes de stub local qui ne sont pas forcément fournis par le fournisseur. Sur ce point, Google se différencie des autres par le fait que ses solutions sont généralement basées sur des systèmes Open Source, de ce fait l'écosystème qui gravite autour fournit assez rapidement des moyens de simuler ces services.

Nous voyons aussi l'émergence de briques comme LocalStack (https://github.com/localstack/localstack) qui permettent de simuler l'ensemble des services AWS. D'autres questions apparaissent alors sur le niveau de confiance que l'on peut avoir dans ces stubs et si un service n'est pas fourni, comment faire pour les implémenter.

Il en va de même pour l'intégration des services FaaS. Il est encore difficile de trouver une implémentation locale de la structure qui embarque certains types de fonctions, mais ce problème tend à se résoudre avec les implémentations des différents frameworks comme Serverless (https://www.serverless.com) ou SAM (https://github.com/awslabs/serverless-application-model) et sa CLI (https://github.com/awslabs/aws-sam-cli). Il va donc falloir utiliser directement l'environnement final. Même si des notions de staging permettent de séparer l'utilisation en test de l'utilisation en production, celles-ci ne s'appliquent pas à tout le catalogue de services Serverless.

Déploiement et versioning

Actuellement, aucun pattern ou outil probant n'est sorti sur la phase de packaging pour garantir l'immutabilité des packages. Nous voyons cependant sortir de nouvelles fonctionnalités d'optimisation de ces packages avec le partage de ressources et de code (AWS Lambda). C'est sur ce type de services que l'on peut observer la plus grande disparité entre les fournisseurs de services.

CONCLUSION

La mise à l'échelle automatique, le pay-as-you-go, la gestion automatique de vos services sont autant de concepts clés qui caractérisent le Serverless. L'utilisation de ce type de services doit être soumis à réflexion afin de comprendre et connaître au mieux vos cas aux limites et de valider son applicabilité. Cette architecture peut ne pas convenir à certaines typologies pour plusieurs raisons comme le besoin de synchronisme dans vos communications. Cependant elle peut, par essence, vous fournir des garanties de haute disponibilité, résilience de vos données, et mise à l'échelle automatique de vos services sans effort supplémentaire de votre part.

Enfin, ce type de services est amené à évoluer rapidement, et le catalogue des produits 100% managés à croître. Chacune des parties (Cloud Providers et utilisateurs) sont gagnantes dans l'utilisation de tels services et garantissent ainsi un avenir florissant qui apportera de l'homogénéité et de la sécurité dans le monde de l'infrastructure et de l'hébergement informatique. •

Yann Coleu
devops
Skale-5

Développement d'applications sur une infrastructure immutable

niveau 100

Quels sont les points à surveiller lorsqu'on développe une application moderne sur une infrastructure immutable, aussi connue sous le terme de Cloud Native ? Quelques réponses dans cet article.

Aujourd'hui, le rythme des fonctionnalités (releases) impose une forte vélocité des développements, ce qui conduit à accélérer le rythme des livraisons, donc raccourcir le risque et le temps issus des déploiements d'un environnement à l'autre. Vient alors l'idée de déplacer la même application d'un bout à l'autre de la chaîne de validation.
Parallèlement, les concepts d'auto-scaling sont des services maintenant largement répandus chez les Cloud Providers, encore faut-il que l'application puisse en profiter …
Autant de raisons pour appréhender les fondamentaux d'un développement Cloud native.
Dans cet article, nous allons rappeler les différents points d'attention à ne pas oublier quand on développe une application qui pourra être déployée dans une infrastructure de production moderne et **Immutable**, donc de dernière génération.

Le paradigme d'infrastructure immutable se caractérise par des applications taguées et cristallisées (Exemple: Le packaging d'une app dans un container Docker) et dont les ressources de calcul sont jetables et éphémères.

Ce changement de paradigme modifie notablement la façon de déployer les applications, et leurs scalabilité par extension des ressources sur une plus large flotte de calcul.
Le point crucial réside dans l'aspect éphémère du concept d'application, on préférera la destruction et la re-création plutôt qu'un soin continu d'un groupe de serveurs ainsi que l'ajout de couches successives au maintien de l'application. On parlera ici du concept « pet vs. cattle ».
(https://medium.com/@Joachim8675309/devops-concepts-pets-vs-cattle-2380b5aab313)
Le monde de l'infrastructure a fortement évolué ces dernières années et les applications doivent respecter plusieurs règles pour exploiter concrètement la puissance de ces nouveaux "datacenters".

La gestion des signaux

Du point de vue du système d'exploitation, toutes les applications prennent la forme de processus. Pour donner une instruction d'arrêt à un processus, le système va lui envoyer un signal TERM. Ce signal ordonne **poliment** à l'application de s'arrêter **dès que possible**. L'application de son côté peut prendre en main le signal et terminer proprement toutes les actions qu'elle est en train d'effectuer. Par exemple, dans le cas d'un service web, il faut s'assurer de terminer toutes les requêtes HTTP en cours de traitement avant extinction du processus. Dans le cas contraire, les utilisateurs vont être sanctionnés d'une ou plusieurs erreurs 500 rendant l'expérience d'utilisation désagréable.
A chaque déploiement, la plateforme va créer de nouvelles instances de l'application puis supprimer les anciennes. Chaque instruction de suppression sera précédée d'une demande d'arrêt (SIGTERM) à l'application. Ce remplacement en roulement permet de changer la version de l'application tout en continuant à servir du trafic. Si les processus ne s'arrêtent pas correctement, certaines requêtes vont échouer pendant la manipulation.

```
go func() {
  wait := time.Second * 30
  sig := make(chan os.Signal, 1)
  signal.Notify(sig, syscall.SIGTERM)
  <-sig
  // Stops here until SIGTERM is caught

  ctx, cancel := context.WithTimeout(context.Background(), wait)
  defer cancel()

  // We received an interrupt signal, shut down.
  if err := srv.Shutdown(ctx); err != nil {
    // Error from closing listeners, or context timeout:
    l.Infof("APP server shutdown error: %v", err)
  }
  l.Infof("APP server successfully shutdown")
}()
```

Dans l'exemple ci-dessus en Go, une goroutine est lancée et se met en attente d'un signal SIGTERM, la fonction srv.Shutdown() ferme proprement les connexions actives en cours sur le serveur.
Peu importe la technologie employée dans le développement d'une application, l'objectif est de s'assurer que le processus réagit parfaitement à la réception d'un signal SIGTERM. Le test se fait facilement avec la commande kill (1)

```
$ ps -a | grep myapp
27229 pts/0    Sl+    0:00 ./myapp
$ kill -TERM 27229
```

Les logs

Les journaux de l'application sont une véritable mine d'or d'informations qu'il faut absolument exploiter en production. Ils peuvent être vus et analysés en deux dimensions, évènementielle et temporelle. La vision **évènementielle** est intéressante car elle renseigne sur une information à un temps donné. Prenons

l'exemple d'une application de gestion de plantes dans une pépinière. Voici ce que pourrait être un message pertinent destiné à un humain en charge des produits à vendre. Le message est clair et à la lecture de l'information, le jardinier corrigera l'alerte en apportant de l'eau au végétal.

```
Attention, 03h21, la plante n°382 manque d'eau
```

La vision des évènements **en séries temporelles** est tout aussi intéressante, si ce n'est plus. Par exemple, un groupe d'évènements intéressera plus le vendeur que les évènements pris unitairement. Il pourrait extraire la somme du chiffre d'affaires de la journée, les horaires où il fait le plus de ventes, etc.

```
Information, 09h24, la plante n°829 a été vendue pour 25€
Information, 10h10, la plante n°026 a été vendue pour 5€
Information, 17h32, la plante n°291 a été vendue pour 40€
```

Pour permettre au pépiniériste d'exploiter les journaux de l'application, il faut ensuite les structurer. Le format le plus adapté est le JSON et les structures pourraient ressembler à ça :

```
{"warn": "info", "time": "0321", "trigger": "need_water", "plant_id": "382"}
{"level": "info", "time": "0924", "trigger": "sold", "plant_id": "829", "price": 25}
{"level": "info", "time": "1010", "trigger": "sold", "plant_id": "026", "price": 5}
{"level": "info", "time": "1732", "trigger": "sold", "plant_id": "291", "price": 40}
```

Il ne reste plus qu'à stocker ces données dans un moteur permettant de les indexer et de les analyser comme Elasticsearch, Logmatic ou autre. Pour ce faire, l'application a juste besoin d'afficher ces lignes sur **la sortie standard**. Des agents comme Filebeat ou Fluentd viendront récupérer le contenu afin de l'injecter dans un moteur de stockage spécialisé.
Attention à ne pas utiliser le disque local pour stocker les fichiers de logs. Les instances des applications étant éphémères, placer de la donnée en local est dangereux voire impossible en production.

La configuration par l'environnement

Dans le paradigme d'infrastructures immutables, les composants applicatifs sont construits une seule fois puis gelés dans une version bien précise. Leurs configurations doivent pourtant changer d'un environnement à l'autre. Qu'il s'agisse d'informations de connexion à une base de données, un niveau de verbosité de logs, etc. Le moyen de plus simple et performant de renseigner à l'application des configurations environnementales est d'utiliser **les variables d'environnement.**

```
prefix := "APP_"
for _, e := range os.Environ() {
  if strings.HasPrefix(e, prefix) {
    // e is in the form key=value
    pair := strings.Split(e, "=")
    k := strings.TrimPrefix(pair[0], prefix)

    // store all keys in lowercase
    c.envs[strings.ToLower(k)] = pair[1]
  }
}
```

Dans l'exemple ci-dessus, en Go, une fonction va parcourir toutes les variables d'environnements passées à l'application. Si la variable commence par "APP_" alors elle stocke la clé et sa valeur correspondante dans un tableau associatif pour une utilisation future.
La véritable force de cette technique est la simplicité de l'implémentation. De plus, le passage d'information via des variables d'environnement est supporté par n'importe quelle plateforme pour lancer des applications (SystemD, Kubernetes, Startup-Scripts chez Google Cloud Platform ou AWS, etc.).
Si toutefois, ce système n'est pas assez évolué ou sécurisé pour les besoins de l'équipe, cette dernière pourrait faire appel à un système de configuration centralisé ou service discovery comme Consul. Le modèle reste le même, bien qu'un petit peu plus complexe à implémenter.
Le point à retenir, quand une application à besoin de configurations, celles-ci doivent absolument provenir de l'environnement sur lequel le service est lancé.

Les sondes de santé

Dans le cadre d'une infrastructure immutable, les instances applicatives sont créées et détruites autant de fois que nécessaire. Le placement et le routage de ces unités se font grâce à des algorithmes de scheduling plus ou moins avancés. Les informations à recueillir pour effectuer un placement optimal sont nombreuses :

- Les besoins de l'application en CPU et RAM
- La cartographie des ressources disponibles
- Les anti-affinités des applications entre elles
- Le datacenter ou zone géographique
- etc.

Quand le scheduler définit et place l'application à un endroit précis, celui-ci doit savoir que l'application s'est correctement initialisée et lancée. L'état du processus système ne définit pas forcément l'état de santé d'une application qui peut bloquer ou réessayer indéfiniment sur quelque chose, tout en restant active au niveau système. C'est donc à l'application d'informer ses pairs qu'elle s'est bien initialisée et que sa tâche de scheduling s'arrête ici.
Pour informer les intéressés sur l'état de santé d'une application, celle-ci doit se munir d'un point d'entrée dédié à son état de santé. Le plus simple pour une application web est d'avoir une route HTTP accessible par exemple sur GET /-/healthy dont le seul but est de répondre par un 200 OK et ainsi indiquer que l'application fonctionne.
Sur certaines plateformes comme Kubernetes, une route supplémentaire sert à savoir si l'application est prête pour recevoir des requêtes et ainsi être placée dans le flux de trafic. Cette route HTTP accessible par exemple sur GET /-/ready doit elle aussi répondre par un 200 OK.
Ces routes sont appelées très régulièrement, il faut donc être extrêmement vigilant sur les performances de ces points d'entrée. De plus, l'application doit définir son propre état de santé **uniquement**. Par exemple, si l'application est dépendante d'une base de données, elle ne doit pas inclure l'état de santé de cette dernière. La raison est simple : si la base de données est défectueuse, toutes les instances de l'application seront en

CONCOURS : FORGE YOUR CODE

by Mobioos Forge Studio®

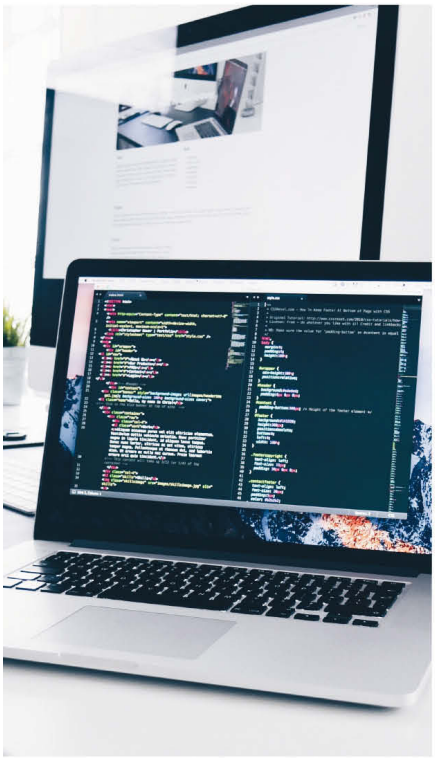
Cet été découvrez l'outil **Mobioos Forge Studio** via un concours de création d'application et tentez de gagner un des prix !

Créez une application en utilisant les capacités de génération de code de Mobioos Forge Studio en choisissant l'une des deux catégories de notre concours : **PRODUCTIVITE** ou **JEU**.

3 mois, 2 catégories, seul ou à 3 maximum :
1 application mobile.

Prérequis :

- Avoir une première expérience en **ASP.Net Core** et **Ionic**.
- Utiliser **Visual Studio 2019** (version gratuite ou payante)



Comment participer ?

1. Constituez votre équipe, d'un maximum de 3 personnes, ou participez seul. Puis choisissez votre catégorie!
2. Inscrivez-vous sur **summerchallenge.mobioos.ai** afin de rejoindre l'espace **Slack** où nous répondrons à vos questions.
3. Téléchargez **Mobioos Forge Studio** sur la Market Place de **Visual Studio https://bit.ly/2JpAoMn**
4. Créez votre application et postez la sur notre store **avant le 8 septembre 2019** à 8h du matin.

Les gagnants seront annoncés dans le numéro de **Programmez!** du mois d'octobre.

Le règlement complet du jeu est sur **summerchallenge.mobioos.ai/rules**

Les prix :

Notre jury, présenté sur le site du concours, choisira les 3 applications gagnantes dans chaque catégorie : **Productivité** et **Jeu**, qui recevront :

1er prix : 2 250 EUR pour l'équipe
2e prix : Oculus Go 64 Gb (1 par personne)
3e prix : 150 EUR carte cadeau Amazon (1 par personne)

mobioos.ai

CONCOURS: FORGE YOUR CODE

en savoir plus sur l'outil

Qu'est-ce que **MOBIOOS FORGE Studio** ?

- Une extension de Visual Studio 2019
- Génération de code de qualité
- Code Factory
- Génération de code multilanguage
- Visual Modeling
- Pas de dépendance au Framework utilisé

Un concours et une technologie à découvrir
La solution Mobioos Forge Studio est le fruit d'une collaboration étroite entre des développeurs ayant plus de 20 ans d'expérience et des laboratoires de recherche en software factory, qui se proposent de répondre au défi de générer des applications à partir d'un modèle sans se soucier du langage utilisé.

Le méta modèle comme fondation de l'application
La plateforme innove et se démarque de la concurrence en s'appuyant sur un outil de modélisation qui ne rentre volontairement pas dans les détails.

En effet la modélisation s'effectue via un manifeste (de type DSL) propre à Mobioos, et couvre toutes les couches d'une application (UI, API, Data, etc.), indépendamment des technologies.

Le méta-modèle permet de définir, entre autres :

- Les fonctionnalités globales de l'application (langues, plateformes supportées, etc.) ;
- Les différents écrans de l'application et les actions associées ;
- Les API disponibles (format REST) ;
- Les bases de données (modèle objet).

Des fonctionnalités de compilation et de validation du méta-modèle permettent de modéliser rapidement, et avec efficacité, l'application tout en garantissant sa cohérence.

Un assistant virtuel à votre service
La conversion du méta-modèle en une application spécifique est réalisée à l'aide d'un assistant de type chatbot.
Il gère, de façon élégante, l'épineux problème délaissé par la méta-modélisation : la spécialisation du modèle vers des complications d'implémentations et la gestion des interactions et contraintes entre les différentes couches d'application.
Après analyse du méta modèle, le chatbot interroge le développeur pour comprendre plus finement les besoins.
Faut-il générer un front end ? Un back end ? Une base de données ?

Chaque réponse peut apporter d'autres questions plus spécifiques, tout en ignorant les questions qui ne seraient pas en phase avec l'application en devenir et l'architecture ciblée.
Cette "interview" permet de dépasser le périmètre fonctionnel et technique du méta-modèle pour aller au fond des vraies problématiques auxquelles ont affaire les développeurs : style d'architecture, Framework, nomenclature du code, tests unitaires, etc.
C'est ce qui différencie en profondeur Mobioos Forge des outils MDA complexes et des produits existants de low code / no code.

Les générateurs de code : une extension du méta modèle

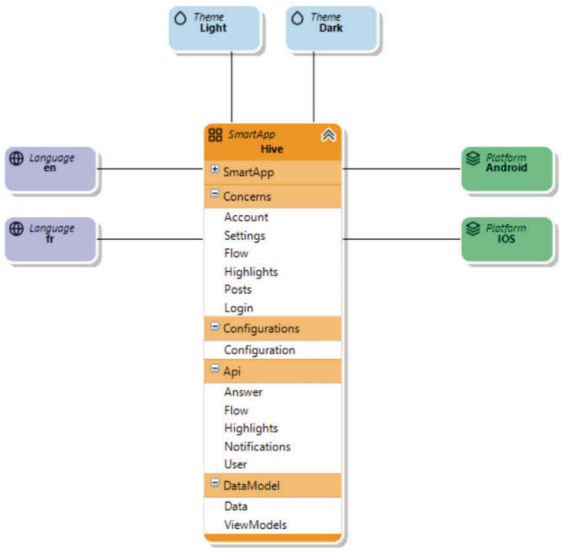
Les générateurs de code jouent un rôle prépondérant durant l'interview. Ils interagissent directement avec le développeur pour réclamer des réponses techniques propres à la technologie qu'ils génèrent. Après l'interview, la génération du code est déléguée aux générateurs. Chaque générateur prend en compte les réponses fournies par l'interview et s'adapte aux contraintes édictées.

L'EndoCode : Un gage de qualité du code

A la fin d'une interview, Mobioos Forge Studio génère 60 à 70% du code. Le code ainsi généré se nomme l'EndoCode, c'est à dire endogène aux générateurs.
De par la fiabilité fournie par les générateurs, 100% du code généré est dit « zéro distorsion ». C'est à dire testé, fiable, robuste, sans bugs et sans dette technique.

Le développeur, porteur de richesse et créativité

Le code développé à la suite d'une génération de code se nomme l'ExoCode, car exogène aux générateurs. La distinction entre EndoCode et ExoCode permet de bien délimiter le périmètre de responsabilité entre l'apport réalisé par Mobioos Forge Studio et celui que doit apporter le développeur.
Ce code est particulièrement relatif aux besoins métiers et aux ajustements. L'ExoCode reflète toute la richesse, la puissance et la créativité des développeurs, là où la machine n'est pas capable d'apporter de solutions



Suivez le code QR ou visitez
summerchallenge.mobioos.ai
pour vous inscrire au concours **Forge Your Code**

LE PRINCIPE

Modélisation › Orchestration › Génération de code › Mise en oeuvre › Finalisation

Mobioos FORGE Studio | Outils IDE et ALM

Visual Studio 2019

mauvaise santé, obligeant l'orchestrateur à recréer d'autres instances indéfiniment.

Pour que votre application soit scalable horizontalement, vous ne devez stocker aucun état persistant et aucune données sur disque.

Exemple code non scalable / scalable :

Contexte :
Store exemple de package
Function NewLocalStore () => Stockage d'objet sur disk
Function NewDataStore () => Stockage d'objet sur datastore (BDD Nosql sur GCP)

Exemple:
Code non scalable stockage sur disque :

```
func main() {
 ....
 store := store.NewLocalStore(JSONEncoding)
 if err := store.Save("numbers", []int{1, 2, 3}); err != nil {
   // Handle err
 }
}
```

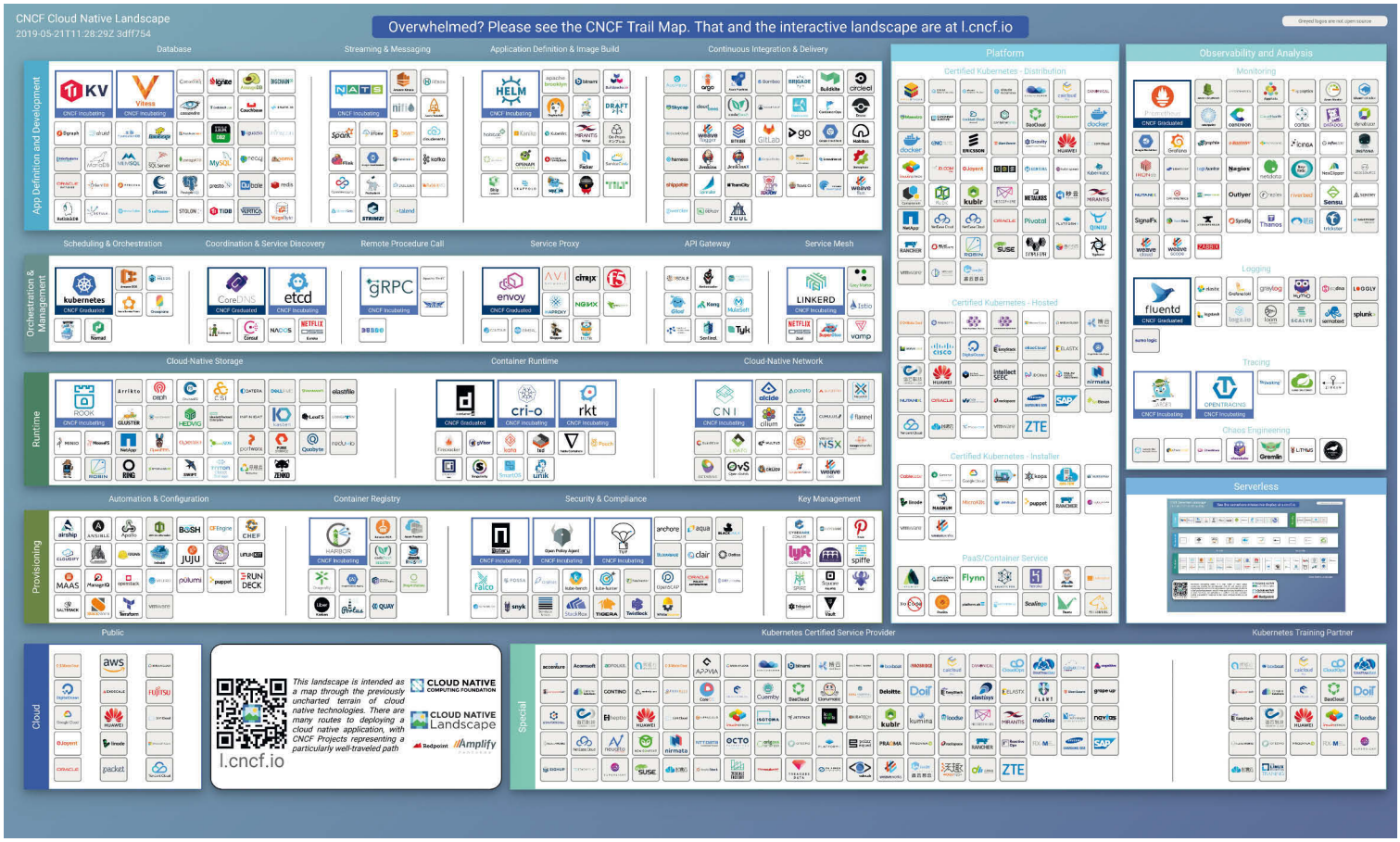
Code scalable stockage de data sur Datastore :

```
func main() {
...
  store := store.NewDataStore(JSONEncoding)
  if err := store.Save("numbers", []int{1, 2, 3}); err != nil {
    // Handle err
  }
}
```

Conclusion

Avec quelques règles simples dans le code, les développeurs sont capables d'avoir un impact extrêmement positif quant à l'utilisation des dernières générations d'infrastructures, améliorant ainsi le comportement et l'analyse de l'application en Production.
Issue des investissements colossaux consentis dans la gestion des infrastructures de nouvelles générations, une ère nouvelle s'ouvre aux développeurs qui sont maintenant acteurs de premier plan du fonctionnement des infrastructures. La connaissance et l'utilisation des Services Managés bénéficient directement à la scalabilité et la résilience de leur application. Pour bien débuter, il suffit de respecter ces quelques règles :

- Gestion des SIGTERM ;
- Logs en sortie standard / erreur au format JSON ;
- Chercher les configurations dans l'environnement ;
- Exposer l'état de santé via des endpoints HTTP.



Le paysage du cloud native selon la Cloud Native Computing Foundation. Version valable à l'impression du numéro.

Interview de Bastien Legras

DIRECTEUR TECHNIQUE GOOGLE CLOUD

Google Cloud Platform est désormais la 3e plateforme cloud du marché. Largement distancée, elle a su se repositionner et compléter son offre. Revenons avec Bastien sur les principales nouveautés du GCP et l'offre Google.

Il y a à peine 3 ans, Google Cloud Platform semblait être en retard, et, depuis, les nouvelles fonctionnalités apparaissent régulièrement. La dernière conférence NEXT a encore dévoilé de nombreuses évolutions. Comment compareriez-vous GCP aux prochaines plateformes du marché ?

Nos dernières annonces réalisées à notre conférence Next'19 en avril confirment effectivement notre volonté de solutions qui soient à la fois constamment plus performantes et pertinentes par rapport aux besoins de nos clients et aux tendances du marché.

Anthos notamment, notre nouvelle plateforme ouverte qui permet d'exécuter une application de n'importe où - simplement, avec souplesse et en toute sécurité - entre assurément dans cette logique. En adoptant les normes ouvertes, Anthos permet d'exécuter ses applications, sans modification, sur des investissements matériels existants sur site ou dans le cloud public.

Nos clients reconnaissent par ailleurs notre machine learning qui s'appuie sur la gestion du big data tout en ayant une approche démocratisée et facilitée du sujet.

Ces annonces confirment en fait les éléments qui font notre différence depuis les tout débuts de notre activité cloud : une approche résolument hybride, un engagement véritable et historique sur l'open source, la sécurité et la fiabilité nécessaires et évidentes, et le fait que le numérique fait partie de notre ADN. Le cloud est à la racine de Google.

Vous avez été les premiers à casser les frontières entre l'IaaS et le PaaS et à proposer un hybride entre les deux mondes. Aujourd'hui, on parle de serverless, de conteneurs, d'infrastructure as code. Quels sont les objectifs de cette approche ? Est-elle adaptée à tous les usages et développement ?

Effectivement, et on en parle beaucoup d'ailleurs, et il faut agir. C'est ce que nous avons fait avec Cloud Run, présenté également à Next : une offre totalement serverless qui s'occupe de la gestion de l'infrastructure, dont notamment le provisioning, la configuration, la scalabilité scaling et la gestion des serveurs. Il augmente ou réduit ses capacités en quelques secondes selon le trafic et permet de s'assurer que la facture corresponde très précisément à l'utilisation des ressources. Veolia l'a du reste mis en place avec succès. Cloud Run peut également s'appuyer sur GKE, ce qui signifie que vous pouvez exécuter des charges serverless sur vos clusters GKE existants et de façon simultanée, abstraire des concepts complexes de Kubernetes.

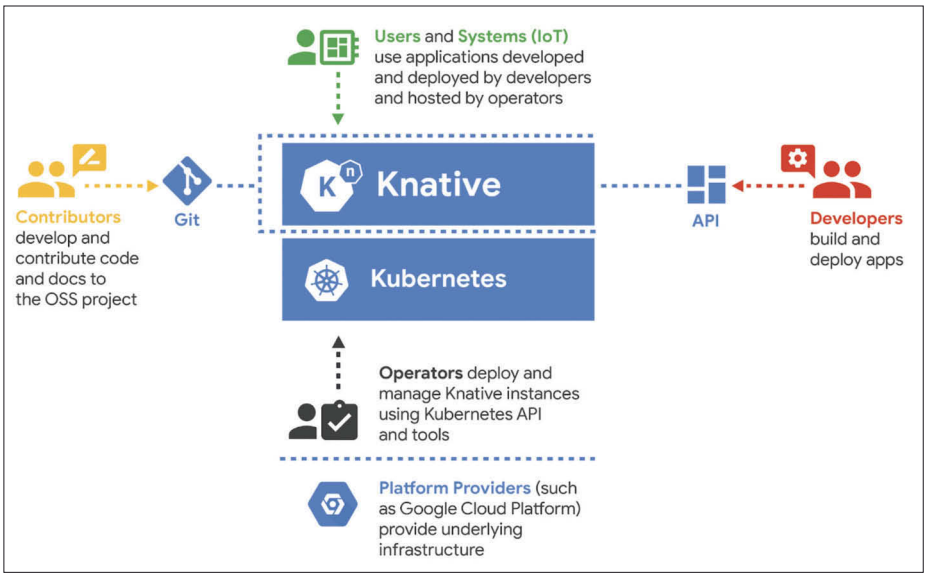
Notre approche est du reste la même que lorsque nous avons lancé Kubernetes: permettre aux développeurs d'écrire, déployer et deboguer simplement et efficacement les applications. L'objectif est de fournir un environnement ouvert, consistant et hybride.

Cette approche permet ainsi soit de migrer ses applications dans le cloud, comme l'ont fait récemment Amadeus ou BlaBlaCar, ou de développer de nouvelles applications directement dans le cloud comme Total peut le réaliser avec ses projets en intelligence artificielle.

Cette approche convient d'ailleurs bien sûr aux entreprises du numériques, mais également aux grands groupes qui nous interrogent et nous plébiscitent de plus en plus.

Comment aujourd'hui peut-on migrer des apps, des codes sur le cloud ? Est-ce conseillé ou faut-il le faire progressivement, ou en faisant du replatforming par exemple avec des services back dans le cloud ?

Le cloud peut être utilisé pour tous les projets : les nouveaux bien sûr comme les anciens, et donc avec une migration. Notre nouvelle plateforme intègre ce que nous avons appelé Anthos Migrate, disponible en version beta, qui permet de migrer automatiquement les machines



virtuelles ou VMs sur site, ou d'autres nuages, directement dans des conteneurs en GKE, en un minimum d'effort. Une seule étape simplifiée est nécessaire pour migrer son infrastructure, sans modification initiale des machines virtuelles ou des applications d'origine. Les tâches fastidieuses d'infrastructure telles que la maintenance des machines virtuelles et les correctifs d'OS, sont facilitées, permettant de se concentrer sur la gestion et le développement des applications.

Autre point pour migrer plus facilement : Velostrata, qui permet une migration transparente en arrière plan, depuis une infrastructure sur site ou depuis d'autres clouds. La technologie permet d'aller plus vite et plus simplement sur les projets de migration de masse, d'autant que les entreprises peuvent valider, exécuter et migrer des applications dans Google Cloud sans avoir besoin de les modifier.

Knative a été une de vos annonces phares, comment pourrait-on le présenter ? Sommes-nous plutôt côté ops ?
Il est vrai que Knative a fait le show, avec Anthos et Cloud Run. Knative c'est un ensemble open source qui permet aux développeurs de construire, déployer et gérer des charges de travail serverless, n'importe où : que ce soit sur site, dans des serveurs tiers et bien sûr sur Google Cloud Platform. Le focus est sur les aspects certes habituels mais difficiles d'exécution des applications, tels que le routing, l'orchestration, la gestion du trafic pendant le déploiement ou l'auto scaling. Knative permet aux opérateurs de construire leur propre plateforme serverless en utilisant ces éléments et s'appuie sur Kubernetes pour plus de flexibilité pour le développeur.
On est là clairement dans la mouvance DevOps en termes de technologie, mais également d'outils mis à disposition.

Pour le dév, kubernetes, le serverless, l'infra as code, le cloud native ou hybride ce sont parfois des notions obscures voire nébuleuses, finalement quels sont les avantages pour eux ?
Le serverless permet aux développeurs d'être plus productifs en se concentrant sur le développement de l'application même sans se soucier de la gestion de l'infrastructure sur laquelle elle va reposer. Nous faisons du reste des investissements continus sur le serverless pour l'ensemble de notre offre de production. Nous avons annoncé à Next'19 Cloud Run, une nouveauté qui vient compléter notre plateforme serverless ainsi que de nouvelles mises à jour à nos offres clés que sont App Engine et Cloud Functions. Ces produits, conjugués à Knative dont je viens de parler, permettent aux développeurs d'écrire le code de la manière dont ils le souhaitent (en fonctions, en apps ou containerisées) et de déployer des charges de travail serverless là où ils le souhaitent : dans Google Cloud bien sûr, sur site ou dans des data centers tiers.

Au niveau programmation, architecture logicielle, qu'est-ce que le développeur doit absolument comprendre pour bien faire les choses ?
En fait, et c'est véritablement l'évolution principale qui est permise par nos technologies, le développeur peut, voire même doit, se concentrer sur les objectifs de son application, sur sa structure pour définir le code et le rendre évolutif. On passe en fait de l'architecture à la structure, et du logiciel aux microservices, le tout permettant de faciliter la gestion du matériel et de l'opérationnel pur et dur pour se concentrer sur les aspects les plus stratégiques du développement, et les plus évolutifs aussi, ce qui rend l'ensemble passionnant.

Comment déboguer efficacement des apps, des workloads cloud ? Qu'est-ce qui est disponible pour les dévs ?
Nous avons une fonctionnalité de débogage qui permet d'inspecter l'état d'une application en temps réel, sans interrompre ni ralentir son exécution. Le développeur peut capturer la pile des appels et les variables à l'emplacement qu'il choisit dans le code source sans que cela n'ait aucune incidence sur les utilisateurs. Cette fonction est multisources, très simple à mettre en place et peut également être utilisée de façon collaborative avec les autres membres de l'équipe.
Et bien sûr Kubernetes propose également des fonctions de débogage en local ou à distance.

La sécurité est un thème récurrent et c'est même un des critères mis en avant par Google. Quels sont les principaux points qui intéressent le développeur ?
D'une part sur le principe, les données des clients de Google Cloud leur appartiennent, exclusivement. Nous n'y touchons pas, n'y accédons pas et en assurons la sécurité. Les données ne sont exploitées que dans le cadre du service rendu à nos clients. D'autre part, nous opérons dans le cloud et le sécurisons depuis bientôt 20 ans, en tirant parti notamment de 8 applications utilisées par plus d'un milliard de personnes. La sécurité et la disponibilité sont des points essentiels pour nous, nous sommes transparents sur le sujet et c'est pour nous un sujet constant.
Très concrètement, nous entendons assurer de façon simple et efficace la sécurisation des applications, ce qui permet in fine leur mise sur le marché plus rapide.
Si nous sécurisons le Cloud, nous permettons également à nos clients de sécuriser "dans le Cloud" : Nous fournissons un ensemble de contrôles pour que nos clients sécurisent leurs applications. Cela passe tout d'abord par la gestion des identités, accès et permissions (Cloud Identity), par la sécurisation du réseau (gestion de VPC, firewall, protection DDOS avec Cloud Armor) et la mise à disposition de chiffrement en transit et en repos par défaut. Nous proposons également des outils pour sécuriser la chaîne de développement logiciel : contrôle de vulnérabilité des images, contrôles de sécurité dans le déploiement (Binary Authorization), analyse automatique des failles logicielles telles que XSS (Cloud Security Scanner), utilisation de machines virtuelles et systèmes d'exploitation sécurisés (Shielded VM). Enfin, nous proposons également un ensemble d'outils de surveillance, de débogage et d'audit qui donnent à nos clients une visibilité complète sur leur environnement (Stackdriver, Cloud Audit Logging, Access Transparency).

Finalement, comment faire du secure by design en cloud ?
Si on prend App Engine par exemple, on peut paramétrer la gestion des identités et des points d'accès, la gestion de trafic de points ou de réseaux ou services spécifiques, bloquer certaines adresses IP ou encore analyser pour définir et remédier à d'éventuelles failles. On n'est bien ici dans le secure by design.
L'utilisation du Cloud facilite la mise en place des principes de la sécurité en profondeur et d'architectures de type "zero-trust". On peut notamment citer la plateforme Cloud Service Mesh qui garantit que les communications de service sont sécurisées par défaut avec peu ou pas de modifications apportées aux applications.
Dans l'ensemble, pour nous, il est nécessaire d'intégrer la sécurité dès la conception puis le développement d'une application, plutôt que l'ajouter après. Cela semble une évidence, ce n'est pourtant pas toujours le cas. •

La demande pour Ruby explose, Java et PHP s'enfoncent

Alors que les entreprises ont manifestement eu un grand besoin de spécialistes du Ruby au mois d'avril, les technologies Java et PHP semblent devenir des technologies qui intéressent de moins en moins les recruteurs, à en croire le baromètre mensuel Hired de la recherche des développeurs.

En avril 2019, les candidats référencés sur la plateforme Hired ayant mis en avant la technologie Ruby se sont vu proposer en moyenne 10 entretiens. C'est 4 de plus que pour la deuxième technologie la plus demandée, le Go, qui est abonné au podium depuis plusieurs mois. La forte demande pour Ruby s'accompagne en outre d'une grande rareté des profils. Seul 1,37 % des candidats inscrits en avril sur Hired au mois d'avril l'avaient mise en avant. Sur les 6 derniers mois, cette proportion est de 3,24 %. Les technologies React, Node et .NET complètent ensuite le top 5 avec respectivement 5,7, 5,6 et 5,6 demandes d'entretiens reçues par les candidats qui les avaient mises en avant. Dans le fond du tableau, on retrouve iOS (3) et Android (3,6), des technologies pour lesquelles la demande fluctue énormément. Elles étaient dans le top 3 il y a encore quelques mois. Elles sont juste devancées par les Java et PHP, toutes deux créditées de 4,2 demandes d'entretiens. Ces deux technologies semblent toutefois devenir abonnées au fond du classement, contrairement à celles qu'elles devancent. À noter d'ailleurs qu'elles comptent parmi les plus représentées sur Hired, Java étant mis en avant par 21,92 des inscrits en avril et PHP par 10,76 % d'entre eux. Du côté des autres technologies, Node, React et Python restent les plus représentées aux côtés de Java et PHP. Elles totalisent à elles cinq plus de 70 % des candidatures enregistrées sur la plateforme en février. De l'autre côté du classement, iOS (2,45 %) et DevOps (0,99 %) ferment le classement avec le Ruby.

Tous les mois, Programmez ! publie en exclusivité le baromètre Hired des technologies les plus recherchées par les entreprises. Elles peuvent permettre aux développeurs de connaître les nouvelles tendances du recrutement pour se former ou se démarquer des autres candidats.

Avril 2019

Technologies demandées	Pourcentage de candidatures développeurs	Technologies demandées	Nombre moyen de demandes d'entretiens
Java	21,92%	Ruby	10,00
Node	14,29%	Go	6,00
React	13,31%	React	5,72
Python	12,33%	Node	5,59
PHP	10,76%	.NET	5,58
Angular	8,22%	Vue	5,46
Android	5,28%	Angular	5,21
,NET	4,70%	DevOps	4,80
Go	4,31%	Python	4,38
Vue	2,54%	PHP	4,22
Ruby	1,37%	Java	4,21
iOS	1,37%	Android	3,78
DevOps	0,98%	iOS	3,00

De Novembre à avril 2019

Technologies demandées	Pourcentage de candidatures développeurs	Technologies demandées	Nombre moyen de demandes d'entretiens
Java	19,56%	Vue	7,28
Node	13,88%	React	6,64
React	13,22%	Node	6,60
Python	13,15%	Ruby	6,49
PHP	11,50%	Go	6,42
Angular	7,87%	Angular	5,87
Android	5,09%	iOS	5,41
,NET	4,36%	Python	5,36
Go	3,64%	,NET	5,36
Vue	3,50%	Android	4,99
Ruby	3,24%	PHP	4,92
iOS	2,45%	DevOps	4,80
DevOps	0,99%	Java	4,74

The Hacking Project : bootcamp en pleine croissance

The Hacking Project est un bootcamp d'apprentissage de la programmation ouvert à toutes et à tous. Aujourd'hui, il est présent dans 22 villes et a déjà formé +1200 personnes. Pour cette année, l'ambition est de dépasser les 30 villes et 2000 personnes formées !

Développeur : un métier recherché ?

Pôle Emploi a publié son enquête « Besoins en main d'œuvre » édition 2019. Les métiers de l'informatique (ingénieur, R&D, chefs de projet, etc.) arrivent à 5e place des projets de recrutement en 2019. Cette place se situe dans le domaine des recrutements non saisonniers. Le chiffre avancé est -47 000. Les recrutements d'agents d'entretien sont en tête avec +86 900 demandes !

… mais un chômage en croissance !

Ce chiffre est théorique et la réalité varie durant l'année. En 2018, Syntec Informatique estimait à environ 28 000 postes nets créés dans l'informatique. Mais un autre chiffre doit inquiéter et interroger : +51 000 informaticiens inscrits à Pôle Emploi fin 2018. Sous le terme informaticien, on retrouve l'ensemble des profils mais certains sont bien plus dynamiques que d'autres.

Le marché est toujours tendu notamment sur les profils développeurs mais attention à ne pas faire croire n'importe quoi et qu'il y a forcément un poste à la fin d'une formation.

Index TIOBE

Quoi de neuf dans l'index TIOBE ? On retrouve grosso modo les mêmes langages, dans l'ordre : Java, C, C++ et Python. PHP connaît tendance à baisse tout comme C#. Au-delà, Objective-C reprend quelques places (11e) et devance toujours Swift (18e).

Mai 2019	Mai 2018	Tendance	Langage	en %	Evolution en %
1	1	=	Java	16.005%	-0.38%
2	2	=	C	14.243%	+0.24%
3	3	=	C++	8.095%	+0.43%
4	4	=	Python	7.830%	+2.64%
5	6	↗	Visual Basic .NET	5.193%	+1.07%
6	5	↘	C#	3.984%	-0.42%
7	8	↗	JavaScript	2.690%	-0.23%
8	9	↗	SQL	2.555%	+0.57%
9	7	↘	PHP	2.489%	-0.83%
10	13	↗	Assembleur	1.816%	+0.82%

Robin Huart
Consultant sénior
INVIVOO

Les slots, une optimisation méconnue

En tant qu'ancien développeur C travaillant dans le calcul hautes performances, je me suis très tôt et naturellement posé la question de la compacité des objets que l'on créait ordinairement en Python. Il m'est apparu rapidement que celle-ci n'était pas optimale dans la plupart des cas, ce qui résulte de choix de conceptions originels voulus et assumés.

niveau 200

Néanmoins, j'ai découvert à cette occasion qu'il existait une façon de résoudre en partie ce problème depuis l'apparition des nouvelles classes en Python 3 : recourir à des déclarations d'attributs appelés slots.

Quelques rappels et observations

Lorsque nous écrivons une classe en Python, nous ne procédons pas à une déclaration statique de ses attributs. Au mieux, dans la méthode *__init__* de l'objet, nous initialisons un certain nombre d'attributs d'instance. Rien ne nous empêche cependant d'en rajouter à la volée par la suite. Illustrons ce propos :

```
>>> class MaClasse:
...     def __init__(self, a, b):
...         self.a = a
...         self.b = b
...
>>> mon_objet = MaClasse(1, 2)
>>> mon_objet.c = 3
```

On voit que l'interpréteur accepte que je rajoute l'attribut c. Comment cela fonctionne-t-il ? Sans nous étendre sur les subtilités des accès aux attributs, c'est l'existence d'un attribut spécial appelé *__dict__* qui rend l'opération possible. En effet, la dernière ligne de code est équivalente à :

```
>>> mon_objet.__dict__['c'] = 3
```

Tous les attributs d'instance rajoutés via une instruction du type *self.x*, que ce soit dans la méthode *__init__* ou ailleurs, finissent dans ce dictionnaire. C'est à partir de cette simple observation que j'ai commencé très tôt à me poser la question de la performance et de la compacité en mémoire des objets ainsi construits. Puisqu'ils sont extensibles, leur implémentation ne peut être optimale au regard de ces critères. CPython est l'implémentation principale de Python qui nous intéresse ici. Elle est obligée d'allouer de la mémoire à l'aveugle, une première fois. Puis d'autres, selon une heuristique que nous n'avons pas besoin de connaître a priori. Et pour ce qui est des performances, devoir aller chercher un attribut dans un dictionnaire Python n'est sans doute pas non plus la manière le plus optimale. C'est en réalité principalement pour cette question de performance que Guido van Rossum a créé les *slots*.

Présentation

Les *slots* sont une manière de **déclarer** les attributs d'instance qui composent nos objets. Depuis le début, j'ai pris soin de parler d'attributs d'instance pour bien faire la distinction avec les autres attributs accessibles depuis un objet, comme les méthodes qu'on peut lui appliquer par exemple, qui sont des attributs de classe.
Dans l'exemple précédent, j'ai défini trois attributs au cours de la vie de mon objet. Voyons comment redéfinir celui-ci avec des *slots*.

```
>>> class MaClasseSlottee:
...
...     __slots__ = ('a', 'b', 'c')
...
...     def __init__(self, a, b):
...         self.a = a
...         self.b = b
...
>>> mon_objet_slotte = MaClasseSlottee(1, 2)
>>> mon_objet_slotte.c = 3
```

C'est tout. Il suffit de rajouter un attribut de classe *__slots__* renvoyant à un itérable avec les noms des attributs d'instance que l'on souhaite manipuler. Aucun autre ne sera accepté :

```
>>> mon_objet_slotte.d = 4
Traceback (most recent call last):
  File "<input>", line 1, in <module>
AttributeError: 'MaClasseSlottee' object has no attribute 'd'
```

Effectivement, l'attribut *d* ne figure pas dans la liste de ceux que nous avons déclarés. Quelles sont les autres conséquences de l'ajout de *__slots__* ? D'abord, la disparition pure et simple de *__dict__* :

```
>>> mon_objet_slotte.__dict__['c']
Traceback (most recent call last):
  File "<input>", line 1, in <module>
AttributeError: 'MaClasseSlottee' object has no attribute '__dict__'
```

Ensuite, les gains en termes d'occupation mémoire et de performance. Le premier aspect est difficile à mesurer en Python, car la fonction *getsizeof* du module sys, prévue à cet effet, se base sur une méthode magique *__sizeof__* dont le comportement par défaut ne va pas procéder à une inspection profonde de l'objet. Voyons ce que cela donne :

```
>>> getsizeof(mon_objet)
56
>>> getsizeof(mon_objet_slotte)
64
```

À première vue, l'ajout des *slots* résulte en un objet plus lourd en mémoire. À première vue seulement. En réalité, c'est juste qu'on a omis un petit détail. Le premier objet que nous avons créé garde ses attributs dans son *__dict__*, alors que ce n'est pas le cas pour le second, puisque son *__dict__* a disparu. Or *getsizeof* appliqué sur l'objet entier ne mesure pas l'occupation mémoire du contenu de *__dict__*. Vérifions par nous-mêmes :

```
>>> getsizeof(mon_objet.__dict__)
112
```

Effectivement vu sa taille, *__dict__* n'était pas mesuré par getsizeof. Cela est en revanche aussi vrai pour *__slots__* dans le second objet, qu'il faut donc compter pour être équitable :

```
>>> getsizeof(mon_objet_slotte.__slots__)
72
```

La comparaison qui a vraiment du sens est donc plutôt 56 + 112 contre 64 + 72 (soit 168 contre 136) ! Comme promis, on observe bien un gain. Néanmoins, il est essentiel de relativiser, car, plus on rajoute d'attributs plus la différence entre les deux approches devient anecdotique par rapport à la taille totale de l'objet. C'est la raison pour laquelle on peut souvent lire et entendre que les *slots* sont utiles quand on doit travailler avec **beaucoup** d'instances d'une même classe dont la taille est relativement petite. C'est vrai, mais pas seulement ! En réalité, comme je l'évoquais plus haut, la principale raison qui a poussé à créer ce système est la question de la performance. Le fait d'éliminer la recherche dans un dictionnaire et d'appeler à la place un descripteur rend l'accès aux attributs plus rapide. On peut mesurer cet effet de manière relativement simple en profilant un bout de code répétant des opérations de lecture/écriture :

```
>>> import timeit
>>> def test(avec_slots=False):
...     instance = MaClasse(1, 2) if not avec_slots else MaClasseSlottee(1, 2)
...     def repeat_basic_operations(instance=instance):
...         for _ in range(10):
...             instance.a, instance.b = 3, 4
...             instance.a, instance.b
...     return repeat_basic_operations
...
>>> for _ in range(10):
...     avec_slots = min(timeit.repeat(test(avec_slots=True)))
...     sans_slots = min(timeit.repeat(test()))
...     printf(f'Le test sans slots prend {(sans_slots/avec_slots - 1)*100:.4}% de temps en plus')
...
Le test sans slots prend 15.21% de temps en plus
Le test sans slots prend 14.62% de temps en plus
Le test sans slots prend 14.76% de temps en plus
Le test sans slots prend 16.23% de temps en plus
Le test sans slots prend 15.6% de temps en plus
Le test sans slots prend 14.8% de temps en plus
Le test sans slots prend 18.42% de temps en plus
Le test sans slots prend 15.78% de temps en plus
Le test sans slots prend 15.76% de temps en plus
Le test sans slots prend 15.27% de temps en plus
```

Ces temps ont été obtenus à partir d'une version 3.6.7 de Python sur un MacBook Pro.

Inconvénients

Tout n'est malheureusement pas parfait et je vais maintenant évoquer les problèmes que je considère comme majeurs avec cette technique.

Écartons d'emblée l'impossibilité de rajouter dynamiquement des attributs aux objets grâce à *__dict__*, car c'est justement ce qui est à l'origine des gains observés. Mais surtout, parce qu'il est tout à fait possible de rajouter *__dict__* parmi les *__slots__* ! On aboutit alors à une sorte d'optimisation partielle qui ne concerne que les attributs qui sont déclarés parmi les *slots*, tandis qu'on peut rajouter dynamiquement tous les autres attributs qu'on souhaite.

Le problème majeur concerne plutôt l'héritage et la réutilisabilité du code. Il y a trois règles à avoir à l'esprit lorsqu'on entreprend de rajouter des *slots* dans un arbre d'héritage :

- Les *slots* d'une classe mère s'ajoutent à ceux de la classe fille ;
- Il ne peut exister qu'une seule classe mère avec une séquence de *slots* non vide ;
- Il suffit qu'une classe dans l'arbre d'héritage omette de déclarer des *slots*, même vides, pour que les instances qui en résulteront aient un *__dict__* .

On peut donc factoriser un certain nombre de *slots* dans une classe mère, ce qui peut s'avérer très utile. En revanche, les deux derniers points posent problème. Lorsqu'on veut définir des *slots* dans une classe, cela implique de repasser dans toutes les classes mères pour leur ajouter soit des *slots* pertinents, quitte à y introduire un *__dict__* s'il est absolument nécessaire de pouvoir rajouter dynamiquement des attributs (malgré les pertes de performance qu'on a vues précédemment).

Dans les cas d'héritage multiple, du fait du second point, cela signifie qu'on se retrouve confronté à une situation difficile. Soit on peut déclarer une séquence de *slots* vides, car la classe n'apporte pas d'attribut (seulement des méthodes), soit nous n'avons aucun moyen d'empêcher la création d'un *__dict__*. Ce dernier inconvénient se retrouve également si les classes dont on hérite se trouvent dans une bibliothèque qui n'a pas prévu ce cas d'utilisation et que nous ne pouvons pas l'éditer. Pour ceux qui ont déjà un peu navigué dans le code de certains modules de la bibliothèque standard (je pense par exemple à *typing* et *collections*) qui définissent des nouveaux types d'objets, c'est la raison pour laquelle vous trouverez beaucoup de définitions de *slots*, la plupart du temps vides.

Conclusion

On a vu que les *slots* sont une technique d'optimisation potentiellement très simple à mettre en œuvre. D'expérience, je sais qu'il n'est pas rare du tout d'avoir des objets dont la structure ne varie jamais et qui dérivent soit directement de la classe *object*, soit d'une ou deux classes mère(s) qu'on peut éditer librement. Dans ces cas-là, je ne vois aucune raison de ne pas profiter des *slots*. D'éventuelles classes filles pourront choisir de réutiliser cette mécanique ou non, libre à leur développeur. Le seul point potentiellement gênant serait que la classe qu'on est en train d'écrire se retrouve à être utilisée comme *mixin* au milieu d'autres qui déclarent des *slots* non vides, mais c'est loin d'être un cas fréquent. Quoi qu'il en soit, gardons à l'esprit qu'une optimisation prématurée est la source de nombreux maux. La meilleure façon de procéder consiste très certainement à rajouter cette optimisation a posteriori et progressivement, en profilant son code, encore et toujours. •

Christophe Villeneuve
Consultant IT pour Hello-design, Mozilla Rep, auteur du livre "Drupal avancé" aux éditions Eyrolles et auteur aux Editions ENI, PHPère des elePHPants PHP, membre des Teams DrupalFR, AFUP, LeMug.fr (MySQL/MariaDB User Group FR), Drupagora...

Une architecture PHP optimisée

Les serveurs web s'appuient principalement sur une architecture AMP (Apache – MySQL – PHP) et ils doivent répondre aux problématiques du « web moderne » tout en garantissant un niveau élevé de performances et de disponibilité des applications. PHP a su répondre aux problématiques d'aujourd'hui des serveurs.

niveau 200

PHP permet de concevoir des sites et des projets internet en utilisant des frameworks, CMS... Mais il peut aussi s'appuyer sur des technologies Open Source complémentaires. Il peut s'interfacer avec les outils 'modernes' disponibles pour améliorer la productivité. L'article va vous montrer un ensemble de techniques pouvant être utilisées dans un développement.
Bien sûr, les sites web d'aujourd'hui (en PHP) à fort trafic doivent répondre aux problématiques modernes et délivrer toutes les X secondes ou minutes, sans perturber l'accès à leurs contenus pour les internautes. Une occasion de voir comment se comporte le langage dans la haute performance.

Prérequis

Pour notre article exemple, nous utiliserons un environnement classique LAMP :

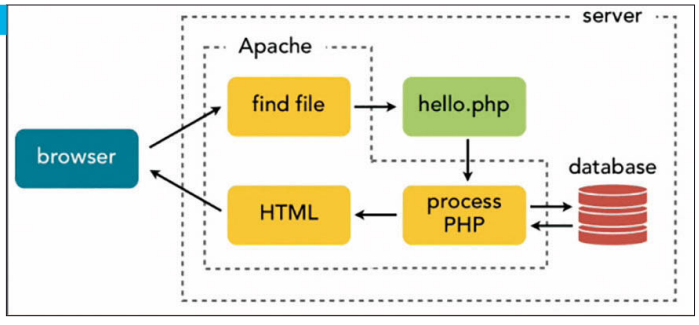
- Linux
- Apache
- MySQL / MariaDB
- PHP - FPM

Cette configuration serveur répond à un problème dit « moderne » pour fournir des données venant des requêtes HTTP, côté front.

Qu'est ce php-fpm ?

PHP-FPM signifie « Fast Process Manager », c'est à dire que PHP s'exécute sur certains processus PHP générés, contrôlés et tués ; ce qui est différent par rapport à un serveur processus unique. Toutefois, le gestionnaire de processus FPM est un processus auquel le serveur Web transmet les demandes.
Le principe d'exécution s'effectue lorsque le navigateur envoie une demande à un serveur PHP. PHP ne constitue pas le point de départ lorsqu'une page web est appelée ; il s'agit plutôt du serveur HTTP, géré par Apache ou par Nginx. Ces serveurs Web doivent ensuite décider comment se connecter au langage pour lui transmettre les informations demandées comme une requête, les données, les en-têtes. 1

1



Lorsque le serveur appelle les pages PHP, la partie trouvée sera le fichier index.php sur lequel le serveur est configuré pour déléguer toutes les demandes.
Jusqu'à présent, l'opération consiste à démarrer un nouveau processus, qui inclut automatiquement PHP, et s'exécute. Cette méthode utilise le module mod_php, qui signifie « php en tant que module », que le serveur Apache utilise et auquel il répond à chaque demande reçue qui provoquait des limites.
Aujourd'hui, le serveur se connectant à PHP peut le faire avec php-fpm qui est proposé par Nginx 1 ou Apache 2.4.
Php-fpm aura la responsabilité de la gestion de PHP pour gérer les processus du langage sur le serveur.
Dans php-fpm pour la responsabilité de la gestion de PHP, les processus incombent au programme PHP sur le serveur. En d'autres termes, le serveur Web ne se soucie pas de savoir où se trouve PHP ni comment il est chargé, tant qu'il sait comment envoyer et recevoir des données.
Ainsi, le serveur peut gérer des charges importantes avec des milliers de connexions simultanées. La configuration se représente de la manière suivante :

Serveur Apache :
Pour le serveur Apache, nous configurons le fichier /etc/apache2/mods-enabled/fastcgi.conf

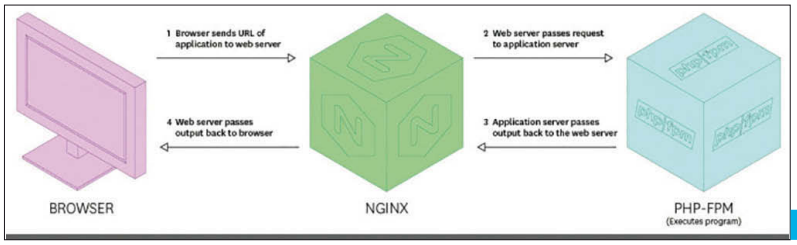
```
<IfModule mod_fastcgi.c>
    AddHandler php7.2-fcgi .php
    Action php7.2-fcgi /php7.2-fcgi
    Alias /php7.2-fcgi /usr/lib/cgi-bin/php7.2-fcgi
    FastCgiExternalServer /usr/lib/cgi-bin/php7.2-fcgi -socket /var/run/php/php7.2-fpm
.sock -pass-header Authorization
</IfModule>
```

Serveur Nginx :
Pour le serveur Nginx Nous configurons le fichier /usr/local/etc/php-fpm.conf

```
location ~ \.php$ {
    try_files $uri =404;
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    fastcgi_pass unix:/run/php/php7.2-fpm.sock;
    fastcgi_index index.php;
    include fastcgi_params;
    fastcgi_param  SCRIPT_FILENAME $document_root$fastcgi_script_name;
}
```

Les 2 exemples communiquent avec le processus PHP par un socket. Ainsi, pour chaque demande entrante, le serveur écrit les

données dans ce fichier et, à la réception du résultat, les renvoie au navigateur. PHP ne reçoit pas directement les demandes envoyées par les navigateurs. Les serveurs Web les interceptent en premier.
Le serveur Web sait comment se connecter au processus PHP et transfère toutes les données de la demande (le colle littéralement) dans PHP. Lorsque PHP a terminé, il renvoie la réponse au serveur Web, qui la renvoie au client (ou au navigateur, dans la plupart des cas). 2 3
Comme vous le voyez avec FPM, PHP peut s'exécuter sur un serveur qui n'est pas un processus unique, mais plutôt sur certains processus PHP qui sont générés, contrôlés et tués par le gestionnaire de processus auquel le serveur Web transmet les requêtes.



2

Server API	FPM/FastCGI
Virtual Directory Support	disabled

3

Optimiser php-fpm ?

La technologie est très puissante pour être suffisante pour un seul serveur et pour répondre à la plupart des cas d'utilisation.
Toutefois, l'optimisation de PHP-FPM devient pertinente lorsque vous mettez en place plusieurs machines, en cluster. C'est à dire plusieurs serveurs pour éviter les interruptions, les latences trop importantes et avoir la capacité à monter en charge en cas de pic de charge.
Cette optimisation va nous permettre de gérer facilement des milliers de connexions simultanées.

Comment faire ?

L'emplacement du fichier de configuration php-fpm sera différent suivant le système d'exploitation de votre serveur :

```
Debian / Ubuntu / Mint : /etc/php/7.2/fpm/php-fpm.conf
CentOS / Readhat : /etc/nginx/conf.d/monSite.conf
```

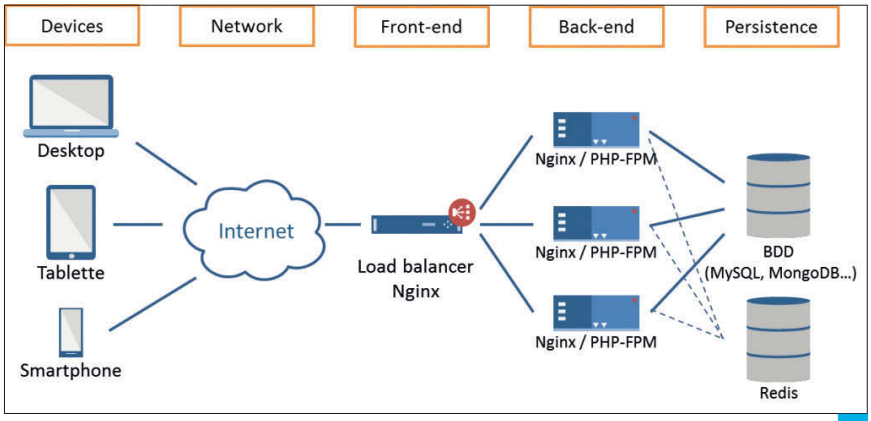
Voici à quoi ressemblent les premières lignes de ce fichier :

```
pid = /run/php/php7.2-fpm.pid
error_log = /var/log/php7.2-fpm.log
emergency_restart_threshold 10
emergency_restart_interval 1m
process_control_timeout 10s
```

Le fichier de configuration fournit des informations importantes pour notre serveur. Tout d'abord, l'emplacement de l'identifiant du processus « pid ». Ensuite l'emplacement du fichier des logs errors. Les lignes emergency_restart_threshold et emergency_restart_interval vont gérer la vie du serveur en détectant si 10 processus enfants échouent dans une minute, le processus principal doit redémarrer automatiquement. L'option process_control_timeout, indique aux processus enfants d'attendre autant de temps avant d'exécuter le signal reçu du processus parent.
Au final, pour servir les requêtes web, le php-fpm crée un nouveau pool de processus, qui aura une configuration séparée. Dans mon cas, le nom du pool s'est avéré être www et le fichier que je voulais modifier était /etc/php/7.2/fpm/pool.d/www.conf.
Voyons à quoi ressemble ce fichier : [www]

```
user = www-data
group = www-data
```

Ce fichier va stocker les droits propriétaires et le groupe du répertoire permettant ainsi au processus PHP de pouvoir écrire dans les fichiers de logs et télécharger des documents, etc.
Enfin, nous arrivons au fichier source pour effectuer le réglage du gestionnaire de processus (pm). En général, vous verrez les valeurs par défaut suivantes :



4

```
pm = dynamic
pm.max_children = 5
pm.start_servers = 3
pm.min_spare_servers = 2
pm.max_spare_servers = 4
pm.max_requests = 200
```

La première ligne accepte trois types de valeurs possibles :
- Static : nombre fixe de processus PHP qui sera maintenu quoi qu'il arrive ;
- Dynamic : possibilité de spécifier le nombre minimum et maximum de processus php-fpm ;
- Ondemand : les processus sont créés et détruits à la demande.

Ne négligez pas ces paramètres car ils peuvent être assez désastreux au niveau de votre budget, de la performance et de la robustesse. Bref : soyez précis !

Répartition de la charge

Lorsque vous souhaitez ajouter des serveurs supplémentaires pour répartir la charge de vos pages web, vous êtes confronté à la manière dont le serveur va privilégier un serveur plutôt qu'un autre. Vous devrez configurer la répartition de charge, le fameux « load balancing », pour distribuer les données entre les différentes machines d'un même groupe. 4
Le but est d'obtenir une architecture PHP de haute performance

avec la possibilité d'ajouter des serveurs supplémentaires, en mode scalabilité horizontale, pour répondre aux pics de charge. La répartition de la charge, ainsi que le rôle de load balancer sera joué par le serveur Nginx, Apache, HAProxy… (en fait vous aurez le choix). La mise en place de cette architecture « haute performance » gérera les données venant du côté front. Après, le navigateur demandera l'accès à une page comme point d'entrée et le load balancer se chargera de répartir la charge proprement dite entre un ou plusieurs serveurs de la manière suivante :

APACHE

Pour Apache, il est nécessaire d'activer mod_proxy_balancer et mod_proxy_fcgi.

```
<Proxy balancer://phpfpmlb>
  BalancerMember fcgi://10.0.0.12:9000
  BalancerMember fcgi://10.0.0.34:9000
</Proxy>
# Redirect PHP execution to the balancer
<FilesMatch \.php$>
  SetHandler "proxy:balancer://phpfpmlb"
</FilesMatch>
```

NGINX

```
http {
    upstream myapp1 {
        server srv1.example.com;
        server srv2.example.com;
        server srv3.example.com;
    }
    server {
        listen 80;
        location / {
            proxy_pass http://myapp1;
        }
    }
}
```

HAPROXY

Le fichier à configurer est le suivant : /etc/haproxy/haproxy.cfg

```
frontend Local_Server
  bind 192.168.1.18:80
  mode http
  default_backend My_Web_Servers

backend My_Web_Servers
  mode http
  balance roundrobin
  option forwardfor
  http-request set-header X-Forwarded-Port %[dst_port]
  http-request add-header X-Forwarded-Proto https if { ssl_fc }
  option httpchk HEAD / HTTP/1.1rnHost:localhost
  server web1.votreSite.com  192.168.1.101:80
  server web2.votreSite.com  192.168.1.102:80
  server web3.votreSite.com  192.168.1.103:80
```

Traitement des requêtes

PHP-FPM utilise plusieurs workers pour traiter en parallèle les requêtes, permettant ainsi de gérer la charge de connexions.
Avec PHP 7, la performance de PHP-FPM a été améliorée pour obtenir des performances optimales et scalables.
Cette architecture permet à une requête entrante, fournie par le serveur HTTP, de rendre une réponse qui sera renvoyée par ce dernier. Le load balancer ne s'occupe que de choisir le bon serveur applicatif ou celui ayant le moins de charge sur le cluster configuré.
La vraie complexité est de gérer les sessions utilisateur qui, de base, sont gérées via le système de fichiers du serveur. Dans notre cas, c'est impossible puisque l'utilisateur peut être dirigé vers un serveur A pour sa première requête, vers un serveur B pour sa seconde, etc.

Les sessions

La problématique principale lorsque nous utilisons PHP-FPM avec un load balancer, concerne la gestion des sessions utilisateurs qui est gérée par le système de fichiers du serveur. C'est à dire que nous ne contrôlons pas le serveur qui sera interrogé surtout si l'un d'eux a été utilisé pour l'authentification.
La solution est d'exploiter un serveur de cache distribué qui permettra de centraliser l'ensemble des sessions, peu importe le serveur qui les crée ou les utilise.
Il existe de nombreux systèmes qui répondent à cette problématique : memcached, redis… ou encore des systèmes proposés directement avec la base de données.
En pratique, les sessions utilisateurs sont liées à un jeton, ou token. Chaque requête possède un jeton. Ainsi, peu importe le back-end qui recevra le jeton, il pourra retrouver les données des sessions sur le même serveur.

Conclusion

Cette architecture se veut simple et rapide à mettre en œuvre pour gérer un site web à fort trafic. Toutefois, même si PHP-FPM répond à de nombreux cas d'usages, les réglages serveurs peuvent être nécessaires par rapport à la performance et à la haute disponibilité souhaitée.
Bien sûr, vous pouvez ajouter des composants supplémentaires comme RabbitMQ pour les traitements asynchrones concernant l'envoi de mails, la génération de fichiers, d'exports, etc.
Enfin, la configuration proposée facilite aussi le delivery, c'est à dire la livraison en continu, sans interrompre les services et le site web lors de mises à jour du projet à n'importe quel moment de la journée. •

Aurélie Vache
Cloud Dev(Ops) chez Continental
Duchess France & DevFest Toulouse Leader
Toulouse Data Science core-team & Marraine
Elles bougent - @aurelievache

Je crée ma CLI en Go en quelques minutes

niveau 100

Dans notre travail au quotidien nous avons l'habitude d'utiliser de chouettes outils en ligne de commande, nous sommes entourés de CLI (docker, kubectl, terraform, aws ...). Nous allons voir dans ce tutoriel, qu'en seulement quelques minutes il est possible de créer une CLI avec le langage Go et la bibliothèque Cobra. Vous êtes prêt ?

POURQUOI GO ?

Go, créé en 2009, et utilisé en production par des milliers d'entreprises dans le monde (incluant Uber, Lyft, Slack, Pinterest ... et bien sûr Google), est en train de devenir le langage de prédilection du Cloud.
Il est facile à installer, bien fourni en termes de bibliothèques et d'outils, rapide à compiler et peu gourmand en mémoire.
Contrairement à d'autres langages, la montée en compétence est assez douce et lorsque l'on vient du monde Java, je trouve qu'elle l'est davantage.
Avec son écosystème truffé d'outils et de CLI, on aurait tort de ne pas mettre le langage Go dans notre boîte à outils.
Si vous ne vous êtes pas encore mis à Go, vous pouvez commencer en faisant le tutoriel intéractif "tour de go" : https://tour.golang.org
Comme vous allez le voir dans cet article, écrire une CLI avec GO est un jeu d'enfant !

Prérequis :

Go !

La première étape consiste, si vous ne l'avez pas déjà fait, à installer Go sur votre machine.
Pour cela vous pouvez suivre la procédure d'installation sur le site officiel ou bien passer par GVM.
GVM est un version manager pour Go, très pratique, qui permet de mettre à jour votre version de Go en spécifiant quelle version vous désirez.

Installation :

- Pour bash :

```
bash < <(curl -s -S -L https://raw.githubusercontent.com/moovweb/gvm/master/bin scripts/gvm-installer)
```

- Pour zsh :

```
zsh < <(curl -s -S -L https://raw.githubusercontent.com/moovweb/gvm/master/bin scripts/gvm-installer)
```

Utilisation :

```
$ gvm
Usage: gvm [command]

Description:
 GVM is the Go Version Manager

Commands:
 version    - print the gvm version number
 get        - gets the latest code (for debugging)
 use        - select a go version to use (--default to set permanently)
 diff       - view changes to Go root
 help       - display this usage text
 implode    - completely remove gvm
 install    - install go versions
 uninstall  - uninstall go versions
 cross      - install go cross compilers
 linkthis   - link this directory into GOPATH
 list       - list installed go versions
 listall    - list available versions
 alias      - manage go version aliases
 pkgset     - manage go packages sets
 pkgenv     - edit the environment for a package set
```

La commande de GVM qui va nous intéresser tout particulièrement est la commande **gvm install**, elle s'utilise comme ceci :

```
$ gvm install [version] [options]
```

Installation de Go :

```
$ gvm install go1.12.4 -B
$ gvm use go1.12.4 --default
```

Dans votre .zshrc ou .bashrc veuillez à bien régler vos variables d'environnement **$GOROOT** et **$GOPATH**. Voici un exemple :

```
[[ -s "$HOME/.gvm/scripts/gvm" ]] && source "$HOME/.gvm/scripts/gvm"
export GOPATH=$HOME/go
export GOBIN=$GOPATH/bin
export PATH=${PATH}:$GOBIN
```

Ça y est, Go est installé, son gestionnaire de version également, il est temps de rentrer dans le vif du sujet et de créer notre première CLI.

Cobra, c'est plus fort que toi !

Cobra est à la fois une bibliothèque pour créer de puissantes applications CLI modernes et un programme permettant de générer des applications et des fichiers de commandes.
Utiliser Cobra est facile. Tout d'abord, utilisez la commande **go get** afin de télécharger la dernière version. Cette commande installera l'exécutable du générateur cobra avec la bibliothèque et ses dépendances :

```
$ go get -u github.com/spf13/cobra/cobra
```

Le binaire de Cobra est maintenant dans le répertoire bin/ de votre **$GOPATH**, qui est lui- même dans votre PATH, donc utilisable directement.
Nous allons commencer par générer notre première application grâce à la commande **cobra init** suivie du package et du nom de votre app. La commande va générer l'application avec la bonne structure de fichier et les imports.

```
$ cd $GOPATH/src
$ cobra init github.com/scraly/hello-world
Using config file: /home/scraly/.cobra.yaml
Your Cobra application is ready at
/home/scraly/git/src/github.com/scraly/hello-world

Give it a try by going there and running `go run main.go`.
Add commands to it by running `cobra add [cmdname]`.
```

Votre application est initialisée, un fichier main.go a été créé ainsi qu'un package cmd/ :

```
$ cd github.com/scraly/hello-world
$ tree
.
├── cmd
│   └── root.go
├── LICENSE
├── main.go

1 directory, 3 files
```

A l'exécution de notre CLI nous voulons afficher :
- une courte description ;
- une longue description ;
- l'aide de notre application.

Pour cela, il suffit de modifier le fichier **cmd/root.go** :

```
// rootCmd represents the base command when called without any subcommands
var rootCmd = &cobra.Command{
  Use:   "hello-world",
  Short: "A brief description of your application",
  Long: `A longer description that spans multiple lines and likely contains
examples and usage of using your application. For example:
Cobra is a CLI library for Go that empowers applications.
This application is a tool to generate the needed files
to quickly create a Cobra application.`,
  // Uncomment the following line if your bare application
  // has an action associated with it:
  //  Run: func(cmd *cobra.Command, args []string) { },
}
```

Maintenant, nous voulons que notre petite application ait la commande start ; pour cela nous allons nous servir de la commande add de la CLI de Cobra :

```
$ cobra add start
Using config file: /home/scraly/.cobra.yaml
start created at /home/scraly/git/src/github.com/scraly/hello-world/cmd/start.go
```

> A noter que les noms de commandes doivent être au format camelCase.

Une fois que vous avez exécuté ces trois commandes, vous obtenez une structure d'application similaire à celle-ci :

```
$ tree
.
├── cmd
│   ├── root.go
│   └── start.go
├── LICENSE
├── main.go
```

À ce stade, vous pouvez exécuter **go run main.go** pour exécuter votre application :

```
$ go run main.go
A longer description that spans multiple lines and likely contains
examples and usage of using your application. For example:

Cobra is a CLI library for Go that empowers applications.
This application is a tool to generate the needed files
to quickly create a Cobra application.

Usage:
  hello-world [command]

Available Commands:
  help        Help about any command
  start       A brief description of your command

Flags:
      --config string   config file (default is $HOME/.hello-world.yaml)
  -h, --help            help for hello-world
  -t, --toggle          Help message for toggle

Use "hello-world [command] --help" for more information about a command.
```

Et **go run main.go start** pour lancer la commande start :

```
$ go run main.go start
```

```
start called
```

Si nous avions créé une autre commande, avec par exemple **cobra add config**, nous pourrions également tester la commande en saisissant **go run config.go**.
Nous allons donc maintenant modifier le fichier **cmd/start.go**, avec notre propre code, afin que notre application affiche le message que nous voulons pour la commande start :

```
var startCmd = &cobra.Command{
    Use:  "start",
    Short: "This command will show you a hello world message",
    Long: `Welcome in start command, this cmd will display to you a hello world message.`,
    Run: func(cmd *cobra.Command, args []string) {
        fmt.Println("Hello world, start have been called!")
    },
}
```

Testons l'aide de la commande start dès à présent :

```
$ go run main.go start -h
Welcome in start command, this cmd will display to you a hello world message.

Usage:
 hello-world start [flags]

Flags:
 -h, --help   help for start

Global Flags:
    --config string   config file (default is $HOME/.hello-world.yaml)
```

Puis exécutons la commande start afin de voir le message mis à jour :

```
$ go run main.go start
Hello world, start have been called!
```

Nous pouvons également afficher le message d'aide de notre commande start :

```
$ go run main.go start -h
Welcome in start command, this cmd will display to you a hello world message.

Usage:
 hello-world start [flags]

Flags:
 -h, --help   help for start

Global Flags:
    --config string   config file (default is $HOME/.hello-world.yaml)
```

Votre application est prête, il ne vous reste plus qu'à la builder/ à l'installer dans votre $GOPATH puis à l'utiliser :

```
$ go install
$ hello-world start
Hello world, start have been called!
```

Envie d'aller plus loin ?

Nous allons rajouter une commande **say** et une sous commande **hello**, afin de faire dire à notre petit programme : hello world !
Ajoutons notre commande say :

```
$ cobra add say
say created at /Users/uidn3817/go/src/github.com/scraly/hello-world/cmd/say.go
```

Editons ce fichier afin d'avertir l'utilisateur que la commande say ne s'utilise pas seule. Il suffit de remplacer la méthode Run par celle-ci :

```
RunE: func(cmd *cobra.Command, args []string) error {
    return errors.New("Provide item to the say command")
},
```

Puis ajoutons la sous commande hello :

```
$ cobra add hello -p 'sayCmd'
hello created at /Users/uidn3817/go/src/github.com/scraly/hello-world/cmd/hello.go
```

Modifions le fichier hello.go qui vient d'être généré, afin que la CLI puisse afficher "hello world !" :

```
 Run: func(cmd *cobra.Command, args []string) {
    fmt.Println("hello world !")
 },

$ go install

$ hello-world say
Error: Provide item to the say command
Usage:
 hello-world say [flags]
 hello-world say [command]

Available Commands:
 hello      A brief description of your command

Flags:
 -h, --help   help for say

Global Flags:
    --config string   config file (default is $HOME/.hello-world.yaml)

Use "hello-world say [command] --help" for more information about a command.

Provide item to the say command

$ hello-world say hello
hello world !
```

Et voilà, vous avez codé votre CLI en Go en quelques minutes, c'est un bon début ! :-)
Si cet article vous a plu, rendez-vous dans des prochains épisodes pour aller plus loin avec votre première CLI et vos premières applications en Go ! •

Julie Hourcade
Consultante DevOps
Osaxis

Etat de l'art des solutions de registry Docker

niveau 100

Docker Registry est une application côté serveur utilisée pour versionner, stocker et distribuer des images Docker.Un registry permet de contrôler l'origine d'une image et facilite le déploiement continu. Il existe différents registry, aussi bien en mode auto hébergé qu'en mode SaaS.
Le plus connu d'entre eux est Docker Hub, les images officielles telles que Ubuntu, Debian et Alpine provenant en réalité de ce registry.

Comprendre les images Docker

Une image Docker est un manifeste qui définit le contenu d'un conteneur Docker. Cet article n'est pas destiné à expliquer le fonctionnement d'une image mais il est essentiel de comprendre les normes de nommage des images Docker.

Les images les plus communes sont Debian, Ubuntu et Alpine pour les distributions, mais il en existe également pour des langages particuliers comme par exemple PHP, Node ou Golang. Ces images sont des images dites officielles et sont fournies par Docker. Il est fortement conseillé de dériver de ces images plutôt qu'utiliser des images non sûres. Depuis peu, il est désormais possible de trouver des images créées par des éditeurs officiels et offrant donc plus de garanties. Pour toutes recherches à ce sujet, le site internet de Docker Hub est particulièrement bien conçu : https://hub.docker.com

Pour comprendre le fonctionnement d'une image, une des solutions consiste à analyser la composition de son nom. Pour les images officielles, il est par exemple possible de faire simplement "docker pull debian". Toutefois, en réalité le nom complet de cette image est library/debian:latest. Tout d'abord, "library" désigne le registry où se trouve l'image, puis "debian" désigne le nom de son repository, puis pour finir "latest" désigne son tag (Le tag pouvant être assimilé à une version). L'image Debian possède plusieurs tags comme par exemple "buster" pour sa version 10 ou encore "buster-slim" pour une version allégée de sa version 10.

Dans cet article, c'est principalement la partie nom du registry qui se révèle intéressante. Dans les exemples précédents, le registry était "library" qui représente le registry par défaut mais il en existe plusieurs. Certaines images sont rattachées à un autre registry, par exemple Nexus 3 est rattaché au registry de Sonatype, son éditeur. L'image s'appelle donc sonatype/nexus3.

Quelles sont les fonctions attendues pour un bon registry Docker ?

Avoir un registry privé permet de contrôler la provenance des images Docker utilisées dans les projets. Cela permet aussi de garder la main sur tous les déploiements. Mais celui-ci doit également répondre à un certain nombre de caractéristiques.

Il faut tout d'abord savoir si une solution auto-hébergée est totalement nécessaire ou si une solution SaaS serait plus adaptée. Dans le cas où le réseau interne nécessite un contrôle des flux internet assez restrictif, un registry auto-hébergé devient une nécessité. En contrepartie, une solution SaaS se révèle être un choix judicieux pour limiter les actions humaines de la part d'un administrateur, le support étant généralement un service proposé. Il est de plus important de noter que certaines solutions aussi bien SaaS qu'auto-hébergées ont un coût de licence, cette donnée pouvant impacter le choix d'une des catégories de produit.

La sécurité est un aspect important pour un registry, ce dernier devant être en mesure de fournir une authentification efficace telle qu'un support AD/LDAP. Certaines solutions proposent plutôt une authentification via token. Certains registry proposent également de scanner les images Docker présentes pour déceler les vulnérabilités. A noter enfin que quelques solutions intègrent également Docker Notary qui est un système destiné à signer les images pour plus de sécurité.

L'intégration continue et le déploiement continu sont tous deux des sujets importants, le choix d'un registry privé pourrait faciliter le déploiement d'image. Certaines plateformes proposent de construire l'image automatiquement lorsqu'un changement est détecté sur un outil de versionning comme Git.

La facilité d'utilisation constitue également un point à prendre en considération. Nous avons abordé la facilité de support des solutions SaaS mais l'aspect utilisateur doit aussi être pris en compte. Une interface graphique "User Friendly" permet de gérer les images contenues dans le registry sans trop de difficultés. Pour autant, beaucoup de solutions ne proposent pas la suppression d'images ou de garbage collector, ce qui est bien regrettable.

Dernier point à aborder, la disponibilité. Pour les solutions SaaS, la question ne se pose généralement pas, la haute disponibilité étant gérée par le fournisseur. Mais pour les solutions auto-hébergées, ce n'est pas toujours le cas. La disponibilité des images étant essentielle pour le bon fonctionnement des services, c'est un point à particulièrement soigner.

Tour d'horizon des différentes solutions

(voir tableau ci-contre)

Le projet Docker Distribution, construisez votre propre solution !

Si aucune des solutions ne vous convient ou qu'une fonctionnalité vous manque, le projet Docker Distribution est pour vous. Ce projet Open Source, disponible sur le GitHub de Docker, constitue une base pour vous permettre de développer votre propre solution.

Sans interface graphique et avec un système d'authentification assez sommaire, il fournit néanmoins tous les composants nécessaires à un bon registry. Il peut être installé via l'image Docker officielle distribution/registry:2.0 mais ne convient en aucun cas pour une utilisation en production. Le projet Docker Distribution reste cependant une solution parmi les autres.

Conclusion

L'intégration d'un registry Docker à une infrastructure me paraît essentiel en ces temps où la mouvance DevOps est en plein essor. L'automatisation est une problématique très en vogue et le registry en est un élément incontournable. Le choix d'une solution est à étudier en fonction des besoins spécifiques de l'entreprise, des compétences de ses équipes, mais aussi en fonction de son infrastructure. •

Solutions SaaS	Authentification AD/LDAP	Build automatique	Scan vulnérabilités	Remarques	Payant
GitLab CE	Oui	Oui mais avec GitLab CI	Non	Le registry est intégré à GitLab donc très pratique si on utilise déjà le produit. A noter que GitLab existe aussi en version auto-hébergée	Non
GitLab EE					Oui
Quay	Oui	Non	Oui	Proposé avec une interface graphique bien pensée. Quay appartenait anciennement à CoreOS mais a été racheté par RedHat récemment. Des changements sont en cours.	Oui
Docker Hub	Avec un compte Docker Hub	Peut être lié avec GitHub et BitBucket	Non	Docker Hub est le registry le plus utilisé mais reste limité. La création d'un compte est obligatoire et ne permet pas l'authentification AD/LDAP. Cependant, il reste un bon choix pour le partage d'images publiques.	Les registry publics sont gratuits mais pas les privés
Solutions Auto-Hébergées	**Authentification AD/LDAP**	**Build automatique**	**Scan vulnérabilités**	**Remarques**	**Payant**
Docker Trusted Registry	Oui	Non	Oui	Docker Trusted Registry reste la solution logique pour les entreprises utilisant la version Entreprise de Docker. Etant la version officielle, il propose la plupart des fonctionnalités attendues. La fonctionnalité qui manque principalement à DTR est la construction automatique des images. A noter qu'il est possible d'avoir une version SaaS de Docker Trusted Registry.	Oui (avec Docker EE)
Nexus	Oui	Non	Non	Nexus n'est pas seulement un registry Docker et propose de plus du stockage pour différents types d'artefacts. C'est une solution assez lourde et ne convient pas si le besoin se restreint à un registry Docker. A noter aussi que l'interface graphique reste très sommaire.	Non
Artifactory	Oui	Non	Non	En ce qui concerne le registry, Artifactory et Nexus sont similaires. Seule différence notable, Artifactory ne propose pas le registry dans sa version gratuite.	Oui
Harbor	Oui	Non	Oui	Très récemment proposée en version Open Source par VMWare, c'est l'une des solutions les plus intéressantes. Mais comme tout produit très jeune, il subsiste encore pas mal de bugs. L'installation est particulièrement laborieuse.	Non
Portus	Oui	Non	Oui	Proposé par openSUSE, son installation se révèle bien moins délicate que celle de Harbor, son principal concurrent. En effet, il existe une simple image sur le Docker Hub (opensuse/portus). Portus propose un bon nombre de fonctionnalités aussi bien pour améliorer la sécurité que la disponibilité.	Non

Jean-Nicolas BERGER
Architecte Données chez
Sogeti / Cap Gemini
Expert SQL Server depuis 15 ans
https://blog.sqlserver.fr

SQL Server : les index à la page

niveau 200

Les index sont un élément facultatif dans la théorie mais essentiel en pratique dans les moteurs de données. Rentrons dans les entrailles du stockage de SQL Server pour observer comment tout cela s'articule.

Les index sont un élément majeur dans les bases de données. Ils permettent notamment d'accéder plus rapidement aux informations lors des recherches. Dans la partie relationnelle classique du moteur de données Microsoft SQL Server, on distingue les index clustered et les index non clustered. Voici par l'exemple une explication détaillée de ces index et du stockage interne associé au niveau du moteur SQL Server.

Dans un premier temps, il convient de comprendre le mode de stockage primaire des données au niveau de SQL Server. Ce stockage s'appuie sur des blocs de 8Ko appelés pages. Les données sont systématiquement stockées sous forme de pages. Les fichiers de données sur lesquels s'appuient les bases de données contiennent ces pages de 8Ko. Plusieurs types de pages existent, allant de la GAM (Global Allocation Map) aux pages d'index et données.

Lors de la sauvegarde d'une base de données, toutes les pages tout ou partie utilisées sont prises en compte et agrégées au sein du fichier de sauvegarde. Les pages non utilisées (par exemple celles qui était utilisées pour le stockage de tables supprimées) sont ignorées lors de la phase de sauvegarde.

Lors de la restauration d'une base de données, le moteur va d'abord reconstituer des fichiers de taille identique à ceux qui étaient en place lors de la sauvegarde. Ensuite, les différentes pages sauvegardées seront replacées à l'endroit exact qu'elles occupaient précédemment. Cela permet de ne pas avoir à retoucher l'enchaînement des pages tel qu'il est stocké au sein de l'entête (96 premiers octets) de chaque page.

Dans la nouvelle version SQL Server 2019, une nouvelle fonction sys.dm_db_page_info est disponible et documentée (enfin ! c'est rare sur ce type d'outils internes…). Elle permet d'avoir une vision détaillée de l'entête des pages, et notamment de l'enchaînement de celles-ci. Elle sera utilisée sous forme d'exemple d'application dans le contexte du stockage des données dans les index.

Hormis pour quelques cas particuliers (tables InMemory, types de données de plus de 8Ko, …), les données des tables sont stockées soit sous forme d'index clustered, soit sous forme de segments (heaps). Il est important de comprendre qu'il s'agit là d'un choix binaire : soit la définition de la table présente un index clustered (et il ne peut y en avoir qu'un seul), auquel cas c'est au niveau de cet index que sont stockées les données de la table, soit cette définition ne présente pas d'index clustered, auquel cas les données sont stockées sous forme de segment.

Segments de données

Considérons dans un premier temps la table définie par le script suivant :

```
CREATE TABLE [dbo].[TableSansIndexClustered]
(
    [Id] INT IDENTITY NOT NULL,
    [CleNumerique] INT CONSTRAINT PK_TableSansIndexClustered PRIMARY KEY NONCLUSTERED,
    [CleTexte] VARCHAR(10)
)
GO
CREATE NONCLUSTERED INDEX [Index_Non_Clustered_Texte] ON [dbo].[TableSansIndexClustered]
 ([CleTexte])
GO
```

Cette table ne présente aucun index clustered, et a donc ses données stockées sous forme d'un segment :

```
SELECT object_id,name,index_id,type,type_desc,is_unique
FROM sys.indexes where object_id=OBJECT_ID('[dbo].[TableSansIndexClustered]')
```

object_id	name	index_id	type	type_desc	is_unique
613577224	NULL	0	0	HEAP	0
613577224	PK_TableSansIndexClustered	2	2	NONCLUSTERED	1
613577224	Index_Non_Clustered_Texte	3	2	NONCLUSTERED	0

Pour l'exemple, la table sera remplie avec des données rentrées dans un ordre aléatoire :

Id	CleNumerique	CleTexte
1	30932	Num30932
2	436768	Num436768
3	171267	Num171267
4	172152	Num172152
5	599390	Num599390
6	940831	Num940831
7	443539	Num443539
8	320856	Num320856
9	508173	Num508173
10	217366	Num217366
..	…	…

Plusieurs méthodes permettent d'identifier les pages associées à cet « index 0 ». Elles sont dans l'état actuel non documentées et peuvent de fait être sujettes à des évolutions avec rupture de compatibilité. La commande DBCC IND fait partie de ces commandes, et permet d'identifier la liste des pages allouées pour un index donné. Elle nécessite dans un premier temps de récupérer l'identifiant de la base de données et celui de l'objet.

```
SELECT DB_ID() as [DB],OBJECT_ID('[dbo].[TableSansIndexClustered]') as [Object]
```

DB	Object
5	613577224

Le listing des pages consommées par cet index donne :

```
DBCC IND (5,613577224,0)
```

1

On constate en première ligne la présence d'une page IAM (Index Allocation Map), qui contient la référence des autres pages. Il y a ensuite la liste des pages de données utilisées par l'index.

Voici quelques-uns des éléments retournés par la fonction sys.dm_db_page_info :

```
SELECT page_id,page_type_desc,object_id,index_id,is_iam_page
from sys.dm_db_page_info (DB_ID(),1,319,'DETAILED')
```

page_id	page_type_desc	object_id	index_id	is_iam_page
319	IAM_PAGE	613577224	0	1

```
SELECT page_id,page_type_desc,object_id,index_id,is_iam_page
FROM sys.dm_db_page_info (DB_ID(),1,4064,'DETAILED')
```

page_id	page_type_desc	object_id	index_id	is_iam_page
4064	DATA_PAGE	613577224	0	0

En regardant d'un peu plus près le contenu de la page contenant les données, on peut observer l'ordre de celles-ci :

```
DBCC PAGE (5,1,4064,1)
```
2

Les données sont dans l'ordre dans lequel elles ont été entrées. C'est le principe du segment de données.

Index non clustered

L'observation des données des pages de la clé primaire (non clustered pour notre table) permet d'expliquer de manière plus claire l'organisation d'un index.

Pour identifier les pages de données de l'index 2 (clé primaire PK_TableSansIndexClustered vue précédemment), nous pouvons encore utiliser DBCC IND, ou bien cette fois-ci utiliser la fonction sys.dm_db_database_page_allocations, elle aussi non documentée :

```
SELECT allocated_page_page_id, page_type_desc,
    page_level, previous_page_page_id, next_page_page_id
FROM sys.dm_db_database_page_allocations
    (DB_ID(),OBJECT_ID('[dbo].[TableSansIndexClustered]'), 2, NULL, 'DETAILED')
WHERE page_type in (1,2) -- INDEX_PAGE ou DATA_PAGE
ORDER BY page_level DESC, previous_page_page_id
```

En nous limitant aux pages de données, on obtient le début de liste suivant :

allocated_page_page_id	page_type_desc	page_level	previous_page_page_id	next_page_page_id
11432	INDEX_PAGE	2	NULL	NULL
10656	INDEX_PAGE	1	NULL	11433
11433	INDEX_PAGE	1	10656	11434
11434	INDEX_PAGE	1	11433	11435
11435	INDEX_PAGE	1	11434	11436
11436	INDEX_PAGE	1	11435	11437
11437	INDEX_PAGE	1	11436	11438
11438	INDEX_PAGE	1	11437	NULL
10640	INDEX_PAGE	0	NULL	10664
10664	INDEX_PAGE	0	10640	10665
10665	INDEX_PAGE	0	10664	10666
10666	INDEX_PAGE	0	10665	10667
10667	INDEX_PAGE	0	10666	10668
10668	INDEX_PAGE	0	10667	10669

La valeur maximale de page_level correspond à la page racine de l'index. La page 11432 est donc ici la première entrée lorsque l'on parcourt cet index.

```
DBCC PAGE (5,1,11432,3)
```
3

PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel
1	319	NULL	NULL	613577224	0	1	72057594043301888	In-row data	10	NULL
1	4064	1	319	613577224	0	1	72057594043301888	In-row data	1	0
1	4065	1	319	613577224	0	1	72057594043301888	In-row data	1	0
1	4066	1	319	613577224	0	1	72057594043301888	In-row data	1	0
1	4067	1	319	613577224	0	1	72057594043301888	In-row data	1	0
1	4068	1	319	613577224	0	1	72057594043301888	In-row data	1	0
1	4069	1	319	613577224	0	1	72057594043301888	In-row data	1	0
1	4070	1	319	613577224	0	1	72057594043301888	In-row data	1	0

1

```
Slot 0, Offset 0x60, Length 27, DumpStyle BYTE

Record Type = PRIMARY_RECORD        Record Attributes =  NULL_BITMAP VARIABLE_COLUMNS
Record Size = 27
Memory Dump @0x0000000B9E78060

0000000000000000:   30000c00 01000000 d4780000 03000001 001b004e  0.......Ôx.........N
0000000000000014:   756d3330 393332                               um30932

Slot 1, Offset 0x7b, Length 28, DumpStyle BYTE

Record Type = PRIMARY_RECORD        Record Attributes =  NULL_BITMAP VARIABLE_COLUMNS
Record Size = 28
Memory Dump @0x0000000B9E7807B

0000000000000000:   30000c00 02000000 20aa0600 03000001 001c004e  0.......  ª.........N
0000000000000014:   756d3433 36373638                             um436768

Slot 2, Offset 0x97, Length 28, DumpStyle BYTE

Record Type = PRIMARY_RECORD        Record Attributes =  NULL_BITMAP VARIABLE_COLUMNS
Record Size = 28
Memory Dump @0x0000000B9E78097

0000000000000000:   30000c00 03000000 039d0200 03000001 001c004e  0...................N
0000000000000014:   756d3137 31323637                             um171267

Slot 3, Offset 0xb3, Length 28, DumpStyle BYTE

Record Type = PRIMARY_RECORD        Record Attributes =  NULL_BITMAP VARIABLE_COLUMNS
Record Size = 28
Memory Dump @0x0000000B9E780B3

0000000000000000:   30000c00 04000000 78a00200 03000001 001c004e  0.......x .........N
0000000000000014:   756d3137 32313532                             um172152

Slot 4, Offset 0xcf, Length 28, DumpStyle BYTE
```

2

FileId	PageId	Row	Level	ChildFileId	ChildPageId	CleNumerique (key)	KeyHashValue	Row Size
1	11432	0	2	1	10656	NULL	NULL	11
1	11432	1	2	1	11433	120782	NULL	11
1	11432	2	2	1	11434	241563	NULL	11
1	11432	3	2	1	11435	362344	NULL	11
1	11432	4	2	1	11436	483125	NULL	11
1	11432	5	2	1	11437	603906	NULL	11
1	11432	6	2	1	11438	724687	NULL	11

3

FileId	PageId	Row	Level	ChildFileId	ChildPageId	CleNumerique (key)	KeyHashValue	Row Size
1	10656	0	1	1	10640	NULL	NULL	11
1	10656	1	1	1	10664	450	NULL	11
1	10656	2	1	1	10665	899	NULL	11
1	10656	3	1	1	10666	1348	NULL	11
1	10656	4	1	1	10667	1797	NULL	11
1	10656	5	1	1	10668	2246	NULL	11
1	10656	6	1	1	10669	2695	NULL	11
1	10656	7	1	1	10670	3144	NULL	11
1	10656	8	1	1	10671	3593	NULL	11

4

FileId	PageId	Row	Level	CleNumerique (key)	HEAP RID	KeyHashValue	Row Size
1	10640	0	0	1	0xD01B000001001E00	(8194443284a0)	16
1	10640	1	0	2	0x672500000100A800	(61a06abd401c)	16
1	10640	2	0	3	0x151F00000100DE00	(98ec012aa510)	16
1	10640	3	0	4	0xB81E00000100AE00	(a0c936a3c965)	16
1	10640	4	0	5	0x922000000100E100	(59855d342c69)	16
1	10640	5	0	6	0x302700000100C200	(b9b173bbe8d5)	16
1	10640	6	0	7	0xF41B000001001F00	(40fd182c0dd9)	16
1	10640	7	0	8	0x232300000100FB00	(c9fb1da9313f)	16
1	10640	8	0	9	0xA027000001009A00	(30b7763ed433)	16
1	10640	9	0	10	0x7E28000001009800	(d08358b1108f)	16
1	10640	10	0	11	0xDA1C000001001E00	(29cf3326f583)	16
1	10640	11	0	12	0x4A2000000100	(11ea04af9866)	16

5

Le contenu de cette page nous indique que les premières valeurs de la clé CleNumerique seront trouvées en descendant vers la page 10656, qu'à partir de CleNumerique=120782, il faudra aller vers la page 11433, et ainsi de suite.

```
DBCC PAGE (5,1,10656,3)
```
4

On continue à descendre, les premiers enregistrements seront dans la page 10640, puis dans la page 10664 à partir de CleNumerique=450, …

Dans la page feuille 10640, on obtient un exemple de niveau final (feuille) d'index non clustered d'une table sans index clustered.

```
DBCC PAGE (5,1,10640,3)
```
5

D'une part, on remarque que les données sont bien rangées, sur

l'ensemble des pages de cet index, suivant la clé numérique.
D'autre part, on constate que mis à part cette clé d'index, chaque élément d'une page feuille contient un HEAP RID. RID signifie ici Row IDentifier. Ainsi, le premier HEAP RID de 0xD01B000001001E00 de l'exemple, pour CleNumerique=1, se décompose de la manière suivante (attention à l'inversion des octets) :

- La partie gauche, contient le numéro de page (&x1BD0 = 7120) ;
- La partie centrale contient le numéro de fichier (1) ;
- La partie de droite contient le numéro de slot (&x1E = 30).

Allons vérifier :

```
DBCC PAGE (5,1,7120,3)
```

Extrait du résultat :

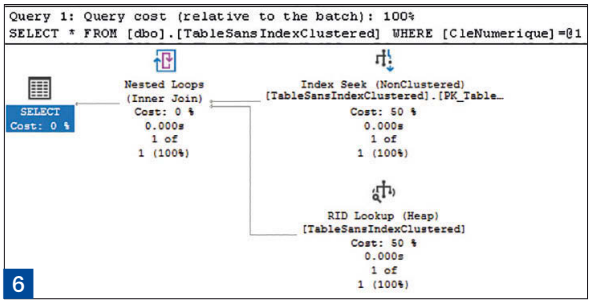
```
Slot 30 Offset 0x3a4 Length 23

Record Type = PRIMARY_RECORD     Record Attributes =  NULL_BITMAP VARIABLE_COLUMNS
Record Size = 23
Memory Dump @0x000000967FEF83A4

0000000000000000:  30000c00 63c20000 01000000 03000001 0017004e  0...cÂ.............N
0000000000000014:  756d31                                        um1

Slot 30 Column 1 Offset 0x4 Length 4 Length (physical) 4

Id = 49763

Slot 30 Column 2 Offset 0x8 Length 4 Length (physical) 4

CleNumerique = 1

Slot 30 Column 3 Offset 0x13 Length 4 Length (physical) 4

CleTexte = Num1
```

Cette démarche, consistant à passer par l'index non clustered, en extraire le HEAP RID et l'utiliser ensuite pour aller lire le reste des informations dans la page adéquate, correspond exactement à ce que fait le moteur SQL Server. On peut le constater en demandant l'affichage du plan d'exécution d'une requête simple de recherche, dans lequel on a notamment un RID Lookup pour chaque ligne retournée par la recherche dans l'index PK : 6
Au-delà de cette utilisation individuelle, on remarque aussi que chaque page feuille contient l'identification de la page précédente et de la page suivante. Cela permet au moteur de parcourir une liste d'enregistrements dans l'ordre de la clé d'index (par exemple pour une recherche de plage). Regardons par exemple les informations que l'on peut extraire de la deuxième page feuille de notre index PK :

```
SELECT page_id,prev_page_page_id,next_page_page_id
FROM sys.dm_db_page_info (DB_ID(),1,10664,'DETAILED')
```

page_id	prev_page_page_id	next_page_page_id
10664	10640	10665

Cette page connaît sa précédente et sa suivante dans la liste. Il est en de même au niveau intermédiaire. Cela nous donne donc le schéma suivant d'enchaînement des pages d'un index (le fameux arbre d'indexation …) : 7



6

Index clustered

En prenant une table pour laquelle la clé primaire est définie comme clustered, la liste des index est désormais :

```
SELECT object_id,name,index_id,type,type_desc,is_unique
FROM sys.indexes where object_id=OBJECT_ID('[dbo].[TableAvecIndexClustered]')
```

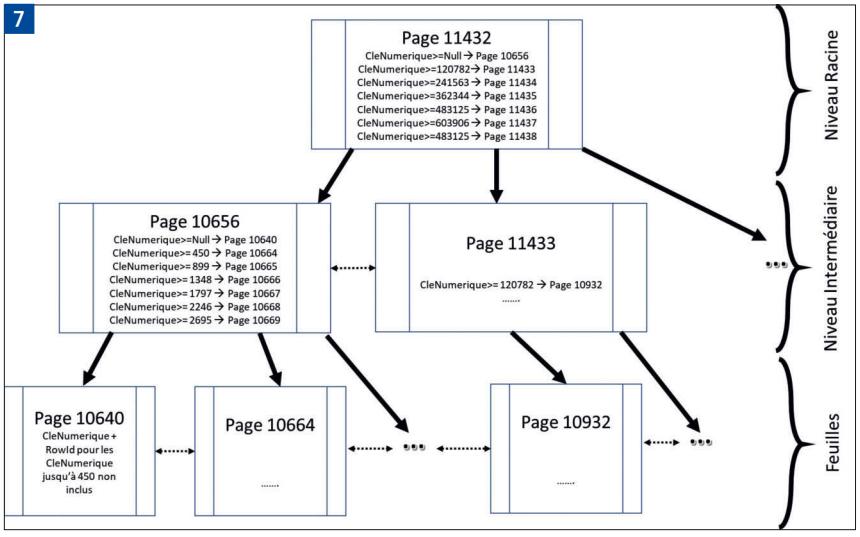
object_id	name	index_id	type	type_desc	is_unique
581577110	PK_TableAvecIndexClustered	1	1	CLUSTERED	1
581577110	Index_Non_Clustered_Texte	2	2	NONCLUSTERED	0

On note qu'il n'y a désormais plus de segment pour stocker les données, car elles sont directement stockées dans l'index clustered (qui a systématiquement pour identifiant 1).
Reprenons comme précédemment notre classement des pages d'index en les triant par niveau et en nous concentrant sur l'index clustered :

```
SELECT allocated_page_page_id, page_type_desc,
    page_level, previous_page_page_id, next_page_page_id
FROM sys.dm_db_database_page_allocations
    (DB_ID(),OBJECT_ID('[dbo].[TableAvecIndexClustered]'), 1, NULL, 'DETAILED')
WHERE page_type in (1,2) -- INDEX_PAGE ou DATA_PAGE
ORDER BY page_level DESC, previous_page_page_id
```

dont les premières lignes de résultat sont

7



allocated_page_page_id	page_type_desc	page_level	previous_page_page_id	next_page_page_id
1112	INDEX_PAGE	2	NULL	NULL
336	INDEX_PAGE	1	NULL	1113
1113	INDEX_PAGE	1	336	1114
1114	INDEX_PAGE	1	1113	1115
1115	INDEX_PAGE	1	1114	1116
1116	INDEX_PAGE	1	1115	1117
1117	INDEX_PAGE	1	1116	1118
1118	INDEX_PAGE	1	1117	1119
1119	INDEX_PAGE	1	1118	1120
1120	INDEX_PAGE	1	1119	1121
1121	INDEX_PAGE	1	1120	1122
1122	INDEX_PAGE	1	1121	1123
1123	INDEX_PAGE	1	1122	1124
1124	INDEX_PAGE	1	1123	NULL
328	DATA_PAGE	0	NULL	344
344	DATA_PAGE	0	328	345
345	DATA_PAGE	0	344	346
346	DATA_PAGE	0	345	347
347	DATA_PAGE	0	346	348
348	DATA_PAGE	0	347	349

On remarque qu'il y a un peu plus de pages de niveau intermédiaire que précédemment (13 pages ici contre 7 précédemment). La raison est présente dans le contenu d'une page de niveau feuille :

```
DBCC PAGE (5,1,328,3)
```

Extrait du résultat :

```
Slot 0 Offset 0x60 Length 23

Record Type = PRIMARY_RECORD      Record Attributes =  NULL_BITMAP VARIABLE_COLUMNS
Record Size = 23
Memory Dump @0x00000096019F8060

0000000000000000:  30000c00 01000000 63c20000 03000001 0017004e  0.......cÂ.........N
0000000000000014:  756d31                                        um1

Slot 0 Column 2 Offset 0x4 Length 4 Length (physical) 4

CleNumerique = 1

Slot 0 Column 1 Offset 0x8 Length 4 Length (physical) 4

Identifiant = 49763

Slot 0 Column 3 Offset 0x13 Length 4 Length (physical) 4

CleTexte = Num1

Slot 0 Offset 0x0 Length 0 Length (physical) 0

KeyHashValue = (8194443284a0)
```

La longueur de cet enregistrement est de 23 caractères (et encore, nous sommes sur le premier, c'est-à-dire un de ceux avec les plus petits nombres et donc les chaînes de caractères les plus courtes), là où précédemment on avait une valeur Row Size = 16.
On peut donc rentrer moins d'enregistrements dans une page feuille, il nous faut donc plus de pages feuilles et par conséquence plus de pages intermédiaires pour les adresser. Par ailleurs, les autres index (non clustered) ne contiennent plus de HEAP RID dans leurs feuilles mais la clé de l'index clustered.

```
SELECT top 1 allocated_page_page_id, page_type_desc,
    page_level, previous_page_page_id, next_page_page_id
FROM sys.dm_db_database_page_allocations
    (DB_ID(),OBJECT_ID('[dbo].[TableAvecIndexClustered]'), 2, NULL, 'DETAILED')
WHERE page_type in (1,2) -- INDEX_PAGE ou DATA_PAGE
    AND page_level=0 -- Feuilles
    AND previous_page_page_id is NULL -- LA première
ORDER BY page_level DESC, previous_page_page_id
```

allocated_page_page_id	page_type_desc	page_level	previous_page_page_id	next_page_page_id
4200	INDEX_PAGE	0	NULL	4216

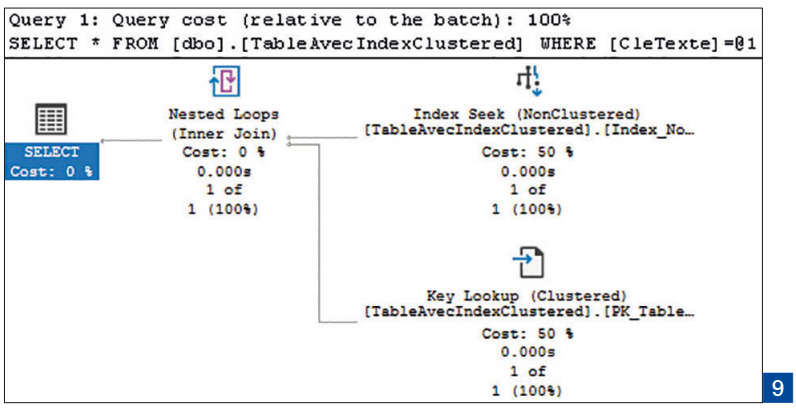
```
DBCC PAGE (5,1,4200,3)
```

8

A partir de cette clé d'entrée, le moteur peut désormais réaliser une recherche « Key Lookup », et non plus un « RID Lookup ». 9
A noter que la recherche RID Lookup peut être plus rapide sur de grosses tables subissant un nombre réduit de modifications. En effet, l'approche via une clé d'index peut consommer plus de lectures de pages (car on peut avoir à descendre un arbre d'index sur plusieurs niveaux, là où un HEAP RID nous emmènerait droit au but). Néanmoins, de nombreux facteurs plaident en faveur d'un index clustered bien choisi, comme la performance des lectures de plages de données, ou encore le comportement lors de modifications de données qui peut apporter dans des segments des sauts de redirection. Le choix de la politique d'indexation d'une base est tout un art. On constate trop souvent des moteurs de données qui n'atteignent pas les performances souhaitées du fait d'un manque d'index, mais aussi très souvent des bases de données avec trop d'index. Comme pour beaucoup de choses, il en faut ni trop, ni trop peu, et le fait de savoir ce qu'il y a sous le capot aide à prendre les bonnes décisions. •

FileId	PageId	Row	Level	CleTexte (key)	CleNumerique (key)	KeyHashValue	Row Size
1	4200	0	0	Num1	1	(281b827146e7)	16
1	4200	1	0	Num10	10	(7064ffd55481)	17
1	4200	2	0	Num100	100	(f03c1d9993d2)	18
1	4200	3	0	Num1000	1000	(d97ce5828f2)	19
1	4200	4	0	Num10000	10000	(2100ec2dadcf)	20
1	4200	5	0	Num100000	100000	(fee4c3cf4f41)	21
1	4200	6	0	Num1000000	1000000	(f6e2c559177a)	22
1	4200	7	0	Num100001	100001	(8fc0d8b708b9)	21
1	4200	8	0	Num100002	100002	(f64d66092b19)	21

8



9

Christophe PICHAUD
Architecte Microsoft chez 'Modern Applications by Devoteam'
christophepichaud@hotmail.com
www.windowscpp.com

XAML Islands : incorporez les contrôles XAML Windows 10 dans vos Apps WinForms et WPF

niveau 200

L'interface Windows 10 WinRT XAML ou UWP est faite en C++. Microsoft souhaite que les développeurs créent des applications XAML et les mettent à disposition sur le store. Il en va de même pour les applications WinForms ou WPF. Pour cela, l'éditeur propose XAML Islands, un SDK qui permet d'intégrer des composants WinRT XAML dans les applications WinForms et WPF. Ceci est valable depuis Windows 10 version 1903.

Comment ça marche

Il y a deux façons de faire, à savoir pour :

- Les applications C++ Win32 ;
- Les applications WinForms, WPF NET Framework et Net Core.

Note

Cet article ne couvre pas en détails les applications C++ Win32. Le focus est fait pour les applications .NET.

Windows 10 version 1903

À partir de Windows 10, version 1903, Microsoft fournit deux façons d'utiliser XAML Islands dans vos applications WPF, Windows Forms et C++ Win32 :

- Le kit de développement logiciel (SDK) Windows fournit plusieurs classes Windows Runtime et interfaces COM que votre application peut utiliser pour héberger tout contrôle UWP qui dérive de **Windows.UI.Xaml.UIElement**. Collectivement, ces classes et interfaces sont appelées l'API d'hébergement XAML UWP, et elles vous permettent d'héberger des contrôles UWP dans n'importe quel élément d'interface utilisateur dans votre application qui a un handle de fenêtre associé (HWND). Pour plus d'informations sur cette API, consultez Utilisation de l'API d'hébergement XAML.
- **Windows Community Toolkit** fournit également des contrôles XAML Islands supplémentaires pour WPF et Windows Forms. Ces contrôles utilisent l'API d'hébergement XAML UWP en interne et implémentent tout le comportement que vous auriez autrement besoin de gérer vous-même si vous avez utilisé l'API d'hébergement XAML UWP directement, y compris la navigation au clavier et les modifications de disposition. Pour les applications WPF et Windows Forms, nous vous recommandons vivement d'utiliser ces contrôles à la place de l'API d'hébergement XAML UWP directement, car ils implémentent beaucoup de détails d'implémentation de l'utilisation de l'API. Notez qu'à partir de Windows 10, version 1903, ces contrôles sont disponibles en tant que preview pour les développeurs.

Les applications de bureau C++ Win32 doivent utiliser l'API d'hébergement XAML UWP pour héberger des contrôles UWP. Les contrôles XAML Islands dans le Windows Community Toolkit ne sont pas disponibles pour les applications de bureau C++ Win32.

Il existe deux types de contrôle XAML Islands dans le Windows Community Toolkit pour les applications WPF et WinForms : contrôles encapsulés et contrôles hôtes.

Contrôles encapsulés

Les applications WPF et Windows Forms peuvent utiliser une sélection de contrôles UWP encapsulés dans le Windows Community Toolkit. Ils sont appelés contrôles encapsulés, car ils encapsulent l'interface et les fonctionnalités d'un contrôle UWP spécifique. Vous pouvez ajouter ces contrôles directement à l'aire de conception de votre projet WPF ou Windows Forms, puis les utiliser comme tout autre contrôle WPF ou Windows Forms dans votre designer. Les contrôles UWP encapsulés suivants pour implémenter XAML Islands sont actuellement disponibles pour les applications WPF et Windows Forms.

Control	Minimum supported OS	Description
InkCanvas InkToolbar	Windows 10, version 1903	Fournit une surface et des barres d'outils associées pour l'interaction utilisateur basée sur Windows Ink dans votre application de bureau Windows Forms ou WPF.
MediaPlayerElement	Windows 10, version 1903	Incorpore une vue qui diffuse et restitue le contenu multimédia tel que la vidéo dans votre application de bureau Windows Forms ou WPF.
MapControl	Windows 10, version 1903	Vous permet d'afficher une carte symbolique ou photoréaliste dans votre application de bureau Windows Forms ou WPF.

En plus des contrôles encapsulés pour les îles XAML, Windows Community Toolkit fournit également les contrôles suivants pour l'hébergement de contenu Web.

Control	Minimum supported OS	Description
WebView	Windows 10, version 1803	Utilise le moteur de rendu Microsoft Edge pour afficher le contenu Web.
WebViewCompatible	Windows 7	Fournit une version de WebView qui est compatible avec d'autres versions du système d'exploitation. Ce contrôle utilise le moteur de rendu Microsoft Edge pour afficher le contenu Web sur Windows 10 version 1803 et versions ultérieures, et le moteur de rendu Internet Explorer pour afficher le contenu Web sur les versions antérieures de Windows 10, Windows 8. x et Windows 7.

Hosting de Controls

Pour les scénarios au-delà de ceux couverts par les contrôles encapsulés disponibles comme WebView et Ink Control, les applications WPF et Windows Forms peuvent également utiliser le contrôle WindowsXamlHost du Windows Community Tookit (https://docs.microsoft.com/windows/uwpcommunitytoolkit/) . Ce contrôle peut héberger n'importe quel contrôle UWP qui dérive de Windows. UI. Xaml. UIElement, y compris tout contrôle UWP fourni par le SDK Windows ainsi que les contrôles utilisateur personnalisés. Ce contrôle prend en charge Windows 10 Insider Preview SDK Build 17709 et versions ultérieures.

Vue d'ensemble de l'architecture

Voici un aperçu de la façon dont ces contrôles sont organisés architecturalement. 1

Les API qui apparaissent au bas de ce schéma sont expédiés avec le kit de développement logiciel (SDK) Windows. Les contrôles encapsulés et les contrôles hôtes sont disponibles via les packages NuGet dans le Windows Community Toolkit.

Prérequis

XAMl Islands requiert Windows 1903.

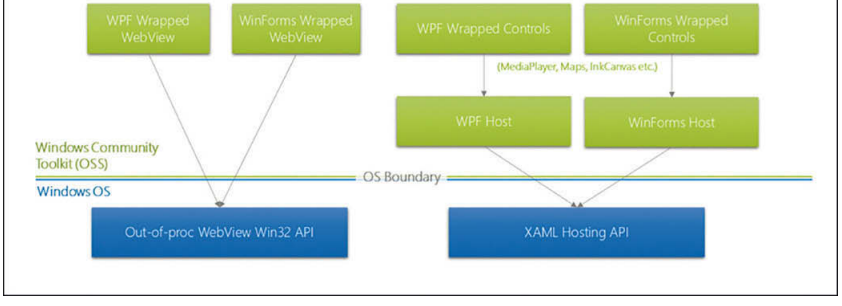
Les samples Microsoft du WCT

Le Windows Community Toolkit contient des exemples Webview, Ink pour WinForms et WPF et aussi des exemples custom XAML controls pour WinForms et WPF.

Exemple d'application WPF :

```
using System;
using System.Windows;
using Microsoft.Toolkit.Win32.UI.Controls.Interop.WinRT;
using windows = Windows;

namespace Microsoft.Toolkit.Sample.Wpf.App
{
  /// <summary>
  /// Interaction logic for MainWindow.xaml
  /// </summary>
  public partial class MainWindow : Window
  {
    private windows.UI.Xaml.Controls.ContentDialog _contentDialog;

    public MainWindow()
    {
      InitializeComponent();
    }

    private void inkCanvas_Loaded(object sender, RoutedEventArgs e)
    {
      inkCanvas.InkPresenter.InputDeviceTypes =
        CoreInputDeviceTypes.Mouse |
        CoreInputDeviceTypes.Pen |
        CoreInputDeviceTypes.Touch;
    }

    private void inkToolbar_Initialized(object sender, EventArgs e)
    {
      // Handle ink toolbar initialization events here
    }

    private void inkToolbar_ActiveToolChanged(object sender, object e)
    {
      // Handle ink toolbar active tool changed events here.
    }

    private async void myMap_Loaded(object sender, RoutedEventArgs e)
    {
      // Specify a known location.
      BasicGeoposition cityPosition =
        new BasicGeoposition() { Latitude = 47.604, Longitude = -122.329 };
      var cityCenter = new Geopoint(cityPosition);

      // Set the map location.
      await myMap.TrySetViewAsync(cityCenter, 12).ConfigureAwait(false);
    }

    private void WindowsXamlHost_Loaded(object sender, RoutedEventArgs e)
    {
      windows.UI.Xaml.Controls.StackPanel stackPanel = new windows.UI.Xaml.Controls.
StackPanel()
      {
       Background = new windows.UI.Xaml.Media.SolidColorBrush(windows.UI.Colors.Black),
      };

      stackPanel.Children.Add(new windows.UI.Xaml.Shapes.Rectangle()
      {
        Width = 50,
        Height = 75,
        Fill = new windows.UI.Xaml.Media.SolidColorBrush(windows.UI.Colors.Blue),
      });

      stackPanel.Children.Add(new windows.UI.Xaml.Shapes.Rectangle()
      {
        Width = 200,
        Height = 30,
        Fill = new windows.UI.Xaml.Media.SolidColorBrush(windows.UI.Colors.Red),
      });

      var button = new windows.UI.Xaml.Controls.Button()
      {
```

1



```
    Width = 160,
    Height = 60,
    HorizontalAlignment = windows.UI.Xaml.HorizontalAlignment.Center,
    Content = "ContentDialog UWP Button",
};
button.Tapped += Button_Tapped;
stackPanel.Children.Add(button);

stackPanel.Children.Add(new windows.UI.Xaml.Shapes.Rectangle()
{
    Width = 25,
    Height = 100,
    Fill = new windows.UI.Xaml.Media.SolidColorBrush(windows.UI.Colors.Green),
});

windows.UI.Xaml.Controls.Flyout flyout = new windows.UI.Xaml.Controls.Flyout();
flyout.Content = new windows.UI.Xaml.Controls.TextBlock()
    { Text = "Flyout content", };

var button2 = new windows.UI.Xaml.Controls.Button()
{
    Width = 300,
    Height = 40,
    HorizontalAlignment = windows.UI.Xaml.HorizontalAlignment.Center,
    Content = "Long UWP Button with Flyout",
    Flyout = flyout,
};
stackPanel.Children.Add(button2);

var comboBox = new windows.UI.Xaml.Controls.ComboBox()
{
    HorizontalAlignment = windows.UI.Xaml.HorizontalAlignment.Center,
};
comboBox.Items.Add("One");
comboBox.Items.Add("Two");
comboBox.Items.Add("Three");
comboBox.Items.Add("Four");
stackPanel.Children.Add(comboBox);

windows.UI.Xaml.Controls.Grid grid = new windows.UI.Xaml.Controls.Grid();
stackPanel.Children.Add(grid);

_contentDialog = new windows.UI.Xaml.Controls.ContentDialog();
_contentDialog.Content = new windows.UI.Xaml.Controls.TextBlock()
    { Text = "ContentDialog content", };
stackPanel.Children.Add(_contentDialog);

var popup = new windows.UI.Xaml.Controls.Primitives.Popup()
{
    Width = 50,
    Height = 50,
    ShouldConstrainToRootBounds = false,
    Child = new windows.UI.Xaml.Controls.TextBlock()
        { Text = "Popup child", },
};
grid.Children.Add(popup);
        windowsXamlHost.Child = stackPanel;
        popup.IsOpen = true;
    }

    private async void Button_Tapped(object sender, windows.UI.Xaml.Input.TappedRouted
EventArgs e)
    {
      await _contentDialog.ShowAsync(windows.UI.Xaml.Controls.ContentDialogPlacement.
Popup);
    }
  }
}
```

Voici le contenu de MainWindow.xaml :

```
<Window x:Class="Microsoft.Toolkit.Sample.Wpf.App.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:xamlhost="clr-namespace:Microsoft.Toolkit.Wpf.UI.XamlHost;assembly=Microsoft.
Toolkit.Wpf.UI.XamlHost"
    xmlns:controls="clr-namespace:Microsoft.Toolkit.Wpf.UI.Controls;assembly=Microsoft.Toolkit.
Wpf.UI.Controls"
   xmlns:controls1="clr-namespace:Microsoft.Toolkit.Wpf.UI.Controls;assembly=Microsoft.
Toolkit.Wpf.UI.Controls.WebView"
    xmlns:x1="clr-namespace:System.Windows.Markup;assembly=System.Xaml"
    mc:Ignorable="d"
    Title="MainWindow" Height="450" Width="800">
  <Grid >
    <TabControl>
      <TabItem>
        <TabItem.Header>Ink Controls (Canvas / Toolbar)</TabItem.Header>
        <DockPanel LastChildFill="True" >
          <controls:InkToolbar  DockPanel.Dock="Top" x:Name="inkToolbar"
            Grid.Row="0" TargetInkCanvas="{x1:Reference Name=inkCanvas}"
            Initialized="inkToolbar_Initialized"
            ActiveToolChanged="inkToolbar_ActiveToolChanged">
            <controls:InkToolbarCustomToolButton x:Name="toolButtonLasso" />
          </controls:InkToolbar>
          <!--Inking area -->
          <controls:InkCanvas x:Name="inkCanvas" DockPanel.Dock="Top" Loaded="ink
Canvas_Loaded"/>
        </DockPanel>
      </TabItem>
      <TabItem>
        <TabItem.Header>Samples</TabItem.Header>
        <xamlhost:WindowsXamlHost x:Name="windowsXamlHost" Loaded="Windows
XamlHost_Loaded" />
      </TabItem>
      <TabItem>
        <TabItem.Header>MediaPlayerElement</TabItem.Header>
        <controls:MediaPlayerElement x:Name="mediaPlayerElement"
            Source="https://mediaplatstorage1.blob.core.windows.net/windows-universal
-samples-media/elephantsdream-clip-h264_sd-aac_eng-aac_spa-aac_eng_commentary-srt
```

```
_eng-srt_por-srt_swe.mkv"
            AutoPlay="True" Margin="5" HorizontalAlignment="Stretch"
            VerticalAlignment="Stretch" AreTransportControlsEnabled="True" />
      </TabItem>

      <TabItem>
        <TabItem.Header>WebView</TabItem.Header>
        <!--WebBrowser /-->
        <controls1:WebViewCompatible Source="http://www.bing.com"
            HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>
        <!--WebBrowser Source="http://www.bing.com"/-->

      </TabItem>
      <TabItem>
        <TabItem.Header>MapControl</TabItem.Header>
        <controls:MapControl x:Name="myMap" Loaded="myMap_Loaded"/>
      </TabItem>
```

```
    </TabControl>
  </Grid>
</Window>
```

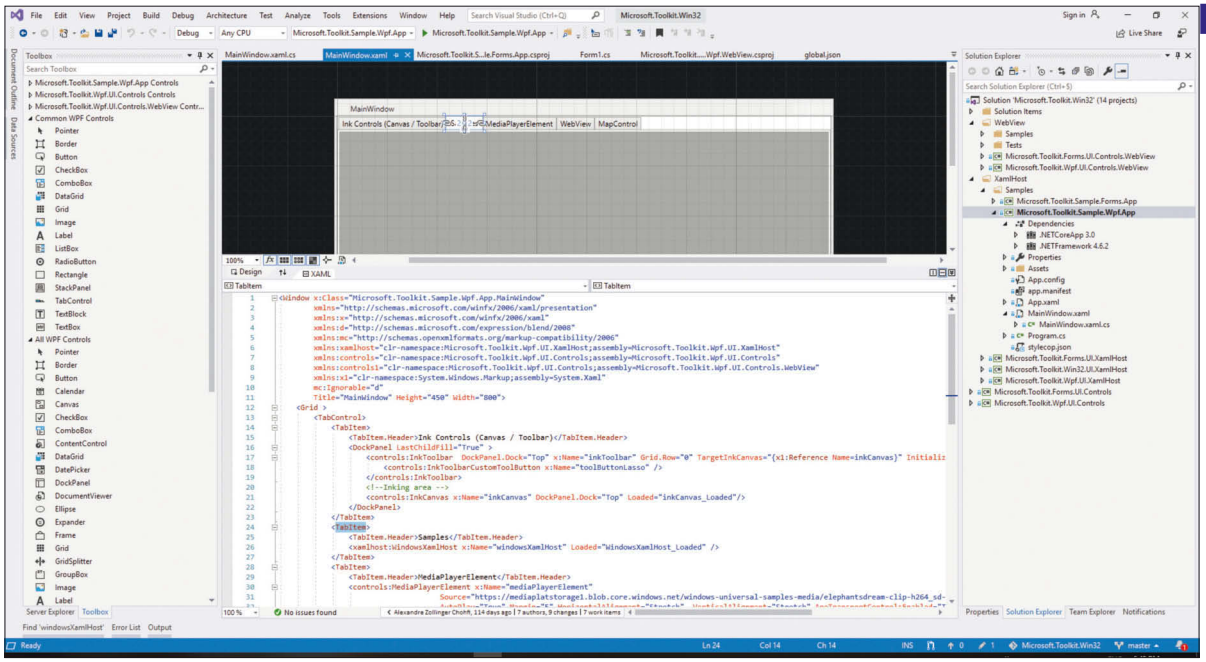
Voici ce que cela donne dans Visual Studio 2019 : 2

Conclusion

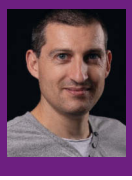
L'utilisation des contrôles UWP XAML C++ Windows dans les applications WinForms ou WPF est un challenge. Techniquement, le Windows Community Toolkit masque la complexité des API COM sous-jacentes.

Videos :

https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/XAML-Islands
https://channel9.msdn.com/Shows/On-NET/Integrating-UWP-components-into-Win32-applications



2



Emeric Martineau
Consultant DevOps à Zenika Nantes

zenika
<animés par la passion>

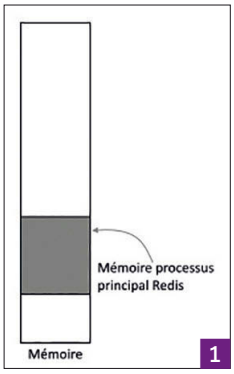
Sous le capot de Redis

Après avoir vu les bases de Redis dans l'article précédent (Programmez! 229), nous allons voir comment celui-ci fonctionne plus en profondeur.

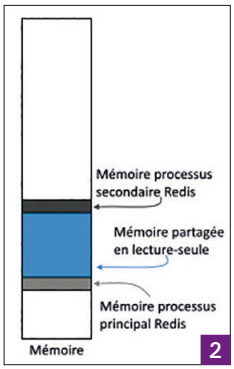
niveau 200

Processus de sauvegarde de la mémoire

Il est important de bien comprendre comment Redis sauvegarde sa mémoire au risque que celui-ci échoue tout le temps et génère des alertes système (notamment concernant les IO). Afin de continuer à traiter les requêtes entrantes, Redis va *forker* son processus et c'est le nouveau processus qui va écrire la mémoire sur disque. De prime abord, on est tenté de penser que le besoin de mémoire va être doublé et que cela peut être problématique dans le cas où il y a beaucoup de données (i.e. 8 Go). Toutefois, l'équipe Redis a fait ce choix volontairement. En effet, les systèmes d'exploitation de type Unix dupliquent la mémoire uniquement lorsque c'est nécessaire. Ils utilisent le C.O.W. (Copy On Write). Cela signifie que le nouveau processus partagera le même espace mémoire jusqu'à ce qu'une écriture sur la zone mémoire survienne. Cette technique permet d'économiser la mémoire et le temps de la sauvegarde. Mais attention, **seulement dans le cas où il n'y a aucune écriture** (1).



Il y a donc un cas où la mémoire peut être doublée, lorsque le *fork* est lancé et qu'il y a une requête d'écriture en entrée (2). Plus il y a de données, plus les données sont volumineuses et plus il y a de requêtes d'écriture, plus la probabilité augmente.



Dans ce cas, sous Linux, par défaut, s'il n'y a pas assez de mémoire disponible, le fork sera tué et la sauvegarde échouera. Alors, le processus primaire Redis relancera un fork qui lui aussi sera tué et ainsi de suite. Pour indiquer à Linux d'utiliser le SWAP si la mémoire vive disponible est trop faible (attention toutefois, si le SWAP est trop faible, un **out-of-memory** peut survenir), il est donc nécessaire d'activer l'overcommit memory à 1.

```
echo 1 > /proc/sys/vm/overcommit_memory
```

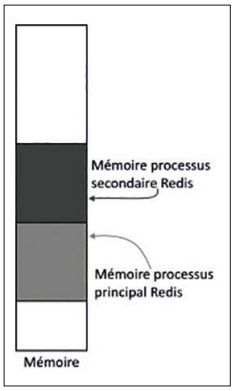
Ou dans /etc/sysctl.conf :

```
vm.overcommit_memory = 1
```

Il y aura aussi un impact en termes de performance (latence).
Il sera bon de surveiller le fichier /var/log/kern.log pour remonter les alertes sur ce type de message :

```
Jun 4 07:41:59 xxx kernel: [70667120.897649] Out of memory: Kill process 23 (redis-server) score 366 or sacrifice child
Jun 4 07:41:59 xxx kernel: [70667120.897701] Killed process 23 (redis-server) total-vm:2532680kB, anon-rss:1416508kB, file-rss:0kB
```

Il faudra aussi veiller à ce que l'espace disque soit suffisant. En effet, si le *fork* échoue lors de l'écriture, il va s'arrêter et le processus Redis primaire va le relancer indéfiniment.
Il y aura donc un nombre important IOPS qu'il faudra surveiller.



Sauvegarde des données pour reprise en cas de crash

Redis stocke sa mémoire dans un fichier RDB (Redis DataBase) et dans un fichier AOF.

```
# Répertoire où seront stockés les fichiers suivants
dir /var/redis/db
appendonly yes
appendfilename « appendonly.aof »

# Redis Database
dbfilename dump.rdb
```

Il faudra donc sauvegarder régulièrement (où en continu) le répertoire /var/redis/db.

Optimisations

Par défaut, Redis peut contenir 16 bases de données différentes. Souvent, une seule base de données est nécessaire. On peut donc indiquer à Redis d'utiliser une seule base de données dans le fichier de configuration afin d'économiser l'allocation mémoire nécessaire :

```
databases 1
```

De plus en plus, les OS installés sur les serveurs sont en 64 bits. De ce fait, on a tendance à installer les applications en 64 bits.
Dans le cadre de Redis, il peut être intéressant d'utiliser la version 32 bits. En effet, si le processus Redis utilise moins de 3,25 Go sous Linux, le fait de passer en 64 bits augmentera l'utilisation mémoire (les pointeurs mémoire passent de 4 octets à 8 octets).
Il est possible d'aller encore plus loin dans l'optimisation, mais cela sera uniquement intéressant en cas de problème de consommation mémoire (https://redis.io/topics/memory-optimization).

Éviter les latences

Il est difficile avant de monitorer Redis d'éviter d'éventuelles latences (réseaux, disque…). La documentation officielle présente une checklist à suivre. Sous Linux, par défaut, il faut activer l'*overcommit*, comme vu précédemment, et désactiver le *transparent huge pages*.

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

Ou dans /etc/sysctl.conf :

```
vm.nr_hugepages = 0
```

Si vous souhaitez comprendre précisément à quoi servent et comment fonctionnent les Huge Pages, vous pouvez vous reporter à l'article de Redhat « Huge Pages » et « Transparent Huge Pages ».

Supervision

Afin de s'assurer du bon fonctionnement de Redis, il faudra superviser :

- le processus (actif ou non),
- la mémoire disponible,
- la taille disque,
- les IOPS,
- la charge CPU (normalement faible),
- les logs.

Il est à noter que Redis peut être supervisé par *systemd* en ajoutant dans la configuration de Redis :

```
supervised systemd
```

Il faut ensuite créer un service, par exemple le fichier /etc/systemd/system/redis.service.

```
[Unit]
Description=Redis
After=syslog.target
[Service]
ExecStart=/usr/local/bin/redis-server /etc/redis/6379.conf
RestartSec=5s
Restart=on-failure
User=redis
[Install]
WantedBy=multi-user.target
```

La supervision se faisant par *systemd*, il est impératif de désactiver le mode démon de Redis car sinon *systemd* va relancer Redis.

```
daemonize no
```

Créer un utilisateur pour Redis :

```
sudo adduser –system –group –no-create-home redis
```

Bien positionner le user sur le répertoire indiqué dans la configuration db (ici /var/redis/db) ainsi que sur le fichier de configuration (/etc/redis/6379.conf). Pour rappel, Redis va écrire dans le fichier de configuration si nécessaire (lorsqu'on modifie sa configuration en ligne de commande par exemple). Ensuite, lancer le service :

```
sudo systemctl enable /etc/systemd/system/redis.service
sudo systemctl start redis.service
```

En cas de crash de Redis, *systemd* relancera automatiquement le service.

Supervision des log

Redis permet d'enregistrer les logs dans un fichier via la clef de configuration logfile.

```
# Specify the log file name. Also the empty string can be used to force
# Redis to log on the standard output. Note that if you use standard
# output for logging but daemonize, logs will be sent to /dev/null
logfile « »
```

Le niveau de log se fait via la configuration :

```
loglevel notice
```

Si l'on souhaite activer le mode *syslog*, il faut modifier ces lignes :

```
# To enable logging to the system logger, just set 'syslog-enabled' to yes,
# and optionally update the other syslog parameters to suit your needs.
# syslog-enabled no

# Specify the syslog identity.
# syslog-ident redis

# Specify the syslog facility. Must be USER or between LOCAL0-LOCAL7.
# syslog-facility local0
```

Ensuite, il faudra surveiller dans les logs les patterns suivants :

Impossible de sauvegarde sur disque

```
# MASTER aborted replication with an error: ERR Unable to perform background save
```

Perte du slave

```
# Connection with replica 127.0.0.1:6001 lost.
```

Supervision des metrics

Redis retourne un certain nombre d'informations utiles via la commande info. Voici une liste intéressante à superviser :

```
$ redis-cli -p 6000 info
# Server
uptime_in_seconds:185
uptime_in_days:0
# Clients
connected_clients:7
blocked_clients:0
# Memory (in byte)
used_memory:2040688
# Persistence
loading:0
rdb_changes_since_last_save:0
rdb_bgsave_in_progress:0
rdb_last_save_time:1542634866
rdb_last_bgsave_status:ok # sinon 'err' en cas d'erreur
# Stats
total_connections_received:11
total_commands_processed:1064
# CPU
used_cpu_sys:0.284100
used_cpu_user:0.214048
used_cpu_sys_children:0.000000
used_cpu_user_children:0.003277
```

Réinitialisation des données

Si vous utilisez Redis uniquement pour du cache, vous pouvez vider la base en vous connectant à Redis via la ligne de commande redis-cli et exécutant la commande flushdb.

Chiffrement des flux de données

Par défaut, Redis communique avec le client ou les autres serveurs en clair. Il est toutefois possible d'utiliser SSH ou Spiped pour cela. Il faudra configurer Redis pour écouter sur une interface loopback type localhost (par exemple 127.0.0.1) et activer le mode protégé. Pour cela ajouter la ligne suivante dans la configuration Redis :

```
protected-mode yes
```

SSH

Pour sécuriser Redis via SSH, il faut créer un tunnel SSH classique :

```
ssh -L 6379:localhost:6379 user@mydomainwhatever.net
```

Spiped

Spiped est un logiciel qui permet de sécuriser des flux sur le même principe qu'un tunnel SSH. Il est toutefois plus tolérant car si une connexion se ferme, les autres restent actives au contraire de SSH. De plus, les performances réseau sont meilleures. Spiped utilise une clef pré-partagée. A vous de jouer ! •

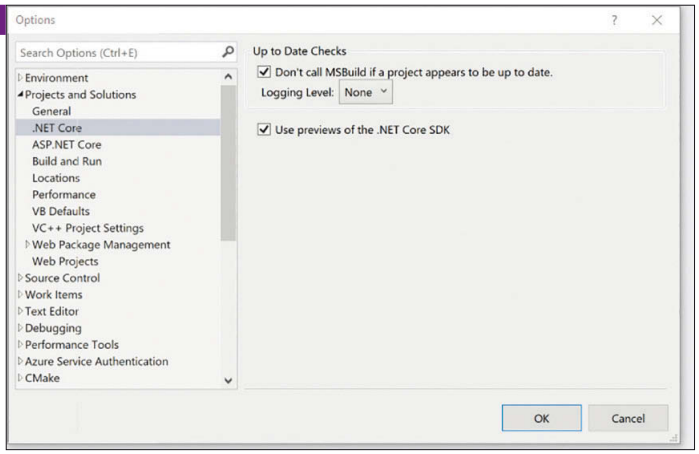
Jean-Pierre Gervasoni

Utilisation de WPF et de WinForms avec .Net Core 3.0 preview

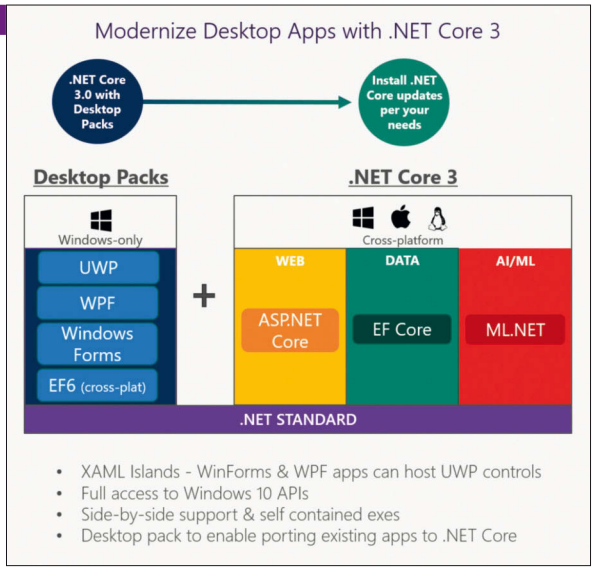
niveau 100

L'annonce en 2018 de .Net Core 3 s'est accompagnée d'une petite surprise, le support de WPF et de WinForms, via un SDK dédié à la plateforme Windows. Comme on le verra, il n'est pas très compliqué d'installer la version preview du SDK et de l'utiliser au travers de Visual Studio 2019 – attention toutefois, les designers WPF et WinForms ne sont pas (encore) disponibles en .Net Core. Nous aborderons les questions de portage d'application .net legacy, ainsi que les gains de performances liés.

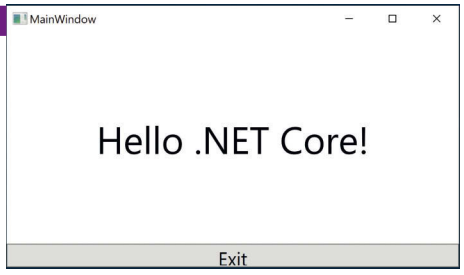
1 *Utilisation des previews de .net Core 3 avec VS2019*



2 *.Net Core résumé en une image*



3 *Hello world DotNet Core 3.0 WPF*



Installation du SDK .Net Core 3 :

A l'heure où cet article est écrit, aucun sdk n'est disponible. Nous pouvons toutefois installer des preview à partir du site https://dotnet.microsoft.com/download/dotnet-core/3.0. L'utilisation avec Visual Studio 2019 est possible en cochant une case dans les Options : 1

.Net Core 3, WinForms et WPF

Avec cette version de .net Core, il est possible d'utiliser WinForms et WPF, mais uniquement sur les versions Windows du SDK. A l'heure actuelle, rien n'est prévu pour les rendre cross-platform, à en croire https://github.com/dotnet/wpf/blob/master/Documentation/contributing.md: « *We also do not intend to accept contributions that provide cross-platform implementations for Windows Forms or WPF.* ». Mais il ne faut jamais dire jamais ?

Pourquoi développer une application WPF ou WinForms en .Net Core ? 2

Habitué des applications WPF dotnet standard, j'y vois a minima au moins un avantage : il n'est plus obligatoire d'installer le bon framework .net, grâce au déploiement autonome. La promesse de gains de performances n'est pas à négliger non plus. Mais à mon avis l'argument le plus pertinent, c'est que même si le framework .net sera toujours supporté, son petit frère .net Core devrait évoluer à un rythme plus élevé. Way go go !

Hello, World ! Et anatomie de projet .Net Core

Les aficionados de la ligne de commande se réjouirons, car pour créer un projet .Net Core 3 WPF, il suffit de tape en ligne de commande :

```
dotnet new wpf
```

Et ensuite pour l'exécuter :

```
dotnet run
```

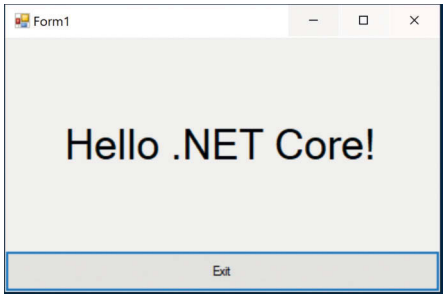
3

Le csproj d'un tel projet est simplissime :

```
<Project Sdk="Microsoft.NET.Sdk.WindowsDesktop">
 <PropertyGroup>
  <OutputType>WinExe</OutputType>
  <TargetFramework>netcoreapp3.0</TargetFramework>
```

```
  <UseWPF>true</UseWPF>
  <AssemblyName>DotnetCore3.DotNetCore3.WPF</AssemblyName>
  <RootNamespace>DotnetCore3.DotNetCore3.WPF</RootNamespace>
 </PropertyGroup>
</Project>
```

Pour un projet WinForms, dans les deux lignes de commande ci-dessus, il suffit de remplacer wpf par winforms : 4



4 *Hello world DotNet Core 3.0 WinForms*

Le csproj associé est très similaire au précédent :

```
<Project Sdk="Microsoft.NET.Sdk.WindowsDesktop">
 <PropertyGroup>
  <OutputType>WinExe</OutputType>
  <TargetFramework>netcoreapp3.0</TargetFramework>
  <UseWindowsForms>true</UseWindowsForms>
  <AssemblyName>DotnetCore3.DotNetCore3.WinForms</AssemblyName>
  <RootNamespace>DotnetCore3.DotNetCore3.WinForms</RootNamespace>
 </PropertyGroup>
</Project>
```

Rien ne les distingue, si ce n'est les propriétés UseWindowsForms et UseWPF. A noter que les deux peuvent être présentes et valant true, par exemple dans le cas d'un dialogue WinForms hébergeant un contrôle WPF.

Avec des csproj aussi minimalistes, comment sont listés les fichiers sources ? Il semblerait qu'il suffise de rajouter un fichier au niveau du csproj pour qu'il soit automatiquement détecté et inclus dans la solution, en fait cela va même plus loin. Prenons l'exemple du projet WPF ci-dessus :

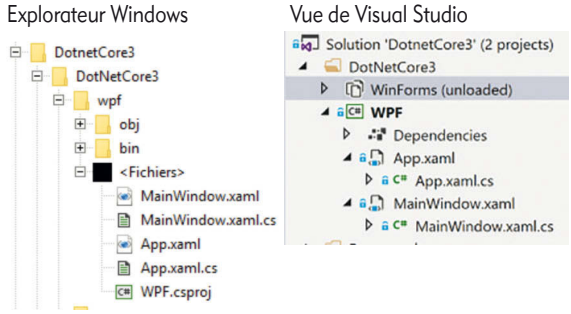


Figure 5 *: Structure du projet WPF avant ajout d'un fichier*

Maintenant, rajoutons via l'explorateur Windows un répertoire *test* dans lequel nous plaçons un fichier vide *test.cs*. Voici ce que l'on obtient sans autre intervention : 6

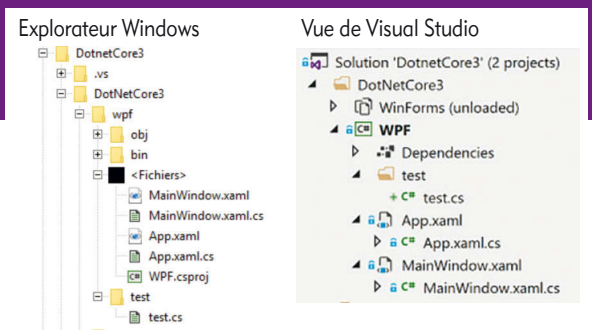


Figure 6 *: Structure du projet WPF après ajout d'un fichier*

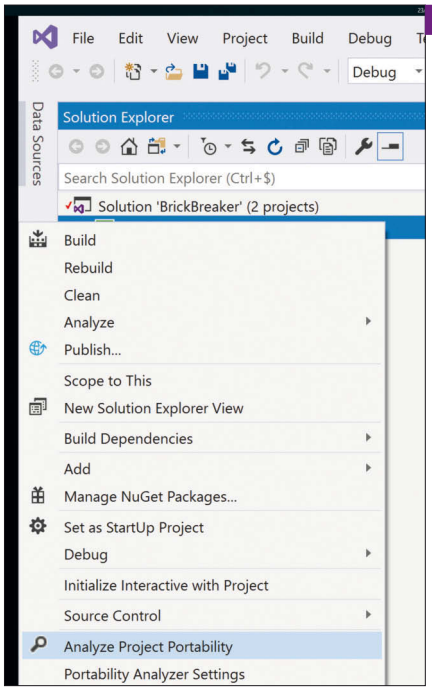
A noter que cela marche de manière récursive – sacré gain de productivité. Plus besoin d'ajouter à la main les fichiers via Visual Studio.

Portage d'une application WinForms vers .Net Core 3

J'ai forké sous GitHub un projet WinForms minimaliste inspiré du fameux *Breakout*, appelé *BreakoutWinForms* disponible sous https://github.com/nitrogene/BreakoutWinForms. Le portage lui-même a été effectué en suivant les instructions figurant dans le billet suivant https://devblogs.microsoft.com/dotnet/how-to-port-desktop-applications-to-net-core-3-0/. Cela se résume en :

- Installation du sdk .Net Core 3.0 et de Visual Studio 2019 ;
- Partir d'une solution qui builde, sans bugs ;
- Mettre à jour les package NuGet le cas échéant ;
- Lancer « .Net Portability Analyzer » (disponible dans le Marketplace de Visual Studio) pour détecter les APIs non supportées par DotNetCore, puis ensuite refactorer le code, changer de package NuGet, pour éviter ces APIs : 7
- Si vous avez des packages NuGet, faites un clic-droit sur le fichier *package.config* et « *Migrate packages.config to PackageReference* » ;
- Créer un projet dans Visual Studio du même type (WPF ou WinForms) ;
- Dans le cas du fork de Breakout, tout ce que j'ai eu à faire pour migrer le code, fut de copier/coller les fichiers d'un projet à l'autre. Pour un vrai projet, je vous encourage vivement à aller voir le lien précédent.

On trouvera sur le lien suivant un autre exemple de portage d'une application WPF en .Net Core 3.0 : https://weblog.west-wind.com/



7 *.Net Portability Analyzer*

posts/2019/Apr/24/First-Steps-in-porting-Markdown-Monster-to-NET-Core-30?

A noter que le designer WinForms n'est pas encore opérationnel, et le sera via un update de Visual Studio 2019, https://github.com/dotnet/winforms: "*Note: The Windows Forms visual designer is not yet available and will be part of a Visual Studio 2019 update. See here for a workaround invoking the Classic Framework Designer*".

Le lien mentionné est le suivant https://github.com/dotnet/winforms/blob/master/Documentation/winforms-designer.md.

Le designer WPF ne semble pas non plus opérationnel.

Quid des performances de .Net Core ?

Comme Saint Thomas d'Aquin, je ne crois que ce que je vois, du coup, j'ai eu envie de comparer les performances d'un même code C# selon différents runtimes. L'outil idéal pour cela est la librairie BenchmarkDotNet, disponible via Github https://github.com/dotnet/BenchmarkDotNet ou Nuget.

Lors de mes recherches sur ce sujet, je suis tombé sur un billet de Stephen Toub datant d'Avril 2018, détaillant des gains de performances entre .Net Core 2.0 et 2.1. Le billet original est disponible ici :

https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-core-2-1/.

J'ai rassemblé ces différents tests sur Github : https://github.com/nitrogene/Benchmark. Voici quelques extraits. Le lecteur curieux se reportera à la page Github associée au dépôt pour trouver le code testé et des comparaisons exhaustives. Mon environnement de test est le suivant :

```
BenchmarkDotNet=v0.11.5, OS=Windows 10.0.17763.437 (1809/October2018Update/Redstone5)
Intel Core i7-6700HQ CPU 2.60GHz (Skylake), 1 CPU, 8 logical and 4 physical cores
  [Host]    : .NET Framework 4.7.2 (CLR 4.0.30319.42000), 64bit RyuJIT-v4.8.3761.0
  Job-BFNISR : .NET Core 2.0.9 (CoreCLR 4.6.26614.01, CoreFX 4.6.26614.01), 64bit RyuJIT
  Job-OXVCGU : .NET Core 2.1.9 (CoreCLR 4.6.27414.06, CoreFX 4.6.27415.01), 64bit RyuJIT
  Job-AQDXUX : .NET Core 2.2.4 (CoreCLR 4.6.27521.02, CoreFX 4.6.27521.01), 64bit RyuJIT
  Job-KTSQXX : .NET Core 3.0.0-preview4-27615-11 (CoreCLR 4.6.27615.73, CoreFX 4.700.19.
21213), 64bit RyuJIT
  Clr       : .NET Framework 4.7.2 (CLR 4.0.30319.42000), 64bit RyuJIT-v4.8.3761.0

Platform=X64  Runtime=Clr
```

Method	Job	Toolchain	Mean	Ratio
Sha256	Default	.NET Core 3.0	48.95 us	0.52
Sha256	Clr	Default	94.14 us	1.00

Tableau 1 *Calcul de hash*

Method	Job	Toolchain	Mean	Ratio
Where	Default	.NET Core 3.0	49.06 us	0.40
Where	Clr	Default	123.55 us	1.00

Tableau 2 *Requête Linq Where*

Method	Job	Toolchain	Mean	Ratio
Sin	Default	.NET Core 3.0	204.11 us	0.41
Sin	Clr	Default	493.33 us	1.00

Tableau 3 *Math.Sin*

Method	Job	Toolchain	Mean	Ratio
StringEquals	Default	.NET Core 3.0	12.18 ns	0.57
StringEquals	Clr	Default	21.24 ns	1.00
StringIndexOf	Default	.NET Core 3.0	18.69 ns	0.18
StringIndexOf	Clr	Default	101.66 ns	1.00
IndexOfAny	Default	.NET Core 3.0	18.87 ns	0.14
IndexOfAny	Clr	Default	138.91 ns	1.00
StringToLowerChangesNeeded	Default	.NET Core 3.0	59.11 ns	0.28
StringToLowerChangesNeeded	Clr	Default	211.58 ns	1.00
StringToLowerAlreadyCased	Default	.NET Core 3.0	41.64 ns	0.19
StringToLowerAlreadyCased	Clr	Default	214.63 ns	1.00
StringSplit	Default	.NET Core 3.0	257.87 ns	0.72
StringSplit	Clr	Default	358.82 ns	1.00
StringConcatCharEnumerable	Default	.NET Core 3.0	14,554.08 ns	0.50
StringConcatCharEnumerable	Clr	Default	28,958.84 ns	1.00

Tableau 4 *Manipulations de chaînes*

Method	Job	Toolchain	Mean	Ratio
StringFormat	Default	.NET Core 3.0	121.52 ns	0.61
StringFormat	Clr	Default	199.90 ns	1.00
StringBuilderAppend	Default	.NET Core 3.0	2,965,858.10 ns	0.37
StringBuilderAppend	Clr	Default	8,052,928.96 ns	1.00
Int32Formatting	Default	.NET Core 3.0	35.79 ns	0.46
Int32Formatting	Clr	Default	78.21 ns	1.00
Int32Parsing	Default	.NET Core 3.0	22.37 ns	0.20
Int32Parsing	Clr	Default	110.41 ns	1.00
DoubleFormatting	Default	.NET Core 3.0	268.78 ns	0.46
DoubleFormatting	Clr	Default	580.71 ns	1.00
BigIntegerFormatting	Default	.NET Core 3.0	56.49 ns	0.65
BigIntegerFormatting	Clr	Default	87.08 ns	1.00
DateTimeOffsetFormatR	Default	.NET Core 3.0	80.27 ns	0.32
DateTimeOffsetFormatR	Clr	Default	253.87 ns	1.00
DateTimeOffsetFormatO	Default	.NET Core 3.0	129.25 ns	0.17
DateTimeOffsetFormatO	Clr	Default	754.35 ns	1.00
ConvertFromBase64String	Default	.NET Core 3.0	1,648.02 ns	0.79
ConvertFromBase64String	Clr	Default	2,079.47 ns	1.00
ConvertFromBase64Chars	Default	.NET Core 3.0	1,645.37 ns	0.79
ConvertFromBase64Chars	Clr	Default	2,082.23 ns	1.00

Tableau 5 *Formatting et parsing*

Method	Job	Toolchain	Mean	Ratio
fIPAddressNetworkToHostOrder	Default	.NET Core 3.0	0.0145 ns	0.001
IPAddressNetworkToHostOrder	Clr	Default	10.0213 ns	1.000
UriAllocations	Default	.NET Core 3.0	716.4724 ns	0.62
UriAllocations	Clr	Default	1,164.3810 ns	1.00
SocketReceiveThenSend	Default	.NET Core 3.0	41,577,366.1111 ns	0.67
SocketReceiveThenSend	Clr	Default	61,856,634.5588 ns	1.00

Tableau 6 *Networking*

Conclusion

Bien que la version de .Net Core 3.0 ne soit pas encore définitive, ses performances sont déjà impressionnantes. Avec le support de WinForms, WPF et Entity Framework 6, je pense que beaucoup d'applications vont être portées vers ce framework, d'autant plus que des outils sont fournis pour nous guider dans cette tâche. Pour aller plus loin, je vous conseille de surveiller les dépôts Github suivants : .Net Core Repos : https://github.com/dotnet/core/blob/master/Documentation/core-repos.md

WPF : https://github.com/dotnet/wpf

WinForms : https://github.com/dotnet/winforms

Christophe PICHAUD
Architecte Microsoft chez
'Modern Applications by Devoteam'
christophepichaud@hotmail.com
www.windowscpp.com

Introduction à ASP.Net Core 3.0

ASP.Net Core est le framework multiplateformes et open source de Microsoft pour faire des applications Web. Cela peut être des sites web, des services Web API, des apps IoT et cela tourne sur le cloud ou sur des serveurs on-premise.

Choix de Framework

Le tableau ci-dessous compare ASP.NET Core et ASP.NET 4.x. :

ASP.NET Core	ASP.NET 4.x
Cible Windows, macOS, ou Linux	Cible Windows
Razor Pages est le modèle de développement recommandé pour créer des UI Web avec ASP.NET Core 3.0. Voir aussi MVC, Web API, and SignalR.	Utilise Web Forms, SignalR, MVC, Web API, WebHooks, or Web Pages
Plusieurs versions par machine	Une version par machine
Développement avec Visual Studio, Visual Studio for Mac, ou Visual Studio Code zn utilisant C# ou F#	Développement op avec Visual Studio using C#, VB, ou F#
Meilleure performance que ASP.NET 4.x	Bonne performance
.NET Framework ou .NET Core runtime	Utilise le .NET Framework runtime

Pourquoi ASP.NET Core?

Asp.Net 4.x est un framework très utilisé. ASP.NET Core est la nouvelle version, plus rapide, plus modulaire. On y trouve les fonctionnalités suivantes :

- Une API unifiée pour les UI web et les web APIs ;
- Architecturé pour la testabilité ;
- Les pages Razor facilitent et rendent plus productifs les scénarios axés sur la page de codage ;
- Possibilité de développer et d'exécuter sur Windows, macOS et Linux ;
- Open source et axé sur la communauté ;
- Intégration des frameworks modernes, côté client et des workflows de développement ;
- Un système de configuration basé sur l'environnement, prêt pour le Cloud ;
- Injection de dépendances intégrée ;
- Un pipeline de requêtes HTTP léger, performant et modulaire ;
- Possibilité d'héberger sur IIS, Nginx, Apache, docker ou auto-Host dans votre propre processus ;
- Versionning des applications côte à côte avec .NET Core ;
- Outillage qui simplifie le développement Web moderne.

Fonctionnalités .NET Core 3 dans les diverses Preview

Pour vous donner une idée de la dynamisme du projet ASP.NET Core, voici la liste des fonctionnalités proposées dans les diverses Preview de .NET Core 3.

.NET Core 3.0 Preview 2 propose :

- Razor Components ;
- Le streaming SignalR client-server ;
- Les Pipes sur HttpContext ;
- Des templates de host génériques ;
- Des mise à jour sur les routing Endpoint.

.NET Core 3.0 Preview 3 propose :

- Améliorations des composants Razor :
  - Template de projet simple,
  - Nouvelle extension.razor,
  - Intégration des routing Endpoint,
  - Prerendering,
  - Composants Razor dans des DLL de classes Razor,
  - Gestion des évènements améliorée,
  - Forms & validation.
- Compilation au Runtime ;
- Template de Worker Service ;
- Template gRPC ;
- Template Angular mis à jour pour Angular 7 ;
- Authentification pour les Single Page Applications ;
- Intégration SignalR avec les routing endpoints ;
- Support SignalR Java client pour le polling long.

.NET Core 3.0 Preview 4 propose :

- Composants Razor renommés Blazor server ;
- Blazor client sur WebAssembly est officiel ;
- Resolution de nom de components basée sur @using ;
- _Imports.razor ;
- Template Nouveau composant ;
- Reconnexion au même serveur ;
- Reconnexion stateful après prerendering ;
- Composants interactifs de rendu stateful depuis les pages et vues Razor ;
- Détection quand une app fait du prerendering ;
- Configuration du client SignalR pour les apps Blazor server ;
- Amélioration des fonctionnalités de reconnexion SignaR ;
- Configuration du client SignalR pour les apps Blazor server ;
- Options additionnelles pour le service de registration MVC ;
- Mises à jour des routing Endpoint ;
- Nouveau template gRPC ;
- Design-time build pour gRPC ;
- SDK New Worker.

Réaliser des application web UI et des web APIs avec ASP.NET Core MVC

ASP.NET Core MVC fournit des fonctionnalités pour créer des API Web et des applications Web :

- Le modèle MVC (Model-View-Controller) rend vos API Web et vos applications Web testables ;
- Les pages Razor sont un modèle de programmation basé sur une page qui rend l'UI Web plus facile à réaliser et plus productive ;
- Les balises Razor sont une syntaxe productive pour les pages Razor et les vues MVC ;
- La prise en charge intégrée de plusieurs formats de données et la négociation de contenu permet à vos API Web d'atteindre un large éventail de clients, y compris les navigateurs et les appareils mobiles ;
- Le modèle de binding mappe automatiquement les données des requêtes HTTP aux paramètres de la méthode d'action ;
- Le modèle de validation effectue automatiquement la validation côté client et côté serveur.

Développement client

ASP.NET Core intègre facilement les frameworks clients (JS, TS) les plus connus, comme Blazor, Angular, React, and Bootstrap.

ASP.NET Core et.NET Framework

ASP.NET Core 2.x peut tourner sur.NET Core ou.NET Framework. Les applications ASP.NET Core pour .NET Framework ne sont pas cross-platforms - elles ne tournent que sur Windows.

ASP.NET Core 3 ne tourne que sur.NET Core. Les avantages de cibler .NET Core sont :

- Cross-platform. Tourne sur macOS, Linux, et Windows ;
- Meilleures performances ;
- Exécution Side-by-side ;
- Nouvelles APIs ;
- Open source.

Introduction à ASP.NET Core MVC

ASP.NET Core MVC est un Framework riche pour la création d'applications Web et d'API à l'aide du pattern Model-View-Controller.

Le pattern MVC

Le pattern d'architecture Model-View-Controller (MVC) architectural sépare une application en 3 groupes de composants : Models, Views, et Controllers. À l'aide de ce modèle, les demandes des utilisateurs sont acheminées vers un contrôleur qui est responsable de l'utilisation du modèle pour effectuer des actions utilisateur et/ou récupérer les résultats des requêtes. Le Controller choisit la vue à afficher à l'utilisateur, et lui fournit toutes les données de modèle qu'il exige. 1

Responsabilités du modèle

Le Model dans une application MVC représente l'état de l'application et toute logique métier ou opérations qui doivent être exécutées par elle. La logique métier doit être encapsulée dans le modèle, ainsi que toute logique d'implémentation pour la persistance de l'état de l'application. Les vues fortement typées utilisent généralement des types ViewModel conçus pour contenir les données à afficher sur cette vue. Le contrôleur crée et remplit ces instances ViewModel à partir du modèle.

Responsabilités des vues

Les vues sont chargées de présenter le contenu via l'interface utilisateur. Elles utilisent le moteur de vue Razor pour incorporer du code .NET dans le balisage HTML. Il devrait y avoir une logique minimale dans les vues, et toute logique en elles doit se rapporter à la présentation de contenu. Si vous trouvez la nécessité d'effectuer une grande partie de la logique dans les fichiers de vue afin d'afficher des données à partir d'un modèle complexe, envisagez d'utiliser un composant de vue, ViewModel ou un modèle de vue pour simplifier la vue.

Responsabilités du contrôleur

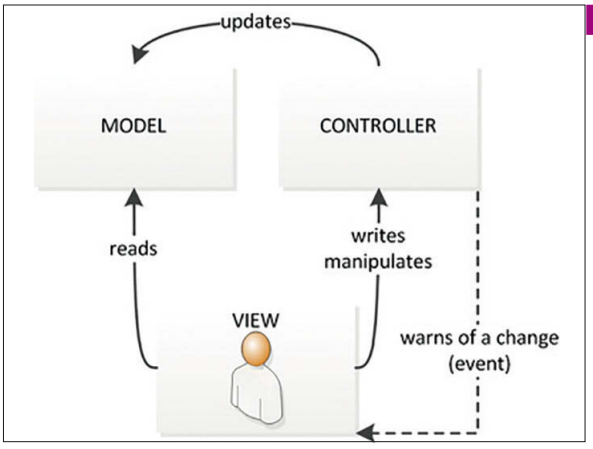
Les contrôleurs sont les composants qui gèrent l'interaction de l'utilisateur, travaillent avec le modèle et sélectionnent finalement une vue à restituer. Dans une application MVC, la vue affiche uniquement les informations; le contrôleur gère et répond à l'entrée et à l'interaction de l'utilisateur. Dans le modèle MVC, le contrôleur est le point d'entrée initial. Il est responsable de la sélection des types de modèles à utiliser et de la vue à restituer (d'où son nom). Il contrôle la façon dont l'application répond à une requête.

ASP.NET Core MVC

ASP.NET Core MVC est un Framework léger de présentation, open source, hautement testable, optimisé pour une utilisation avec ASP.NET Core. ASP.NET Core MVC fournit un moyen basé sur des modèles pour créer des sites Web dynamiques qui permet une séparation nette des responsabilités. Il vous donne un contrôle total au travers des balises, prend en charge le développement compatible TDD et utilise les dernières normes Web.

ASP.NET Core MVC fournit les services suivants :

- Routing ;
- Model binding ;
- Model validation ;
- Dependency injection ;
- Filters ;
- Areas ;
- Web APIs ;
- Testability ;
- Razor view engine ;
- Strongly typed views ;
- Tag Helpers ;
- View Components.



1

Architecture ASP.NET Core 2

Hébergement : processus natif

Au bas de la figure est la couche d'hébergement. L'hébergement est une petite couche de code natif qui est responsable de trouver et appeler l'hôte natif. Plusieurs hôtes sont disponibles, dont le ASP.NET Core Module, Kestrel et DOTNET.exe. Le ASP.NET Core module permet aux applications ASP.NET Core d'être hébergées dans IIS. Dotnet. exe est un outil de ligne de commande qui peut être utilisé pour créer et exécuter vos applications ASP.NET à des fins de développement et de test. Kestrel est un serveur HTTP multiplateforme haute performance basé sur la bibliothèque d'e/s asynchrones libuv. Il offre les meilleures performances pour héberger votre application ASP.NET Core. Cependant, Kestrel n'offre pas le même niveau de fonctionnalités comme IIS. Par exemple, si votre application doit utiliser l'authentification Windows, vous devez utiliser Kestrel en conjonction avec IIS.

Si vous en avez besoin, vous pouvez créer un hôte personnalisé pour héberger votre application ASP.NET. Cela pourrait être une Application WPF, un outil de ligne de commande ou un service Windows.

Runtime: CLR Native Host

La couche Runtime configure et démarre le CLR et crée le domaine d'application pour le code managé pour tourner dedans. Lorsque l'application d'hébergement s'arrête, la couche d'exécution est chargée de nettoyer les ressources utilisées par le CLR, puis l'arrêter. Dans la couche Runtime, les machines Windows auront une implémentation de code natif du CLR. Vous avez également la possibilité d'exécuter l'implémentation mono du CLR. Le projet mono offre des hôtes natifs pour macOS, Linux et Windows.

Runtime : point d'entrée managé

Le point d'entrée managé est écrit en code natif. Le but principal de cette couche est de trouver et de charger les assemblages requis. Une fois les assemblies chargés, le point d'entrée managé est appelé et l'application commence à s'exécuter.

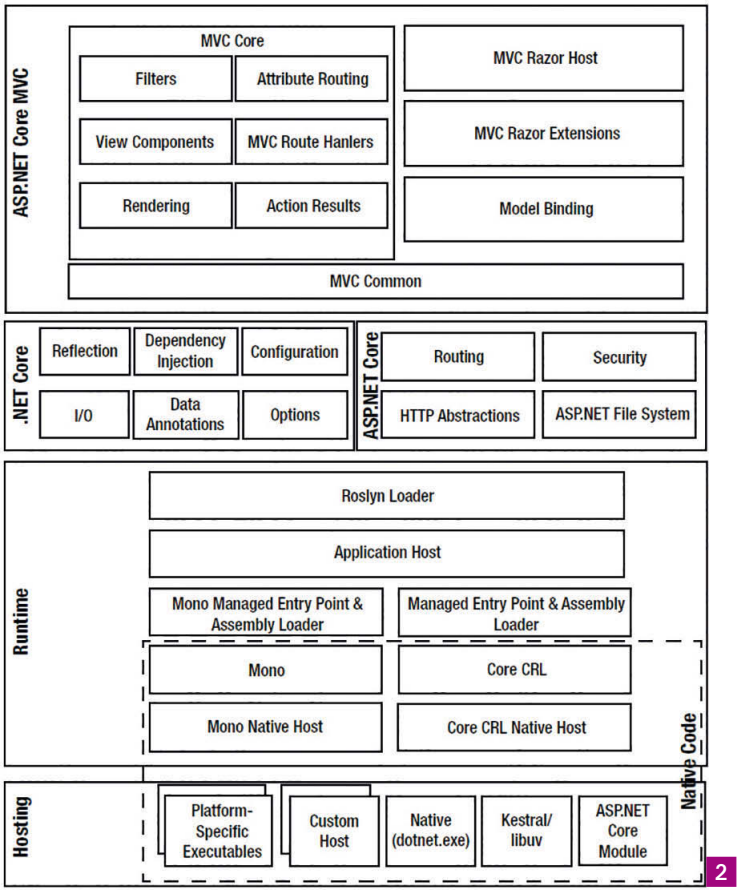
Runtime : hôte d'application

La couche d'hébergement d'application du runtime est la première couche dans laquelle un développeur Web est généralement impliqué. L'hôte de l'application lit le fichier .csproj du projet et détermine les dépendances. Il peut localiser des assemblies à partir de nombreuses sources, y compris NuGet et les assemblies compilés en mémoire par les services du compilateur Roslyn. Dans une application ASP.NET Core, cette couche créera également le pipeline ASP.NET Core et chargera les composants middleware spécifiés.

Runtime : Roslyn Loader

Le chargeur Roslyn est responsable du chargement et de la compilation des fichiers sources. Avec ASP.NET Core, le code de l'application n'a pas besoin d'être compilé avant son déploiement. Il peut être déployé en tant que code source et compilé à la demande.

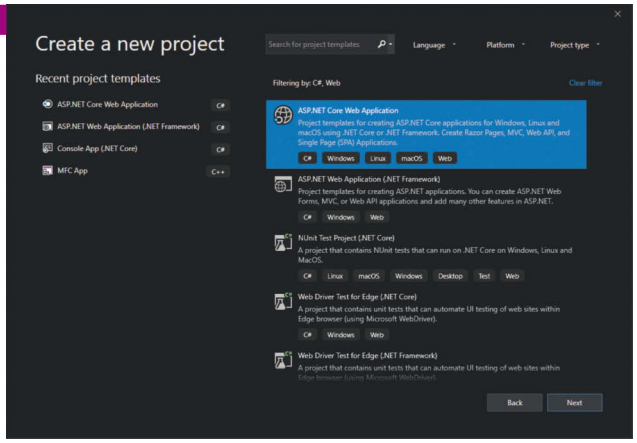
Dans la pile ASP.NET Core, il existe de nombreux composants requis par ASP.NET Core MVC. Certains des plus importants sont indiqués dans le schéma ci-dessus.



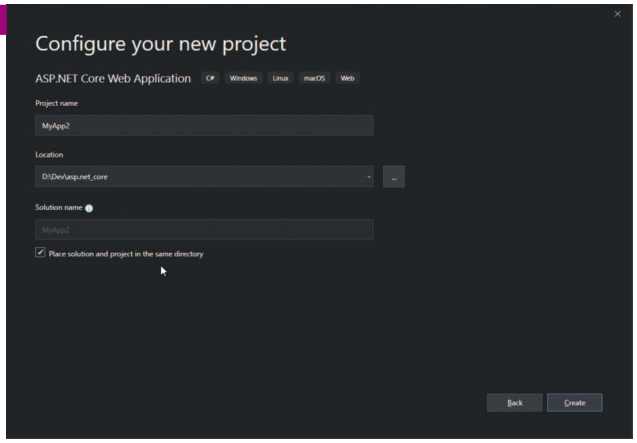
2

- Routage : ce composant contient la logique permettant de mapper les requêtes HTTP composant d'application ou de ressource statique. Il vous permet de personnaliser l'URL pour votre application ou service Web.
- Sécurité : ce composant est constitué d'une collection de fournisseurs basés sur OWIN connus collectivement comme ASP.NET Identity. ASP.NET Identity prend en charge plusieurs authentifications et les normes d'autorisation, y compris l'authentification SAML, OAuth et Windows, ainsi que des bases de données utilisateur custom avec l'authentification des cookies. Cette fonctionnalité vous permet d'intégrer rapidement votre application à des fournisseurs d'identité externes tels que Facebook, Microsoft et Google ou d'intégrer votre application à Active Directory. ASP.NET Identity inclut également la prise en charge des fonctionnalités de sécurité avancées comme l'authentification à deux facteurs.
- Abstractions http : ce sont de nouveaux composants qui font partie d' ASP.NET Core. Ils créent une couche d'abstraction qui garantit des API et des comportements cohérents, quel que soit l'endroit où vous hébergez votre application ASP.NET Core.
- Système de fichiers ASP.NET Core : cela fournit la gestion des fichiers statiques pour vos applications Web. Ceci est nécessaire car contrairement aux versions antérieures de ASP.NET, qui ont délégué le chargement et la portion de fichiers statiques tels que des images et des fichiers CSS sur le serveur Web, ASP.NET Core avait besoin d'un ensemble standard d'abstractions qui seraient cohérentes entre les différents serveurs Web et systèmes d'exploitation.

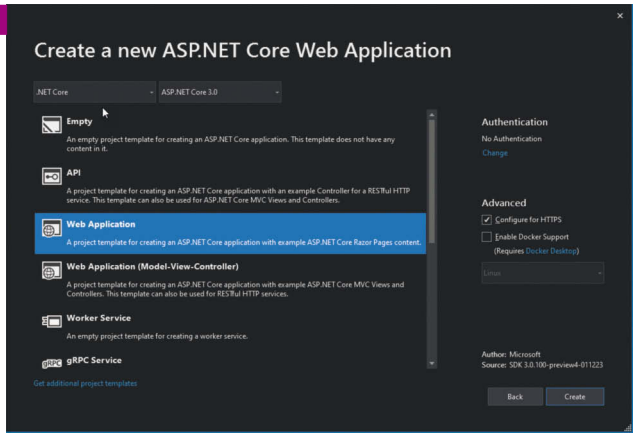
3



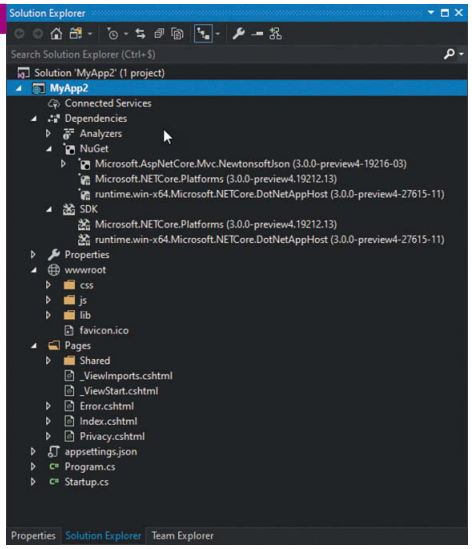
4



5



6



Créer une application ASP.NET Core avec Visual Studio 2019

Les nouveaux développements doivent se faire avec des pages Razor. C'est la préconisation Microsoft. Lançons Visual Studio 2019 et nous allons créer une nouvelle Application Web ASP.Net Core. 3 4

Nous choisissons une Web Application non MVC. 5

Voici le contenu du Solution Explorer : 6

Le projet cible .NET Core 3 preview 4. On remarque le folder Pages et les pages Razor… C'est très minimaliste.

Une page Razor

Une page Razor c'est :

- Un fichier cshtml en HTML avec des balises Razor @xxxx ;
- Un fichier cshtml.cs avec du code en C#.

Voici le contenu de la page index.cshtml :

```
@page
@model IndexModel
@{
  ViewData["Title"] = "Home page";
}

<div class="text-center">
  <h1 class="display-4">Welcome</h1>
  <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps
with ASP.NET Core</a>.</p>
</div>
```

Voici le contenu du fichier index.cshtml.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace MyApp2.Pages
{
  public class IndexModel : PageModel
  {
    public void OnGet()
    {

    }
  }
}
```

Conclusion

Le framework ASP.NET Core est riche et performant. C'est une composante essentielle à connaître pour tout développeur .NET. C'est le point d'entrée vers les architectures micro-services. Nous le verrons dans le prochain dossier… •

Denis Duplan
sociologue et développeur à ses heures.
Blog : http://www.stashofcode.fr

niveau 200

Mandelbrot en WebAssembly

Partie 2

Ainsi donc, grâce à WebAssembly, il serait possible de faire tourner des programmes fulgurants dans une page Web. Mazette ! Pourquoi ne pas tester la proposition en réalisant un petit programme qui requiert traditionnellement une certaine puissance de calcul, comme une représentation de l'ensemble de Mandelbrot ?

Générer les pixels avec le module

Si vous ne connaissez pas encore la manière dont la représentation d'un ensemble de Mandelbrot peut être générée, je vous recommande de visionner l'explication citée dans les références de cet article sur l'excellente chaîne YouTube Numberphile – ça change du prof de maths barbu : Vous y apprendrez qu'il s'agit de calculer au plus n fois `z=z2+c` en s'interrompant prématurément dès que $|z| \geq 2$. Le nombre d'itérations qu'il a fallu réaliser sert alors à déterminer l'intensité de la couleur du pixel dont les coordonnées `(x,y)` sont telles que `z=x+i*y`. Généralement, on pose que c correspond au point `(0,0)`. La fonction `mandelbrot` exportée par le module prend plusieurs arguments :

$width, $height	Dimensions de l'image dont il faut calculer la couleur des pixels
$minX, $maxX	Intervalle des abscisses correspondant à la largeur de l'image
$minY, $maxY	Intervalle des ordonnées correspondant à la hauteur de l'image
$maxN	Nombre maximum d'itérations

Ce qui se traduit par le module suivant :

```
(module
    (memory (import "imports" "memory") 1)
    (func (export "mandelbrot")
        (param $width i32)
        (param $height i32)
        (param $minX f64)
        (param $maxX f64)
        (param $minY f64)
        (param $maxY f64)
        (param $maxN i32)
        ;; ...
    )
)
```

Maintenant que nous savons comment réaliser des boucles, le code Wasm de la fonction est assez simple à écrire. Les commentaires sont en anglais, car wat2wasm achoppe sur les caractères accentués :

```
(func $mandelbrot (export "mandelbrot")
    (param $width i32)
    (param $height i32)
    (param $minX f64)
    (param $maxX f64)
    (param $minY f64)
    (param $maxY f64)
    (param $maxN i32)
    (local $i i32)
    (local $j i32)
    (local $dx f64)
    (local $dy f64)
    (local $x f64)
    (local $y f64)
    (local $a f64)
    (local $b f64)
    (local $c f64)
    (local $n i32)
    (local $index i32)
    (set_local $dx (f64.div (f64.sub (get_local $maxX) (get_local $minX)) (f64.convert_u/i32
(get_local $width))))
    (set_local $dy (f64.div (f64.sub (get_local $maxY) (get_local $minY)) (f64.convert_u/i32
(get_local $height))))
    (set_local $j (get_local $height))
    (set_local $y (get_local $minY))
    (set_local $index (i32.const 0))
    (loop
        (set_local $i (get_local $width))
        (set_local $x (get_local $minX))
        (loop
            (set_local $a (f64.const 0.0))
            (set_local $b (f64.const 0.0))
            (set_local $n (get_local $maxN))
            (block
                (loop
                    ;; $c = $a (just push it)
                    get_local $a
                    ;; Set $a = $a * $a - $b * $b + $x
                    (f64.mul (get_local $a) (get_local $a))
                    (f64.sub (f64.mul (get_local $b) (get_local $b)))
                    (set_local $a (f64.add (get_local $x)))
                    ;; Set $b = 2 * $c * $b + y
                    (f64.mul (f64.const 2.0))      ;; This pops $a
                    (f64.mul (get_local $b))
                    (set_local $b (f64.add (get_local $y)))
                    ;; Set $n -= 1
                    (set_local $n (i32.sub (get_local $n) (i32.const 1)))
                    ;; Break if $a * $a + $b * $b >= 4.0
                    (f64.add (f64.mul (get_local $a) (get_local $a)) (f64.mul (get_local $b)
(get_local $b)))
                    (br_if 1 (f64.ge (f64.const 4.0)))
                    ;; Break if $n == 0, else loop
                    (br_if 1 (i32.eqz (get_local $n)))
```

```
                br 0
            )
        )
        ;; Store (0xFF000000 | (($n * 255 / ($maxN - 1)) & 0xFF)) at index $index
        get_local $index
        (i32.trunc_u/f64 (f64.div (f64.convert_u/i32 (i32.mul (get_local $n) (i32.const
255))) (f64.convert_u/i32 (i32.sub (get_local $maxN) (i32.const 1)))))
        (i32.or (i32.const 0xFF000000))
        i32.store
        ;; Same thing with S-expressions :
        ;;(i32.store
        ;;   (get_local $index)
        ;;   (i32.or
        ;;      (i32.const 0xFF000000)
        ;;      (i32.trunc_u/f64 (f64.div (f64.convert_u/i32 (i32.mul (get_local $n)
(i32.const 255))) (f64.convert_u/i32 (i32.sub (get_local $maxN) (i32.const 1)))))
        ;;   )
        ;;)
        ;; Set $index += 4
        (set_local $index (i32.add (get_local $index) (i32.const 4)))

        ;; Set $x += $dx
        (set_local $x (f64.add (get_local $x) (get_local $dx)))
        ;; Loop if $i -- != 0, else exit
        (set_local $i (i32.sub (get_local $i) (i32.const 1)))
        (br_if 0 (i32.ne (get_local $i) (i32.const 0)))
    )
    ;; Set $y += $dy
    (set_local $y (f64.add (get_local $y) (get_local $dy)))
    ;; Loop if $j -- != 0, else exit
    (set_local $j (i32.sub (get_local $j) (i32.const 1)))
    (br_if 0 (i32.ne (get_local $j) (i32.const 0)))
  )
)
```

La seule chose qu'il convient de commenter, c'est le stockage des pixels. Ce dernier mobilise l'instruction `i32.store`, qui prend deux opérandes : un indice dans la mémoire, et la valeur à stocker à cet indice dans la mémoire. La mémoire en question est implicitement la seule autorisée dans un module – en l'état actuel de WebAssembly, un module est pour l'heure limité : une mémoire au plus, une table au plus, etc. Deux remarques :

- ces opérandes doivent être empilés dans cet ordre : d'abord l'indice, puis la valeur ;
- l'indice est exprimé en octets, ce qui signifie que s'il s'agit de stocker un `i32` dans un tableau d'entiers codés sur 32 bits à la position `$n`, l'indice doit être `4 * $n`.

Partant, le code est le suivant :

```
get_local $index
(i32.trunc_u/f64 (f64.div (f64.convert_u/i32 (i32.mul (get_local $n) (i32.const 255)))
(f64.convert_u/i32 (get_local $maxN))))
(i32.or (i32.const 0xFF000000))
i32.store
```

N'est-il pas possible de simplifier en utilisant plus de S-expressions ? De fait, il est parfaitement possible d'écrire :

```
(i32.store
    (get_local $index)
    (i32.or
        (i32.const 0xFF000000)
        (i32.trunc_u/f64 (f64.div (f64.convert_u/i32 (i32.mul (get_local $n) (i32.const 255)))
(f64.convert_u/i32 (i32.sub (get_local $maxN) (i32.const 1)))))
    )
)
```

Si la forme précédente a été conservée, c'est pour illustrer le fonctionnement de `i32.store` étape par étape.

Afficher les pixels avec le programme

Avant d'en venir au code du programme, en voici un qui est minimaliste. Il vous permet de tester ce que le module génère dans wat2wasm, dans le cas d'une image de 11×11. Il vous suffit de le copier-coller dans la fenêtre en bas à gauche de l'outil :

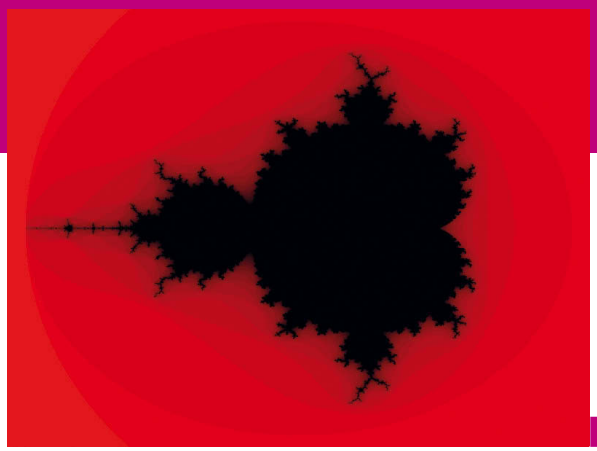
```
var width = 11, height = 11;
var importObject = {
    imports: {
        memory: new WebAssembly.Memory ({ initial: Math.ceil (width * height * 4 / 65536) })
    }
};
const wasmInstance = new WebAssembly.Instance (wasmModule, importObject);
wasmInstance.exports.mandelbrot (width, height, -2.1, 1.1, -1.4, 1.4, 20);
var i, j, pixels;
pixels = new Uint32Array (importObject.imports.memory.buffer);
for (j = 0; j != height; j ++) {
    for (i = 0; i != width ; i++)
        console.log (`(${i}, ${j}) = ${pixels[i + j * width].toString (16)}`);
}
```

Le programme que nous envisageons est plus ambitieux. Il s'agit d'afficher l'image dans une page Web à l'aide d'un canvas.

Pour ceux qui l'ignorent encore, un canvas est un objet de l'API Canvas qui permet de manipuler une image dans une page Web. Cette API permet de dessiner des primitives de tout genre, de manière très haut-niveau par un simple appel de méthode – comme par exemple `rect()` – ou un peu plus bas-niveau, en décrivant des chemins. Ce qui nous intéresse est encore plus bas-niveau : c'est la possibilité de dessiner des pixels.

D'emblée, il faut préciser que dessiner au pixel comme nous allons le faire n'est pas généralement la meilleure solution. En effet, il est bien plus rapide d'appeler la méthode `rect()` pour dessiner un rectangle de 1×1 que d'écrire la séquence octets décrivant un pixel dans un `Uint32Array` représentant la surface. C'est a priori assez étonnant, mais cela s'explique par la manière dont la technologie fonctionne.

Pour autant, c'est ce que nous allons faire. Rappelons l'idée de départ : balancer un paquet de données à un module et s'en faire balancer un en retour, après un traitement qui dépote. Nous ferons l'hypothèse que les performances seraient outrageusement dégradées si jamais le module devait appeler une fonction du programme à chaque pixel, et donc que le temps perdu à recopier l'ensemble des pixels générés par le module dans un tableau représentant la surface du canvas sera amplement compensé par la rapidité avec laquelle le module calculera les couleurs des pixels en question. On verra si l'hypothèse se vérifie…



3 *Le résultat à l'écran.*

Pour commencer, il faut créer le canvas. S'il est possible de simplement ajouter une balise `canvas` dans le code HTML, procédons plutôt dynamiquement pour décortiquer le processus :

```
var width = 801, height = 601, canvas;
canvas = document.createElement ("canvas");
canvas.setAttribute ("width", width);
canvas.setAttribute ("height", height);
document.body.appendChild (canvas);
```

Ce code crée un objet `HTMLCanvasElement` correspondant à un élément `<canvas>` et recupère un objet `CanvasRenderingContext2D` dont les propriétés et les méthodes permettent de dessiner en 2D. L'élément est alors ajouté au document.

Dès lors, il est possible d'appeler la fonction `mandelbrot ()` exportée par le module, pour qu'elle génère autant de pixels que l'image à produire en comporte. Les paramètres concernant l'échelle de l'espace complexe correspondant et le nombre limite d'itérations ont été choisis pour des considérations esthétiques, après quelques essais :

```
response.instance.exports.mandelbrot (width, height, -2.1, 1.1, -1.4, 1.4, 20);
```

Enfin, c'est le grand moment. Au retour de l'appel, l'objet `WebAssembly.Memory` importé par le module contient les pixels. Il suffit de les recopier intégralement dans la surface de l'image. Pour cette opération, les buffers sont manipulés sous la forme optimale de tableaux d'entiers codés sur 32 bits :

```
inPixels = new Uint32Array (importObject.imports.memory.buffer);
context2d = canvas.getContext ("2d");
imageData = context2d.getImageData (0, 0, width, height);
outPixels = new Uint32Array (imageData.data.buffer);
outPixels.set (inPixels.slice (0, width * height - 1));
context2d.putImageData (imageData, 0, 0);
```

Dans ce code, la méthode `Uint32Array.slice ()` est utilisée car `Uint32Array.set ()` recopie l'intégralité du tableau qui lui est fourni. Or dans notre cas, la taille de ce tableau est un multiple d'une page (64 Ko), qui a toutes les chances d'excéder le nombre de pixels `width * height`.

Attention ! Le tableau des pixels est donc un tableau d'entiers codés sur 32 bits, ce qui signifie que chaque pixel est représenté sous la forme de quatre octets. Or on n'écrit pas les octets composant un mot de 32 bits dans le même ordre en mémoire selon que la machine est en little ou big endian. Soit un pixel dont la couleur lue en une fois en mémoire sous la forme d'un mot de 32 bits doit être `0x12345678`. Un PC est généralement little endian. Cela signifie que les octets composant ce mot doivent être stockés en mémoire dans cet ordre d'adresse croissante : `0x78` (transparence A), `0x56` (bleu B), `0x34` (vert G) et `0x12` (rouge R), c'est-à-dire ABGR et non RGBA. Ce serait l'inverse sur une machine big endian, typiquement un Mac doté d'un processeur PowerPC.

Tout cela pour dire que si les codes du programme et du module fonctionnent sur n'importe quelle machine, choix a été fait d'adopter la logique d'une machine little endian dans le module lors de la composition de la couleur stockée en mémoire par `i32.store`. A cette occasion, il a été tenu compte du fait que, comme le précise la spécification de WebAssembly, cette instruction stocke en little endian.

L'image générée ne sera pas correcte sur une machine big endian – le rouge et la transparence seront intervertis. Qui souhaite résoudre ce problème devra détecter si la machine est big ou little endian en écrivant deux octets à des adresses consécutives, puis en lisant d'un coup ces octets en lisant un mot de 16 bits : si ces octets figurent en ordre inversé dans le mot de 16 bits, la machine est little endian :

```
var word = new Uint8Array (2);
word[0] = 0x01;
word[1] = 0x23;
word = new Uint16Array (word.buffer);
if (word[0] == 0x0123)
    // Machine big endian
else
    // Machine little endian
```

Tout le code qui vient d'être présenté est à exécuter au terme de la résolution de promesses qui s'enchaînent, du chargement du fichier du binaire du module à l'instanciation de ce dernier. Ces opérations ont été présentées au début de cet article.

Rendez-vous sur le site de l'auteur pour tester, et récupérer par la même occasion tout le code JavaScript qui vient d'être présenté. Si vous êtes sur une machine little endian, vous devriez voir la représentation suivante de l'ensemble de Mandelbrot s'afficher (Figure 3).

C'est bien beau, mais ça dépote ?

Pour l'heure, ce qu'il est possible de dire, c'est que ce n'est pas bien gros : Le fichier du binaire du module fait 420 octets, ce qui va certainement saturer notre bande passante. Toutefois, un module JavaScript équivalent minifié à l'aide de l'outil d'Andrew Chilton produit un fichier de 281 octets :

```
export default function (r,n,f,o,t,a,u){var e,i,c,d,h,v,w,y,A,M,U,b;for(e=new Uint32Array
(r*n),w=(o-f)/r,y=(a-t)/n,v=t,i=0,d=0;d!=n;d++){for(h=f,c=0;c!=r;c++){for(A=0,
M=0,b=u;U=A,b--,!((A=A*A-M*M+h)*A+(M=2*U*M+v)*M>=4)&&b;);e[i++] =4278190080
+Math.trunc(255*b/u),h+=w}v+=y}return e}
```

Bon, ce n'est sans doute pas d'un si petit module qu'il est possible de conclure sur l'intérêt d'économiser de l'espace. Par contre, il devrait être possible d'en tirer sur son intérêt en termes de performance. Toutefois, la modestie du module se révèle encore une fois contraignante, car elle interdit de tirer des conclusions sur l'intérêt d'améliorer les performances durant les phases qui séparent la fin du chargement de l'exécution – ce qui est avancé comme un gros avantage sur JavaScript. Enfin, tâchons tout de même voir ce qu'il en est de la vitesse de calcul…

Pour le savoir, une version JavaScript de la fonction `mandelbrot ()` du module a été programmée. Afin qu'elle soit placée approximativement sur le même pied d'égalité que son homologue, elle accède à un objet `Uint32Array` créé une fois pour toutes, tout comme le modu-

le peut accéder à un objet `WebAssembly.Memory` créé au chargement de la page Web. Tout bêtement, il s'agit de commander le calcul des images d'une animation qui a vocation à être jouée le plus rapidement possible, tout en restant calée sur la fréquence de rafraîchissement de l'écran. Le paramètre choisi pour faire varier la charge de calcul est le nombre maximum d'itérations, qui correspond à `$maxN` dans le code de notre module. Vous pouvez tester le résultat sur la reprise de l'article sur le site de l'auteur. Même en chargeant la mule sur le petit portable utilisé pour l'occasion, aucune différence notable n'apparaît. La fonction JavaScript semble tout aussi performante que le module WebAssembly. « *Si j'aurais su, j'aurais pas venu !* ». Tout ça pour produire un module plus gros et pas plus rapide que le programme JavaScript équivalent ? WebAssembly, c'est donc la déception sur tous les plans. Cependant, il convient encore de rappeler que le problème choisi dans cet article n'est pas à la hauteur de la solution dont il vient de servir à tester l'efficacité. Qui recherche « webassembly performance » dans Google tombe sur des benchmarks plus sophistiqués qui peuvent laisser entendre que dans certains cas, WebAssembly peut présenter un intérêt.

Conclusion

Il faut être honnête : l'écriture du module n'a pas été des plus simples. Quand bien même wat2wasm assemble en live le code Wasm et signale à cette occasion les erreurs de syntaxe et d'utilisation de la pile, il a fallu introduire quelques fonctions de débogage ad hoc pour bien détecter ce qui pouvait aller de travers en dépit d'une syntaxe correcte. A ma décharge, je découvrais Wasm dans un certain niveau de détail à cette occasion, et je devais faire avec une documentation qui n'est pas des plus claires. En particulier, j'espère que mes explications vous éviterons de perdre du temps sur les instructions de contrôle… en attendant que je présente plus en détail comment elles fonctionnent dans un prochain article.

Est-il intéressant de s'intéresser à WebAssembly ? Comme toujours en matière d'innovation technologique : oui, mais prudemment. A ce jour, comme le précise l'excellent site *Can I use?*, cette technologie a été adoptée par les principaux navigateurs. Toutefois, force est de constater que les réalisations sont rares : d'une recherche portant sur « webassembly demo » sur Google, il ne ressort pas grand-chose. Par ailleurs, la plus-value en termes de performance reste à démontrer, mais il est vrai qu'il ne faut pas se limiter à cette question, les objectifs de WebAssembly rappelés ici étant bien plus généraux.

WebAssembly n'en est qu'à ses débuts ; ceux qui sont en charge de l'implémentation de WebAssembly sur les navigateurs sont encore à la tâche. Ainsi qu'en témoigne une liste figurant sur le site GitHub des développeurs officiels de WebAssembly, les fonctionnalités dont il est question d'enrichir la technologie sont nombreuses, allant de la possibilité de charger un module comme n'importe quel module JavaScript, à celle pour une fonction de retourner plusieurs valeurs, en passant par celle de manipuler directement le DOM – et plus généralement d'accéder aux diverses API – dans le contexte d'une page Web.

Bref, regarder pour savoir de quoi il en retourne, et s'amuser un peu. Pour le reste, attendre et voir. •

Références

Mandelbrot expliqué par une prof pas barbue
https://www.youtube.com/channel/UCoxcjq-8xIDTYp3uz647V5A

Documentation officielle de WebAssembly
https://webassembly.org/docs/modules/#imports

WebAssembly Reference Manual
https://github.com/sunfishcode/wasm-reference-manual/blob/master/WebAssembly.md#import-section

WebAssembly Binary Toolkit
https://github.com/webassembly/wabt

Présentation de Node.js par Mosh Hamedani
https://www.youtube.com/watch?v=TlB_eWDSMt4

Repère des développeurs officiels de WebAssembly :
https://github.com/WebAssembly

Réinventer le CPU dans le contexte du navigateur : la fin de JavaScript ?

Comme j'ai déjà eu l'occasion de le faire remarquer ici, le Web a constitué pour les développeurs une régression incroyable. Le fait est qu'il a fallu tout simplement réinventer la roue pour la faire tourner dans une architecture Representational State Transfer (REST). En 2018, nous en sommes encore à essayer de revenir au niveau de confort qu'offrait la programmation avant le Web, en réinventant des concepts aussi élémentaires que la compilation et la liaison. La seule différence, c'est que nous nous « aidons » en cela des toolchains horriblement complexes à base de npm, de webpack, etc, qu'il n'est possible d'utiliser simplement que dans des IDE extrêmement lourds.Bien évidemment, il faut voir les bons côtés de la chose. Les apports de l'architecture REST sont considérables. La connectivité généralisée qu'elle a permis d'établir avec le Web sert à la fois de support et de moyen pour des innovations sidérantes. Citer npm comme à l'instant, ce sera pour critiquer le fait que c'est un outil en ligne de commandes qui entraîne la création d'une myriade de fichiers à la moindre installation d'un package – un vrai gaspillage de bande passante et d'espace disque -, mais certainement pas pour critiquer le fait que l'outil donne accès à d'innombrables packages produits par la Terre entière, une sorte de bibliothèque d'Alexandrie.Tout de même, dans le domaine du développement, nous sommes loin, très loin, d'avoir atteint le niveau de maturité qui permettrait de tirer pleinement parti de REST.Pour revenir à WebAssembly, si l'on prend du recul sur l'évolution de cette technologie du Web que constitue le navigateur, on constate que la tendance a depuis toujours été d'en faire un OS dans l'OS. Avec WebAssembly, on passe à un autre niveau qui consiste à faire du navigateur une machine dans la machine. Ce projet est ancien : c'est celui de la machine virtuelle. Toutefois, il n'était pas jusqu'à ce jour question de pouvoir écrire du bytecode à la main, ou d'en générer à partir de n'importe quel langage, car il n'existait pas de langage intermédiaire pour cela. C'est l'innovation qu'introduit WebAssembly. On en est donc là dans le retour en arrière : on se met à programmer en assembleur dans le contexte d'un navigateur comme on programmait un CPU de base avant le Web. Inévitablement, les évolutions à venir de WebAssembly tiendront plus du recyclage que de l'innovation.Il n'en reste pas moins que WebAssembly a tout le potentiel pour bouleverser l'écosystème du développement front-end pour le Web, tout particulièrement en réduisant drastiquement ce qu'il sera encore nécessaire de programmer en JavaScript. Qu'on y songe : à compter du moment où un module WebAssembly pourra aussi bien dialoguer avec l'environnement d'exécution du navigateur comme c'est un des objectifs rappelé ici, et qu'il sera possible de compiler un source écrit dans un langage quelconque pour produire un tel module, JavaScript ne sera certainement plus aussi incontournable qu'il ne l'est aujourd'hui. Mais pour que tout bascule, il faudra une démonstration incontestable du gain en productivité que représente la généralisation du recours à WebAssembly.

Vincent Rivière
Développeur à l'Université Paris 1 Panthéon-Sorbonne
http://vincent.riviere.free.fr/
https://www.youtube.com/c/Vretrocomputing

niveau 200

Programmation système sur Atari ST

Partie 3

La dernière fois, je vous ai présenté les différentes couches du système d'exploitation de l'Atari ST, le TOS. Après avoir détaillé la structure des exécutables, nous allons voir comment faire appel à ces couches depuis vos programmes assembleur.

Appels système

Une fois qu'un programme est chargé en mémoire, comment fait-il pour faire appel au système d'exploitation ? Réponse : avec les instructions « **trap** » du 68000. En effet, le TOS peut être qualifié de système monolithique. En temps normal, les programmes s'exécutent en mode utilisateur. Mais les instructions trap déclenchent l'exécution de routines privilégiées en mode superviseur. Rappelez-vous, j'ai commencé par vous présenter les différentes couches du TOS. Eh bien, chacune d'entre elles s'appelle avec un trap spécifique. Mais attention, il faut respecter les **conventions d'appel** propres à chaque couche.

GEMDOS

C'est l'instruction « **trap #1** » qui permet d'appeler le GEMDOS. Avant l'appel, les paramètres doivent être passés sur la pile, en ordre inverse. Et après l'appel, ils doivent être retirés. Les habitués auront reconnu la convention d'appel standard du langage C. Le premier paramètre (donc chronologiquement, le dernier empilé) est le **numéro de fonction** du GEMDOS. Par exemple, la fonction Malloc() permet d'allouer de la mémoire supplémentaire. Elle porte le numéro $48, ce qui au passage représente le nombre hexadécimal 48, soit 72 en décimal. Et elle prend en paramètre la quantité de RAM souhaitée, en octets.

```
move.l   #300,-(sp)   ;Allouer 300 octets
move.w   #$48,-(sp)   ;Numéro de fonction de Malloc()
trap     #1           ;Appeler le GEMDOS
addq.l   #6,sp        ;Corriger la pile
```

Le numéro de fonction est toujours spécifié sous forme de *mot* (d'où l'instruction *move.w*), et précède toujours l'instruction *trap*. Dans le cas de Malloc(), la documentation indique que la fonction prend un seul paramètre de type *long mot*. Pour information, le fait d'empiler les paramètres à l'envers permet à la fonction appelée de les retrouver à l'endroit sur la pile. Comptons maintenant la **taille en octets** des paramètres empilés. On a un long mot (4 octets) qui indique la taille du bloc à allouer, et un mot (2 octets) qui indique le numéro de fonction à appeler. Ce qui fait un total de 6 octets passés sur la pile. C'est pour cela que l'on doit corriger la pile de précisément 6 octets après le trap. Attention, toutes les fonctions GEMDOS ne prennent pas le même nombre de paramètres. Donc pour chaque numéro de fonction, il faudra refaire cette petite gymnastique afin de déterminer de combien d'octets la pile doit être corrigée. Pas de panique, les bonnes documentations donnent toujours un exemple d'appel pour chaque fonction. Mais il est toujours bon de s'exercer à refaire ces calculs de pile par soi-même pour bien comprendre les tenants et aboutissants.

Après l'appel, on retrouve le résultat dans le registre D0. Une valeur sous forme de *long mot négatif* indique une erreur. Voir la documentation pour tous les codes possibles.

Mais attention, il y a un piège ! L'appel au trap #1 préserve certains registres, mais pas tous. Les registres non préservés sont appelés « **scratch registers** » ou « **clobbered registers** ». Si ces registres contiennent des données utiles, il faut les sauvegarder quelque part avant l'appel. Les registres préservés lors d'un trap #1 sont : D3-D7/A3-A7. Et les registres non préservés sont : D0-D2/A0-A2. Méfiez-vous, ces derniers ne sont pas systématiquement écrasés par le trap #1, mais ça peut arriver. Donc ne faites jamais confiance au contenu de ces registres après un trap #1 (à part bien sûr D0 qui contient le résultat du trap).

BIOS et XBIOS

Bonne nouvelle, le BIOS et le XBIOS s'appellent de la même manière que le GEMDOS. Seul le numéro de trap change. Le BIOS utilise « **trap #13** », alors que le XBIOS utilise « **trap #14** ».

Cas du GEM

Attention, ça se corse. Le GEM utilise une convention d'appel différente. Les couches VDI et AES s'appellent toutes les deux avec « **trap #2** ». Mais les paramètres ne sont pas passés par la pile, au lieu de cela ils sont passés par des **tableaux** référencés par des **registres**. Voyons cela en détail.

VDI

D'abord, il faut définir des tableaux de mots pour les paramètres. Leurs noms sont normalisés.

```
contrl:   ds.w  12    ;Paramètres de contrôle
intin:    ds.w  128   ;Entiers en entrée
ptsin:    ds.w  128   ;Points x,y en entrée
intout:   ds.w  128   ;Entiers en sortie
ptsout:   ds.w  128   ;Points x,y en sortie
```

Ensuite, il faut définir un « **bloc de paramètres VDI** ». Il s'agit d'un bloc de pointeurs vers les tableaux définis précédemment.

```
vdipb:    dc.l    contrl,intin,ptsin,intout,ptsout
```

Pour appeler la VDI, il faut commencer par passer les paramètres dans les tableaux. Le numéro de fonction VDI se met dans contrl[0], et les éventuels autres paramètres se passent via intin[] et ptsin[]. Le nombre de paramètres réellement passés doit en plus être renseigné dans contrl[]. Là aussi, il faut scrupuleusement respecter la documentation de chaque fonction.
Ensuite, il faut utiliser « trap #2 » de cette manière :

```
move.l    #vdipb,d1    ;Bloc de paramètres VDI
move.w    #115,d0      ;Identifiant de la couche VDI
trap      #2           ;Appeler le GEM
```

Après l'appel, on retrouve le résultat dans les tableaux intout[] et ptsout[]. Les registres non préservés sont les mêmes qu'avec le GEMDOS.
Tous les appels à la VDI font référence à une « station de travail virtuelle » qui représente écran, clavier et souris. Mais j'en reste là pour l'instant.

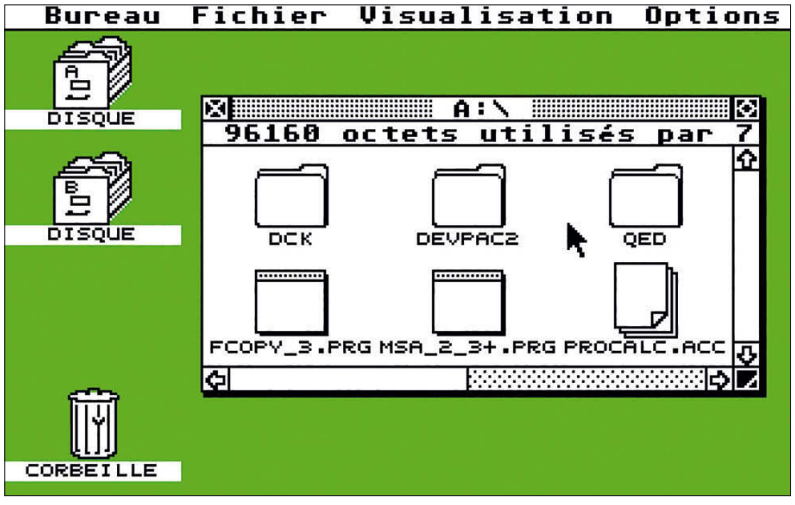
AES

C'est presque la même chose que pour la VDI, mis à part que les tableaux et l'identifiant de la couche AES sont différents.

```
control:    ds.w  10    ;Paramètres de contrôle
global:     ds.w  16    ;Informations diverses
int_in:     ds.w  128   ;Entiers en entrée
int_out:    ds.w  128   ;Entiers en sortie
addr_in:    ds.w  128   ;Adresses en entrée
addr_out:   ds.w  128   ;Adresses en sortie
aespb:      dc.l  control,global,int_in,int_out,addr_in,addr_out

move.l    #aespb,d1    ;Bloc de paramètres AES
move.w    #200,d0      ;Identifiant de la couche AES
trap      #2           ;Appeler le GEM
```

Et à nouveau, on retrouve le résultat dans les tableaux int_out[] et addr_out[].

Le bureau GEM



A noter que contrairement aux autres couches du TOS, il est interdit d'appeler l'AES en mode superviseur. Seul le mode utilisateur est autorisé.

Line-A

Cette API est assez spéciale, car elle repose sur une fonctionnalité du 68000 portant le même nom : *Line-A*, ou *A-line*. Chaque instruction (ou *opcode*) du 68000 est codée sur un mot de 16 bits, suivi éventuellement de mots supplémentaires pour les paramètres. Les opcodes commençant par $A déclenchent systématiquement une exception, comme les trap. Les 12 autres bits de l'opcode peuvent servir de paramètre. Ainsi, Atari a défini 16 fonctions dans la Line-A, de $A000 à $A00F. On les utilise de cette manière :

```
dc.w      $a009    ;Appeler la fonction Line-A numéro 9
```

Attention, comme pour les autres couches, les registres D0-D2/A0-A2 sont susceptibles d'être modifiés.
La fonction $A000 « Initialization » est essentielle : elle retourne dans D0 et A0 un pointeur vers une zone que l'on appelle « variables Line-A ». En offsets positifs, on trouve des variables qui servent de paramètres lors de l'appel aux autres fonctions. Et en offsets négatifs, on trouve des variables qui fournissent des informations sur le pilote d'écran, tels que la position de la souris, la résolution de l'écran, etc.
Je me contenterai de citer les fonctions $A00A « Hide mouse » et $A009 « Show mouse » qui sont très utilisées. Attention toutefois, Atari a clairement déclaré que l'API Line-A était obsolète. Mais en pratique, elle fonctionne bien pour les modes vidéo du ST. Par ailleurs, cette API pose problème avec les processeurs ColdFire, mais c'est une autre affaire.

Terminer un programme

Nous savons maintenant comment appeler les différentes couches du TOS. Mais comment termine-t-on un programme ? Avec des fonctions du GEMDOS. Le plus simple est d'utiliser la fonction **Pterm0()**. Elle a 2 particularités. Comme son numéro est 0, on peut le spécifier implicitement avec l'instruction « clr.w » au lieu de « move.w », cela permet d'économiser 1 mot. Et comme cette fonction ne rend jamais la main (puisqu'elle termine le programme en cours), ce n'est pas la peine d'essayer de corriger la pile après l'appel. Donc la manière la plus simple de l'appeler est :

```
clr.w   -(sp)   ;Pterm0()
trap    #1      ;GEMDOS
```

Comme sur la plupart des systèmes d'exploitation, les processus renvoient un code de retour au programme appelant. Par définition, Pterm0() renvoie systématiquement le code 0 (à ne pas confondre avec l'autre 0 qui est le numéro de sa fonction). Si on souhaite renvoyer un code de retour différent, il faut utiliser la fonction Pterm() qui porte le numéro $4C et qui fonctionne de manière similaire, avec évidemment un paramètre supplémentaire.

```
; Hello, World! Programme complet.
    move.l    #msg,-(sp)    ;Chaîne
    move.w    #9,-(sp)      ;Cconws()
    trap      #1            ;GEMDOS
    addq.l    #6,sp

    clr.w     -(sp)         ;Pterm0()
    trap      #1            ;GEMDOS

msg:  dc.b    "Hello, World!",13,10,0
```

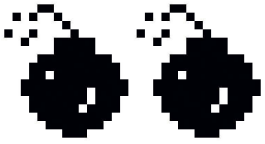
Mode superviseur

Je vous ai dit que par défaut, les programmes s'exécutent en mode utilisateur. Cela permet un certain niveau de protection, car dans ce mode, il est impossible d'accéder par erreur à certaines adresses, telles que les variables système et les registres hardware. Tout accès non autorisé se solde par une **Bus Error**, autrement dit : 2 bombes. Notez que cette protection est une spécificité de l'Atari ST : son rival de toujours, l'Amiga, ne dispose de rien de tel, donc sur cette machine n'importe quel programme buggé risque d'endommager le système, même en mode utilisateur.

Donc sur Atari ST, si on veut volontairement accéder aux variables système et faire des accès directs au hardware, il faut passer en mode superviseur. Pour cela, il y a 2 possibilités :

- La fonction XBIOS $26 « **Supexec()** » pour exécuter une seule fonction en mode superviseur ;
- La fonction GEMDOS $20 « **Super()** » pour passer le programme principal en mode superviseur, et éventuellement revenir plus tard en mode utilisateur.

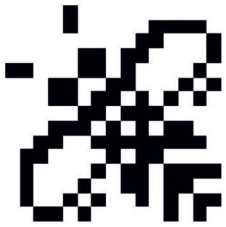
Attention, en mode superviseur, les erreurs ne pardonnent pas : en cas de bug, on risque de corrompre les variables internes du TOS ou de bloquer la machine. Pas de panique, pour que tout rentre dans l'ordre, il suffit de faire un reset.



Variables système

Sur toutes les machines à base de 68000, les **vecteurs d'exception** sont situés à partir de l'adresse 0. Il s'agit de variables qui pointent vers les routines de traitement des trap, des interruptions, etc. Sur Atari ST, il y a en plus d'autres vecteurs à partir de l'adresse $100 qui sont utilisés par le coprocesseur MFP 68901. Et par ailleurs, le TOS gère des **variables système** dont les adresses sont officiellement documentées par Atari et situées à partir de l'adresse $400. Et ça tombe bien, car le hardware de l'Atari ST interdit tout accès aux adresses inférieures à $800 en mode utilisateur. Toutes les données situées dans cette zone sont protégées contre d'éventuelles erreurs des programmes. Donc pour y accéder, il faut impérativement passer en mode superviseur.

Cookies



Depuis le TOS 1.6 du STe, le système possède une « **cookie jar** » (littéralement : un pot à biscuits). Il s'agit en fait d'un tableau de variables nommées qui fournissent des informations sur le système. Chaque cookie possède un nom sur 4 caractères, et une valeur sur 4 octets. Par exemple, le cookie « _CPU » indique le modèle exact du processeur. L'adresse de la cookie jar est indiquée dans la variable système située à l'adresse **$5A0**. Donc vous l'aurez deviné, il faut être en mode superviseur pour accéder aux cookies.

Accès direct au hardware

Sur Atari ST, au plus bas niveau, on accède aux périphériques via des adresses situées dans la plage $FF8000-$FFFFFF. Et comme je vous l'ai dit, cette zone est uniquement accessible en mode superviseur. En général, il est conseillé d'accéder à ces périphériques via les fonctions du XBIOS, ou mieux, indirectement via les couches supérieures. Mais si on le souhaite, on peut accéder directement au hardware, à condition d'être en mode superviseur. C'est bien entendu cette méthode qui est abondamment utilisée par les jeux et démos pour obtenir des performances maximales, en s'affranchissant du système d'exploitation, mais au détriment de la compatibilité.

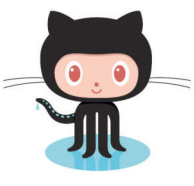
Conclusion

Nous venons de faire un bref tour d'horizon des différents composants du TOS. Même si c'était juste un aperçu de ce système, vous avez maintenant une idée assez précise de son fonctionnement et de la manière de l'utiliser depuis vos programmes. Pour aller plus loin, vous devrez impérativement consulter la documentation, mais grâce aux concepts clés que je vous ai expliqués ici, vous vous sentirez immédiatement en terrain connu. Et bien sûr, vous n'êtes pas obligés de programmer en assembleur ! Tous les langages, tels que BASIC ou C, fournissent des solutions pour appeler facilement les fonctions du système.

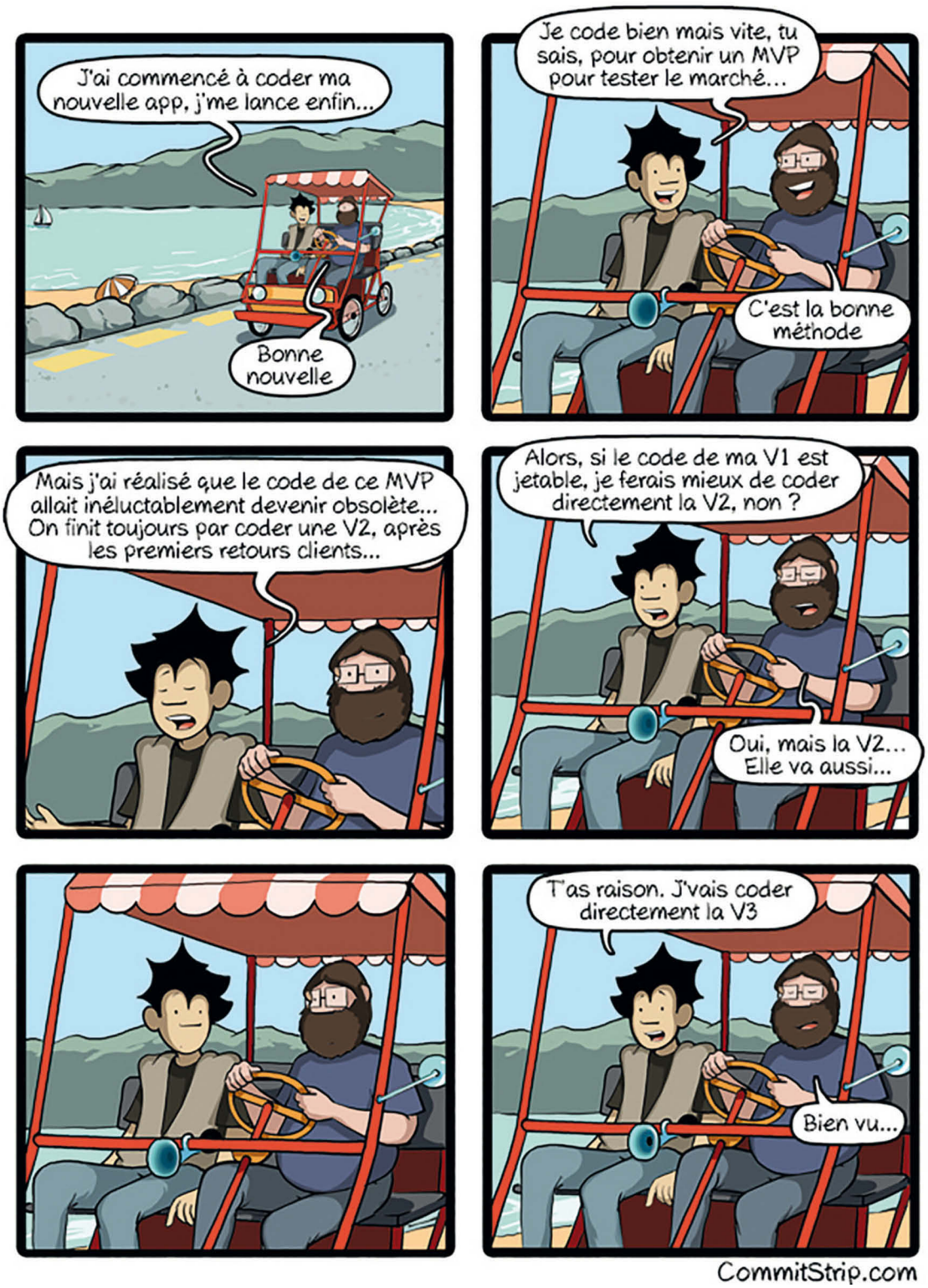
Sachez que toutes les couches du TOS peuvent être étendues, ou même remplacées par des implémentations complètement différentes. Par exemple, FreeMiNT remplace la couche GEMDOS par sa propre implémentation qui supporte le multitâche préemptif. NVDI et fVDI remplacent la couche VDI pour améliorer les performances graphiques et supporter des cartes graphiques alternatives. XaAES et MyAES remplacent la couche AES pour proposer une interface graphique améliorée. Même le XBIOS peut être remplacé pour supporter du matériel alternatif. Mais dans tous les cas, si vous écrivez des programmes propres qui respectent l'API du système d'exploitation, alors vous avez toutes les chances pour qu'ils fonctionnent correctement sur l'ensemble des ordinateurs Atari et de leurs clones.

•

Retrouvez tous les codes sources de Programmez!
sur https://github.com/francoistonic/ et sur programmez.com



Codebase de startup



Une publication Nefer-IT, 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08 - Directeur de la publication & Rédacteur en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Ont collaboré à ce numéro : ZDNet
Nos experts techniques : L. Piot, A. Blind, C. Maneu, A. Vasseur, C. Michel, S. Houel, B. Legras, Y. Coleu, R. Huart, C. Villeneuve, A. Vache, J. Hourcade, J-N Berger, C. Pichaud, E. Martineau, J-P Gervasoni, D. Duplan, V. Rivière
Couverture : ©Maxiphoto - **Maquette** : Pierre Sandré.
Publicité : François Tonic / Nefer-IT - Tél. : 09 86 73 61 08 - ftonic@programmez.com.
Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.
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, mai 2019

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

Cybersécurité :
RESTEZ INFORMÉ

L'actualité quotidienne
News, avis d'experts, témoignages, livres blancs, etc.
https://www.solutions-numeriques.com/securite/

La newsletter
Chaque lundi, comme 40 000 professionnels et décideurs, recevez la synthèse des informations.
C'est gratuit, inscrivez vous :
https://www.solutions-numeriques.com/securite/inscription/

L'annuaire en ligne
Trouvez l'éditeur de solution, le prestataire de services qu'il vous faut.
https://www.solutions-numeriques.com/securite/annuaire-cybersecurite/

Conférence & Exposition

3e édition

AIPARIS BY Corp

11-12 JUIN 2019

Palais des Congrès de Paris

Rencontrez l'ensemble de l'écosystème français de l'Intelligence Artificielle

5 000+ VISITEURS

110+ PARTENAIRES

80+ SPEAKERS

LEAD THE EVOLUTION*

* Soyez à la pointe de l'évolution

Réservez votre accès gratuit sur www.aiparis.fr