PROGRAMMEZ!

Le magazine des développeurs

07-08/2020 N°241 23e année

DÉVELOPPEUR FREELANCE

/ Comment devenir freelance ?
/ Paradis ou enfer ?
/ Des développeurs témoignent !

/ Des formulaires réactifs avec **Angular**

/ **Kubernetes** sous Linux

/ **Phalcon :** le nouveau framework PHP à connaître

Tout SAVOIR sur Red Hat OpenShift

M 04319 - 241H - F: 6,50 € - RD

Le seul magazine écrit par et pour les développeurs

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

Hors-série n°1 SPÉCIAL ÉTÉ

HORS-SÉRIE N°1 ÉTÉ 2020
DISPONIBLE DÈS LE 3 JUILLET !

sudo reboot -n

+ 21 000 pages publiées, 241 numéros, quelques hors-séries ici et là. Programmez! entame sa 23e année. Le magazine a traversé les époques, les crises et les évolutions.

Kernel panic

Une double crise a frappé. La première est le Covid-19. La seconde est la crise de Presstalis. Presstalis est ce que l'on appelle une messagerie qui prend en charge et distribue les magazines et journaux dans des milliers de kiosques dans toute la France. Mais le manque de réforme profonde du système a fini par faire exploser la messagerie et nous avons vécu des semaines très difficiles : non-diffusion du magazine, menace sur le versement des ventes aux éditeurs, dont Nefer-IT. Cet argent est le résultat des ventes en kiosque. Il s'agit de notre argent. Les coûts toujours plus importants de la distribution font que nous récupérons de moins en moins d'argent.

Un magazine est une équation subtile à trois facteurs : le kiosque, les abonnés et la publicité/sponsoring. Le kiosque fait partie de la culture de Programmez! et vous êtes nombreuses/nombreux à nous lire grâce aux kiosques, aux relais et aux rayons presse des grandes surfaces ! Et nous continuerons à être présents.

Reboot !

À partir de ce numéro, nous rebootons la périodicité du magazine. Et la révolution est profonde. Après 23 ans de parution mensuelle, Programmez! va désormais se séparer en deux branches distinctes :

Programmez! devient bimestriel. Au lieu de sortir chaque mois, le numéro « normal » sortira désormais tous les 2 mois. Soit 6 numéros par an.

Programmez! hors-séries tous les trimestres ! Votre nouveau rendez-vous : tous les 3 mois, Programmez! vous proposera un hors-série thématique. 1 thème = 1 hors-série. Soit 4 numéros par an.

En résumé, vous aurez 10 numéros de Programmez! par an : 6 numéros « normaux » (bimestriel) + 4 numéros thématiques (trimestriel).

Cette nouvelle cadence est là pour vous proposer toujours plus de diversité dans les technologies. Les hors-séries approfondiront, sur 84 pages, un langage, une plateforme.

Les prochaines builds Programmez!

Hors-série n° 1 spécial été 2020 : dès le 3 juillet
N° 242 : 4 septembre

Bienvenue dans la nouvelle matrice.

François Tonic - ftonic@programmez.com

SOMMAIRE

Prochain numéro

HORS-SÉRIE N°1 : SPÉCIAL ÉTÉ DÈS LE 3 JUILLET

Stopcovid : l'application se dévoile

Promise de longue date par le gouvernement, l'application de contact tracing StopCovid a finalement été présentée sur Gitlab le 12 mai. Une partie du code source a été mise en ligne, ainsi que la documentation décrivant le projet et l'organisation de celui-ci. Un bémol néanmoins : si le gouvernement avait promis une application entièrement open source, la documentation du projet précise que certaines parties ne seront pas publiées « car correspondant à des tests ou à des parties critiques pour la sécurité de l'infrastructure. » Une définition de l'open source qui n'a pas l'air de mettre tout le monde d'accord, mais le projet n'en est plus à une critique près.

MàJ de la rédaction : pas mal de flottement dans la disponibilité du code. Mi-mai, seul le code et le SDK du protocole Robert était disponible. Le code de l'app est disponible depuis fin mai. Lien : https://gitlab.inria.fr/stopcovid19

La fondation Eclipse traverse l'atlantique

La fondation Eclipse, notamment connue pour son travail sur l'IDE éponyme, a annoncé qu'elle pliait ses bagages et déménagerait son siège canadien vers l'Europe, plus précisément à Bruxelles. Selon les dirigeants, ce déménagement s'explique par l'intérêt croissant des institutions européennes pour l'open source au cours des dernières années. La fondation gardera ses bureaux en Amérique du Nord, mais entend étoffer ses équipes européennes au cours des années à venir et resserrer les liens avec les membres européens de son écosystème.

New IP : la Chine propose de refaire Internet à sa façon

Les entreprises du secteur télécoms chinois et Huawei ont présenté l'année dernière une refonte en profondeur des normes TCP/IP, qui constituent la base technique du réseau Internet. Ce projet de refonte, baptisé New IP, n'a pas vraiment du goût de tous : de nombreux critiques ont relevé que New IP embarquait des fonctionnalités de censure du réseau allant à l'encontre de la philosophie d'origine du réseau. Le RIPE, registre régional européen s'est d'ailleurs fendu au mois d'avril d'une colonne sur son blog résumant tout le mal qu'elle pensait de cette nouvelle proposition. Le sujet risque en effet d'être discuté âprement au cours du prochain World Telecommunication Standardisation Assembly, prévu en novembre en Inde. C'est à cette occasion que les membres de l'Union Internationale des Télécommunications décident de leurs prochains sujets d'étude, et New IP pourrait bien se retrouver à l'ordre du jour si personne n'y fait attention.

Zoom s'offre Keybase

L'application de visioconférence Zoom a annoncé le rachat de la société Keybase, qui se spécialise dans l'identité numérique et la gestion de clefs de chiffrement. Le montant du rachat n'a pas été dévoilé, ni la façon dont Zoom comptait intégrer les fonctionnalités développées par Keybase. Suite aux nombreuses critiques ayant visé la sécurité de Zoom au cours des derniers mois, le rachat sera probablement mis à profit pour muscler la sécurité de l'application de visioconférence américaine.

Valve laisse traîner du code source, mais se veut rassurant

Fin avril, le code source de versions des jeux Team Fortress 2 et Counter Strike : Global Offensive se sont retrouvées diffusées sur le site 4chan. Le genre de scénarios dont l'éditeur, Valve, se serait sûrement bien passé. Mais celui-ci l'assure : malgré les nombreux appels à la vigilance, les risques de piratage pour les joueurs sont minimes et le code diffusé correspondrait à des versions datant de 2017.

Le minage de bitcoin ne fait plus autant recette

Tremblez ou réjouissez-vous, le troisième « Bitcoin halving » est arrivé. Le 11 mai à 19h23, un groupe de mineurs chinois a miné le 630 000e bloc de la blockchain bitcoin, occasionnant une baisse automatique des récompenses prévues pour les mineurs : celles-ci sont divisées par deux, passant de 12,5 à 6,25 nouveaux bitcoins ainsi crées. C'est un principe inscrit dans le protocole Bitcoin qui veut que les récompenses offertes aux mineurs soient divisées par deux tous les 210 000 blocs créés par le réseau et cet événement intervient plus ou moins régulièrement tous les 4 ans.

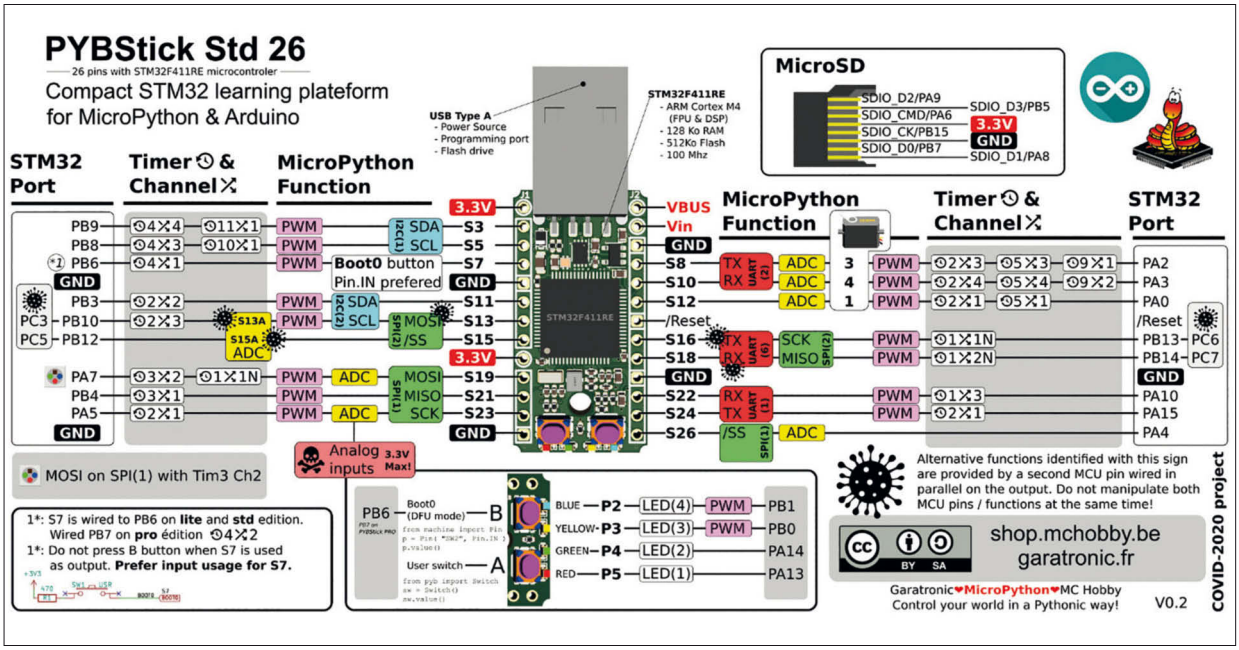
L'appel d'offres 5G repoussé (au moins) au mois de juillet

Les enchères pour l'octroi des fréquences 5G n'étaient déjà pas en avance, mais la pandémie de coronavirus n'a rien arrangé. L'Arcep a ainsi annoncé que le processus, initialement prévu pour le mois d'avril, serait repoussé. Deux scénarios sont envisageables selon le président de l'Arcep : une tenue des enchères au mois de juillet, ou au mois de septembre. « La date exacte dépendra de la vitesse de sortie du confinement », a précisé le président devant le Sénat. Cela laissera le temps aux opérateurs et au régulateur de peaufiner les détails du processus, ou aux détracteurs qui accusent la 5G de tous les maux de saboter quelques pylônes de plus.

PYBStick26 : série Programmez!

Dans quelques semaines, la première carte de prototypage « Programmez! » pour les développeurs, étudiants et makers sera disponible ! C'est un projet que nous avions depuis plusieurs années. Nous étions d'abord partis sur une Attiny ou une Arduino, mais finalement, pourquoi ne pas faire beaucoup mieux? La Pybstick26 était le choix parfait : polyvalente, performante, compatible Arduino ET MicroPython ! Preview de la carte. **La rédaction**

Avertissement : *par rapport à la première pré-série, la Covid-19 a pesé sur la chaîne logistique et des difficultés d'approvisionnement de certains composants ont obligé nos partenaires à modifier certaines ressources.*



Le PYBStick est une carte de petite dimension que l'on peut comparer à une Arduino Nano ou certains ESP, à peine 5 cm sur 1,8 ! Les headers sont fournis, mais pas soudés. La carte utilise la PYBStick standard 26.

Cortex M4, port SD !

Petite, mais puissance ! La carte embarque un Cortex M4 32 bits 100 MHz (STM32F411RE). Largement plus puissant que les Atmel des Arduino. Il embarque en standard 128 Ko de RAM et 512 Ko de stockage. La carte propose 17 GPIO dont 2 SPI et 2IC. 15 broches sont alimentées (broches PWM). On dispose aussi de 3 broches UART. Attention : la carte s'utilise uniquement en 3,3 V !

La carte propose, par défaut, un connecteur microSD. On peut utiliser des SD jusqu'à 32 Go ! Surtout, on peut utiliser la carte comme d'une clé USB, mais son premier usage est de pouvoir étendre le système de fichiers et d'y mettre directement les codes et librairies nécessaires pour développer son objet !

Pour faciliter son utilisation, on enfiche la board directement sur un port USB classique (type A). Et si tout se passe, le volume monte automatiquement… La carte dispose d'un mode DFU qui permet de mettre à jour le firmware.

Quels usages ?

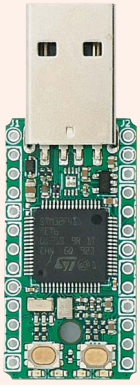
La carte est polyvalente. Vous pouvez coder votre projet en Micro-Python et en Arduino. Si vous êtes habitués à la programmation Arduino, vous ne serez pas dépaysé ! Micro-Python connaît un énorme succès auprès des makers et des développeurs pour créer des objets intelligents, de la domotique, des prototypes.

Les usages sont illimités : il faut juste trouver l'idée. Vous pouvez utiliser des centaines de capteurs du marché. La seule limite est le voltage : 3,3 V et non 5V (même si certaines broches sont tolérantes). En Micro-Python, il faudra faire attention aux librairies.

Par exemple, pour utiliser un écran OLED, vous devrez, selon le modèle, récupérer et déployer SSD1306.py. Il est possible de contrôler des Neopixel, bandes LED, un servomoteur, écran, moteur pas-à-pas, utilisation des composants en I2C ou SPI, créer un bouton reset, etc.

Une documentation en Français sera disponible dès la disponibilité de la carte.

Bref, vous avez l'embarra des choix.



Partenaires techniques :

Garatronic
MCHobby

Open Hardware
Design et fabrication en France

Principales spécifications

- Dimensions : 35x18mm
- poids 4,76 g dont 1,89 g pour le connecteur USB seul
- PCB 1,6 mm FR4 4 couches
- finition étain pur (ROHS)
- brasure SAC305 (ROHS)
- 44 composants
- 132 vias (trous métallisés)
- 17 E/S
- niveau logique 3.3V
- MCU : Cortex M4 32 bits, avec DSP.

Où et quand ?

Disponibilité : fin juin
Où : directement sur
www.programmez.com

Tous les détails dans le hors-série n°1 spécial été de Programmez! : dès le 3 juillet

Roadmap des langages

Chaque mois, Programmez! vous propose un panorama des agendas de sorties des versions des langages, frameworks, etc. Malgré le Covid-19, les développements reprennent leur vitesse de croisière.

Disponible

Flutter 1.17 & Dart 2.8

La nouvelle version de Flutter se focalise sur 2 éléments : les performances et la mise à niveau de Dart et de son nouvel outil de package. En même temps que le framework, Dart 2.8 a été déployé. La 1.17 supporte Metal pour améliorer les performances sous iOS (Metal étant natif à l'OS). On trouve aussi un nouvel outil de tracking réseau. L'équipe annonce aussi de nombreux bugs corrigés. En tout, la communauté a fixé 800 requêtes, géré 3 000 pull requets. Toujours sur iOS, l'équipe évoque une baisse de la charge CPU et GPU, tout en améliorant les performances globales. La taille des applications a été réduite. On notera aussi : Google Fonts, diverses améliorations autour de Material, fonction expérimentale de fast start. Dart arrive en version 2.8. Change log Flutter 1.17.0 : https://flutter.dev/docs/development/tools/sdk/release-notes/changelogs/changelog-1.17.0

Node.js 14

Cette nouvelle version propose d'intéressantes nouveautés et améliorations :
- Moteur V8 passe en version 8.81,
- API de stockage local async (expérimentale),
- Version expérimentale de WASI,
- Le rapport de diagnostic se stabilise,

Le support LTS devrait arriver en octobre 2020.

Next.JS 9.4

La nouveauté principale est l'introduction d'une nouvelle expérience de rechargement à chaud, Fast Refesh, activé par défaut. Le rechargement à chaud n'était pas d'une qualité suffisante pour être activé par défaut. Par exemple, l'ancienne implémentation de rechargement à chaud n'était pas résistante aux erreurs de compilation ou d'exécution et effectuait un rechargement complet de l'application en cas de faute de frappe lors de la modification de votre CSS ou JavaScript. La régénération statique incrémentielle arrive en version bêta. Il s'agit d'un mécanisme permettant de mettre à jour les pages existantes, en les restituant en arrière-plan.

TypeScript 3.9

L'équipe TypeScript promet des améliorations sur les performances, le temps de compilation. La v3.9 doit corriger un comportement anormal sur certaines déclarations de fonctions. Le processus d'inférence de la 3.9 doit corriger cela. TypeScript 4.0 devrait arriver en août prochain. Tous les détails ici : https://github.com/microsoft/TypeScript/issues/38510

Bootstrap 4.5

Cette version corrige des bugs des versions précédentes et promet une petite optimisation sur le poids des fichiers. Les exemples et la documentation sont maintenant disponibles en fichier ZIP. Cette version continue à s'appuyer sur jQuery qui sera retiré avec la v5.

2e semestre

Kotlin 1.4

Ete

La principale nouveauté est un nouvel algo d'inférence de type (en préversion dans la 1.3). Le nouvel algorithme déduira des types dans de nombreux cas où l'ancienne inférence vous obligeait à les spécifier explicitement. Par exemple, dans l'exemple suivant, le type du paramètre lambda it est-il correctement déduit comme étant String?

```
val rulesMap: Map<String, (String?) -> Boolean> = mapOf(
  "weak" to { it != null },
  "medium" to { !it.isNullOrBlank() },
  "strong" to { it != null && "^[a-zA-Z0-9]+$".toRegex().matches(it) }
)
```

Le nouvel allocateur d'objets fonctionnera jusqu'à deux fois plus rapidement sur certains benchmarks, selon JetBrains. Cette version inclura aussi un nouveau compilateur IR (intermediate representation) côté backend Kotlin / JS. Mais l'éditeur prévient qu'il y aura un problème de compatibilité binaire.

Swift 5.3

Automne

La grosse nouveauté sera l'arrivée officielle sur Windows. Côté Linux, le langage arrive sur de nouvelles distributions : Ubuntu 20.04, CentOS 8 et Amazon Linux 2 !
Au-delà, la version 6 apportera pas mal de nouveautés et d'améliorations :
- Possibilité de porter le langage sur de nouvelles plateformes (ARM ?) ;
- Des améliorations sur le gestionnaire de paquets ;
- Temps de compilation réduits ;
- Meilleure autocomplétion du code.

Aucun agenda n'a pour le moment été communiqué.

Java 15

septembre

Java 15 supportera les blocs de texte. Cette fonctionnalité est attendue depuis les premières préversions sorties avec Java 13. Les blocs de texte sont des littéraux chaînes à plusieurs lignes. C'est tout simple, mais cela permet d'insérer beaucoup plus facilement, dans le code Java, des extraits de code en HTML, XML, SQL, JSON, etc. Java 15 fera aussi du ménage en retirant Nashorn, le moteur JS d'Oracle. GraalVM rend obsolète le moteur.

Python 3.9.0

Octobre

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

.Net 5

Novembre

Elle doit réunifier les différents .Net. L'ambition est d'être disponible sur Windows, Linux, macOS, iOS, Android, tvOS, webassembly, etc. Disponible en version preview.

PHP 8.0

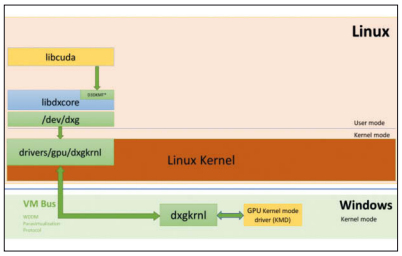
Décembre

La prochaine grande version de PHP devrait arriver fin 2020. Tout n'est pas encore fixé, mais nous connaissons les premières nouveautés et évolutions du langage :
- Forte possibilité d'une casse de compatibilité avec les versions 7.x : sur ce point, on attend les précisions ;
- Les types Union vont arriver avec PHP 8, qui permettront de définir plusieurs types pour les arguments reçus par une fonction, ainsi que pour la valeur qu'elle retourne. En plus du type self, le type static va devenir un type de valeur retournée valide ;
- Le compilateur JIT de PHP 8 promet d'importantes améliorations de performances.

Microsoft BUILD 2020 : .Net MAUI, WSL + DirectX

La conférence technique BUILD était cette année virtuelle. Durant 2 jours, les équipes Microsoft ont proposé de nombreuses sessions. Et comme chaque année, des nouveautés, plus ou moins disponibles, ont été annoncées. Voici ce que nous avons retenu.
Attention : la roadmap peut changer ou glisser de quelques mois.

Le sous-ensemble Linux gagne DirectX



Jusqu'à présent, une des limitations de WSL est la difficulté à utiliser des applications à interface graphique. Les équipes de l'éditeur travaillent à l'accélération graphique et de DirectX dans WSL. On aura droit au dxgkrnl, un nouveau pilote Linux pour pouvoir utiliser des cartes graphiques directement sous WSL en passant par une couche d'abstraction/virtualisation du GPU. Jusqu'à présent, cette couche était réservée aux OS Windows. Sur la partie OpenCL, OpenGL, Microsoft travaille le sujet. Sur Vulkan, pour le moment, rien de concret. Bref, on pourra utiliser de la 3D, de l'accélération GPU, mais le travail est loin d'être terminé et apparaît compliqué à implémenter.

.Net Multi-platform APP UI

.Net MAUI sera la couche d'interface (native) multiplatorme de .Net sur Windows, macOS, Android. Il s'agit d'avoir un seul projet avec l'ensemble des assets, une seule base de codes. L'objectif est d'intégrer ce nouveau composant dans .Net 6 prévue à l'automne 2021. Les premières bêta devraient arriver vers fin de l'année. MAUI est l'évolution de Xamarin.forms. Avec l'unification des différents .Net à partir de .Net 5, Microsoft veut éclaircir la couche d'interface. Et Xamarin va peu à peu se fondre dans cette logique et le nom disparaître. La migration de forms à MAUI devrait être assez simple, car la base technique est identique.

En attendant, Microsoft a annoncé une version majeure de Xamarin.forms dans les prochains mois et sera régulièrement mis à jour en attendant son intégration dans .Net 6.

Projet « reunion » + WinUI 3.0

Depuis plusieurs années, les développeurs se perdent dans les différents modèles de développements applicatifs Windows. Il reste aujourd'hui deux grands modèles : Win32 et UWP. Microsoft retente sa chance pour unifier les modèles et proposer une approche « universelle ». Objectif : 1 seul modèle de développement d'applications ciblant l'ensemble des plateformes (sous Windows 10) en réduisant l'écart entre Win32 et UWP. Que l'on développe en C++, langage .Net, React Native. Reunion doit supporter l'ensemble des librairies, frameworks, packages actuels et futurs. Et les API réservées à UWP seront aussi disponibles en Win32.
Cette approche aidera (dixit Microsoft) les développeurs à garder à jour les apps et à intégrer les nouveautés. Et surtout, les dévs qui ne voulaient pas entendre parler de UWP, y auront accès (même indirectement), notamment sur toute la partie interface. Mais cela ne signifie pas la fusion entre les deux « mondes », pas pour le moment. WinUI 3.0 fait partie de cette nouvelle approche Win32/UWP. WinUI est un framework complet d'interface utilisateur. Il sera inclus dans .Net 5. Ainsi une app Win32 pourra exploiter les interfaces de Windows 10, que ce soit sur les anciennes apps (à moderniser donc) et les nouvelles. Il est bien entendu natif à Windows et inclura les dernières évolutions de Fluent Design System. Il supporte les langages .Net et C++. Il est utilisable avec .Net sans être dépendant de ce dernier, un point important. Et surtout, WinUI est 100 % C++ !

Outil C#/WinRT

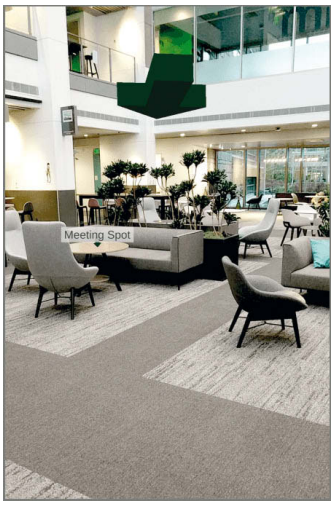
C#/WinRT est un toolkit pour pouvoir utiliser les API WinRT (RT pour runtime) directement dans son code C#. Ce que l'on appelle une projection. La projection est une couche intermédiaire pour la conversion et l'interopérabilité. Tout n'est pas encore disponible. Dans une future version, il est prévu de proposer le support des types WinRT à C#. Il supportera WinUI 3.0.

Visual Studio

Côté Visual Studio, Microsoft a dévoilé la 16.6 de VS 2019 en version finale et la préversion de la prochaine 16.7. La 16.6 étend le support de la librairie standard de C++ 20, amélioration sur le snapshot de debugging, le designer WinForms supportant .Net Core.
La future 16.7 apportera des améliorations notables sur Git et une meilleure expérience utilisateur, l'arrivée des Address Sanitizer de C++, Intellisense Code Linter pour identifier et fixer le code en défaut, de nouvelles options dans la combinaison Contrôle + x et l'ajout de nouvelles actions rapides (quick actions). IntelliCode aura de nouvelles recommandations pour les fonctions. On aura aussi pas mal de choses sur la partie XAML (WPF, UWP et Xamarins.forms).
Ce post présente l'ensemble des améliorations :
https://devblogs.microsoft.com/visualstudio/visual-studio-2019-v16-6-and-v16-7-preview-1-ship-today/

Azure Spatial Anchors

Parmi les nombreuses nouveautés d'Azure, nous retenons le service Spatial Anchors. Sous ce nom peu clair se cache un service multi-plate-forme dédié aux usages/apps de réalité mixte. Les objets virtuels restent à leur emplacement même si on bouge, si nous sommes en contexte multi-utilisateur. Ce qui peut être très intéressant dans des jeux multijoueurs. Ce service peut aussi aider à conserver l'état d'un contenu virtuel dans un contexte réel. Ce service supporte Unity (Android, Hololens, iOS), iOS, Android, Hololens et Xamarin.

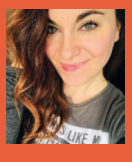


Pour en savoir plus :
https://docs.microsoft.com/fr-fr/azure/spatial-anchors/

En résumé, la roadmap des principales annonces :

1er semestre 2020 //BUILD 2020	2e semestre & fin 2020	Vers l'infini et au-delà : 2021 -> 202x
WinUI 2.4 VF	WinUI 3.0 VF	.Net MAUI VF
WinUI 3.0 PV	.Net 5 VF	Évolution de WinUI.
Project Reunion PV	.Net MAUI PV	Net 6 VF
C# 9.0 PV	Xamarins.forms vNext	
Windows Terminal 1.0 VF	Visual Studio 2019 16.7 VF	
Uno Platform 3.0		
C#/WinRT (outil) PV		
Visual Studio 2019 16.6 VF		
Visual Studio 2019 16.7 PV		
Azure Spatial Anchors		

VF : version finale - PV : préversion



Shirley Almosni-Chiche
Agent de carrières IT, Recruteuse Freelance et Agitatrice Sociale

Océane Roudier
RH Freelance, co-organisatrice de NewCrafts Conference, membre du CJD

LOST IN RECRUITMENT

Nous avions initialement prévu une série d'articles RH spécifiques au monde de la tech en nous basant sur nos expériences respectives pour traiter de sujets tels que les modèles d'entreprises actuels, les différents statuts, les évolutions possibles, les boosters de carrière… Cependant, en revoyant notre copie, nous avons réalisé que l'actualité de ces derniers mois avait profondément bouleversé l'ordre établi et que si tous nos apprentissages et observations glanés ces 8 dernières années n'avaient pas volé en éclat, ils avaient tout-de-même du plomb dans l'aile ! Ce n'est donc pas un article RH mais plutôt un rapport d'étonnement de la part de 2 professionnelles des Ressources Humaines que nous vous livrons. Evidemment, nos propos n'engagent que nous !

Reboot professionnel

Nous sommes recruteuses dans la tech et à l'arrêt. Nous ne sommes pas des héros, ne sauvons pas de vies, ne nous sacrifions pas pour les autres à l'inverse des professions essentielles en cette période de pandémie. Nous sommes souvent qualifiées de spammeuses, harceleuses, techniquement nulles, sans valeur-ajoutée ni même valeurs humaines. Le milieu nous est plutôt hostile. Avouez : vous n'êtes pas tendres avec nous ! Mais nos jobs ne sont pas réellement « bullshit »*, puisque sans nous, l'entreprise ne fonctionne pas vraiment.

Nous sommes des fourmis. Dans un métier comme le nôtre, tout prend un temps fou : comprendre les profils, les technos, rencontrer du monde, animer une communauté, convaincre des personnes de nous rencontrer, de considérer une opportunité, de la saisir, et tout cela avec la conviction de leur offrir une meilleure situation. Se débarrasser de l'image dégradante qui nous colle à la peau et acquérir un minimum de considération dans ce milieu est un travail colossal et continu. Et c'est grâce à notre passion pour ce job que nous trouvons chaque jour l'énergie pour continuer, nous améliorer, asseoir notre confiance en nous et faire fi des critiques.

C'est aussi un travail intense, qui prend aux tripes. Nous accompagnons des humains dans leurs ambitions, leurs questionnements, leurs doutes. Nous sommes ces gens de passage dans une vie, plutôt aux bons moments, les évolutions. Nous sommes présentes du mieux que nous pouvons pour rendre les transitions professionnelles agréables, les prises de poste réussies. Nous ne sommes pas nombreux.ses à fonctionner ainsi, presqu'aussi rares que les masques FFP2 en avril 2020, mais nous nous efforçons de nous rendre visibles et utiles aux autres parce que nous croyons en l'utilité de notre fonction.

Mais au moment où nous écrivons, nous sommes à l'arrêt et observons. Cette situation de crise, de « guerre » diront certains, a tout bouleversé. Si certaines entreprises sont allées au-delà d'un discours superficiel de « marque employeur » et ont réellement tout fait pour conserver leurs collaborateurs, faire durer les projets coûte que coûte, proposer des missions en interne qui seront utiles à l'ensemble de l'entreprise une fois la crise passée, d'autres ont abîmé des mois voire des années de travail en mettant fin à des périodes d'essai comme on déchire un livre à la préface et en piétinant leur marque employeur. Dans notre réseau professionnel, on a pu entendre et lire des gens perdus, désabusés, à la recherche d'explications, d'oreilles bienveillantes, de solutions. Derrière ces périodes d'essai rompues, il y a des vies humaines, des loyers à payer, de la colère, de la peur, une honte sociale et familiale. Il y a aussi des recruteurs qui ont bossé pour rien, cru en ces nouvelles recrues, engagé leur responsabilité morale pour que, finalement, tout ce travail disparaisse en quelques secondes sur la base de décisions bancales de crise, et sans obligation de justification. Et dans un grand flou légal. **

Mea culpa ?

Dans un contexte chaotique comme celui-ci où il n'y a plus de notion de temps, de prévision, de business plan réaliste et où l'ennemi est invisible, il devient vital pour les entreprises de « personnifier » les coupables : parmi eux, on retrouvera bien sûr le recruteur. Il a trop ou pas assez recruté, fait des mauvais choix, etc. Si on vit mal, la crise c'est à cause de lui, assurément ! Sa fonction de porte d'entrée et de fournisseur de « ressources humaines » cristallise sur lui la croissance ou la décroissance de l'entreprise. En temps normal, il subit quotidiennement cette pression du chiffre, du nombre de rendez-vous, du nombre de recrutements à réaliser en un temps record parfois. Et à côté de ça, il subit aussi celle des candidats. Et là, en temps de crise, c'est encore sur lui qu'on fait peser le non-staffing de certains, le manque de profils disponibles « pile-poil » comme on réclame à l'instant T. Et c'est lui qui se sent coupable d'avoir fait quitter un job pour un autre à quelqu'un qui aujourd'hui n'a plus d'emploi.

En parallèle des reproches, plus vraiment d'activité. On ne recrute plus. Et on se demande à quelle sauce on sera mangées demain. Nous faudra-t-il à nouveau « fournir » des candidats comme s'ils étaient de simples ressources interchangeables pour aider le marché à redémarrer de plus belle ?

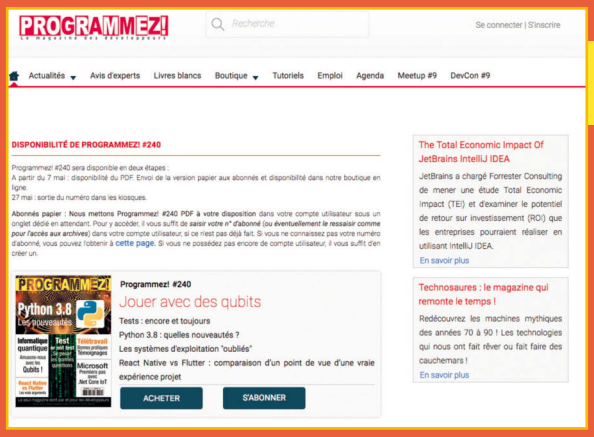
Nous demandera-t-on des comptes en fin d'année sur le turnover 2020 ? Quelqu'un sera-t-il conscient du fait qu'on ne reconstruit pas une marque employeur à la vitesse à laquelle on l'a balayée ?
Parce que le recrutement est une des vitrines d'une entreprise et affiche les signaux d'une bonne ou d'une mauvaise santé économique, il nous faudra rallumer la machine, s'épuiser, ramer pour que l'entreprise puisse à nouveau produire, conquérir, gagner.
D'ailleurs, si on jette un œil à l'étymologie du terme "recrutement", on trouve un sens militaire : recruter les troupes, lever des armées, faire passer la recrue au statut d'arme en action. Une armée conséquente est puissante, impressionne et fait reculer l'ennemi. Ainsi, le recruteur a cette responsabilité, ce devoir presque pressant, constant, guerrier de faire grandir l'armée pour gagner la bataille, pour briller aux yeux de la concurrence.
Après la crise, il faudra relancer la machine à fond pour signaler : Regardez ! Nous sommes toujours là et en pleine forme !! Gare à vous !!

On se relève mais « on n'oubliera pas »

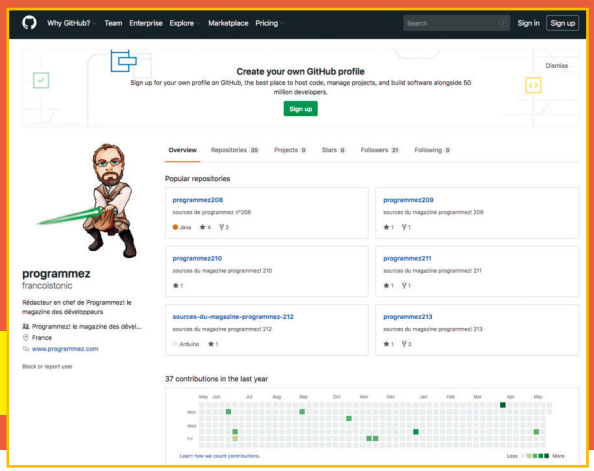
Vous l'aurez compris, nous avons vécu depuis le début de la crise une querelle profonde avec notre système de valeur. Cette crise sanitaire est pour nous une véritable crise intérieure : on remet tout en question. Impuissantes, « on se lève et on se casse » à la cuisine pour s'enfiler une tisane au miel ou un verre de vin blanc, suivant l'heure.
Mais ça y est, au bout de plusieurs semaines, nous avons notre feuille de route, peu importe ce qui adviendra.
On sera du côté de ce qu'on trouve juste, avant tout. On se permettra le luxe du choix. On sera du côté des candidats qui cherchent à mettre leurs compétences au service de quelque chose qui a du sens, des entreprises clean, des projets et des produits en lesquels on croit. Nous travaillerons de nouveau avec les boîtes qui ont géré la crise humainement plutôt qu'avec un Excel, qui ont pris le mot solidarité au pied de la lettre à rebours de la solidarité "de marché" glorifiée dans la presse économique et politique.
Peu importe si quand l'une d'entre nous a pointé du doigt publiquement ces périodes d'essai rompues sur les réseaux sociaux, moyens de contestation de confinement, elle a reçu les remarques suivantes : « c'est trop tôt pour juger », « tu ne joues pas collectif », « ces personnes trouveront facilement un job, d'autres perdent la vie ». Nous assumerons de ne plus bosser pour d'autres valeurs que celles qui nous habitent et pour des boîtes qui piétinent notre travail.
Le monde tech en lequel on croit, il existe, et on sera là pour participer à son redémarrage, à sa reconstruction. C'est peut-être présomptueux mais on pense qu'il faudra des gens entiers, comme nous. On fera en sorte de comprendre comment ça marche et de vulgariser, d'apporter du conseil. On donnera notre énergie. La reprise prendra du temps mais on ne lâchera pas.
Et dès qu'on pourra, quand on aura compris comme le marché redémarre, on rédigera ce fameux dossier RH promis à Programmez! Rendez-vous dans quelques mois pour un point de parcours !!!
Humainement, Shirley & Océane

** Bullshit Jobs, par David Graeber, éd. Les Liens qui libèrent.*
*** Selon le Code du Travail, employeurs comme salariés peuvent mettre fin à une période d'essai à tout moment et sans obligation de motiver la décision. Cependant, Me Virginie Langlet, avocate en droit du travail précise dans Le Parisien du 3 avril 2020 : « Dans la mesure où la période d'essai a pour but unique de tester les capacités professionnelles du salarié, elle ne peut être rompue que pour un motif lié à ses aptitudes : en aucun cas la rupture ne peut être décidée pour une autre cause. En conclusion, ni l'état de crise sanitaire, ni le confinement lié au coronavirus, ne peuvent justifier la rupture de la période d'essai. Le prétexte de l'état de crise sanitaire, ou même des difficultés économiques de l'entreprise qui en découlent ne sont pas un motif valable de rupture de la période d'essai ».*

Où récupérer les codes sources des articles ?



programmez.com ←



→ github.com/francoistonic

François Tonic

Free-lance : le devenir, le rester !

Être indépendant, ne pas dépendre d'un boss, prendre les missions que l'on souhaite, se valoriser à sa juste valeur, s'éclater, s'organiser comme on veut… voilà quelques idées que nous entendons souvent et parfois avec une certaine naïveté. C'est quoi être développeur free-lance en 2020 ? Surtout avec la crise que nous avons vécue et qui nous impactera encore plusieurs mois.

Fin avril, des chiffres de l'Apec indiquaient une baisse de 38 % des annonces sur la partie R&D et ingénieurs informatique. Les métropoles et régions traditionnellement actives sur les technologies et l'IT sont les plus touchées. Les entreprises arrêtent ou reportent les embauches, projets purement et simplement arrêtés ou annulés. Le télétravail a redéfini les rôles et les projets.

Depuis quelques années, les plateformes de mises en relations pour les indépendants se sont multipliées. Et les plateformes plus généralistes publient aussi des offres d'emplois pour des free-lance. Par exemple, pour février 2020, presque 400 offres étaient proposées par indeed.fr.

Combien de free-lance informatiques en France ? C'est difficile d'être précis, car c'est assez mouvant entre les nouveaux arrivants et celles et ceux qui prennent un poste à plein temps. Selon une étude Malt de 2019, environ 930 000 free-lance IT en France, avec une croissance des demandes dans le secteur de la banque ou dans certains intégrateurs.

Pas d'improvisation

Free-lance cela ne s'improvise pas. Vous devez avoir une démarche :

- Définir le projet et vos ambitions
- Avoir une idée du chiffre d'affaires dans les premiers mois et même l'année à venir
- Quels clients potentiels ou clients déjà démarchés ?
- Régler les questions légales : statut, compte bancaire professionnel, assurances, locaux.
- Matériel, licences (si nécessaire).
- Soigner votre site web, réseaux, etc.
- Préparer vos argumentaires et documents marketing.

Le plus difficile n'est pas de se lancer (même si franchir le Rubicon n'est jamais facile), mais de tenir dans la durée, affronter les mois difficiles, les chutes d'activité, les pics. Et il faut sortir de sa zone de confort : c'est-à-dire de son quotidien connu et rassurant.

> « *Il faut savoir faire des projets pour payer les factures et pas pour se faire plaisir* »

Avoir des missions = savoir se vendre = faire du marketing

Un free-lance par définition ne dépend d'aucune entreprise. Il doit trouver les missions. Une évidence, mais qu'il faut rappeler. Trouver des missions, des clients n'est pas à sous-estimer, surtout durant les premiers mois. Si vous avez un solide réseau, des connaissances auprès d'entreprises, d'ESN, des communautés, cela peut vous aider à vous faire connaître et à trouver plus rapidement des missions.

Les plateformes free-lance de mise en relation sont aussi un autre moyen pour être vu et potentiellement trouver des missions. La visibilité est un enjeu important. Il faut aller dans les meetups, se faire connaître des communautés, aller dans les conférences développeurs, suivre les réseaux sociaux, proposer des sessions, etc.

Etre free-lance c'est accepter des missions alimentaires. Ce sont des missions pas toujours intéressantes du point de vue technique ou développement, mais elles permettent de vivre, de payer les charges et factures. Le récurrent est un point important. Avoir quelques missions de plusieurs jours par mois permettent cette récurrence et donc la facturation.

Cette récurrence pourra aussi vous permettre d'accepter ensuite des missions qui vous intéressent réellement.

Soigner sa présentation

Quand vous êtes indépendant, votre carte de visite/plaquette marketing c'est vous. Soyez attentive/attentif à votre CV, vos pages sociales, votre site web, etc. Préparez votre carte de visite, vos plaquettes de services, etc. Ayez tout le matériel de promo nécessaire. Tenez à jour vos références.

Si vous travaillez en sous-traitance pour une ESN, vous devrez sans doute suivre des

NE PAS OUBLIER LE STATUT ET LES PETITS DÉTAILS

Être indépendant ne signifie pas : aucun statut, aucune obligation. Vous devrez choisir entre les différents statuts possibles en France : entreprise individuelle, EURL, SASU. Dans certains cas, vous pouvez éventuellement être autoentrepreneur, mais attention aux plafonds de chiffre d'affaires, à la protection sociale et aux points retraite.
Le portage salarial est une solution qui a l'avantage de simplifier la création de l'activité, mais attention au contrat proposé et au % pris par la société de portage.
Vous aurez aussi à payer la T.V.A. eh oui, on oublie souvent ce petit détail. Et les montants dus peuvent vite devenir importants si on ne gère pas correctement la comptabilité. Les règles sur la T.V.A. peuvent changer selon le statut et le mode d'imposition choisi.
Attention aussi à :

- Imposition des bénéfices.
- Affiliation à un régime de sécurité sociale et de retraite.
- Tenue de la comptabilité.

Même si ce n'est pas obligatoire, nous vous conseillons de consulter un expert/cabinet comptable qui pourra vous conseiller et tenir la comptabilité. Il pourra aussi vous aider dans certaines démarches et déclarations. Ne sous-estimez pas la charge de travail de la compta, facturation et administration.
N'oublions pas non plus les assurances responsabilité civile professionnelles, la multirisque professionnelle et éventuellement des garanties spécifiques. Ces assurances doivent vous couvrir, les locaux utilisés, le matériel, les déplacements, etc.

règles et recommandations. Vérifiez les clauses du contrat et les obligations.

Locaux/bureau

C'est une question récurrente : où travailler quand on est free-lance ? Plusieurs réponses possibles :

- À son domicile si vous êtes en télétravail. Aménagez-vous un espace bureau/poste de travail pour le dév, mais aussi l'administratif.
- Espace co-working : possibilité de louer un bureau dans un espace dédié.
- En entreprise : quand vous êtes en mission, en régie.

Ne sous-estimez pas les déplacements si vous êtes amenés à vous déplacer. Voir le retour terrain d'Aymeric sur le co-working.

Se former !

La compétence technique est quelque chose qui évolue tout le temps, surtout dans le développement. À vous d'être à jour : formez-vous, testez les nouveautés, passez des certifications. La veille technologique est un élément central dans la vie d'un free-lance. Cela permettra d'évoluer et d'apporter de nouvelles compétences. Vous prouverez votre capacité à réagir et à rester à jour.

Le TJM : un tarif incontournable ?

Le tarif journalier sera votre référence. C'est lui qui vous servira de base de facturation si vous facturez à la journée. Même en cas de montant forfaitaire, le TJM issu du forfait sera intéressant à calculer.
Comment souvent, le TJM varie selon de nombreux facteurs : compétence, les technologies/difficulté du projet, les déplacements, selon la ville ou la région concernée. Certaines technologies/certains langages auront un TJM plus élevé, car les compétences sur le marché ne sont pas nombreuses, et inversement, si les compétences sont nombreuses (en ESN ou en free-lance), le TJM subira une décote.
Selon quelques chiffres donnés par la plateforme malt.fr(1), voici une idée d'un TJM sur plusieurs technos (en €) :

	TJM	Paris	Lyon	Lille
Dév Python	565	624	563	471
Dév Java	548	615	505	509
Dév PHP	440	492	440	436
Dév jeux	394	433	422	319
Dév Wordpress	382	405	375	384

Comme on peut le voir, le TJM est variable. Et il faut le pondérer selon les compétences, l'abondance de l'offre dans la région, si c'est un client fidèle, un nouveau client, etc.
N'oubliez pas d'inclure/calculer : les transports si nécessaire, les repas, les nuitées si cela vous oblige à vous déplacer, etc.
Attention, le TJM n'est pas un maître étalon comme vous le verrez dans les différents témoignages. Mais il demeure un indicateur à connaître même si vous ne faites pas de TJM.

Plusieurs types d'activité

Le free-lance peut agir sous différentes formes et donc facturer différemment :

1 Type de missions

- Mission en régie : vous êtes majoritairement chez le client. Vous agissez en votre nom ou pour le compte d'un prestataire (typiquement une ESN).
- Mission majoritairement en télévrail : vous travaillez essentiellement dans vos locaux, déplacements ponctuels.

2 Type de contrat/facturation

- Base forfaitaire : vous avez défini un budget global accepté par le client. Une clause doit permettre une facturation supplémentaire si les développements dépassent l'ordre de mission ou le budget.
- Base TJM : vous facturez la journée.
- Vous êtes en sous-traitance pour une ESN : vous pouvez agir en qualité de développeur pour un prestataire (intégrateur, ESN par exemple). Vous facturez le prestataire et non le client. Vous agissez au nom du prestataire.
- Vous passez par une plateforme de mise en relation pour trouver une mission. La plateforme prend une commission.
- Facturation via une société de portage : la société de portage vous prend une commission pour effectuer pour vous la facturation, le paiement, etc. Attention : soyez prudent(e) à la commission et aux frais annexes.

La durée du forfait ou de la régie sera à préciser. Certaines missions peuvent durer plusieurs mois et devenir du (quasi) plein temps. D'autres missions seront récurrentes : x jours par mois pour tel client.

Vous organiser !

L'organisation est une nécessité. Si vous n'êtes pas habitué à établir des plannings, à caler des jours précis pour un client, etc., vous allez devoir vite changer d'attitude et de mentalité. Le free-lance, et même en télétravail, la discipline et l'organisation sont 2 clés de la réussite ou de l'échec. Vous devez organiser les jours, les semaines, les mois, surtout si vous avez plusieurs missions et de la récurrence.
Tenez à jour vos plannings de missions et votre agenda. Ayez une rigueur dans la conduite de la mission : documents techniques, développements, respect du planning, etc. Soyez transparent(e). Avertissez le plus tôt possible si vous sentez qu'il y aura un problème de développement ou un conflit d'agenda.

(1) https://www.malt.fr/t/barometre-tarifs/tech/developpeur-backend/developpeur-python

Facturation/paiement

Théoriquement, les paiements se font généralement à 30 ou 45 jours. Cela peut varier selon les entreprises. Pour les administrations, ces délais sont régulièrement plus longs, 60 jours, tout comme pour certaines entreprises qui le disent immédiatement. En cas de retard, de non-paiement, faites préalablement des relances « amicales ». Bien souvent, il s'agit d'un retard de la compta ou d'un oubli. Si vous sautez à la gorge immédiatement, ce sera à double tranchant pour votre réputation. Soyez un minimum souple surtout s'il s'agit d'un gros client (potentiel ou récurrent).

Les entreprises demandent l'ouverture d'un compte fournisseur. C'est très fréquent avec les grandes entreprises. Soyez attentif à ce détail qui peut prendre plusieurs semaines (sic !) et oblige à remplir un dossier et fournir des documents. Ayez toujours sous la main un K-Bis si besoin et votre tampon officiel. Pour les administrations/institutions publiques, les factures sont envoyées désormais sur le portail Chorus. À vous de créer votre compte et d'y injecter les factures et d'en faire le suivi.

Un seul gros client ?

C'est l'éternelle question en business : un seul gros client ou plusieurs petits. L'avantage du gros client qui fait 90 % de votre activité est que vous savez que vous n'avez pas à chercher ailleurs et que chaque mois, la facturation tombe. C'est top et rassurant. Mais quand le client arrête de travailler avec vous et que vous n'avez pas d'autres pistes/missions pour pallier le manque à gagner, vous risquez une baisse drastique de vos revenus et très rapidement. Nous l'avons vécu en SSII, nous connaissons le confort et l'inconfort de la situation.

L'idéal serait d'avoir 50-50 : 50 % gros client, 50 % missions ponctuelles.

La plateforme Malt

Aujourd'hui, Malt est présent dans plusieurs pays dont la France, l'Allemagne et l'Espagne. +180 000 free-lance sont référencés sur la plateforme dont 40 % dans les domaines de l'IT et de la donnée. En face, +90 000 entreprises sont inscrites et des milliers de propositions de missions sont déposées chaque mois. « 80 % des entreprises du CAC40 sont inscrites » précise Camille Léage de Malt. Malt essaie de privilégier les missions longues, d'au moins 3 mois.

« Le Covid-19 est une réalité. On constate que 80 % des missions sont préservées, mais la demande entrante a chuté de 30 %. » poursuit Camille.

Plus globalement, Camille insiste sur le fait que le free-lance n'est pas un statut protégé et qu'il est encore peu représenté. Peut-être que la crise actuelle ouvrira le débat, notamment sur la forme de représentativité du monde free-lance. Le free-lance est encore très individuel et il est souvent décrit comme un mercenaire.

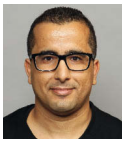
« Le TJM est un indicateur. Il ne représente pas forcément le montant final. Nous avons tendance à dire qu'il ne faut pas (se) brader. Mais il est plus facile de négocier à la baisse qu'à la hausse. Combien de jours me faut-il pour avoir le même niveau de salaire qu'avant ? Il faut organiser son temps pour pouvoir réaliser des projets personnels. » explique Camille.

Malt est là pour aider et conseiller, avec un suivi des free-lance. Pour exemple, sur les notes de frais, la plateforme fait de la pédagogie pour faire comprendre la notion de frais, que le free-lance n'est pas corvéable, que ce n'est pas un salarié de l'entreprise.

FREE-LANCE : UNE ORIGINE MULTIPLE

D'où vient le terme « free-lance » ? L'une des références les plus connues est le roman de Walter Scott (1819), Ivanhoé. Le soldat met au service du roi sa lance libre. En réalité, le terme apparaît dès le Moyen Âge avec les mercenaires qui sont des « lances libres ». C'est-à-dire des troupes ou des individus qui mettent au service d'un seigneur ou d'un roi, leurs armes (dont la lance). La notion de mercenaire existe depuis l'Antiquité.

Mais c'est véritablement Ivanhoé qui va populariser le terme qui va peu à peu de répandre et déborder dans le langage courant.



Riadh Mnasri : « *quitter sa zone de confort* »

Je suis Riadh MNASRI développeur et technical leader sénior. Je suis passionné par le développement logiciel et je code en Java depuis plus de 15 ans. Je m'intéresse aussi à Kotlin depuis 3 ans. Je suis adepte des pratiques Craftmanship (TDD, Clean Code, partage des connaissances...) et je m'intéresse également à la programmation fonctionnelle.

Depuis combien de temps es-tu free-lance ? Pourquoi avoir choisi le free-lance ? Après un passage en entreprise/ESN ?

Je suis free-lance depuis plus de 3 ans. Avant d'être free-lance, j'ai fait de nombreuses expériences dans plusieurs ESN et une expérience chez un client final. Après plusieurs années, je me suis rendu compte de la difficulté de certaines SSII à trouver les missions intéressantes techniquement, qui correspondent à mes attentes et qui contribuent à mon épanouissement technique et relationnel.

J'ai vu aussi qu'en tant que sénior qu'on a tendance à stagner techniquement dès 2 à 3 ans d'expérience et qu'il n'est pas évident d'avoir l'accord de son manager ESN pour changer de mission. En effet, l'intérêt de l'ESN est que tu restes le plus longtemps possible dans une mission.

Donc les propositions de changement restent rares surtout si le consultant reste passif en attente d'une opportunité.

J'avais aussi pratiquement les mêmes problématiques, lors de mon passage chez un client final en interne. Il est très difficile de changer de projet quand on a commencé à avoir une certaine expertise et des connaissances approfondies du sujet, notamment sur l'aspect fonctionnel.

Du point de vue formation, les propositions étaient généralement rares et j'étais toujours obligé de m'investir à fond avec mes propres moyens afin de rester à jour et compétitif sur le marché des talents ayant des compétences pointues.

Je me suis dit pourquoi ne pas tenter l'expérience free-lance. J'avais envie de sortir de ma zone de confort, de choisir mes missions et de tenter

d'apporter mon expertise à des clients potentiellement intéressés par mes compétences. J'avais également envie de tenter l'expérience entrepreneuriale, qui est une expérience très stimulante et qui présente des challenges importants à relever.

Qu'est-ce qui a été le plus difficile au lancement ? Trouver le bon statut ? Démarrer réellement l'activité ? Comprendre comment fonctionne le monde du free-lance ?

Le plus difficile au début, c'est d'abord, la décision de quitter sa zone de confort. On se demande si on n'est pas en train de faire une grosse erreur. Sacrifier la sécurité de son emploi n'est pas une décision facile. Une fois la décision prise, on passe au concret. C'est à dire essayer de trouver la démarche à suivre pour créer son entreprise. J'ai commencé par acheter un bouquin « Se mettre à son compte... en 10 étapes. » Pour essayer de comprendre. J'ai vite compris que je ne pourrai pas faire la démarche de création d'entreprise tout seul, car elle me paraissait longue et fastidieuse.

Grâce à la recommandation d'un ami free-lance, j'ai contacté un expert comptable qui a essayé de m'expliquer la démarche et je lui ai confié la mission de création de mon entreprise.

Le choix du statut n'est pas toujours simple, j'avais commencé en SASU (Société par Actions Simplifiée Unipersonnelle), car j'avais quelques économies et je ne voulais pas me verser un salaire dans l'immédiat. Après avoir commencé mon activité, j'ai compris que pour mon cas, il serait plus avantageux de changer vers le statut EURL (Entreprise Unipersonnelle à Responsabilité Limitée), car elle a le taux d'imposition le plus avantageux surtout quand on décide de se verser un salaire.

Comment définis-tu le montant de ton TJM ou de ton forfait ? Quels points regardes-tu en priorité ? Fais-tu du forfait ?

Avant de définir son TJM, il faudrait déjà définir un positionnement. Il faudrait se poser les questions suivantes : quel poste devrais-je cibler (développeur sénior, architecte technique, technical leader, scrum master, expert sur une technologie....) ? Il faudrait aussi analyser l'état du marché : demander à des collègues, lire des rapports des plateformes de freelancing telles que Malt,...

Ainsi, en partant de l'état du marché et du niveau de ses compétences, on peut fixer un TJM qui soit cohérent. C'est à dire ni trop haut ni trop bas. De mon côté, ce que je regarde en premier ce sont mes compétences actuelles, ma marge de progression et l'intérêt que je porte pour une mission.

J'ai toujours travaillé en régie, mais je ne suis pas contre de tenter l'expérience du forfait du moment où le client potentiel est sensibilisé à l'importance de la qualité logicielle et qu'il ne soit pas uniquement intéressé par les Jours/Homme et les délais.

Est-ce qu'il t'arrive de travailler en sous-traitant d'un intégrateur/ESN ? Si oui, comment se déroule la relation généralement ?

Les free-lance en France n'ont pas toujours la possibilité de signer des contrats directement avec le client final. Généralement, il y a des sociétés référencées, qui font l'intermédiaire et qui prennent des commissions, qui peuvent aller jusqu'à 15 à 20 %. De mon côté, la collaboration se passe généralement bien, dans le respect du contrat et des règles de l'art dans le domaine. Le marché informatique est un marché où tout le monde finit par se connaître et se croiser. Donc, avoir une bonne réputation est très important. Un client qui ne paie pas sera rapidement connu et évité. Pareil pour les free-lance, si on fait du mauvais travail, notre réputation se trouve ternie.

Se faire plaisir en choisissant les projets, avoir des clients récurrents pour payer les factures et avoir un salaire, comment cela se passe au quotidien ?

Free-lance, c'est d'abord cette liberté. Liberté de choisir ses missions, diversifier ses interventions (développement, coaching, audit, formation...). Cela a un prix. Il est primordial d'essayer de rester compétitif sur le marché de l'emploi. L'investissement sur la formation continue est très important pour garder un niveau technique qui corresponde aux attentes des clients. Personnellement, je consacre un budget non négligeable pour me former, assister aux conférences, m'abonner aux magazines spécialisés et acheter les livres les plus intéressants.

Du point de vue financier, la crainte du free-lance, c'est un client qui ne paie pas. D'autre part, les factures sont généralement payées dans les 30 jusqu'à 45 jours après la date de leur envoi. Par conséquent, il est très important d'avoir des économies pour 2 à 3 mois, dans la mesure du possible, afin de pouvoir attendre sereinement le paiement des factures. Du point de vue administratif, il y a certaines choses à faire régulièrement. Par exemple, préparer sa note de frais mensuelle, payer sa TVA... Ces tâches ingrates peuvent être source de stress, mais on s'y habitue rapidement.

Travailler chez le client, dans un bureau partagé, chez soi, rencontrer d'autres développeurs, on a l'impression que le free-lance est solitaire, est-ce réellement le cas ?

Jusqu'à aujourd'hui, je n'ai pas eu cette impression d'être solitaire, car j'ai toujours travaillé en régie. Je me suis toujours considéré comme un salarié de l'entreprise. Je n'ai pas encore testé le mode forfait et le statut full remote. C'est peut-être avec ce mode de fonctionnement, on pourrait, probablement, sentir une certaine solitude. Pour moi, free-lance reste avant tout un statut juridique, il n'empêche pas, de nouer des contacts avec des collègues, de s'intégrer à des communautés techniques, de partager ses connaissances et de continuer à apprendre et affiner son expertise.

Est-ce que le Covid19 a eu un contact sur ton activité, et donc ton chiffre d'affaires ?

Malheureusement, à cause du Covid 19, mon client a arrêté les missions de tous les indépendants. C'est le risque majeur quand on est free-lance. Plus l'intercontrat dure, moins ce statut est intéressant financièrement. Je profite donc du confinement, pour affiner mon expertise sur plusieurs sujets et me former sur d'autres technologies que je n'avais pas le temps d'étudier quand j'étais en mission. J'espère que la crise est passagère et que l'activité va reprendre dès la fin du confinement.

Avec ton expérience, qu'est-ce que tu ferais différemment ?

Ce que je ferais différemment, c'est essayer de faire plus de formation, car j'aime bien partager mes connaissances. J'aimerais bien animer des sessions de formation, pour les développeurs sur des sujets qui me tiennent à cœur.

Quels conseils pourrais-tu donner ?

Ce que je conseille : s'assurer de la maîtrise des bases, passer le temps qu'il faut pour bien maîtriser les technologies et ne pas se contenter d'être un simple utilisateur. J'insiste sur ce point, car avec tous les « buzz words » qu'on entend un peu partout, on a tendance à oublier l'essentiel. Un bon free-lance, c'est quelqu'un, qui est certes bon techniquement, mais doit être aussi sensible à la qualité logicielle, car quand on vend une prestation, on a envie que le client soit content. Content parce qu'on livre dans les temps, mais surtout que la qualité soit au rendez-vous.

Thierry LERICHE
(Architecte et tech lead, @ThierryLeriche)

Le TJM n'est plus mon premier critère

J'ai 42 ans, marié, deux enfants, techlead et architecte logiciel sur les écosystèmes Java et JavaScript. J'ai étudié dans une école d'ingénieur, où j'ai également enseigné plusieurs matières pendant plus de dix ans. Je suis passionné de tech et j'aime partager cette passion. J'écris régulièrement des articles, notamment dans Programmez!, et pas seulement dans mes technos habituelles. Je suis aussi l'un des concepteurs de profil4.com, un site de référence qui permet d'établir son profil de communication et ceux des membres de son équipe.

Depuis combien de temps es-tu freelance ? Pourquoi avoir choisi le freelance ?
J'ai 20 ans d'expérience et je suis devenu freelance il y a tout juste dix ans. J'ai donc passé la moitié de ma carrière avec ce statut. Avant cela, j'ai un peu travaillé pour des start'ups. Mais j'ai surtout passé du temps dans des SSII/ESN qui m'envoyaient en régie chez leurs clients. J'ai eu la chance de ne pas travailler pour les pires SSIIs. Pour autant, en étant au quotidien chez les clients, je profitais finalement plus de la culture d'entreprise du client que de celle de mon véritable employeur.

Assister à des conférences ou même acheter un simple livre était compliqué. Et c'est sans parler des augmentations et primes promises mais jamais données… Pour parler chiffres, disons que le passage en freelance m'a permis de gagner en Net ce que j'avais en Brut jusque-là.

J'avais déjà eu plusieurs opportunités de passer freelance, comme on dit, mais j'avais toujours de bonnes raisons de ne pas franchir le pas. Et finalement, je l'ai fait juste après la naissance de ma première fille. Ce n'est pas le moment le plus évident. Mais quelque chose avait changé. En outre, j'avais anticipé pour me créer une petite réserve de trésorerie. La décision, longtemps reportée, s'est finalement prise en quelques heures, avec l'accord de ma femme.

Qu'est-ce qui a été le plus difficile au lancement ? Trouver le bon statut ? Démarrer réellement l'activité ? Comprendre comment fonctionne le monde du freelance ?
Il y a dix ans, dans le contexte du développement informatique en prestation, le plus intéressant était de créer une SARL/EURL. Le choix du statut était donc assez simple. Depuis, plusieurs lois de finance successives ont réellement changé la donne. Si je devais créer mon entreprise en 2020, j'opterais surement pour une SASU. Le système d'auto-entrepreneur n'est pas intéressant si on travaille à plein temps car on dépasse très rapidement les seuils.

J'ai passé beaucoup de temps pour trouver le nom de mon entreprise. Je voulais qu'il me corresponde et que le nom de domaine soit disponible. J'en rigole avec le recul. J'aurais pu la nommer ABCD que ça n'aurait pas fait une grande différence. Pour le choix de mon comptable (obligatoire), j'ai suivi les conseils de collègues indépendants. J'ai signé avec le cabinet EKO, pour lequel j'avais eu un très bon feeling, et dont les tarifs sont contenus. Je n'ai jamais changé. Dites que vous venez de ma part.

La partie facile a été de créer un compte pro. Je l'ai fait dans mon agence bancaire habituelle en quelques minutes. J'étais beaucoup plus en peine pour créer la société et écrire les statuts. Je suis à l'aise avec du code mais je me sens perdu dès qu'il y a de la paperasse. J'ai délégué cette tâche à mon comptable, le cabinet EKO. Après tout, il travaillait déjà avec d'autres freelances dans la même situation. Deux jours plus tard, tout était réglé.

Comment définis-tu le montant de ton TJM ou de ton forfait ? Quels points regardes-tu en priorité ? Fais-tu du forfait ?
C'est une question très difficile. Avec un TJM trop bas, tu tombes sur des missions souvent pourries, dans lesquelles ton travail ne sera pas reconnu. Mais avec un TJM trop haut, tu fermes la porte aux clients n'ayant pas le budget nécessaire. Il y a un juste milieu à déterminer au cas par cas. Au début, n'ayant pas encore une trésorerie conséquente, j'ai été moins regardant sur le TJM, et sur l'intérêt des missions. Mais je me suis constitué un petit capital sur le compte pro, que j'économisais (report à nouveau) et fur et à mesure des exercices. Cela m'a donné de la latitude dans le choix de mes missions et de mes tarifs.

Aujourd'hui, le TJM n'est plus mon critère principal. Je m'en sers principalement pour faire le tri. D'autant qu'il m'arrive régulièrement d'effectuer des prestations gratuites pour des clients choisis. Je me concentre sur le projet en lui-même. Est-il intéressant ? Vais-je me lever chaque matin en ayant du cœur à l'ouvrage ? Travailler avec l'équipe sera agréable ? Le poste en lui-même est un critère essentiel. Et quid du temps de transport ? Pas envie de traverser l'Ile-de-France deux fois par jour. Le télétravail est aussi un point que j'apprécie.

Je fais finalement assez peu de forfait, sauf quand je suis à 100% en télétravail, et encore… J'interviens principalement sur des gros projets, plusieurs en même temps, avec une démarche agile, en régie.

Est-ce qu'il t'arrive de travailler en sous-traitant d'un intégrateur / ESN ? Si oui, comment se déroule la relation généralement ?
Les ESN représentent 90% de mes clients. Elles sont quasiment obligatoires pour travailler avec les grands comptes. Elles se chargent de trouver les missions, assurent la logistique et t'aident sur le long terme. C'est une sorte de partenariat. La relation client n'est pas ma spécialité. Au passage les ESN prennent une commission d'environ 15%, ce qui me paraît normal.

Il faut noter que beaucoup de clients n'acceptent de travailler qu'avec des entreprises référencées, dont des ESN. Il faudra donc être un sous-traitant. D'autres clients, refusent à tort de travailler avec des freelances. Mais cela change.

Dans certains cas, il peut y avoir plusieurs niveaux de sous-traitance mais ça devient vite compliqué. Je fuis comme la peste tout ce qui ressemble à du portage salarial, car il y a eu trop d'arnaques.

Se faire plaisir en choisissant les projets, avoir des clients récurrents pour payer les factures et avoir un salaire, comment cela se passe au quotidien ?
C'est assez agréable, sauf quand un client se fait attendre et qu'il faut le relancer. Ça m'est déjà arrivé pour plusieurs factures. Je suis compréhensif. Mais quand ça se reproduit, avec de la mauvaise foi, je demande une partie des paiements en avance. Au quotidien, mon travail n'est pas si différent que lorsque j'étais salarié en SSII. Pour mes clients, je suis un prestataire comme les autres. Au mieux, cela les intrigue ou les impressionne.

Travailler chez le client, dans un bureau partagé, chez soi, rencontrer d'autres développeurs, on a l'impression que le freelance est solitaire, est-ce réellement le cas ?
Oui et non. Quand tu es en full remote, ça peut devenir compliqué. C'est super important de mettre en place des rituels. Par exemple tu peux programmer des machines à café virtuelles tous les jours. L'idée est surtout de discuter avec ses collègues, de préférence en vidéo et pas juste en audio. Il y a un espace de cooworking dans ma ville (en banlieue parisienne) où je peux croiser régulièrement d'autres freelances. Ça permet de voir du monde au lieu de rester cloitré.

Dans mon cas, le gros de mes interventions s'effectue en régie chez le client. Et au quotidien, ce n'est pas très différent d'un statut de salarié classique, mis à part qu'on n'est pas invité à tous les événements. Dans tous les cas, on reste un prestataire, avec les avantages et les inconvénients.

Est-ce que le Covid19 a eu un impact sur ton activité ?
Ce virus a réellement changé la donne. Il y aura un avant et un après, et pas seulement pour les freelances. Le télétravail est certainement un des aspects qui évoluera le plus. Le terme de « coronaviré » est apparu pendant la crise. Il désigne toutes les personnes ayant été remerciées purement et simplement. Dans le petit monde du développement, cela aidera aussi à se faire une idée de certaines SSII… De nombreux projets et missions ont été reportés, mis en pause, ou carrément annulés. J'en ai moi-même fait les frais. Mais ça devrait reprendre doucement à la fin du confinement.

Durant cette période, je travaille sur plusieurs projets à titre bénévole, j'écris des articles, je me forme sur de nouvelles technos, etc. Je profite de la vie de famille. Ce n'est donc pas du temps perdu, même si c'est long.

L'impact sur le CA est évident. Mes revenus liés aux prestations sont tombés à zéro du jour au lendemain. Dans mon cas plus spécifique, j'édite plusieurs produits qui me permettent déjà de vivre confortablement et dont l'activité, bien que ralentie, n'a pas de raison de s'arrêter durant la crise.

Avec ton expérience, qu'est-ce que tu ferais différemment ?
Toujours facile de revoir sa copie après coup. Il y a certains clients et équipes avec lesquels je refuserais de travailler, soit parce que les projets n'étaient finalement pas ceux décrits, soit parce que je n'ai pas adhéré à la culture de l'équipe, soit parce que ça a été complexe financièrement. J'ai pendant longtemps eu du mal à prendre des vacances ou m'offrir des petits plaisirs. J'ai souvent fait passer ma vie privée/familiale au second plan. J'ai dépensé beaucoup d'énergie pour des projets/équipes qui n'en avaient pas besoin. Avec le recul, je réalise que j'aurais pu m'organiser autrement.

Je crois que ça ne sert à rien de revenir là-dessus, mis à part pour ne plus faire les mêmes erreurs. Quand tu deviens freelance, tu deviens entrepreneur par la même occasion. Il faut aller de l'avant, en restant fidèle à ses principes, si possible…

Quels conseils pourrais-tu donner ?
Il faut se faire accompagner dès le départ. Le choix du cabinet comptable est crucial. Certaines démarches administratives (urssaf, rsi, mutuelle, etc.) m'ont parfois poussé au bord des larmes. Il existe de nombreuses aides à la création d'entreprises et il ne faut pas hésiter à en profiter. Vous y avez droit. Il faut en discuter discrètement avec les collègues ayant déjà franchi le pas. Chaque cas sera particulier.

Le choix de devenir indépendant n'est pas sans conséquence. Il y a des avantages et des inconvénients. Il ne faut surtout pas le faire pour le fric. Ce serait sûrement la pire des raisons. Une grosse pression va peser sur vos épaules et, par extension, sur votre foyer. La vie ne sera pas rose. Vous aurez des libertés et des facilités. Mais vous aurez aussi des obligations et des contraintes. Et c'est vous qui porterez les risques, notamment en cas d'inter-contrat.

La carrière d'un salarié classique, d'un prestataire dans une bonne SSII, ou d'un freelance ne va pas évoluer de la même manière ou dans le même sens. Il faut en tenir compte.

Emilien Pecoul : « il faut savoir se vendre ! »

Emilien est développeur indépendant depuis 2013. Il code plutôt sur les environnements .Net. Sur son temps libre, il travaille sur son projet de startup.

Depuis combien de temps es-tu freelance ? Pourquoi avoir choisi le freelance ?
Après cinq ans dans le milieu des startups et PME, je suis devenu freelance, principalement pour porter les bonnes pratiques et ce que je voulais développer. Je n'ai pas beaucoup apprécié les pratiques ni la qualité logicielle que je voyais. Durant différents meetups, j'ai vu des freelances et j'ai discuté avec eux.

La démarche n'est pas toujours facile. Ce fut ton cas ?
Oui et non. J'ai rapidement trouvé un comptable qui m'a aidé dans les démarches, les statuts. La création du compte pro a été le plus facile. Mais se lancer ce n'est pas facile. Car je n'avais pas de chômage ni réellement d'économies. Mais j'ai franchi le pas. J'ai fait un twitt et rapidement j'ai trouvé ma première mission. Pour le statut, j'ai pris l'EURL. En 2013, le choix n'était pas aussi large qu'aujourd'hui.

Etre indépendant, c'est parler de TJM, de forfait, de régie, de sous-traitance. Comment gères-tu ?
Soyons direct. Aujourd'hui, je propose des conditions tarifaires qui me conviennent. Par rapport à mes anciens jobs, je suis environ à +50 % et surtout je fais essentiellement du télétravail, avec en moyenne 4 jours par semaine.
Personnellement, je suis plutôt en TJM, j'ai toujours eu du mal avec le forfait. Je m'adapte selon la mission et l'expertise demandée.

J'ai la possibilité de faire le tri dans les missions. Si un projet est vraiment intéressant, je peux m'adapter. L'importance de l'expertise est un facteur important dans le TJM. Je juge aussi bien le projet, que les technologies et l'ambiance. Je définis mon « prix » en discutant avec d'autres freelances et me mettant volontairement dans une fourchette haute. Pour éviter les entreprises qui cherchent uniquement des pisseurs de code.

> « *Ne vous bradez pas ! Formez-vous en permanence ! Ne travaillez pas à plein temps.* »

Tu fais de la sous-traitance pour des prestataires et des ESN ?
Oui comme beaucoup de freelances, nous n'avons pas toujours le choix. Globalement, je suis à 50-50. Souvent, notamment dans les grandes entreprises, il peut y avoir un problème de référencement. Mais dans tous les cas j'ai la possibilité de défendre mes valeurs et de les assumer, de vraiment apporter de la valeur et de l'expertise à mes clients, je me sens beaucoup plus épanoui.

Est-ce que tu prends des missions purement alimentaires ?
Par chance, jusqu'à présent, j'ai pu éviter. Mais il faut avouer que la crise du Covid-19 rend la situation tendue. Mais j'ai toujours refusé de casser les prix. Cela ne m'intéresse pas.

Qu'est-ce qui est le dur pour toi : gérer le quotidien, te vendre ?
Le côté administratif n'est pas forcément mon truc. En qualité de freelance, je suis une entreprise même si elle comprend qu'une personne, moi. Mais les procédures ne sont pas toujours très adaptées. C'est un univers que les freelances ne connaissent pas réellement. L'accompagnement est essentiel.
Après, il faut savoir se vendre, se mettre en avant. Il faut faire de la prospection, être visible, soigner son site web, les réseaux sociaux. Bref : il faut faire du marketing si on veut éviter les intermédiaires.

Freelance rime-t-il avec solitude ?
Non, la solitude c'est forcément un choix. Si on participe à des meetups, aux conférences de développeurs, et qu'on est actif dans les communautés de freelances, on ne se sent pas du tout seul. Au contraire on développe des relations beaucoup plus durables et choisies.

Des freelances évoquent des ruptures de contrats, le report de projets. Quel est l'impact du Covid-19 sur ton activité ?
C'est une réalité. J'ai eu des missions qui se sont arrêtées. Mais cela fait parti du jeu. C'est le risque quand on est indépendant. Il faut être prudent et mettre de l'argent de côté. Mais comme dans toute crise : il y a des entreprises qui chutent et d'autres qui prospèrent. J'ai eu la chance de retrouver très rapidement une nouvelle mission qui débutera seulement en juin, pour pouvoir passer plus sereinement le confinement.

La startup d'Emilien est là pour aider le freelance, notamment sur la gestion avec les administrations : https://www.kickbanking.com

Emilien a également écrit un livre pour aider les développeurs à se lancer en freelance ou améliorer la gestion de leur structure existante : https://www.thebookedition.com/fr/code-freelance-p-368891.html

Vous pouvez retrouver Emilien sur twitter : https://twitter.com/Ouarzy

Anaïs Payet

Être honnête et transparente dans mes propositions

*Anaïs Payet est développeuse freelance d'applications mobiles iOS et Android. Elle a passé un DUT en informatique, suivi d'un diplôme d'ingénieur informatique obtenu en 2015. Elle est des Duchess et co-organisatrice de Mix-It la conférence Lyonnaise avec des crêpes et du coeur. Son twitter: @Sianay_*

Comment et pourquoi es-tu devenue freelance ?
Dès la fin de mes études, j'ai eu cette volonté d'être indépendante pour pouvoir organiser mon temps et être libre de choisir les projets sur lesquels je voudrais travailler. J'ai eu une première expérience en 2015, mon premier client était tout simplement mon ancien employeur où je venais d'effectuer mon alternance, ensuite cela m'a permis d'enchaîner d'autres petites missions. Début 2016, j'ai eu une opportunité à l'étranger, j'ai donc arrêté le freelance. Après deux ans et demi passés à travailler et voyager à l'étranger, je ne me voyais pas reprendre un CDI en rentrant en France. J'ai donc cherché directement une mission de quelques mois et je l'ai décrochée très vite. Ainsi depuis mi 2018, je continue mon parcours en freelance.

Les développeuses sont encore assez peu nombreuses. Quelle est la situation chez les freelances selon ton expérience ?
Je dirais malheureusement que la situation chez les freelances (je parle bien entendu pour mon domaine) est encore plus inégalitaire au niveau de la parité… Ce qui est logique dans un sens car il y a déjà peu de développeuses sur le marché donc la proportion de celles qui se mettent à leur compte est encore plus rare. Je ne dis pas non plus qu'il n'y a pas de développeuses freelances, mais celles qui sont à leur compte sont, à mon sens, le plus souvent très expérimentées alors qu'on retrouverait plus de débutants/intermédiaires du côté masculin.

Freelance c'est créer une entité juridique, gérer l'administration, trouver des clients / missions, c'est quoi ton quotidien ?
J'ai un peu une aversion pour le côté administratif… Même si on ne peut pas totalement y échapper, il ne faut pas hésiter à prendre un comptable pour s'alléger un peu de cette tâche, ce que je me suis empressée de faire. Je ne passe quasiment pas de temps à chercher des clients ou nouvelles missions, les plateformes et le réseau font le job à ma place, je ne consacre donc que du temps à répondre aux demandes et faire des estimations. Ainsi la majorité de mon quotidien est consacré à mes missions avec mes clients (c'est ce que l'on souhaite ^ ^).

TJM, forfait, sous-traitance : ce n'est pas toujours simple pour un freelance. Travailles-tu plutôt en sous-traitance pour une ESN ? Quand tu dois définir un TJM ou éventuellement un forfait, comment définir un tarif ?
Je n'ai jamais eu à passer par une ESN pour le moment, je décroche toujours mes missions directement avec le client. Il m'arrive d'utiliser des plateformes comme malt.fr mais la plupart du temps cela se fait via mon réseau. Je travaille au TJM, un tarif que j'aligne plus ou moins sur mes collègues freelances. J'évite au maximum le travail au forfait, que je trouve très peu adapté à la réalité des projets informatiques. Dans ces cas-là, on essaye de convenir d'une solution intermédiaire avec le porteur de projet quand c'est possible.

Mission purement alimentaire ou mission pour se faire plaisir, comment assurer son salaire et se faire plaisir ?
Dans mon cas, je dirais que le plus difficile c'est de savoir dire non : savoir refuser une mission qui ne correspond pas vraiment à nos valeurs avec la peur de ne rien retrouver derrière si on la refuse. Quand on peut se le permettre, c'est un vrai luxe. J'essaie vraiment d'orienter mes missions et mon image vers des secteurs qui me tiennent à cœur, par exemple l'éducation, le social, l'environnement. Avec le temps et l'enchaînement de missions dans ces domaines, on acquiert de plus en plus de crédibilité, d'expérience et ainsi on augmente ses chances de décrocher d'autres missions qui correspondent donc à ces valeurs qui nous sont chères.

Penses-tu que la formation est un élément clé quand on est freelance ?
Oui, je pense sincèrement qu'il est très important de se former et de continuer à se former quand on est freelance et cela de diverses façons : cours en ligne, conférences, livres, etc. Le client compte sur nous en tant qu'expert de notre domaine et il est important de savoir le guider, de l'orienter vers les bons choix. Il y a l'expérience bien sûr mais cela ne fait pas tout, il ne faut pas se reposer uniquement sur ses acquis, aussi on ne doit pas avoir peur de s'aventurer sur de nouveaux terrains.

L'autre aspect important c'est aussi savoir se vendre, soigner son marketing. Quelle est ton approche ?
Je pense que j'ai encore beaucoup à apprendre dans ce domaine, j'ai une nature introvertie donc cela ne facilite en rien la tâche. Pour le coup, il faut essayer différentes approches pour pouvoir comparer ce qui fonctionne et ce qui fonctionne moins et apprendre de ses erreurs. Il ne faut pas hésiter à demander du feedback quand les choses n'aboutissent pas avec un client. Je ne dirais pas que je "sais" me vendre mais je mise sur le fait d'être honnête et transparente dans mes propositions et pour l'instant cela me réussit.

Covid-19, confinement, nous avons vu l'impact de la crise sur l'économie, l'arrêt de nombreux projets IT. Qu'en est-il pour toi ?
J'ai plutôt de la chance de mon côté, mon activité n'a pas été trop impactée. Je travaille majoritairement avec des clients dans le secteur de l'éducation et le numérique a eu à jouer un rôle important durant cette crise. Je pense que si j'avais eu besoin de rechercher de nouveaux clients cela aurait été compliqué dans le contexte actuel.

Aymeric Weinbach
Nomade du conseil en architecture logicielle dans les nuages - **ZeCloud**

Les meilleurs lieux pour poser son PC et travailler dans Paris.

Aujourd'hui si vous êtes développeur vous pouvez travailler de n'importe où, il vous suffit d'un pc portable d'un bon wifi et le plus important d'un bon café. Vous pouvez travailler aux quatre coins du monde, mais déjà voilà ma sélection pour travailler aux quatre coins de Paris. Toutes ces adresses ont été testées et approuvées par moi.

Avertissement : La crise du Covid19 fait que la plupart de ces lieux sont fermés pendant que j'écris cet article et je ne sais pas si tous rouvriront dans les mêmes conditions.

Mes critères de sélection sont un bon café (si le café n'est pas bon je n'y retournerai pas), un bon wifi, ou, s'il n'y a pas de wifi, il faut qu'on y capte bien la 4G et un endroit plutôt calme. Je pratique régulièrement les lieux de coworking, mais j'ai peu d'habitudes. Je change régulièrement et l'endroit où je me pose dépend de ma journée. C'est pratique de se poser entre 2 rendez-vous parisiens par exemple. Si j'ai des rendez-vous clients à 11h et à 16h, je ne vais pas faire une heure de transport pour rentrer chez moi. Ou si j'ai un meetup prévu le soir, ça me permet de me poser dans le quartier quelques heures avant.

Les coworking

La formule, généralement appliquée, des coworking est de payer au mois un abonnement l'équivalent d'un loyer. C'est ce qui se rapproche le plus d'un bureau traditionnel mis à part qu'au lieu de prendre un bail, vous prenez un abonnement et en fonction de la formule, vous pouvez arrêter quand vous voulez. J'ai peu pratiqué cette formule, parce que j'ai un appartement avec un vrai bureau et un bon espace de travail, mais elle peut être intéressante si vous devez passer plusieurs mois dans une ville ou si vous n'avez pas un vrai bureau. Par contre c'est la formule qui nécessite le plus d'organisation : vous devez prendre un abonnement et réserver votre espace pour votre poste de travail à l'avance sur internet et vous avez la possibilité de domicilier votre courrier. Le plus connu c'est WeWork dans Paris. Ils ont 19 lieux et sont présents un peu partout dans le monde. Les espaces de travail sont assez luxueux et plutôt bien aménagés. Le café y est souvent bon. Je n'ai testé que 2 WeWork : Neuilly et Opéra (Paris).

Les cafés coworking

La formule est simple. Et vous aurez un large choix. Vous payez à l'heure ou à la journée, et vous pouvez consommer autant de cafés/boissons chaudes et de snack que vous voulez. L'aménagement des lieux est entre le café et le coworking. C'est plus chaleureux que des bureaux traditionnels et plus calme qu'un café. Vous n'avez pas besoin de réserver à l'avance, vous arrivez et repartez quand vous voulez et dans certains cas vous pouvez même demander des salles de réunion avec suppléments. Les prix sont en moyenne de 5 € par heure.

Hubsy : un bon café, toute sorte de boissons chaudes diverses et variées, pensez à demander la boisson surprise. Près d'Arts et métiers, 41 Rue Réaumur (possibilités d'avoir des salles de réunions). Près de République, 9bis Rue Lucien Sampaix (possibilités d'avoir des salles de réunions)

Anti Café (14 lieux à Paris).

Anti Café Beaubourg 79 Rue Quincampoix. Anti Café Louvre, 10 Rue de Richelieu (salle de réunion au rez-de-chaussée).

Nuage Café 14 Rue des Carmes.

Le 10h10 Près d' Arts et métiers 10h10 Beaubourg : 210 Rue Saint-Martin. Près du Sentier 10h10 Clery Sentier : 19 Rue de Cléry.

Les incubateurs

Station F : Station F est le plus gros incubateur de Paris. 30 programmes de Startups permettent d'accéder à un bureau. Même si vous ne faites pas partie d'un des programmes startups qui vous permettent d'y accéder, vous serez sûrement amené à y venir pour rencontrer une startup ou pour un meetup.

Il y a aussi deux endroits où vous pouvez vous y poser pour travailler : l'Anti café Station F (5 € de l'heure avec du bon café et des pâtisseries). La Félicita : un immense espace restaurant qui est friendly laptop. Vous pouvez vous y poser avec votre portable pour y travailler la journée avant un meetup et vous y restaurer.

Morning coworking & Numa 39 rue du Caire, Paris.

Historiquement Numa était « la cantine », un coworking café. Quand l'adresse était gérée par numa on pouvait s'installer au rez-de-chaussée qui était un café cantine et on pouvait s'y installer sans réserver. N'y étant pas retourné depuis la reprise par morning coworking, je ne peux vous confirmer que c'est toujours le cas, mais vous pouvez réserver des salles en passant par le site.

Les (vrais) cafés

La formule est simple ce sont de vrais cafés, il vous suffit de prendre un café et de vous poser quelques heures avec votre laptop. Ceux de cette sélection sont laptop friendly les jours de semaine, mais pensez à consommer raisonnablement chez eux et à ne pas trop occuper l'espace aux heures de pointe.

Le café Lomi. 3ter rue Marcadet — 75018 Paris.

Du bon café, des pâtisseries, une ambiance cosy et du wifi, tout ce qu'il vous faut pour écrire du code ou finir cet article pour Programmez!.

Mel Mich et Martin 8 Rue Saint-Bernard.

Près de la place de la Nation, Ambiance chaleureuse dans ce café/restaurant/brocante avec une grande terrasse.

Hexagone Café 121 rue du Château.

Près de la gare Montparnasse, un très bon café pour se poser avec son laptop dans le coin de l'hexagone sur la rue. Il vous permettra de trouver certainement l'inspiration pour votre nouvelle appli.

Quelques autres lieux

Le pavillon des canaux 39 quai de la Loire, 75019 Paris.

Tiers lieu au bord du bassin de la Villette, vous pouvez vous y poser avec votre laptop en terrasse au bord du canal de l'Ourcq ou dans cette maison transformée en café/restaurant/coworking et participer à un des évènements culturels qui y sont organisés.

La Recyclerie 83 Boulevard Ornano, 75018 Paris.

Ancienne gare de la petite ceinture. Tiers lieu consacré à l'économie circulaire et à l'écoresponsabilité, vous pouvez aussi vous y poser avec votre laptop pendant quelques heures, manger boire et participer à un atelier ou une conférence, ou visiter la ferme urbaine et son système d'aquaponie.

Les bons plans

Pour les étudiants ou si vous démarrez une activité, vous n'avez peut-être pas ou peu d'argent à investir dans un coworking.

Les bibliothèques de la BNF

La Gaieté Lyrique 3bis Rue Papin.

Assez calme avant 16/17h. Le lieu dispose d'un coin bibliothèque/café et dans le centre de Paris où vous pouvez vous poser à toutes heures et gratuitement avec votre laptop. Et vous pouvez prendre un café ou une bière avant un meetup, ou aller voir leurs expos, puisque c'est aussi un des centres d'arts numériques et de musique moderne de la ville de Paris.

Pour les versaillais

The Stray Bean coffee 6 Rue Royale.

Un vrai et bon café, une ambiance cosy, une petite terrasse à côté de la gare rive gauche.

Pouce Café 32 Rue d'Anjou.

Un café coworking à 2 pas de la gare rive gauche avec du bon café et des salles de réunion.

Red Hat OpenShift de A à Z

Sébastien Blanc
Sébastien est directeur de l'expérience développeur chez Red Hat. Il se considère comme un Passion-Driven-Developer avec comme objectif principal de partager au maximum sa passion en donnant des conférences ou bien encore en écrivant des articles. Avant d'être Developer Advocate, Sébastien a passé 15 ans dans l'ingénierie logicielle, essentiellement orienté autour de Java EE. Durant son temps libre, il aime apprendre le code aux enfants et pratique le Kobudo, un vieil Art Martial japonais.
@sebi2706

Jaafar Chraibi
Jaafar est Principal Technical Marketing Manager chez Red Hat spécialisé sur OpenShift et publie du contenu technique (articles, conférences) pour faciliter l'adoption de cette plateforme. Il a travaillé de nombreuses années en tant qu'Architecte spécialiste d'OpenShift et Middleware et accompagné de grands clients français dans leur adoption des technologies de conteneurs dans le secteur bancaire, aéronautique, industriel et télécoms. Il est passionné de nouvelles technologies, toujours curieux d'apprendre, et est féru de guitare à ses heures perdues.@ChraibiJc

Laurent Broudoux
Laurent est Architecte Solution chez Red Hat France, où il est spécialisé sur le développement des applications cloud-natives et la transformation des applications existantes. Avant de rejoindre Red Hat en 2016, il a été Architecte SOA pendant plus de 10 ans dans les Services Financiers où il a défini des stratégies de transformation via l'usage des API et de l'Intégration. Durant son temps libre, il aime s'occuper de sa famille et jardiner. Il est aussi le lead developer du projet open source Microcks.io : une plateforme permettant de simuler et tester vos API.
@lbroudoux

Qu'est-ce qu'OpenShift Container Platform ?

Vous avez certainement entendu parler de conteneurs et de Kubernetes, et vous vous demandez ce qu'est OpenShift. Nous espérons que cet article vous aidera à comprendre ce que cette plateforme apporte aux développeurs et comment elle permet d'héberger vos applications critiques en production, comme le font aujourd'hui plus de 2000 clients. À travers un ensemble d'exercices pratiques que nous vous avons spécialement concoctés, vous pourrez aussi vous familiariser avec OpenShift de manière très pratique. Bonne lecture !

OpenShift Container Platform est une distribution Kubernetes certifiée par la Cloud Native Computing Foundation. Ce qui la distingue principalement des autres distributions ce sont ces différents aspects :

- OpenShift offre aux développeurs un large catalogue de services prêts à l'emploi, supportant les technologies les plus évoluées du moment telles que les Opérateurs Kubernetes, ou encore les charts Helm pour simplifier le déploiement d'applications et de services.
- OpenShift est une plateforme « clés en main » architecturée par Red Hat pour être « Enterprise-Ready », c'est-à-dire une plateforme fiable et robuste pour héberger des centaines d'applications en production (certains clients ont plus de 25 000 conteneurs actifs par jour sur OpenShift), fournissant nativement des services de production tels que le monitoring, la gestion de logs centralisés, la sécurité et l'authentification, etc.
- OpenShift est 100% open source, avec notamment un projet communautaire upstream accessible à tous (https://www.okd.io/) et une communauté ouverte d'utilisateurs et contributeurs de plus de 540 entreprises, appelée OpenShift Commons (https://commons.openshift.org/participants.html) et qui ne cesse de grandir chaque année.
- OpenShift se déploie de la même manière aussi bien dans vos datacenters (« on-pre-

mises ») ou chez les différents Cloud Providers, dont certains la proposent en service managé tels qu'IBM, Azure, Google ou AWS.

- Red Hat fournit et maintient à jour un « registry » de conteneurs permettant de bénéficier des mises à jour de failles de sécurité et de maintenance, plutôt que d'avoir à gérer cela par vous même.

Le diagramme 1 donne un aperçu des composants disponibles en standard avec OpenShift.

L'EXPÉRIENCE DÉVELOPPEUR ET LE CATALOGUE PaaS

OpenShift existe depuis 2011 et a toujours proposé des services à destination des développeurs sous forme de PaaS (Platform-as-a-Service). En effet, Red Hat offre en standard des images de conteneurs permettant de développer sur OpenShift en Java, Node.js, PHP, Ruby, Python, Go, etc.

Un développeur peut très rapidement déployer du code existant depuis un référentiel Git (processus source-to-image) ou des applications déjà buildées depuis les binaires, soit depuis une ligne de commande, soit depuis l'interface graphique intuitive et facilitant la prise en main pour des néophytes.

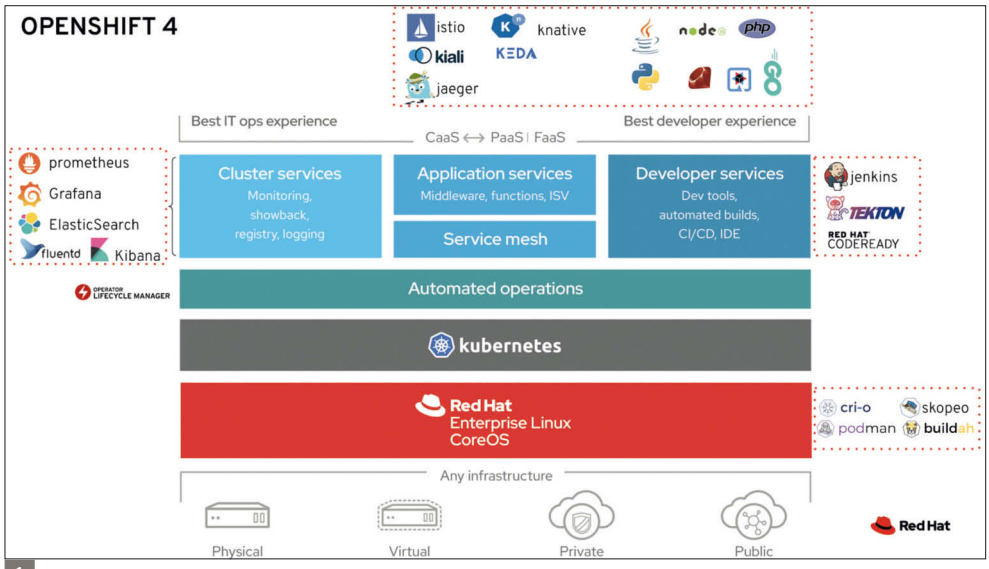
La « Developer Console » offre également une interface graphique très utile pour visualiser l'architecture de vos applications conteneurisées, leurs composants et inspecter les caractéristiques telles que les métriques, variables d'environnements, etc.

Codeready Workspaces, l'environnement intégré de développement « IDE-as-a-service », est accessible depuis un navigateur web et offre des capacités similaires à un véritable IDE traditionnel (complétion de code, analyse de code, debug, etc.). Il permet de reproduire très rapidement tout l'environnement d'une application en containers sous forme de « workspaces », facilitant par exemple les revues de code ou les patchs applicatifs d'une version donnée : (https://developers.redhat.com/products/codeready-workspaces/overview) 2

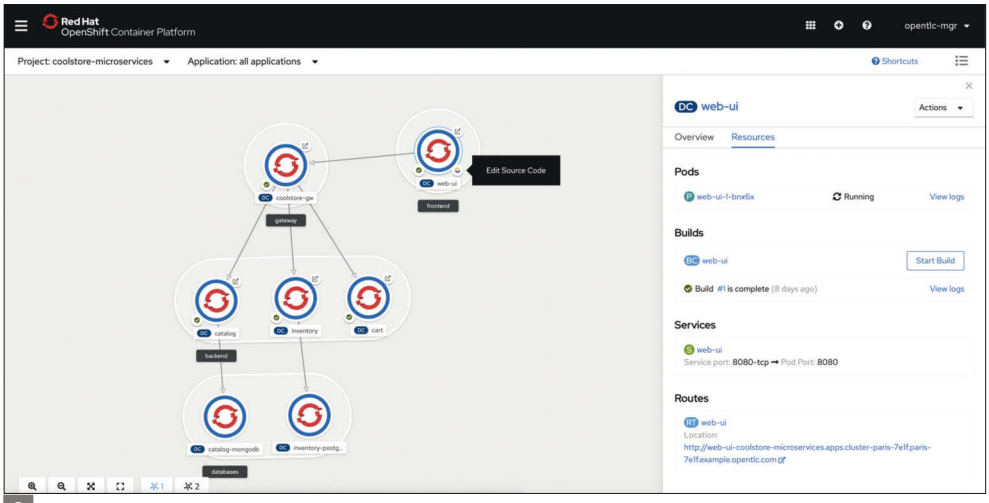
OPERATORHUB, LA MARKETPLACE POUR LES DÉVELOPPEURS

OpenShift 4 intègre de manière extensive la notion d'"Opérateurs" Kubernetes, dont Red Hat (CoreOS) est à l'origine, et qui connaît désormais un franc succès auprès des éditeurs de solutions en conteneurs.

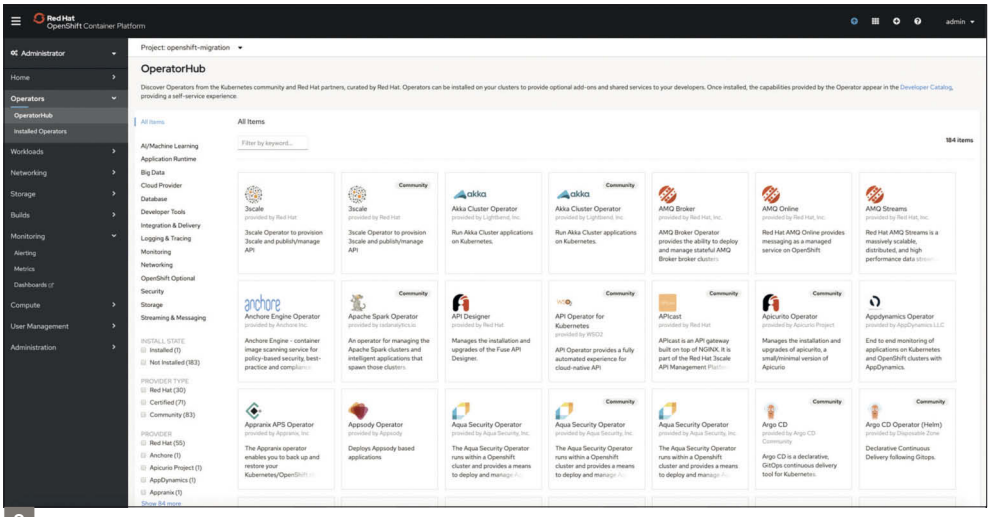
Le principe d'un « Opérateur » est simple : on peut le voir comme un « super-exploitant » ayant une expertise dans l'exploitation d'une solution donnée pour en réaliser l'installation, les mises à jour, la



1 *OpenShift 4 est la version actuelle de la plateforme, et intègre un certain nombre d'innovations décrites ci-dessous, suivant de le schéma ci-dessus de haut en bas.*



2 *La « Developer Console » facilitant l'interaction avec les composants conteneurisés de votre application*



3 *Plus d'infos :* https://marketplace.redhat.com/en-us et https://operatorhub.io/

résilience en cas de panne, sauf que tout ce savoir est automatisé dans un conteneur qui gère le cycle de vie de la solution donnée. Il existe aujourd'hui plus de 300 « Opérateurs » dans la marketplace OpenShift couvrant des domaines très divers tels que : l'intelligence artificielle, la sécurité, les bases de données, la gestion de logs, le stockage de données, les serveurs d'applications…
Ces « Opérateurs » apportent aux développeurs la simplicité du cloud (one-click install) tout en intégrant les meilleures pratiques d'exploitation de la solution déployée, donnant plus d'autonomie aux développeurs pour provisionner des services. 3

APPLICATION SERVICES

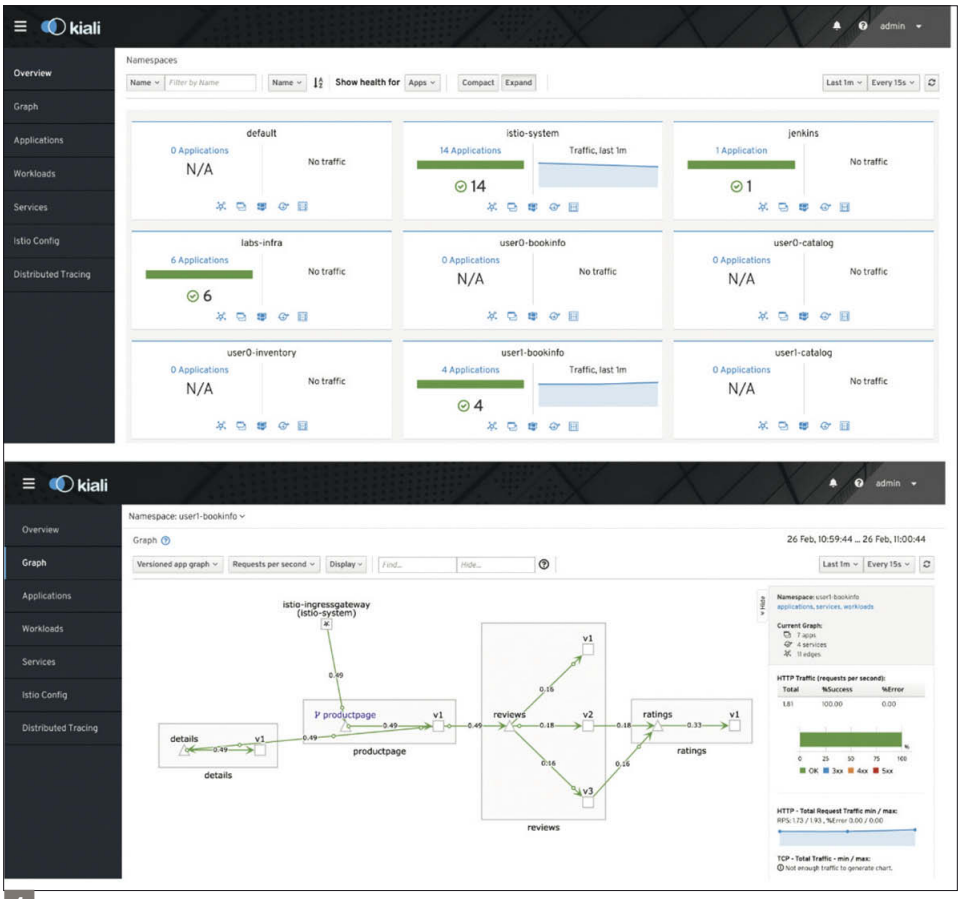
Au-delà des +300 services disponibles dans le catalogue OpenShift, fournis par Red Hat et des éditeurs tiers qui ont certifié leurs solutions pour OpenShift, il existe des services d' « infrastructure » natifs à destination des développeurs, tels qu'OpenShift Service Mesh (basé sur istio) favorisant l'adoption d'architectures en microservices, ou encore Kiali qui est une interface graphique de monitoring des microservices (visualisation du trafic, nombre de requêtes…) ou bien Jaeger permettant de faire du tracing distribué (standard OpenTracing) pour suivre la propagation d'une requête dans un service mesh. OpenShift permet aussi de faire nativement du « Serverless » avec Knative, nous aurons d'ailleurs l'occasion de créer un composant serverless en partie 4. 4

CLUSTER SERVICES

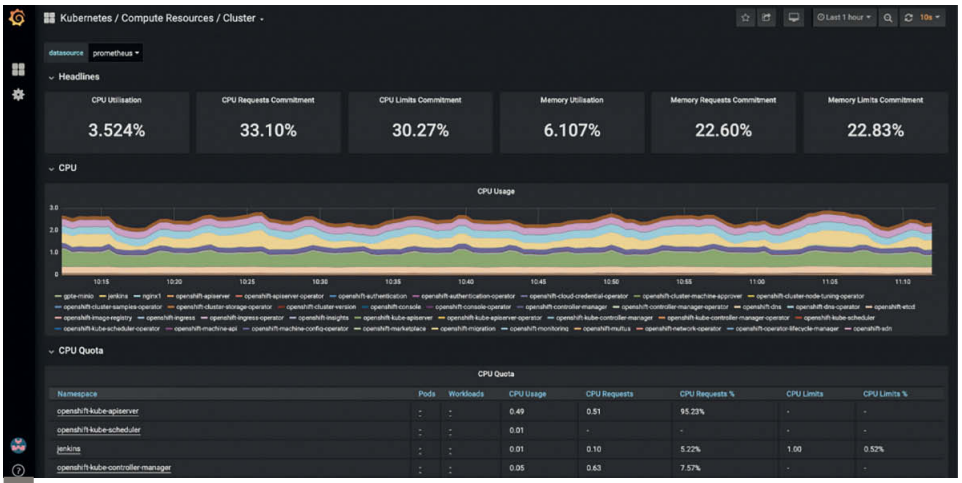
OpenShift fournit en standard un ensemble de services indispensables pour avoir une plateforme de production : monitoring basé sur Prometheus outil Open Source de collecte de métriques, et Grafana, outil Open Source de création de tableaux de bords de monitoring, mais aussi une stack ElasticSearch, Fluend et Kibana (EFK) permettant d'avoir des logs centralisés. Enfin, il existe des services techniques de type serveur d'authentification natif, facilitant l'intégration avec des annuaires d'identité d'entreprise. 5

OPEN HYBRID CLOUD ET MACHINES VIRTUELLES EN NATIF DANS OPENSHIFT

Afin de faciliter la cohabitation du monde « legacy » de machines virtuelles et celui d'applications



4 *La console intégrée OpenShift Service Mesh (Istio, Kiali, Jaeger) facilite le monitoring des microservices*



5

conteneurisées, OpenShift permet nativement de déployer (ou d'importer) des machines virtuelles classiques et de les faire tourner directement à l'intérieur de la plateforme : l'intérêt majeur est que les conteneurs et les machines virtuelles sont tous deux gérés par Kubernetes/OpenShift, y compris leur réseau et le stockage. Cela rompt avec les difficultés classiques où l'on a du mal à migrer une application, car ses dépendances ne peuvent être portées facilement vers un autre environnement, tel que le Cloud par exemple.

Plus d'infos : https://www.openshift.com/blog/blog-openshift-virtualization-whats-new-with-virtualization-from-red-hat
Tous ces services expliquent pourquoi OpenShift est la plateforme Kubernetes et PaaS de référence du marché, comme peuvent en témoigner les nombreux clients l'utilisant en production pour des services critiques et des cas d'usages variés : https://www.openshift.com/learn/success-stories/
Il est temps désormais de passer à la pratique, en réalisant les différents exercices proposés dans les parties suivantes !

Ma 1ere application

La 1re étape consiste à provisionner un environnement OpenShift pour pouvoir effectuer les exercices. Pour cela nous allons utiliser un service gratuit destiné à l'apprentissage d'OpenShift à cette adresse : https://learn.openshift.com/playgrounds/openshift42/. L'application exemple sera détaillée juste après, et le référentiel de l'application qui sera utilisé est: https://github.com/redhat-france-sa/openshift-by-example

Démarrer l'environnement

Il faut cliquer sur "Start Scenario" et suivre les étapes. Rappelez-vous simplement qu'il y aura 2 types de login : soit "**developer**", soit "**admin**", et les mots de passe seront présentés dans l'environnement.

Attention : cet environnement sera provisionné à la demande. Il est également éphémère. Il sera détruit dès que vous fermerez la fenêtre du navigateur.

Ainsi, il est conseillé de faire chaque partie de bout en bout, mais nous vous **fournissons des scripts qui permettent de se mettre à niveau pour démarrer chaque partie séparément dans le répertoire "shortcuts" du référentiel Git**. Par exemple, si vous souhaitez reprendre directement à la partie 3, il vous suffira d'utiliser le script "complete-part2.sh".

Présentation de l'application

Pour vos premiers pas sur OpenShift, nous avons souhaité une application simple, mais suffisamment représentative d'une petite application que vous pourriez avoir à déployer en entreprise. Notre choix s'est alors porté sur le trio de technologies suivantes :

- Une interface web portée par un développement réalisé en Angular,
- Des services ou API portés par un composant Java développé à l'aide de Spring Boot,
- Une base de données NoSQL orientée Documents telle que MongoDB.

Ces composants communiquent sur les ports HTTP et TCP par défaut utilisés par ces frameworks ou technologies. 1

La thématique de cette application ? Le recensement de vos stocks de fruits ! L'été arrivant, il est l'heure de faire le plein de vitamines ;-) Notre composant Spring Boot portera donc de simples API permettant de créer et récupérer des stocks de différents fruits, persistés dans la base MongoDB.

L'ensemble des sources des composants de l'application ainsi que les ressources utilisées dans ce dossier sont sur le GitHub : https://github.com/redhat-france-sa/openshift-by-example

Vision conteneurisée sur Kubernetes

Au démarrage de notre dossier, cette application ne contient aucune spécificité liée aux conteneurs et à la plateforme Kubernetes. Nous allons réaliser cette conteneurisation ensemble pas-à-pas en utilisant les différents outils mis à disposition par OpenShift. Néanmoins afin de vous donner une vision finale de la cible, voici un schéma de l'architecture technique une fois l'application déployée sur notre cluster : 2

Nous construirons et déploierons ensemble au moins 3 conteneurs (1 pour chaque composant de l'application), nous ferons en sorte qu'ils puissent communiquer les uns avec les autres dans le respect de l'architecture et nous exposerons l'application à l'extérieur du cluster en nous appuyant sur une couche de routage proposée nativement par OpenShift (ou Ingress).

Déploiement de l'application

Création du projet

Pour déployer notre application "fruits-catalog", nous allons nous connecter à la console OpenShift en choisissant la perspective **Developer**. Depuis cette console, vous aurez en partie centrale la possibilité de choisir le projet sur lequel vous souhaitez travailler.

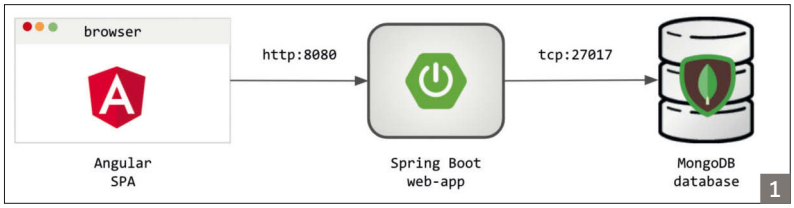
Nous allons créer un nouveau projet que nous appellerons "**fruits-catalog-dev**". 3

À l'initialisation, la vue **Topology** de ce projet est vide et la console vous propose d'y ajouter des éléments. Ce que nous allons nous empresser de faire.

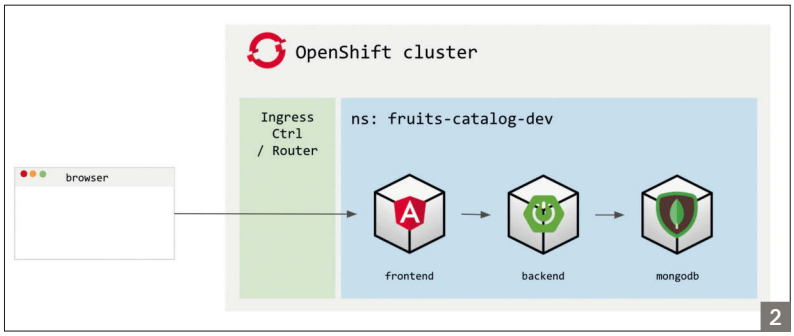
Déployer la base de données

Nous allons commencer par déployer la base de données MongoDB de notre application. Pour cela, nous allons partir de la vue **+Add** de la perspective **Developer** de la console OpenShift et nous allons choisir l'option **From Catalog**.

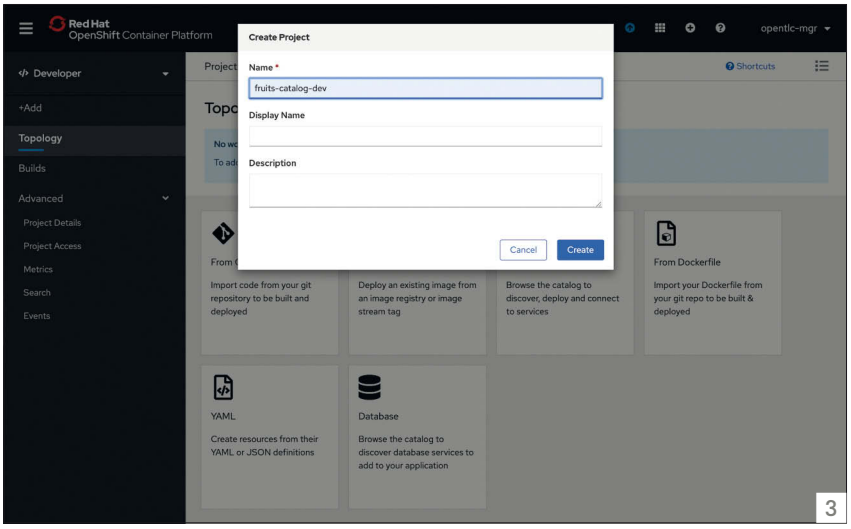
Le Developer Catalog d'OpenShift est une vue structurée des services offerts aux développeurs pour facilement instancier de nouveaux composants, de nouvelles applications ou déployer des



1



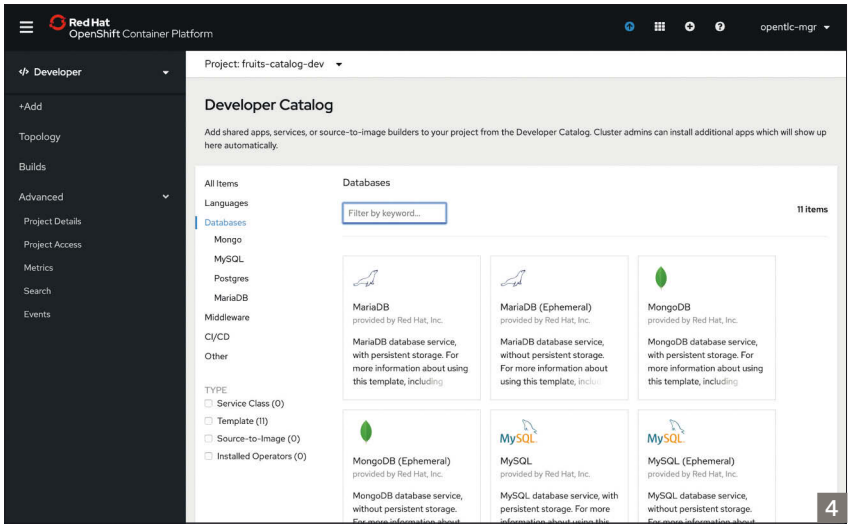
2



services managés en mode CaaS (Container as a Service). Pour notre base de données, c'est ce type de service que nous allons choisir en nous rendant dans la section **Databases** et en choisissant le service **MongoDB** fourni par Red Hat. 4

La console va vous permettre de cliquer sur **Instantiate Template** avant de vous présenter ensuite un formulaire vous permettant éventuellement de personnaliser certains éléments. Pour notre base, nous allons rester sur les défauts et cliquer **Create** en bas du formulaire. 5

Après quelques secondes d'attente et de retour sur la vue **Topology**, vous devez maintenant avoir la vision d'un déploiement nommé `mongodb`. En sélectionnant ce déploiement, vous allez pouvoir parcourir les ressources associées dans le volet qui apparaît sur la droite. Vous voyez notamment qu'OpenShift a instancié un ensemble d'éléments attachés : un `Pod` et un `Service`. Nous expliquerons ces notions par la suite. Votre base est déployée et disponible ! 6

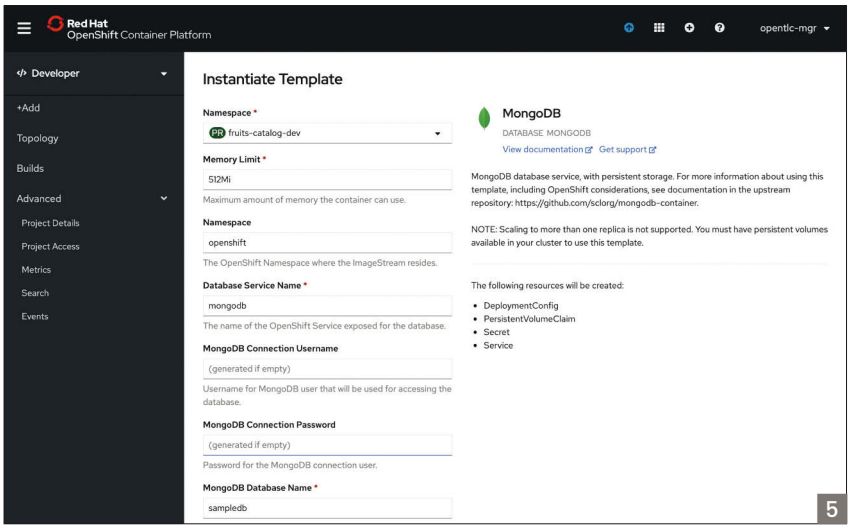


Déployer le composant "backend"

Le composant `backend` est développé en Java en utilisant le framework Spring Boot. Nous allons le déployer sur OpenShift en illustrant les fonctionnalités de PaaS: c'est OpenShift qui va se charger de construire une image de container pour nous !

Commencez par vous rendre dans le répertoire où se trouvent les sources de ce composant et lancez la commande Maven permettant de packager le composant en un JAR. Le résultat se trouve dans le répertoire /target.

```
$ cd backend
$ mvn package
--------OUTPUT:------
[INFO] Scanning for projects...
[INFO]
[INFO] ------------< com.redhat.openshift.samples:fruits-catalog >------------
[INFO] Building Fruits Catalog 1.0.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[...]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ fruits-catalog ---
[INFO]  Building jar: /Users/lbroudou/Development/github/openshift-by-example/backend/target/fruits-catalog-1.0.0-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.2.6.RELEASE:repackage (default) @ fruits-catalog ---
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.037 s
[INFO] Finished at: 2020-04-15T09:20:05+02:00
[INFO] ------------------------------------------------------------------------
```



Pour interagir avec OpenShift et lui transmettre notre composant, nous allons utiliser l'outil ligne de commande odo. Odo est normalement déjà présent sur l'environnement en ligne, mais vous pouvez l'installer si besoin :
https://docs.openshift.com/container-platform/4.3/cli_reference/openshift_developer_cli/installing-odo.html

```
$ odo version
--------OUTPUT:------
odo v1.1.2 (f7fcb5396)

Server: https://api.cluster-lemans-7d9e.lemans-7d9e.example.opentlc.com:6443
Kubernetes: v1.16.2
```

Nous voyons que odo réutilise la connexion au cluster Kubernetes existante. Odo suit une démarche basée sur la notion de composant ; nous allons donc devoir créer un nouveau composant de type java:8 qui s'appellera backend, qui se basera sur le JAR que nous venons de construire et exposera sur le port 8080. Rien de plus simple avec odo :

```
$ odo create java:8 backend --app fruits-catalog --binary target/fruits-catalog-1.0.0-SNAPSHOT
.jar --port 8080/tcp
--------OUTPUT:------
Validation
 ✔ Validating component [41ms]

Please use `odo push` command to create the component with source deployed
```

Comme indiqué par le message, la commande odo push permet de valider les changements et réellement déployer le composant dans le cluster OpenShift :

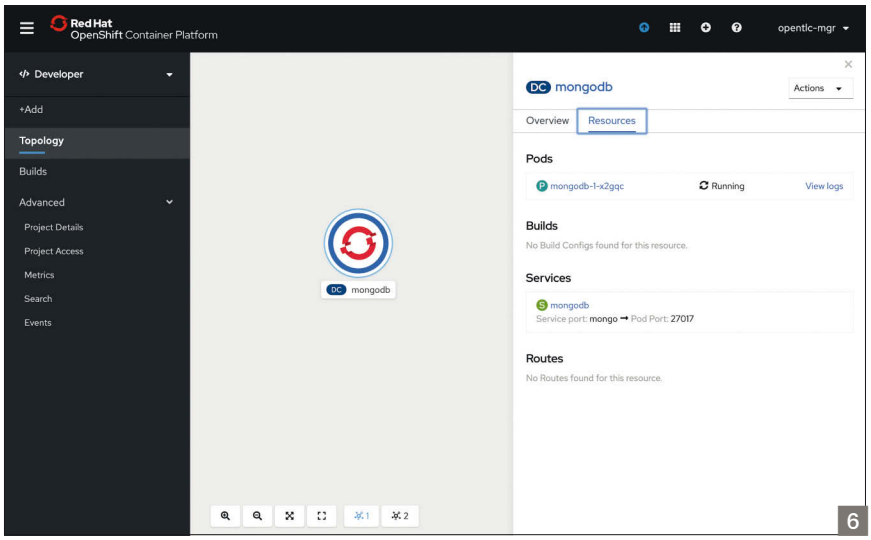
```
$ odo push
--------OUTPUT:------
Validation
 ✔ Checking component [67ms]

Configuration changes
 ✔ Initializing component
 ✔ Creating component [215ms]

Pushing to component backend of type binary
 ✔ Checking files for pushing [2ms]
 ✔ Waiting for component to start [24s]
 ✔ Syncing files to the component [11s]
 ✔ Building component [2s]
```

Une fois le composant déployé, la commande odo log permet de récupérer facilement les logs du composant courant - c'est à dire celui correspondant au répertoire dans lequel on se trouve :

```
$ odo log -f
--------OUTPUT:------
2020-04-15 07:59:34.890  INFO 8150 --- [           main] org.mongodb.driver.cluster               :
Cluster description not yet available. Waiting for 30000 ms before timing out
2020-04-15 08:00:04.892 ERROR 8150 --- [           main] o.s.b.web.embedded.tomcat.TomcatStarter  :
 Error starting Tomcat context.
[...]
org.springframework.beans.factory.BeanCreationException: Error creating bean with name
'mongo' defined in class path resource [org/springframework/boot/autoconfigure/mongo/
MongoAutoConfiguration.class]: Initialization of bean failed; nested exception is com.mongodb.
MongoTimeoutException: Timed out after 30000 ms while waiting to connect. Client view of
cluster state is {type=UNKNOWN, servers=[{address=localhost:27017, type=UNKNOWN,
state=CONNECTING, exception={com.mongodb.MongoSocketOpenException: Exception opening
```



6

```
socket}, caused by {java.net.ConnectException: Connection refused (Connection refused)}}]
2020-04-15 08:00:04.916  INFO 8150 --- [           main] o.apache.catalina.core.StandardService   :
Stopping service [Tomcat]
```

Vous voyez en inspectant les logs que notre composant ne démarre pas correctement… C'est tout à fait normal pour le moment, car nous ne lui avons pas indiqué comment trouver la base de données. Nous y remédierons plus tard.

Déployer le composant "frontend"

Pour déployer ce troisième et dernier composant, nous allons utiliser une autre méthode - nous aimons la diversité chez Red Hat ;-). Nous allons utiliser un autre outil ligne de commande oc - pour OpenShift Client. oc est en fait un super-ensemble de l'outil kubectl : toutes les commandes kubectl sont présentes également dans oc mais ce dernier présente quelques ajouts intéressants.

```
$ oc version
--------OUTPUT:------
Client Version: openshift-clients-4.3-31-gc25fb9c7
Server Version: 4.3.5
Kubernetes Version: v1.16.2

$ kubectl version
--------OUTPUT:------
Client Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.2",
GitCommit:"59603c6e503c87169aea6106f57b9f242f64df89", GitTreeState:"clean",
BuildDate:"2020-01-23T14:21:36Z", GoVersion:"go1.13.6", Compiler:"gc",
Platform:"darwin/amd64"}
Server Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.2",
GitCommit:"b3bfb5a", GitTreeState:"clean", BuildDate:"2020-03-02T08:50:52Z",
GoVersion:"go1.12.12", Compiler:"gc", Platform:"linux/amd64"}
```

Vous pouvez notamment utiliser indistinctement kubectl ou oc pour consulter la liste des Pods déployés sur le cluster :

```
$ kubectl get pods
--------OUTPUT:------
NAME                          READY   STATUS      RESTARTS   AGE
backend-fruits-catalog-1-deploy   0/1     Completed   0          19m
```

```
backend-fruits-catalog-1-hngr9   1/1   Running     0   19m
mongodb-1-deploy                 0/1   Completed   0   26m
mongodb-1-x2gqc                  1/1   Running     0   26
```

Nous avons pour le moment deux Pods en Running : un pour la base et un pour le composant backend. Pour déployer le dernier composant frontend, nous allons utiliser une fonctionnalité PaaS d'OpenShift qui permet de déléguer la construction de l'image de container à OpenShift en lui disant d'aller se sourcer directement dans le référentiel de sources Git. Cette fonctionnalité se nomme **Source To Image** (ou S2I).
Pour cela, oc possède une commande new-app qui permet de référencer un template de départ que l'on configure en indiquant l'adresse de notre référentiel Git. Comme notre composant est un développement Angular qui se construit avec NPM, nous allons utiliser simplement un template de base nodejs qui contient déjà les outils permettant le build automatique :

```
$ oc new-app nodejs~https://github.com/redhat-france-sa/openshift-by-example.git
--context-dir=/frontend --name frontend --labels='app.kubernetes.io/part-of=fruits-
catalog,app.openshift.io/runtime=nodejsversion=v1'
--------OUTPUT:------
--> Found image e86504c (2 weeks old) in image stream "openshift/nodejs" under tag
 "10-SCL" for "nodejs"

   Node.js 10
   ----------
  [...]
--> Success
   Build scheduled, use 'oc logs -f bc/frontend' to track its progress.
   Application is not exposed. You can expose services to the outside world by executing
one or more of the commands below:
    'oc expose svc/frontend'
   Run 'oc status' to view your app.
```

Wouah ! Nous voyons qu'une simple commande permet de créer de nombreuses ressources dans le cluster ! La commande new-app permet d'abstraire la notion d'application multi-composants et ainsi d'instancier très facilement des applications stéréotypées selon des modèles que vous pouvez définir. Explorons quelques objets spécifiques à OpenShift…
Le BuildConfig est créé pour sauvegarder les éléments de configuration de la construction de notre composent frontend. Nous voyons simplement ici que c'est un build de type Source qui est configuré avec un référentiel Git, avec 1 Build en cours :

```
$ oc get bc
--------OUTPUT:------
NAME     TYPE    FROM  LATEST
frontend Source Git   1
```

Les Builds réalisent explicitement l'opération de construction d'une image de container en réalisant le processus **Source To Image**. Ils possèdent un numéro et chaque nouveau déclenchement incrémente ce numéro :

```
$ oc get builds
--------OUTPUT:------
NAME      TYPE    FROM       STATUS    STARTED       DURATION
frontend-1  Source  Git@fdec53c  Complete  2 minutes ago  2m8s
```

L'ImageStream représente une portion de la registry interne à OpenShift qui est réservée pour accueillir les images de conteneurs résultant des différents Builds d'un composant. Un ImageStream pourra accueillir différents tags permettant d'identifier les images produites. Par convention, le tag latest désigne toujours la dernière image produite :

```
$ oc describe is/frontend
--------OUTPUT:------
Name:          frontend
Namespace:       fruits-catalog-dev
  [...]
Tags:          1

latest
 no spec tag

 * image-registry.openshift-image-registry.svc:5000/fruits-catalog-dev/frontend@sha256:dc
352cb7524140b879c1d1b99b7146a054d640cc0726b95fafcfa51c0269c4bc
    About an hour ago
```

Finalement nous pouvons exposer notre service frontend au monde extérieur comme recommandé dans le message de la précédente commande new-app :

```
$ oc expose svc/frontend
--------OUTPUT:------
route.route.openshift.io/frontend exposed
```

Cela a pour effet de créer une Route (Url) qui nous permettra d'accéder à l'application depuis l'extérieur du cluster :

```
$ oc get routes
--------OUTPUT:------
NAME     HOST/PORT
PATH SERVICES PORT     TERMINATION   WILDCARD
frontend   frontend-fruits-catalog-dev.apps.cluster-lemans-7d9e.lemans-7d9e.example.
opentlc.com        frontend   8080-tcp            None
```

Regardons maintenant les logs de notre nouveau Pod frontend :

```
$ kubectl get pods
--------OUTPUT:------
NAME                          READY   STATUS      RESTARTS   AGE
backend-fruits-catalog-1-deploy   0/1   Completed   0          19m
backend-fruits-catalog-1-hngr9    1/1   Running     0          19m
frontend-1-build                  0/1   Completed   0          47m
frontend-1-deploy                 0/1   Completed   0          45m
frontend-1-2pgrf                  1/1   Running     0          45m
mongodb-1-deploy                  0/1   Completed   0          26m
mongodb-1-x2gqc                   1/1   Running     0          26m

$ oc logs frontend-1-2pgrf
--------OUTPUT:------
Environment:
    DEV_MODE=false
```

```
    NODE_ENV=production
    DEBUG_PORT=5858
Launching via npm...
npm info it worked if it ends with ok
npm info using npm@6.13.4
npm info using node@v10.19.0
npm info lifecycle fruits-catalog-ui@0.0.1~prestart: fruits-catalog-ui@0.0.1
npm info lifecycle fruits-catalog-ui@0.0.1~start: fruits-catalog-ui@0.0.1

> fruits-catalog-ui@0.0.1 start /opt/app-root/src
> node index.js

CONFIG ERROR: Can't find backend service config!
Define `BACKEND_SERVICE` env variable.
Node app is running at localhost:8080
```

Celui-ci est également en erreur, car nous ne lui avons pas indiqué comment joindre le service backend et les API permettant d'écrire dans la base de données. C'est la dernière étape du déploiement : la configuration.

Configurer le tout !

Afin de lier nos composants ensemble, nous devons les configurer à l'aide de trois notions différentes présentes dans Kubernetes : les Secrets, les ConfigMap et les variables d'environnement.
Un Secret a déjà été créé pour nous lorsque nous avons déployé la base MongoDB. Nous pouvons voir qu'il contient des données sensibles encodées en base64 - les infos de connexion à la base :

```
$ oc get secret/mongodb -o yaml
--------OUTPUT:------
apiVersion: v1
data:
 database-admin-password: S0EzQkhHd1JRSkprcTJTaw==
 database-name: c2FtcGxlZGI=
 database-password: T3c2TlBxNnJ2bWFjVmZpaw==
 database-user: dXNlcjM4VQ==
kind: Secret
metadata:
 annotations:
  template.openshift.io/expose-admin_password: '{.data["database-admin-password"]}'
  template.openshift.io/expose-database_name: '{.data["database-name"]}'
  template.openshift.io/expose-password: '{.data["database-password"]}'
  template.openshift.io/expose-username: '{.data["database-user"]}'
```

Nous allons pouvoir réutiliser ces informations pour lier notre composant backend à la base. Nous devons tout d'abord créer une ConfigMap qui permettra d'injecter dans le composant la configuration de la connexion :

```
$ oc create -f ../manifests/backend-configmap.yml -n fruits-catalog-dev
--------OUTPUT:------
configmap/backend-config created
```

Cette ConfigMap définit un seul fichier application.yml qui porte une simple propriété correspondant à la chaîne de connexion à la base de données. Cette chaîne de connexion est définie grâce à des variables d'environnement délimitées par ${MONGODB_...} :

```
$ oc get cm/backend-config -o yaml
--------OUTPUT:------
apiVersion: v1
data:
 application.yml: 'spring.data.mongodb.uri:
mongodb://${MONGODB_DATABASE_USER}:${MONGODB_DATABASE_PASSWORD}@mongodb:
27017/sampledb'
kind: ConfigMap
```

Nous devons maintenant indiquer à OpenShift que cette ConfigMap doit être associée à notre composant et qu'elle doit être injectée à un endroit précis dans le répertoire /deployments/config. Cela se fait grâce à la création d'un Volume pour déterminer un point de montage du contenu de la ConfigMap :

```
$ oc set volume dc/backend-fruits-catalog --add --name=backend-config-volume --
mount-path=/deployments/config --type=configmap --configmap-name=backend-config
--------OUTPUT:------
deploymentconfig.apps.openshift.io/backend-fruits-catalog volume updated
```

Finalement, les variables d'environnement utilisées par la ConfigMap doivent, elles aussi, être injectées dans notre composant backend. On réalise cela en disant qu'il faut réutiliser le contenu du Secret de la base et injecter le tout en tant que variables d'environnement préfixées par MONGODB_ :

```
$ oc set env dc/backend-fruits-catalog --from=secret/mongodb --prefix=MONGODB_
--------OUTPUT:------
deploymentconfig.apps.openshift.io/backend-fruits-catalog updated
```

En inspectant les logs de notre composant backend, on voit que celui-ci est maintenant capable de se connecter à la base et de démarrer correctement :

```
$ odo log -f
--------OUTPUT:------
2020-04-15 08:03:42.012  INFO 31 --- [}-mongodb:27017]
org.mongodb.driver.connection          : Opened connection
[...]
2020-04-15 08:03:49.691  INFO 31 --- [         main]
c.r.s.f.c.FruitsCatalogApplication     : Started FruitsCatalogApplication in
26.205 seconds (JVM running for 30.293)
```

Dernière configuration nécessaire: indiquer au frontend, l'URL sur laquelle le service backend peut être contacté. Cela se fait simplement par l'ajout d'une variable d'environnement. On va utiliser ici le nom court du service et son port :

```
$ oc set env dc/frontend BACKEND_SERVICE=backend-fruits-catalog:8080
--------OUTPUT:------
deploymentconfig.apps.openshift.io/frontend updated
```

Avant de tester notre application en live, jetons un coup d'œil aux logs de notre composant frontend. Pour cela, il faut récupérer son nom dans la liste des Pods puis consulter les logs avec la commande oc logs "nom_du_pod" :

```
$ oc logs frontend-2-892cg
--------OUTPUT:------
```

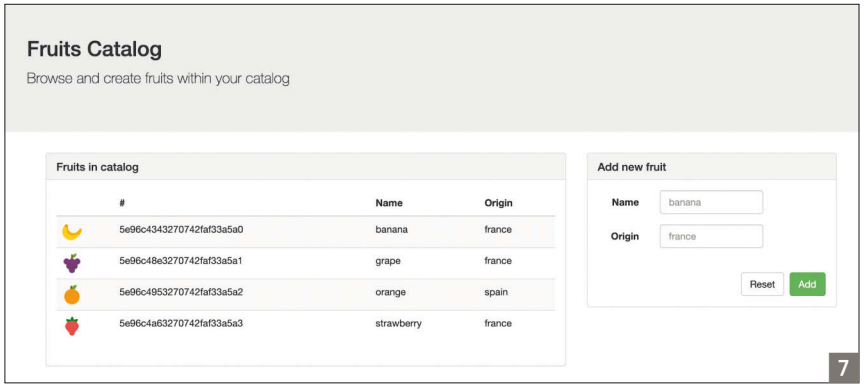
```
[...]

Backend service defined at `backend-fruits-catalog:8080`. Proxying all request to /api/*
Node app is running at localhost:8080
```

Finalement, récupérerons l'URL de la route associée à notre service frontend dans la colonne HOST/PORT de la commande suivante :

```
$ oc get route/frontend
--------OUTPUT:------
NAME     HOST/PORT
PATH SERVICES PORT     TERMINATION   WILDCARD
frontend   frontend-fruits-catalog-dev.apps.cluster-lemans-7d9e.lemans-7d9e.example.
opentlc.com        frontend   8080-tcp            None
```

Et ouvrons la dans notre navigateur favori (éventuellement en forçant le protocole à http://). Tadam ! Vous avez déployé votre première application sur OpenShift et avez maintenant accès à vos différents stocks de fruits. Vous pouvez en recenser plusieurs - en utilisant expressément banana, apple, grape, pear, lemon, pineapple, cherry, orange ou strawberry si vous voulez voir de jolies petites icônes 7



7

Les concepts

En déployant et en configurant nos trois composants (le backend, le frontend et la base de données) nous avons utilisé, sans forcément le savoir, une grande partie des concepts de base de Kubernetes et d'OpenShift. Voyons cela ensemble dans le détail :

- Le projet : Nous avons créé un projet fruits-catalog-dev, un **project** dans Openshift correspond en fait à un **namespace** dans Kubernetes.
- Notre base de données MongoDB a été créée à l'aide du Developer Catalog, ce service se base sur les Templates, fournis par Openshift. Un Template permet de définir un ensemble de primitives Kubernetes, comme les Secrets, un Service ou bien encore un Persistent Volume en fournissant également des paramètres. Ces paramètres, nous les retrouvons dans l'interface utilisateur lorsqu'on instancie notre objet du catalogue. Au final, voici les principales primitives créées à la fin du processus :
  - Un **Pod** : On peut le considérer comme le plus petit dénominateur de l'écosystème Kubernetes. Un pod contiendra notre composant applicatif conteneurisé, disposant de sa propre adresse IP et de son espace de stockage. Mais attention les Pods sont éphémères, en cas de défaillance du nœud sur lequel tourne le pod, lors du redémarrage du nœud, de nouveaux pods, avec une nouvelle adresse IP et espace de stockage seront créés.
  - Un **Service**: Cette primitive va permettre de jouer le rôle de façade pour un ou plusieurs pods. Un service dispose d'une adresse IP stable même en cas de redémarrage. Un service sélectionne ses pods par rapport à un sélecteur de label.
  - Un **Secrets**: Cette primitive permet de stocker de manière opaque des éléments de configuration qu'il chiffre en base 64.
  - Un **PersistentVolume et PersistentVolumeClaim (PV / PVC)** : Un PersistentVolume permet de définir un espace de stockage, le type restant abstrait. Le PersistentVolumeClaim quant à lui est une ressource utilisée par un pod pour indiquer qu'il souhaite disposer d'un espace de stockage, il fait le lien entre notre PersistentVolume et notre pod.
- Pour notre backend, nous avons utilisé une autre approche (ODO) pour instancier notre composant, mais le résultat en termes de primitives créées est le même : nous avons un **Pod** et un **Service**. Mais tout comme notre composant MongoDB, si vous regardez la typologie vous pouvez voir que nos deux composants sont de type **DeploymentConfig** qui est une primitive Kubernetes introduite par Openshift. Elle permet de configurer notre "usine à pods", le ReplicaController. Cette configuration peut réagir à des événements pour lancer la création d'un nouveau pod, ces événements peuvent par exemple être : un changement de configuration, la création d'une nouvelle image ou bien la mise à jour du code source dans le repo git (git push).
- Et enfin le Frontend nous permet de découvrir 3 autres primitives :
  - **BuildConfig** : cette primitive permet de décrire la manière dont une image doit être construite, suivant la technologie de l'application contenue dans l'image, le processus peut-être en effet différent (Maven pour Java par exemple et npm pour NodeJS).
  - **ImageStream** : c'est le point d'entrée d'une collection d'images classées par tags. Elle peut comprendre des images venant d'un répertoire extérieur, comme Docker Hub, mais aussi le répertoire d'image interne d'OpenShift.
  - **Route** : cette primitive permet d'exposer au monde extérieur un Service en lui attribuant un nom d'hôte publique.

Conclusion

Au travers du déploiement de notre application, vous avez vu dans cette deuxième partie comment OpenShift permet simplement d'utiliser et mettre en musique les différentes primitives de Kubernetes. Ces primitives sont toujours accessibles directement au travers de l'outil kubectl et des API, mais aussi activables de façon plus aisée et rapide au travers de concepts comme le Developer Catalog et les Templates.

Vous avez également pu découvrir les fonctionnalités qu'offre OpenShift pour passer facilement du code source de votre application à un container déployé et managé par la plateforme. Toutes ces étapes ont par ailleurs été réalisées dans un cadre sécurisé et reproductible - en s'appuyant sur les services et les processus configurés par les administrateurs de la plateforme. Nous allons notamment poursuivre le développement de ces notions dans la partie suivante de notre dossier où nous allons vous expliquer comment pousser votre application en production avec OpenShift.

Aller en production

Nous avons jusque'à présent déployé notre application sur l'environnement de développement appelé "DEV". Dans cette section, nous allons voir comment créer un 2e environnement de production nommé "PROD", et comment définir un pipeline CI/CD permettant de promouvoir tous les composants de notre application de "DEV" à "PROD".

Pour faciliter l'opération, OpenShift fournit une notion de **projet**, qui est isolé du reste et permet de matérialiser la notion d'environnement, en utilisant des conventions de nommage pour qualifier les projets. Par exemple pour notre application appelée "fruits-catalog", nous avons déjà créé le projet **"fruits-catalog-dev"**, et nous allons créer le projet **"fruits-catalog-prod"**.

OpenShift intègre nativement des outils pour faire du CI/CD, dont :

- Jenkins: il permet de définir des pipelines "as-code"; en utilisant le DSL Jenkins ou bien un DSL spécifique à OpenShift offrant des primitives telles que openshift.startBuild() permettant de démarrer un build.
- OpenShift Pipelines: basé sur le projet Tekton, cet outil permet de créer des pipelines natifs Kubernetes, en utilisant des images de containers comme briques permettant d'effectuer les tâches nécessaires, tout en permettant de les "orchestrer" ou définir leur enchaînement logique.

La suite de l'article utilisera Jenkins, mais nous vous invitons à consulter ce tutoriel (https://learn.openshift.com/middleware/pipelines) pour vous familiariser également avec Tekton.

Vision conteneurisée sur Kubernetes

Préparation de l'environnement de Production

Cette partie est la suite directe de la partie 2 et se base sur les éléments déjà déployés dans celle-ci. Si vous avez le temps, nous vous conseillons de commencer par cette partie. Si le temps vous est compté, vous pouvez utiliser le "raccourci" explicité dans l'encart.

Vous n'avez pas fait la partie 2 ?

Ou vous ne disposez plus de l'environnement ? Pas de panique ! Pensez à cloner notre référentiel GitHub (https://github.com/redhat-france-sa/openshift-by-example) et rendez-vous dans le répertoire shortcuts/ où se trouvent des scripts "raccourcis". Après vous être connecté sur votre cluster au moyen de la ligne de commande oc, exécutez simplement le script complete-part-2.sh. Vous voilà prêt à poursuivre ce dossier.

Création de l'environnement de "Production"

Nous avons vu dans la 1re partie qu'il existait différents moyens de déployer des composants sur OpenShift: à travers la console graphique, en utilisant la ligne de commande `oc` ou en utilisant la ligne de commande `odo` (revoir la partie 1 si les différents moyens ne sont pas clairs). Pour la suite, nous allons utiliser la ligne de commande `oc`.

Console, ligne de commande ou ressources YAML ?

La question revient souvent : quel outil utiliser pour quel usage ? Eh bien tous ! Typiquement la console sera très utile au début d'un projet lorsque l'on est en mode "exploration" et que l'on évalue les composants que l'on voudra utiliser. La ligne de commande sera un bon complément pour apporter des éléments de configuration comme nous l'avons fait dans la partie I. Cette ligne de commande pourra également être utilisée pour obtenir les représentations YAML des différentes ressources créées afin notamment de pouvoir les sauvegarder en gestion de configuration dans votre référentiel Git. Cela est essentiel pour garantir que vous serez capable de reconstruire votre environnement en toute circonstance !

Tout d'abord, nous allons **créer le projet de production** nommé **"fruits-catalog-prod"** avec la commande ci-dessous. Ce projet va posséder des droits d'accès différents et des configurations différentes de celui de développement. Le suffixe **"-prod"** permet de distinguer le projet de celui de développement.

```
$ oc new-project fruits-catalog-prod
--------OUTPUT:------
Now using project "fruits-catalog-prod" on server
```

Maintenant que le projet est créé, au lieu de déployer chaque composant séparément comme précédemment, nous allons tous les déployer en même temps grâce à une notion de **"Template"** que fournit OpenShift. En effet, un template est un fichier YAML regroupant l'ensemble des ressources à créer telles que BuildConfig, Deployment, Route, Namespace, etc.

Pour cela, il faut aller dans le répertoire "manifests" du référentiel que vous avez cloné, et déployer le fichier template-prod.yml qui contient plusieurs sections décrivant les ressources suivantes qui seront créées:

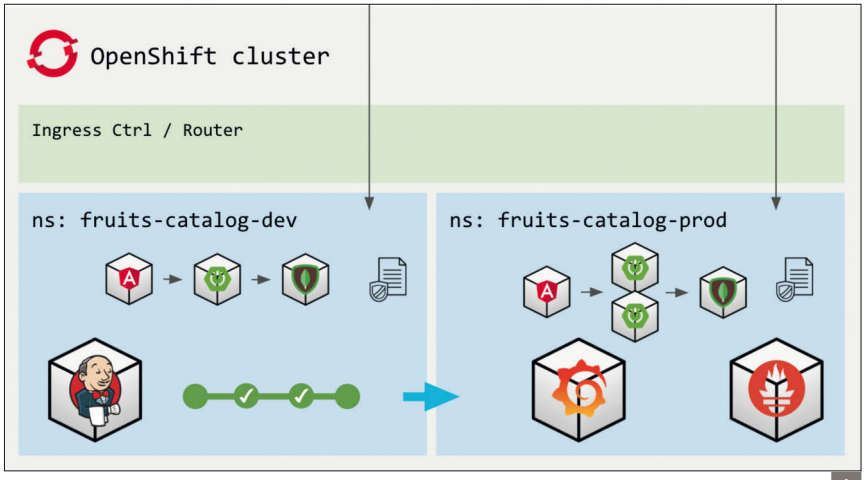
- La base de données `mongodb` qui hébergera les données des fruits ;
- Le composant springboot `backend` qui expose les données de la base mongodb à travers une API REST ;
- Le composant nodejs `frontend` qui contient l'IHM de l'application.

```
$ oc create -f manifests/template-prod.yml
$ oc process fruits-catalog-prod -n fruits-catalog-prod | oc create -f - -n fruits-catalog-prod
```

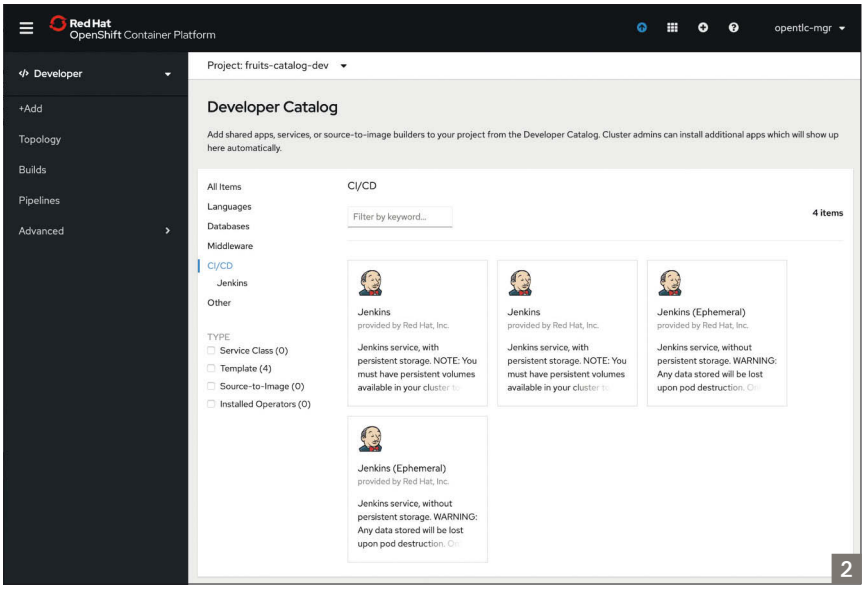
Pipeline de CI/CD du "DEV" à la "PROD"

Dans cette section nous allons créer un pipeline automatisé permettant de construire une nouvelle version de l'application "fruits-catalog" en environnement de développement et de la promouvoir à l'environnement de production en effectuant des mises à jour en "Rolling update", c'est-à-dire sans interruption de service.

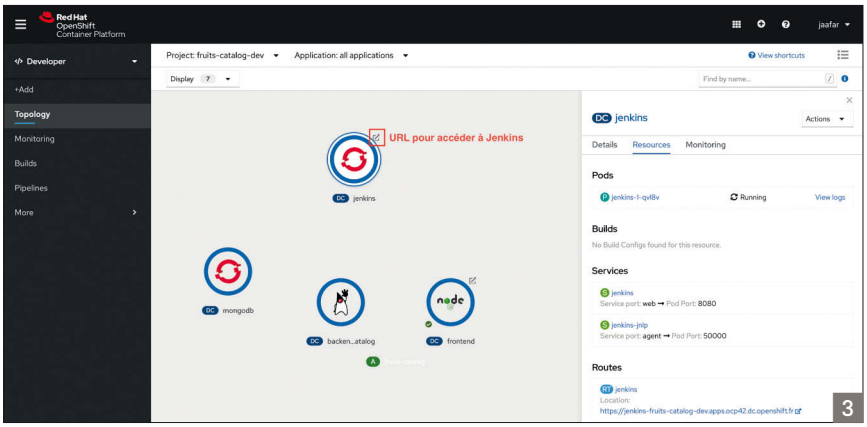
Pour cela, nous allons **déployer une instance de Jenkins dans l'environnement de développement**, qui pilotera le pipeline. Jenkins permet d'écrire en standard des pipelines-as-code en



1



2



3

Groovy, cependant Red Hat fournit une extension DSL (Domain Specific Language) plus riche permettant d'interagir plus facilement avec les primitives d'OpenShift depuis un pipeline Jenkins. 1

Gestion des droits d'accès du compte Jenkins

Le compte technique de Jenkins nécessite d'avoir certains droits (role edit) sur le projet de production pour y déployer des images. On le lui attribue avec la commande suivante **oc adm policy** :

```
$ oc adm policy add-role-to-user edit system:serviceaccount:fruits-catalog-dev:jenkins -n fruits-catalog-prod
--------OUTPUT:------
clusterrole.rbac.authorization.k8s.io/edit added: "system:serviceaccount:fruits-catalog-dev:jenkins"
```

Notre nouveau projet PROD doit également être capable d'aller chercher les images de containers qui auront été produites dans le projet de DEV - images stockées dans un ImageStream. On doit alors également indiquer que les comptes de services du projet PROD endossent le rôle system:image-puller dans le projet de DEV :

```
$ oc adm policy add-role-to-group system:image-puller
 system:serviceaccounts:fruits-catalog-prod -n fruits-catalog-dev
--------OUTPUT:------
Warning: Group 'system:serviceaccounts:fruits-catalog-prod' not found
clusterrole.rbac.authorization.k8s.io/system:image-puller added:
"system:serviceaccounts:fruits-catalog-prod"
```

Notre environnement de production est maintenant prêt à recevoir de nouvelles ressources et les composants de l'application !

Installation de Jenkins

Comme présenté dans l'introduction de cette partie, nous voulons déployer une instance de Jenkins dans l'environnement de développement. Pour ce faire, rendez-vous sur la Console OpenShift, dans la perspective **"Developer"** - soyez vigilant de bien sélectionner le projet fruits-catalog-dev - et choisissez d'ajouter un composant **From Catalog** depuis la vue **+Add**.

Dans la catégorie **CI/CD**, nous allons choisir le 1er composant **Jenkins** fourni par Red Hat. 2

La console va alors vous permettre d'instancier le **Template** et vous proposer un formulaire de personnalisation. Ici encore, conservez toutes les informations par défaut et cliquez simplement **Create** en bas du formulaire.

Après 1 à 2 minutes d'attente et de retour sur la vue **Topology**, vous devez maintenant avoir la vision d'un déploiement nommé jenkins. En sélectionnant ce déploiement, vous allez pouvoir parcourir les ressources associées dans le volet qui apparaît sur la droite. Vous voyez notamment qu'OpenShift a instancié un ensemble d'éléments attachés : un Pod, un Service et une Route. Vous allez pouvoir cliquer sur le lien de la route pour accéder à votre instance et vérifier le fonctionnement de Jenkins. 3

L'image Jenkins fournie par Red Hat est directement sécurisée et intégrée avec le modèle RBAC d'OpenShift. Ainsi, pour y accéder, vous devrez utiliser votre login et mot de passe OpenShift. À la première connexion, il vous sera également demandé de confirmer l'accès à vos informations de profil.

La première connexion à Jenkins est souvent longue, car Jenkins met à jour les versions de plug-ins en appelant son référentiel central - pendant ce temps, il oublie de s'occuper des requêtes utilisateur ! Soyez patient, rafraîchissez le navigateur et vous aurez accès à votre instance Jenkins. 4

Création des pipelines

Maintenant que Jenkins est instancié, nous allons pouvoir créer le pipeline en utilisant le fichier pipelines/fruits-cicd-pipeline.yml

```
$ oc create -f pipelines/fruits-cicd-pipeline.yml -n fruits-catalog-dev
```

Voici certains extraits de ce pipeline, illustrant l'usage du DSL OpenShift pour Jenkins:

Extrait du pipeline - Déclenchement des Builds en environnement de développement :

```
stages {
  stage('Build') {
    steps {
      script {
        openshift.withCluster() {
          openshift.withProject() {
            parallel(
              frontend: {
                openshift.startBuild("frontend").logs('-f')

              },
              backend: {
                openshift.startBuild("backend").logs('-f')
```

Dans cet extrait, l'appel à la fonction openshift.startBuild('nom-du-build') permet de déclencher un build source-to-image qui aura pour effet de récupérer le code source depuis git, et de construire une nouvelle image qui sera poussée dans le registry interne d'OpenShift.

Extrait du pipeline - Promotion à l'environnement de Production :

La promotion d'une image d'un environnement à un autre se fait grâce à des tags. Ainsi, dans le projet de "DEV" nous avons défini que lorsqu'une image recevra le tag "promoteToProd" alors celle-ci sera promue à l'environnement de "PROD". Bien entendu vous pouvez personnaliser le nom du tag à utiliser à cet effet. Voici un extrait montrant cela:

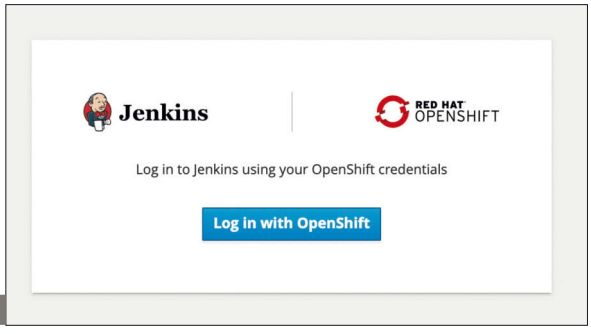
```
stage('deploy to PROD') {
  steps {
    script {
      openshift.withCluster() {
        openshift.withProject() {
          //By tagging the images with "promoteToProd, OpenShift will deploy these versions into PROD environment"
          openshift.tag("fruits-catalog-dev/frontend:latest", "fruits-catalog-dev/frontend:promoteToProd")
          openshift.tag("fruits-catalog-dev/backend:latest", "fruits-catalog-dev/backend:promoteToProd")
        }
      }
    } // script
  } // steps
} // stage
```

Pour visualiser les pipelines, toujours depuis la console "Developer", nous pouvons y accéder en cliquant sur "Builds" puis en sélectionnant "fruits-cicd-pipeline", ce qui nous amènera à cet écran: 5

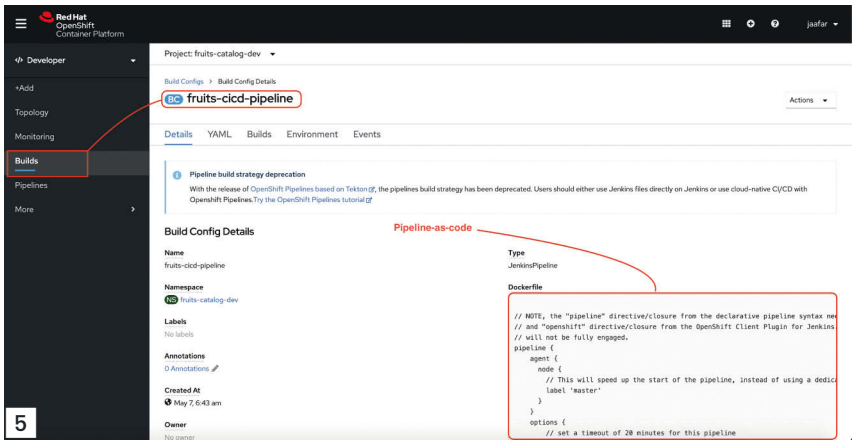
Déclenchement du pipeline de CI/CD

Nous allons désormais pouvoir lancer le pipeline. Pour cela il faut cliquer sur "Actions" en haut à droite et choisir "Start Build". L'intégration bidirectionnelle entre OpenShift et Jenkins permet de visualiser directement le pipeline en cours d'exécution dans l'interface d'OpenShift, comme suit: 6

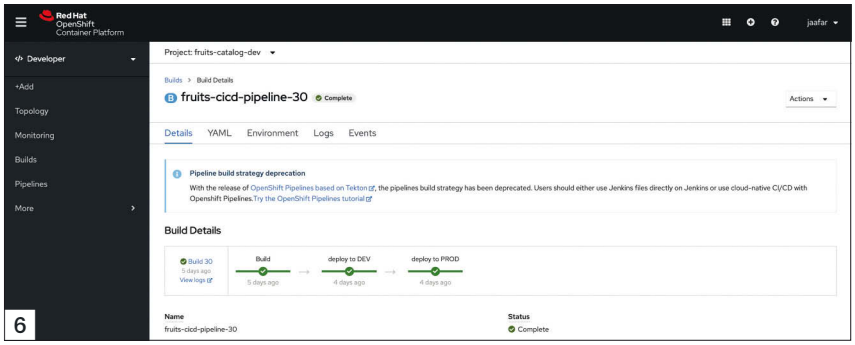
Et voilà! Désormais, en allant au projet "fruits-catalog-prod" on devrait voir les applications nouvellement déployées, notamment en regardant la date de création des pods associés en bout de ligne:



4



5



6

```
$ oc get pods -n fruits-catalog-prod
--------OUTPUT-----------
backend-fruits-catalog-5-j8z6d   1/1   Running   0   14m
frontend-7-ps5mm                 1/1   Running   0   14m
```

Cela indique que le pipeline a effectivement créé de nouvelles images de conteneurs de l'application dans le projet de "DEV", et que celles-ci ont été déployées dans le projet de "PROD" à travers l'instanciation de nouveaux pods.
Etape suivante: maintenant que vous savez automatiser le déploiement de votre application, nous allons voir comment la "monitorer" en production.

Monitoring de l'application

Déploiement de Prometheus et Grafana

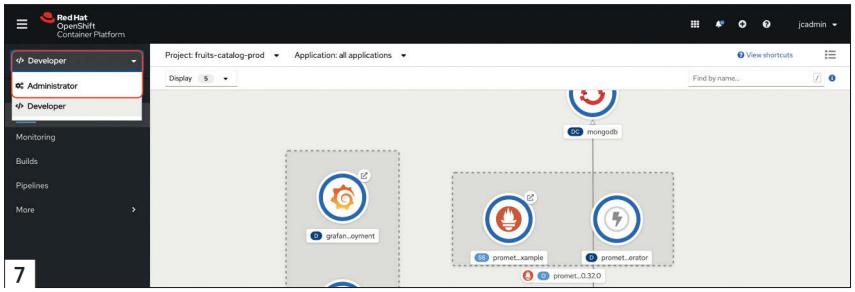
Dans cette section, nous allons déployer Prometheus pour capturer des métriques de notre application et Grafana afin de les visualiser dans des graphiques (dashboards).
Pour cela, nous allons utiliser les Operateurs fournis dans la Marketplace d'OpenShift. Commençons d'abord par y faire un tour. Il faut tout d'abord se "logger" en tant qu'administrateur du cluster.

OperatorHub, la Marketplace d'OpenShift

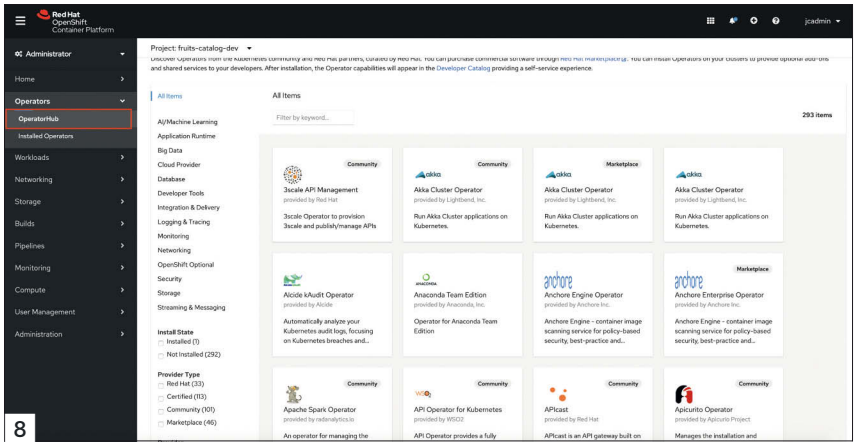
Disclaimer: la première partie de l'article explique ce que sont les opérateurs, leur intérêt dans Kubernetes et dans OpenShift. Je vous invite à relire cette partie si ces notions ne sont pas claires.
Red Hat a beaucoup œuvré pour l'adoption des opérateurs dans l'écosystème Kubernetes et a même réussi à construire une marketplace dédiée afin que tous les éditeurs de logiciels tiers puissent y publier leurs opérateurs: www.operatorhub.io.
Au-delà de cette marketplace publique, OpenShift fournit une marketplace privée avec du contenu testé, validé et certifié, permettant de bénéficier du support par l'éditeur de la solution. Il existe aujourd'hui 293 opérateurs dans cette marketplace à l'heure de la rédaction de cet article.
Ainsi, OpenShift apporte la simplicité du cloud à travers cette marketplace permettant de déployer simplement de nombreuses solutions conteneurisées, tout en bénéficiant de tout le savoir-faire en matière d'exploitation de cette solution, grâce à la notion d'opérateur: en résumé, un "opérateur" est un conteneur qui se déploie aux côtés de la solution qu'il gère, et intègre les bonnes pratiques d'exploitation qui sont automatisées et fournies par l'éditeur lui-même de cette solution. Ainsi l'utilisateur consomme un service de manière simple, mais bénéficie d'une expertise nativement intégrée en simplifiant l'exploitation au maximum (installation, mises à jour, configuration, maintien en condition opérationnelle, etc.).

Opérateurs, OperatorHub.io, Operator Lifecycle Manager, Operator SDK... Oh my... !

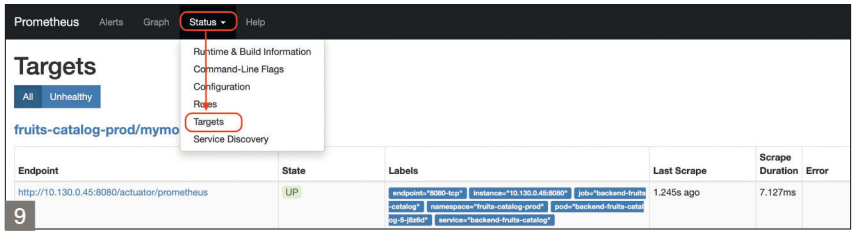
Les Opérateurs sont un concept majeur depuis la version 4.0 d'OpenShift. Les Opérateurs sont avant tout des opérateurs Kubernetes - pouvant fonctionner sur n'importe quelle distribution - mais OpenShift est la seule ayant été totalement architecturée autour de cette notion et offrant le plus de services pour tirer parti de cette technologie le plus facilement possible. Faisons un tour des éléments de l'écosystème Opérateurs intégrés à OpenShift. OperatorHub.io est la marketplace officielle pour les Opérateurs Kubernetes. Cette initiative a été lancée il y a un peu plus de 1 an.



7



8



9

Si vous êtes dans la vue **Developer**, il faut basculer vers la vue **Administrator**. Pour y accéder; il suffit de cliquer sur **Developer** et de sélectionner **Administrator** dans le menu de gauche tel qu'indiqué en : 7
Il faudra ensuite aller à la section **Opérateurs -> OperatorHub** dans le menu de gauche, comme indiqué en rouge ci-dessus : 8
Par souci de concision dans cet article, nous allons instancier les opérateurs à travers des ressources YAML afin d'éviter des copies d'écran à rallonge :)
Nous allons pour cela utiliser les fichiers situés dans le répertoire monitoring du référentiel git:

```
$ oc apply -f monitoring/ -n fruits-catalog-prod
```

Utilisation des "ServiceMonitor" pour monitorer l'application

Pour que Prometheus puisse récupérer (ou "scraper" en anglais) les métriques d'une application, il faut normalement créer une configuration fastidieuse dans Prometheus pour indiquer les caractéristiques de l'application à surveiller, et les injecter manuellement dans Prometheus.
Avec l'opérateur Prometheus, il suffit de créer une ressource de type ServiceMonitor qui indique quelques éléments clés: endpoint à

surveiller; intervalle; et application identifiée par un sélecteur sur les labels, comme décrit ci-dessous:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
 labels:
  k8s-app: prometheus
 name: mymonitor
 namespace: fruits-catalog-prod
spec:
 endpoints:
  - interval: 5s
    path: /actuator/prometheus
    port: 8080-tcp
 selector:
  matchLabels:
    app.kubernetes.io/instance: backend
```

Ainsi dans cet exemple, nous allons surveiller tout service portant un label app.kubernetes.io/instance:backend, sur l'endpoint /actuator/prometheus et le port 8080, et récupérer les métriques à intervalle de 5s. Cela tombe bien, car un endpoint Prometheus est déjà disponible sur le backend Spring Boot, car nous utilisons la librairie micrometer-registry-prometheus. Cet endpoint est disponible sur le port 8080, sur l'URI /actuator/prometheus.

Pour le vérifier nous allons faire un curl dans le pod du composant backend grâce à la commande oc rsh "nom_du_pod" :

```
$ oc rsh backend-4-hn5pm
sh-4.2$ curl http://localhost:8080/actuator/prometheus
# HELP jvm_memory_used_bytes The amount of used memory
# TYPE jvm_memory_used_bytes gauge
jvm_memory_used_bytes{area="heap",id="PS Survivor Space",} 1507360.0
jvm_memory_used_bytes{area="heap",id="PS Old Gen",} 2.5412728E7
status="200",uri="/actuator",} 0.301446996
# HELP tomcat_sessions_expired_sessions_total
# TYPE tomcat_sessions_expired_sessions_total counter
tomcat_sessions_expired_sessions_total 0.0
```

Pour vérifier que la collecte de métriques se passe correctement, vous pouvez ouvrir prometheus et vérifier dans les "targets", comme le montre le schéma ci-dessus : 9

Si tout s'est passé correctement, on devrait voir un statut "UP" en vert devant l'endpoint /actuator/prometheus que nous avons demandé de surveiller.

Visualisation des métriques dans Grafana

Grafana est un outil open-source pour réaliser des tableaux de bord à partir de différentes sources de données, extrêmement populaire dans l'écosystème Kubernetes et Prometheus. Nous avons précédemment installé l'opérateur Grafana, et déployé une instance de Grafana dans le projet **"fruits-catalog-prod"**.

Grafana nécessite une source de données, qui sera dans notre cas Prometheus. Nous allons encore une fois bénéficier des avantages d'utiliser un opérateur, car celui-ci nous permet de créer directement une ressource de type GrafanaDataSource depuis un fichier YAML, et c'est l'opérateur lui-même qui s'occupera d'appeler les primitives Grafana pour créer la source de données.

Cela facilite énormément l'import/export de "datasources" ou de "dashboards" comme nous le verrons plus tard.

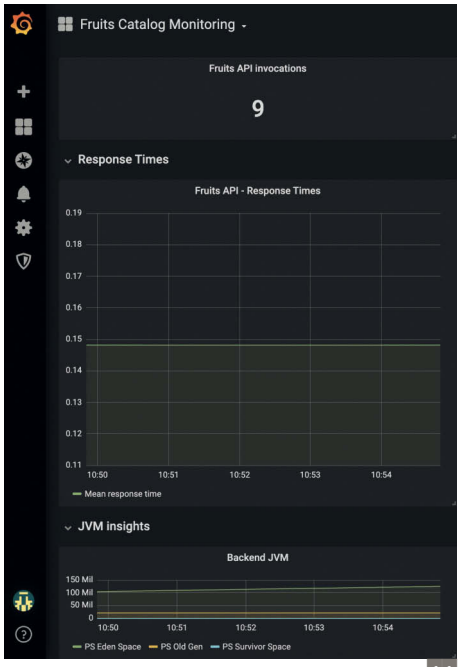
```
$ oc create -f monitoring/dashboards/prometheus-datasource.yaml -n fruits-catalog-prod
```

Il nous reste à créer un dashboard pour notre application, cette fois-ci en créant une ressource kubernetes de type GrafanaDashboard. C'est l'opérateur qui se chargera d'injecter ce dashboard dans l'instance de Grafana.

```
$ oc create -f monitoring/dashboards/fruits-monitoring-dashboard.yaml -n fruits-catalog-prod
```

L'interface graphique d'OpenShift nous permet de créer ou visualiser graphiquement ces mêmes ressources, comme le montre l'écran suivant, dans le menu de gauche **Operators -> Installed Operators** puis en cliquant sur **Grafana Operator** dans la liste: 10

Et voilà! Notre instance de Grafana est correctement configurée, et nous devrions voir le dashboard suivant montrant 3 métriques: nombre d'invocations à l'api backend, temps de réponse de l'api, et utilisation des ressources de la JVM de l'application Backend. 11



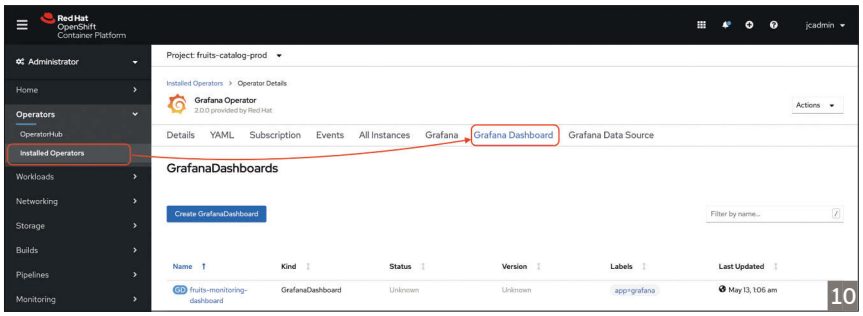
11

Conclusion

Nous avons vu dans cette partie comment OpenShift permet de promouvoir une application d'un environnement de "développement" à un autre environnement de "production", en écrivant un pipeline CI/CD 'as-code' automatisé fournissant un langage adapté pour OpenShift afin de simplifier l'écriture de ceux-ci. Par ailleurs, afin de surveiller notre application en production, nous avons déployé Prometheus et Grafana, tous les deux basés sur des opérateurs Kubernetes fournis par Red Hat dans la marketplace. Ces opérateurs simplifient grandement le déploiement ET la configuration de Prometheus et Grafana, en étendant l'API de Kubernetes avec des objets comme GrafanaDatasource ou GrafanaDashboard pour Grafana, ou encore ServiceMonitor pour Prometheus.

Ainsi, des opérations de configuration qui auraient été fastidieuses et manuelles sont simplement réalisées en créant des ressources à partir de leur définition YAML, dans la pure tradition de Kubernetes.

À vous de jouer maintenant !



10

OpenShift avancé

Jusque-là, nous avons déployé notre application comprenant plusieurs composants (frontend/backend/Base de données) et nous avons mis en place un processus de CI/CD (Continuous integration/Continuous deployment) pour promouvoir les différents éléments dans un environnement de production.

Vous devez maintenant commencer à avoir une bonne vision d'ensemble de la plateforme Openshift et de tout ce qui est possible de faire avec. Dans cette partie, nous allons pousser les capacités de notre plateforme encore plus loin, en introduisant un maillage de services qui traitera les fonctionnalités opérationnelles transverses de notre infrastructure ainsi qu'un nouveau composant serverless.

Préparation de votre cluster

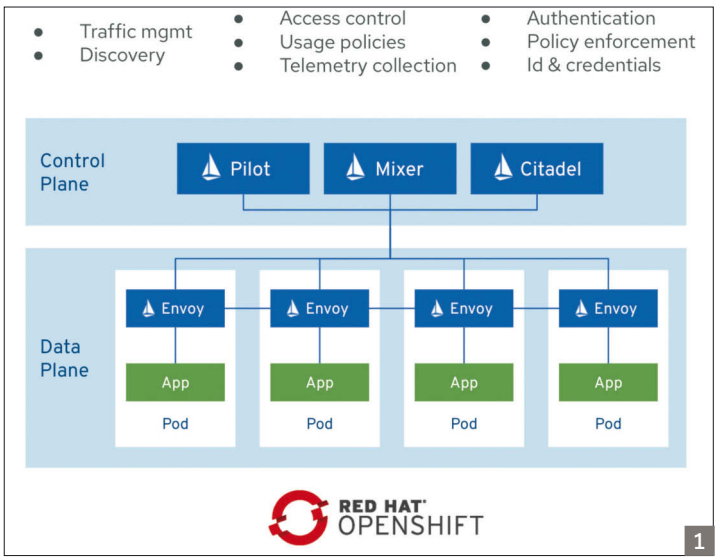
Cette partie est la suite directe de la partie 3 et se base sur les éléments déjà déployés dans cette partie. Si vous avez le temps, nous vous conseillons de commencer par cette partie. Si le temps vous est compté, vous pouvez utiliser le "raccourci" explicité dans l'encart.

Vous n'avez pas fait la partie 3 ?

Ou vous ne disposez plus de l'environnement ? Pas de panique ! Pensez à cloner le référentiel GitHub (https://github.com/redhat-france-sa/openshift-by-example) et rendez-vous dans le répertoire shortcuts/ où se trouvent des scripts "raccourcis". Après vous être connecté sur votre cluster au moyen de la ligne de commande oc, exécutez simplement les scripts complete-part-2.sh et complete-part-3.sh. Vous voilà prêt à poursuivre ce dossier. A l'heure où vous lirez ces lignes il est possible que l'environnement en ligne ne soit pas encore mis à jour pour vous permettre de dérouler cette partie. Si vous êtes pressé, nous vous conseillons d'installer une version complète d'essai sur try.openshift.com.

Dans cette partie, nous allons faire appel à des services complémentaires d'OpenShift qui ne sont pas activés par défaut sur votre cluster. Pour installer ces services, vous aurez besoin des droits cluster-admin sur votre cluster. L'installation de ces services va demander la création de deux nouveaux projets : istio-system pour le Service Mesh et knative-serving pour la partie Serverless. L'installation de ces éléments a été scriptée pour la version 4.3 d'OpenShift et est disponible dans un script "raccourci". Pour l'activer sur votre cluster :

```
$ cd shortcuts
$ chmod u+x prepare-part-4.sh
$ ./prepare-part-4.sh
$ git checkout frontend-v2
[...]
```



1

OpenShift Service Mesh

Pourquoi un Service Mesh ? Et c'est quoi d'ailleurs ?

Les méthodes modernes de développement où l'on recherche toujours plus de vélocité, les besoins de scalabilité des applications ainsi que l'adoption du cloud ont entraîné ces dernières années une évolution de nos architectures applicatives. Nos applications monolithiques d'antan ne répondent plus aux nouveaux enjeux et nous leur préférons aujourd'hui des architectures où les composants présentent un couplage technique faible et des granularités de déploiement et de gestion plus petites.

Sans entrer ici dans des descriptions et discussions autour des tendances en architecture - on pourra citer néanmoins pour référence les 12factor.net, les microservices ou encore les mecha-runtimes - nous faisons le constat que nos applications sont de plus en plus dynamiques et distribuées sur les réseaux. Elles sont alors confrontées à des problématiques de robustesse et résilience, de découverte de leurs dépendances et de leur configuration et posent pas mal de soucis aux développeurs quand il s'agit de comprendre et débugger ce qui se passe…

Il y a quelques années, nous avons cherché à résoudre ces problématiques grâce à des librairies ou frameworks - comme Hystrix, Ribbon, Eureka fournis par Netflix - que l'on intégrait dans nos applications. Bien que répondant au besoin, cette approche a ses limites : le développeur doit maintenir ces nouvelles librairies, elles peuvent être inconsistantes d'un framework à un autre et bien souvent elles ne couvrent que l'écosystème Java.

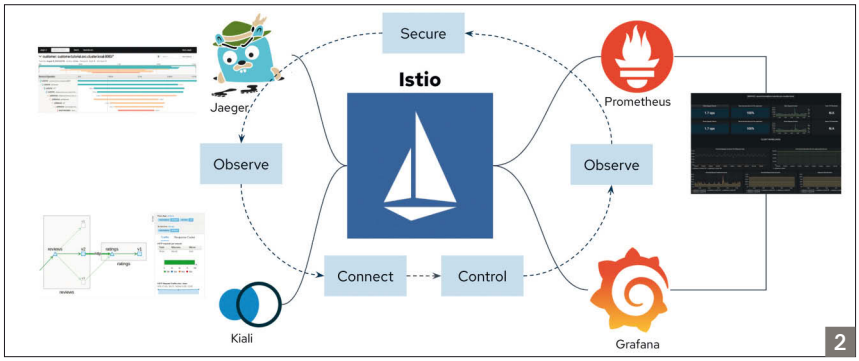
Des esprits brillants ont alors imaginé réutiliser un design-pattern bien connu - le **proxy** - afin de sortir ces préoccupations du code de l'application et permettre à toute technologie d'en bénéficier. Réintroduire un proxy distribué n'aurait en revanche pas de sens dans notre quête de réduction de l'impact des problèmes réseau… L'idée est donc d'introduire ce proxy juste à côté de l'application et Kubernetes permet de faire ça en co-localisant plusieurs containers dans le même Pod. Le principe du Service Mesh était né : c'est **un maillage de proxies**, injectés en side-car dans le Pod de notre application, **permettant d'apporter observabilité, gestion du trafic, sécurité et politiques d'accès** à nos applications. 1

Avec OpenShift, Red Hat a décidé d'inclure OpenShift Service Mesh, un Service Mesh utilisant les éléments de la communauté

Istio.io. OpenShift Service Mesh est une distribution optimisée pour OpenShift qui se déploie notamment grâce à des Opérateurs. La particularité d'OpenShift Service Mesh est également de fournir une suite complète et cohérente pour vous permettre de déployer et gouverner à l'échelle vos Service Mesh sur OpenShift. 2

Nous allons utiliser le Service Mesh pour deux cas d'usages précis :

- La mise en place d'un processus de livraison en *canary-release* d'une nouvelle version de notre composant frontend,
- La sécurisation de la connexion entre notre composant backend et la base de données MongoDB en activant une politique d'encryption des flux via Mutual TLS.



2

Canary-release du frontend

Qu'est que le *canary-release* ? Il s'agit d'un pattern qui permet de mettre en production une nouvelle version de son produit à une population restreinte d'utilisateurs tandis que la majorité des utilisateurs restent sur la version actuelle. Facebook et Twitter sont friands de ce genre de pratique, vous avez sûrement déjà rencontré cette situation où vous avez une nouvelle feature de disponible et pas vos amis (ou bien le contraire et là c'est rageant). C'est ce que nous allons faire avec notre application frontend en déployant une deuxième version qui sera disponible uniquement pour 50% de nos utilisateurs.

Déploiement du frontend v2

Nous avons préparé un fichier YAML de déploiement de notre nouvelle version :

```
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
    app: frontend
    version: v2
    app.kubernetes.io/part-of: fruits-catalog
 name: frontend-v2
spec:
 replicas: 1
 selector:
    matchLabels:
    app: frontend
    version: v2
 template:
    metadata:
    labels:
    app: frontend
    version: v2
    app.kubernetes.io/part-of: fruits-catalog
    app.openshift.io/runtime: nodejs
    deploymentconfig: frontend
    annotations:
    sidecar.istio.io/inject: "true"
```

Il suffit d'appliquer ce fichier sur notre cluster :

```
$ oc apply -f manifests/frontend-v2-deployment.yml -n fruits-catalog-prod
--------OUTPUT-----------
deployment.apps/frontend-v2 created
```

Ce déploiement contient déjà les labels nécessaires pour qu'il soit pris en compte par notre service existant. On patche aussi votre service pour enlever le sélecteur sur la version :

```
$ oc patch service frontend --type json -p '[{ "op": "remove", "path": "/spec/selector/version" }]'
--------OUTPUT-----------
service/frontend patched
```

Si vous retournez sur votre navigateur sur la page du catalogue de fruits et que vous rafraîchissez la page, cela devrait alterner entre la version 1 et la version 2 de manière aléatoire.

Nous allons utiliser Istio pour mettre en place une règle de routage de trafic. Mais tout d'abord nous devons injecter le sidecar proxy dans notre première version :

```
$ oc patch dc/frontend -p '{"spec":{"template":{"metadata":{"annotations":{sidecar.istio.io/
inject:"true"}}}}}'
--------OUTPUT-----------
deploymentconfig.apps.openshift.io/frontend patched
```

Gateway, VirtualService et DestinationRules

Commençons par identifier clairement notre version 1 et notre version 2 au niveau de notre maillage de services, pour cela on utilise un DestinationRule qui permet de contraindre la façon dont on accède à une ressource au sein du ServiceMesh :

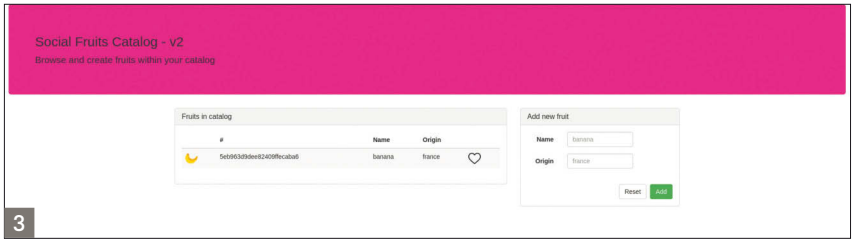
```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
 name: frontend
spec:
 host: frontend
 subsets:
 - name: v1
    labels:
    version: v1
 - name: v2
    labels:
    version: v2
```

```
$ oc apply -f istiofiles/frontend-dr.yml -n fruits-catalog-prod
--------OUTPUT-----------
destinationrule.networking.istio.io/frontend created
```

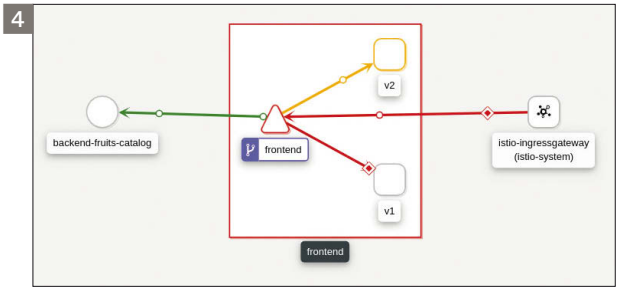
Il nous faut maintenant définir encore 2 éléments : une Gateway pour exposer notre ServiceMesh à l'extérieur, Openshift gère cela en nous créant une nouvelle route. Et puis pour terminer il nous faut

un VirtualService qui va spécifier nos règles de routage. Dans le cas ci-dessous on envoie 50% sur la version 2 et 50% sur la version en utilisant la propriété weight :

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
 name: fruit-gateway
spec:
 selector:
    istio: ingressgateway # use istio default controller
 servers:
 - port:
    number: 80
    name: http
    protocol: HTTP
    hosts:
    - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
 name: fruits-catalog
spec:
 hosts:
 - "*"
 gateways:
 - fruit-gateway
 http:
    - route:
    - destination:
    host: frontend
    subset: v1
    weight: 50
    - destination:
    host: frontend
    subset: v2
    weight: 50
```



3



4

```
$ oc apply -f istiofiles/frontend-vs-gateway.yml -n fruits-catalog-prod
--------OUTPUT-----------
gateway.networking.istio.io/fruit-gateway created
virtualservice.networking.istio.io/fruits-catalog created
```

Il faut maintenant découvrir la nouvelle route qu'Openshift nous a créée :

```
$ oc get route/istio-ingressgateway -n istio-system | grep istio-ingressgateway | awk '{print $2}'
--------OUTPUT-----------
istio-ingressgateway-istio-system.apps.sebination.sebiazure.com
```

Accédez à l'URL et rafraîchissez la page plusieurs fois. À un moment donné, vous devriez voir la version 2 apparaître. Elle est reconnaissable, car la couleur de l'en-tête est différente et "v2" est ajoutée au titre 3

Vous remarquerez aussi qu'à côté de chaque entrée de fruit, il y a maintenant un icône cliquable en forme de cœur. C'est une nouvelle fonctionnalité apparue avec la version 2, la partie backend n'est pas encore présente. Nous allons la déployer à la fin de ce chapitre.

Vérification

Afin de vérifier que tout est bien déployé et fonctionnel, nous vous proposons d'utiliser Kiali, l'outil dédié à l'observation au sein de OpenShift Service Mesh. Kiali a été installé en préambule de ce cas d'usage et la route d'accès à la console peut être trouvée dans le projet istio-system comme ceci :

```
$ oc get route/kiali -n istio-system | grep kiali | awk '{print $2}'
--------OUTPUT-----------
kiali-istio-system.apps.cluster-lemans-7d9e.lemans-7d9e.example.opentlc.com
```

L'image Kiali fournie par Red Hat est directement sécurisée et intégrée avec le modèle RBAC d'OpenShift. Ainsi, pour y accéder, vous devrez utiliser votre login et mot de passe OpenShift. À la première connexion.

Kiali permet d'avoir un contrôle complet sur les éléments de configuration de vos applications dans le Service Mesh : les fameux DestinationRule, VirtualService et autres Gateway. La console vous permet aussi de visualiser les flux entre les composants de votre application et collecter des métriques de fonctionnement. En sélectionnant le menu **Graph** vous aurez alors la possibilité de visualiser les deux versions de votre frontend. 4

Sécurisation MTLS MongoDB

Application

Pour ce cas d'usage, nous allons intégrer les deux composants backend et mongodb à notre Service Mesh et activer la sécurisation par TLS mutuel pour pouvoir les faire bénéficier de deux points particuliers :

- L'**authentification service-à-service** : chaque service va se voir attribuer une identité forte représentant son rôle. Ainsi avant d'être rendue possible, la communication entre services sera assujettie à une vérification d'authentification permettant d'éviter un programme malicieux de se faire passer pour un tiers (et ainsi voler des données),
- L'**encryption service-à-service** : l'infrastructure du Service Mesh gère la génération, la distribution et le renouvellement de certificats aux différents composants devant communiquer. Les flux sont alors cryptés en utilisant les clés distribuées.

Afin de permettre l'entrée dans le Service Mesh du composant mongodb et permettre de mettre en place des politiques de communication, nous devons tout d'abord lui ajouter un label correspondant à son application et un label nommé version. Ici nous définissons une simple v1 grâce à la commande oc patch qui nous permet de modifier une ressource existante :

```
$ oc patch dc/mongodb --type=json -p '[{"op":"add", "path":"/spec/template/metadata/labels/app", "value":"fruits-catalog"}, {"op":"add", "path":"/spec/template/metadata/labels/version", "value":"v1"}]' -n fruits-catalog-prod
--------OUTPUT:------
deploymentconfig.apps.openshift.io/mongodb patched
```

Comme nous avons modifié l'objet, un nouveau déploiement se déclenche. Cette modification n'est en revanche pas suffisante et nous devons maintenant faire entrer les composants dans le Service Mesh. Cela se fait simplement en ajoutant l'annotation sidecar.istio.io/inject=true sur les composants. Sur notre base de données :

```
$ oc patch dc/mongodb -p '{"spec": {"template": {"metadata": {"annotations": {sidecar.istio.io/inject: "true"} } } } }' -n fruits-catalog-prod
--------OUTPUT:------
deploymentconfig.apps.openshift.io/mongodb patched
```

Puis sur notre composant backend. Vous voyez au passage que la commande oc patch permet différentes notations :

```
$ oc patch dc/backend-fruits-catalog -p '{"spec": {"template": {"metadata": {"annotations": {"sidecar.istio.io/inject": "true"} } } } }' -n fruits-catalog-prod
--------OUTPUT:------
deploymentconfig.apps.openshift.io/backend-fruits-catalog patched
```

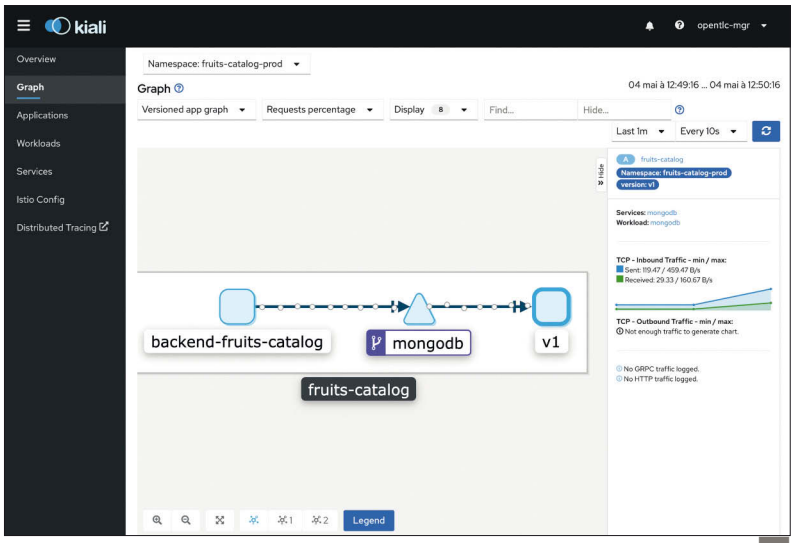
Nous allons maintenant pouvoir appliquer une politique Istio sur notre composant mongodb. Cela se fait de nouveau avec un DestinationRule. Voyons notre règle de destination ci-dessous :

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
 name: mongodb
spec:
 host: mongodb
 trafficPolicy:
  tls:
   mode: ISTIO_MUTUAL
 subsets:
  labels:
 - name: v1
   version: v1
```

De façon assez lisible, nous voyons que cette règle consiste à appliquer une politique de trafic appliquant le TLS mutuel lorsque l'on souhaite à accéder à la version v1 du service mongodb. Tout simplement ! On peut appliquer cette règle via la commande suivante :

```
$ oc apply -f istiofiles/mongodb-dr.yml -n fruits-catalog-prod
--------OUTPUT:------
destinationrule.networking.istio.io/mongodb created
```

L'application de la règle est une chose, mais encore faut-il ouvrir un



5

chemin jusqu'à ce composant dans le Service Mesh. Dans Istio, les "chemins" d'accès (simple ou mettant en œuvre des règles de routage) se gèrent grâce au concept de VirtualService. Nous devons également en créer un pour accéder à la base de données depuis le backend :

```
$ oc apply -f istiofiles/mongodb-vs.yml -n fruits-catalog-prod
--------OUTPUT:------
virtualservice.networking.istio.io/mongodb created
```

Vérification

À nouveau, nous pouvons utiliser Kiali pour vérifier que tout est déployé comme souhaité : 5

Serverless avec Knative

Vous vous rappelez l'icône en forme de cœur dans la version 2 de notre frontend ? Eh bien il est temps d'implémenter le backend pour cette fonctionnalité.

La fonctionnalité est très simple : la possibilité d'aimer un fruit. Le backend ne fera rien de spécial mis à part écrire dans le log que le fruit est aimé et de renvoyer l'instance du fruit.

Cette fonctionnalité ne sera pas utilisée tout le temps, notamment la nuit lorsque les utilisateurs dorment. C'est dans ce genre de cas d'usage où il peut être intéressant d'introduire un composant *serverless*. C'est une fonctionnalité qui ne sera déployée et activée que lorsqu'elle est utilisée. Au bout d'un moment, lorsqu'il n'y a plus de trafic sur ce composant, celui-ci disparaît, vous sauvant ainsi de la mémoire, du temps processeur, et donc vous faites des économies.

Il existe de nombreuses solutions *serverless*. Openshift propose son **opérateur** basé sur la solution *Knative*. C'est un projet initié au départ par Google, mais tous les grands acteurs comme Red Hat participent à ce projet Open Source. Knative propose des primitives Kubernetes orientées serverless. Il se repose donc sur les fondations de Kubernetes et il est ainsi très facile de transformer n'importe quelle ressource Kubernetes en ressource serverless.

Natif avec Quarkus

Nous avons une image toute prête qui contient cette nouvelle fonctionnalité, mais à titre d'information regardons avec quelle technologie ce composant a été réalisé. On sait que ce composant

serverless va devoir se déployer, démarrer puis s'arrêter fréquemment tout en limitant l'utilisation de la RAM et du CPU. D'ailleurs, même en dehors de *serverless* ces contraintes sont valables pour tout type d'applications se disant *Cloud Native*.
En général les développeurs se tournent vers du Go ou du NodeJS pour ce genre de cas d'usage, Java étant trop lent à démarrer et trop gourmand en mémoire.
Mais il y a un nouvel acteur dans le monde Java qui fait oublier toutes ces lacunes de Java : Quarkus. Quarkus est une stack Java qui adopte une approche révolutionnaire en mettant beaucoup plus d'intelligence dans la phase de *build* et qui résulte en des artefacts plus légers, plus rapides et ayant une empreinte mémoire minimale. Mais ce n'est pas tout, avec Quarkus on peut compiler son application en natif grâce à l'utilisation de GraalVM. Le temps de démarrage s'en trouve encore amélioré et l'empreinte mémoire est encore plus réduite qu'avant.
Le composant qu'on va déployer ici démarre en 0,008 seconde ! 6

Création du service Knative

Il nous faut d'abord créer un nouveau projet Openshift afin d'isoler notre composant *serverless* du maillage de service.

```
$ oc new-project fruitless
```

Il suffit maintenant d'appliquer la ressource Knative Service de notre composant et comme vous pouvez le voir celle-ci est très simple :

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
 name: fruits-lover
spec:
 template:
    spec:
    containerConcurrency: 0
    containers:
    - image: sebi2706/fruits-lover:latest
```

```
$ oc apply -f manifests/knative-fruits-lover-service.yaml -n fruitless
```

Et voilà ! C'est tout ! Vous avez déployé votre premier composant *serverless*. Dans un autre terminal, démarrez une commande qui va nous permettre d'observer notre pod knative :

```
$ watch oc get pods -n fruitless
```

Si vous tapez cette commande dans la minute qui a suivi le déploiement du service, vous devriez voir un pod actif. Observez bien, au bout d'une minute le pod va se terminer, car il n'y a aucun trafic sur ce nouveau service. On va le réactiver par notre frontend cette fois-ci.

Configuration du frontend v2

Il faut maintenant configurer le frontend v2 afin qu'il puisse communiquer avec le composant *serverless* :

```
$ oc set env deployment/frontend-v2 LIKE_SERVICE="fruits-lover.fruitless.svc.cluster.local"
```

Invocation du service Knative

Accédez à votre frontend et cette fois cliquez sur le cœur d'un de vos fruits. Observez le terminal qu'on avait ouvert précédemment :

```
NAME                                        READY STATUS    RESTARTS AGE
fruits-lover-kzbjv-1-deployment-584dd85b79-nlzr5 2/2   Running 0     59s
```

On peut voir qu'un pod vient de se créer à l'invocation du service. Encore une fois, au bout d'une minute, si aucun trafic ne parvient sur notre service Knative, le pod sera détruit automatiquement.

N'oubliez pas qu'Istio fait encore un partage de trafic 50% sur la v1 et 50% sur la v2. Si vous souhaitez toujours avoir la version v2, il vous suffit de modifier le manifest istiofiles/frontend-vs-gateway.yml et mettre la v2 à 100%.

Conclusion

Tout d'abord, un grand MERCI d'avoir lu jusqu'au bout, et peut-être suivi les exercices proposés!
Nous voulions vous présenter la plateforme OpenShift : ce qu'elle apportait aux développeurs, en déployant pas-à-pas une application multi-composants (Node.js; Java et MongoDB), puis en créant un pipeline CI/CD automatisé pour la déployer en production avec Jenkins, ensuite en la monitorant en production avec Prometheus et Grafana. Finalement, nous avons utilisé des fonctions plus avancées telles que le Service Mesh (Istio) ou le Serverless, qui sont des fonctionnalités nativement proposées par la plateforme.
OpenShift est depuis plusieurs années une référence pour avoir une plateforme Kubernetes en production, et de nombreux clients l'utilisent pour accélérer leur stratégie "Multi & Hybrid Cloud" permettant d'avoir des applications à la fois "on-premises" dans leur datacenter mais aussi chez les différents fournisseurs de clouds publics. Ainsi, OpenShift joue le rôle de pierre angulaire et de couche d'abstraction entre les différents sous-jacents, favorisant grandement la portabilité des applications des datacenters aux multiples clouds.
Nous n'avons pas abordé d'autres composants de la plateforme tels que MultiCluster Management (Advanced Cluster Management), l'API Management (3scale), ou AMQ Streams (Kafka par Red Hat). OpenShift fournit également de nombreuses solutions complémentaires développées par Red Hat (plusieurs dizaines), au sein de son catalogue de services de type PaaS. Et qui sait ? Cela pourrait faire l'objet de futures éditions de Programmez!. Nous espérons sincèrement que ce dossier vous aura plu et que vous aurez appris de nouvelles choses, et entre temps, happy coding!

6

```
__  ____  __  _____   ___  __ ____  ______
 --/ __ \/ / / / _ | / _ \/ //_/ / / / __/
 -/ /_/ / /_/ / __ |/ , _/ ,< / /_/ /\ \
--\___\_\____/_/ |_/_/|_/_/|_|\____/___/
2020-05-07 06:24:03,132 INFO  [io.quarkus] (main) fruits-lover 1.0.0-SNAPSHOT (po
wered by Quarkus 1.4.1.Final) started in 0.008s. Listening on: http://0.0.0.0:808
0
```

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

Les Enovateurs
"les innovateurs"

Phalcon : un framework performant et robuste compilé en C

Découvrez ce framework encore peu connu, désormais mature, qui monte en puissance de version en version. Ne passez pas à côté de cette pépite !

Prologue

Nous sommes en 2011, Laravel vient de sortir sa première version. La même année, un certain Andrés Gutiérrez conçoit une idée bien précise d'un nouveau type de framework Web : très performant, utile et possédant de nombreuses fonctionnalités.
Dans une interview de 2015, il explique avoir essayé de développer ce framework avec différents langages comme Go, Rust, C… mais sans succès. Il choisit ensuite PHP, et obtient de meilleurs résultats. À ce moment, Andrès possède 9 années d'expérience dans des entreprises qui lui ont demandé de développer des frameworks Web personnalisés. Pendant cette période, il a étudié les concepts de PHP, design patterns, paradigmes… Dans une interview de 2016, il conseille la lecture du site https://phptherightway.com/ ainsi que d'autres livres permettant de devenir un vrai gourou de PHP.

Naissance du framework

Pour créer le framework PHP le plus rapide, Andrès eut l'idée de développer une extension de PHP et non une surcouche. De cette manière le framework est intégré dans l'ADN de PHP. Pour développer une extension, il faut coder en C. La raison en est simple : PHP est un langage écrit en C.
Partant de ce principe, il publia une première version nommée Spark (étincelle) écrite en C. Ce POC lui permit de constituer un groupe de développeurs enthousiastes à l'idée de créer un framework.

Première version de Phalcon

Ils décidèrent ensuite de renommer ce projet « Phalcon », contraction de PHP et Falcon (faucon). Au bout d'un an de travail intensif, ils sortirent une première bêta en novembre 2012 – version 0.4.5.

Naissance de Zephir

Avec cette première version, ils se rendirent compte que peu de développeurs répondaient présents pour développer un framework en C. Partant de cette problématique, ils eurent alors une nouvelle idée : créer un nouveau langage, proche de PHP du point de vue syntaxique, et permettant de développer des extensions en C. C'est ainsi que Zephir naquit en 2013. La version 2.0 de Phalcon a été écrite en Zephir, et continue de l'être depuis.

Grâce à Zephir, Phalcon a gagné en popularité. D'autres développeurs ont rejoint la communauté et contribuent à développer Phalcon plus facilement. Zephir est aussi utilisé dans d'autres projets : des développeurs ont conçu d'autres extensions très performantes et utiles pour leur entreprise.
Pour vous donner une idée de la syntaxe de Zephir, voici un extrait du code source de Phalcon :
Cet extrait correspond à la fonction d'ajout d'un rôle de la classe, permettant de gérer les droits d'accès aux différentes pages d'un projet Web.

```
/**
 * Adds a role to the ACL list. Second parameter allows inheriting access data from other existing role
 *
 * ```php
 * $acl->addRole(
 *    new Phalcon\Acl\Role("administrator"),
 *    "consultant"
 * );
 *
 * $acl->addRole("administrator", "consultant");
 * $acl->addRole("administrator", ["consultant", "consultant2"]);
 * ```
 */
public function addRole(role, accessInherits = null) -> bool
{
    var roleName, roleObject;

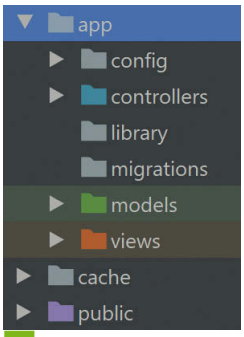
    if typeof role == "object" && role instanceof RoleInterface {
        let roleObject = role;
    } elseif is_string(role) {
        let roleObject = new Role(role);
    } else {
        throw new Exception(
            "Role must be either a string or implement RoleInterface"
        );
    }

    let roleName = roleObject->getName();

    if isset this->rolesNames[roleName] {
        return false;
    }
```

Par rapport à PHP, on constate quelques petites différences :
- Pour définir le type de retour d'une fonction, c'est la flèche qui est utilisée.
- La définition des variables se fait avec le mot-clé var.
- Le mot-clé let permet d'assigner des valeurs aux variables définies.
- Il n'y a pas de $ avant le this.

Pour le reste, c'est du PHP traditionnel.



1

Aujourd'hui

La dernière version de Phalcon, la 4.0.6, date du 16 mai 2020. Quant à Zephir, nous en sommes à la version 0.12.18, sortie le 24 avril 2020. Le 11 novembre 2019, PECL (Gestionnaire d'extension PHP) a validé l'entrée de Phalcon dans son célèbre catalogue. De cette manière, il est désormais plus simple d'installer Phalcon.
Le 15 janvier 2020, j'ai écrit et publié un livre intitulé « Phalcon - Développez des applications Web complexes et performantes en PHP » aux Éditions ENI. Ce livre présente et illustre dans le détail les fonctions de Phalcon, les différentes manières de l'utiliser… ainsi que de nombreux exemples et cas d'utilisations qui vous aideront tout au long de l'avancement de votre projet avec Phalcon.
Phalcon et Zephir ont tous deux beaucoup évolué depuis leurs débuts. Cependant, dans cet article, nous nous concentrerons sur Phalcon et ses fonctionnalités clés.

Les fonctionnalités de Phalcon

Un framework écologique et robuste

Phalcon s'installe comme une extension. En d'autres termes, tous les fichiers qui composent Phalcon sont compilés en seul fichier. Cependant Phalcon doit être compilé en différentes versions afin de fonctionner avec les systèmes 32/64 bits, les différentes versions de PHP, ainsi que les différents systèmes d'exploitation.
La compilation possède de nombreux avantages :
- Tout le code est analysé. S'il y a une erreur, la compilation est arrêtée. Ce qui assure une meilleure stabilité, par rapport à du code interprété.
- Le code est compressé, ce qui le rend à la fois léger et plus simple à installer. Il suffit de copier-coller un fichier .dll ou .so, au lieu de récupérer un nombre conséquent de fichiers (comme le font d'autres frameworks avec composer).
- L'extension est optimisée pour une architecture spécifique. Il est même possible de compiler soi-même Phalcon.
- Lorsque Phalcon est installé sur un serveur Web, toutes les applications peuvent l'utiliser.
- À partir du moment où le daemon du serveur Web est lancé, les classes et fonctions de Phalcon sont accessibles.

Avec tous ces avantages, Phalcon consomme peu de mémoire et de CPU. Pour plus d'informations, je vous conseille de lire ce benchmark : https://blog.phalcon.io/post/benchmarking-phalcon
Ces quelques extraits d'un cas d'utilisation de la RTM (Régie des Transports Métropolitains) démontrent les performances de Phalcon. Le document date de 2018, et vous retrouverez le lien à la fin de l'article.
Voici le problème rencontré par la RTM :
« **la problématique principale de ce site est d'être capable de gérer plusieurs milliers de requêtes par minutes sans rallonger le temps de réponse à celles-ci tout en optimisant au maximum le ratio coûts d'infrastructure/performances.** »
Ils ont donc opté pour la solution Phalcon. Voici leur retour d'expérience :
« Nous avons pu constater que Phalcon s'avère d'une utilité redoutable lorsqu'il est question de performance. »
« Du côté du site à fort trafic, lors de tests de montée en charge, nous avons pu constater que Phalcon tient effectivement ses promesses tant en matière de nombre de requêtes qu'en temps de réponse, même lors de pics de trafic ».

Architecture MVC

Voici à quoi ressemble le squelette de base d'un projet Phalcon. Vous devez certainement reconnaître l'architecture MVC avec les dossiers Models / Views / Controllers. 1
Le dossier config permet de définir les paramètres, les services, les routes et les espaces de noms.
Le répertoire migrations permet de gérer les migrations SQL des bases de données. Le dossier public contient les ressources statiques (JS, Images, CSS…).
Pour finir, le répertoire cache contient les fichiers PHP générés par le moteur de rendu Volt / Phtml.

L'API Rest

Il est plutôt simple de créer une API avec Phalcon. Il suffit de créer un projet de type MICRO avec phalcon-devtools, et le tour est joué.
Voici un exemple d'API simple avec Phalcon :

```
$app = new Micro();

$app->get('/', function () {
    return $this->response->setJsonContent(
        [
          'status' => 'Tout fonctionne',
        ]
    );
});

$app->notFound(function () use($app) {
  $app->response->setStatusCode(404, "Not Found")->sendHeaders();
});

$app->handle($_GET['_url'] ?? '/');
```

Avec la classe Micro, il est possible de créer des routes avec leurs fonctions associées. Comme vous l'aurez sans doute compris, vous pouvez remplacer GET par une autre méthode d'API comme POST, PUT…
Il est aussi possible de gérer les routes inconnues (ex : erreur 404).
Comme en témoigne Maxime :
« Phalcon a tout d'un grand : framework MVC, micro framework pour de l'API… même pour une simple page, pas besoin de composer, le tout via une extension PHP développée en C. »

Le mode CLI - console

Dans la plupart des projets, il faut exécuter certaines tâches en arrière-plan, pour différentes raisons :
- Des tâches longues, pendant lesquelles l'utilisateur peut s'absen-

ter et être averti par mail ou notification lorsque le traitement est fini.
- Des tâches récurrentes : typiquement des traitements de maintenance, comme nettoyer la base de données.
- Des tâches planifiées, pour envoyer des mails automatiques à un instant T.

Pour ce genre de problématique, vous avez deux solutions. La première consiste à se tourner vers une techno supplémentaire. La seconde, à utiliser PHP en console.
Pour éviter d'installer des éléments supplémentaires, je préfère utiliser CRON et appeler un script PHP.
Phalcon propose ce mode console pour répondre à ces différents besoins, et plus encore.
Pour comprendre comment cela fonctionne, vous pouvez créer un projet de type CLI avec phalcon-devtools.
Voici un exemple :

```
phalcon project MesTaches cli
```

Le projet MesTaches est créé. 2
Un projet de type CLI peut facilement être intégré à un projet complet, il suffit d'ajouter les fichiers suivants :
- run
- app/tasks/*
- app/bootstrap.php

Comme vous l'avez sans doute deviné, l'ajout de tâches se fait dans le répertoire tasks en créant de nouvelles classes ou en ajoutant des fonctions dans les classes existantes.
Voici un exemple pour lancer une tâche :

```
php run version
```

Avec cette commande, Phalcon lance la fonction mainAction() de la classe VersionTask.php.

Requête avec la base de données

Phalcon propose différentes manières d'interagir avec une base de données. Il y a :
- ORM – Object Relational Mapping ;
- PHQL – Phalcon Query Language ;
- DAL – Database Abstraction Layer.

Avec le Database Abstraction Layer, Phalcon interroge la base de données en utilisant PDO. C'est la manière classique pour interagir avec une base de données MySQL, PostgreSQL, Sqlite.
Exemple d'utilisation de DAL :

```
$this->db->fetchAll('
  SELECT id, prenom, nom, email
  FROM utilisateurs
  WHERE prenom = :prenom',
  Db::FETCH_ASSOC,
  [
    'prenom' => 'Louise',
    'id'    => 2
  ]
);
```

Pour interroger une table de données avec l'ORM, il faut créer une classe. Dans cette classe, on peut préciser les attributs de la table, les liaisons avec d'autres tables ainsi que des validateurs. L'idée est de manipuler des objets, de passer de table en table plus facilement et de s'assurer de la validité des données insérées en base.
Exemple d'utilisation de l'ORM :

```
use Phalcon\Mvc\Model;

use Phalcon\Validation\Validator\Email as EmailValidator;

class Utilisateur extends Model
{
  public $email;

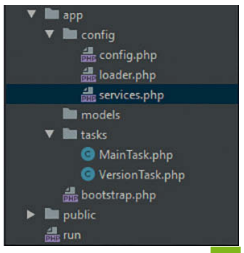
  public $password;

  public function initialize()
  {
    $this->hasMany('id', Cours::class, 'utilisateur_id', [ 'alias' => 'cours' ]);
  }

  public function validation()
  {
    $validator = new Phalcon\Validation();

    $validator->add('email',new EmailValidator(
        [
          'model'   => $this,
          'message' => 'Veuillez entrer une adresse mail correcte',
        ]
      )
    );

    return $this->validate($validator);
  }
}
```



2

Nous arrivons enfin à la méthode la plus performante avec PHQL. Celui-ci est intégré à Phalcon comme un parseur de requête SQL, mais écrit en C. De cette manière, l'analyse et l'exécution des requêtes SQL deviennent plus rapides. De plus, PHQL utilise très peu de mémoire, ce qui le rend thread-safe.
Exemple d'utilisation de PHQL :

```
$sPHQL = 'SELECT * FROM NovaMooc\Models\Utilisateurs
      WHERE prenom like :prenom:';

$aResultat = $this->modelsManager->executeQuery(
  $sPHQL,
  [
    'prenom' => '%a%'
  ]
);
```

Gestion d'un lot de résultats avec la pagination

Phalcon propose aussi un module de pagination efficace et rapide à mettre en place. En effet, Phalcon peut paginer des résultats issus directement de la base de données ou d'un tableau de données. On peut imaginer charger un fichier CSV et le faire paginer par Phalcon.

```
$oConstructeur = $this->modelsManager
        ->createBuilder()
        ->from(Utilisateurs::class);

$oPaginateur = new QueryBuilder(
  [
    'builder' => $oConstructeur,
    'limit'   => 10,
    'page'    => $nPageCourante,
  ]
);

$oPagine = $oPaginateur->paginate();
```

On accède ensuite à tout un tas de propriétés qui permettent de récupérer des données :
- getPrevious() : le numéro de la page précédente.
- getCurrent() : le numéro de page courant.
- getItems() : la liste des éléments courants.
- getNext() : le numéro de page suivante.
- getLast() : le numéro de la dernière page.
- getTotalItems() : le nombre d'éléments à charger.

Gestion des droits d'accès

Phalcon permet de créer une stratégie de droits d'accès assez facilement. Il faut définir une stratégie par défaut : soit tout refuser, soit tout accepter.

```
$oAcl = new Phalcon\Acl\Adapter\Memory();
$oAcl->setDefaultAction(Phalcon\Acl\Enum::DENY);
```

Ensuite, il faut définir les noms et descriptions des rôles.

```
use Phalcon\Acl\Role;
$aRoles = [
  'vendeurs' => new Role(
    'vendeurs',
    'Accès uniquement à la section de vente de produits.'
  ),
  'clients'  => new Role(
    'clients',
    'Accès uniquement à la section d\'achat de produits.'
  )
];

foreach ($aRoles as $oRole) {
  $oAcl->addRole($oRole);
}
```

La dernière étape consiste à indiquer à Phalcon quels sont les droits possédés par chaque type d'utilisateurs.

```
use Phalcon\Acl\Component;
$aActionParRole = [
  'vendeurs' => [
    'vendeurs' => [ 'profil' ],
    'produits' => [ 'nouveau', 'édition' ]
  ],
  'clients'  => [
    'clients'  => [ 'profil' ],
    'produits' => [ 'acheter' ]
  ],
  '*' => [
    'index'   => [ 'index', 'connexion', 'déconnexion' ],
    'erreurs' => [ '*' ]
  ]
];

foreach ($aActionParRole as $sRole => $aRole) {
  foreach ($aRole as $sControleur => $aActions) {
    $oComponentControleur = new Component($sControleur);
    foreach($aActions as $sAction){
      $oAcl->addComponent($oComponentControleur, $sAction);
      $oAcl->allow($sRole, $sControleur, $sAction);
    }
  }
}
```

L'étoile symbolise « tous ». Donc selon le tableau ci-dessus, tous les utilisateurs ont accès aux pages index, connexion et déconnexion du contrôleur index. Et tous les utilisateurs ont le droit d'accéder à toutes les actions du contrôleur erreurs.
Il faut ajouter chaque contrôleur dans l'objet ACL avec la fonction addResource.
Il existe ensuite deux stratégies possibles : indiquer les actions autorisées, ou préciser les actions interdites pour un utilisateur donné.
Pour autoriser un utilisateur, il suffit d'utiliser la fonction allow. Pour lui refuser l'accès, il faut utiliser la fonction deny.
Pour éviter les trois foreach imbriqués, Phalcon propose de sérialiser l'objet ACL afin de le stocker en session ou dans un fichier.
Après avoir effectué tous ces paramétrages, il suffit d'utiliser la fonction isAllowed pour savoir si un utilisateur a le droit d'accéder à une page.
Par exemple :

```
$oAcl->isAllowed('clients', 'produits', 'acheter');
```

Par rapport aux paramètres précédents, la fonction répond true car un utilisateur a le droit d'acheter un produit.

```
$oAcl->isAllowed('clients', 'produits', 'nouveau');
```

Dans ce cas, la fonction répond false car ce n'est que le vendeur qui a le droit de créer un nouveau produit.

nouveau !

Boutique Programmez!

Les anciens numéros disponibles

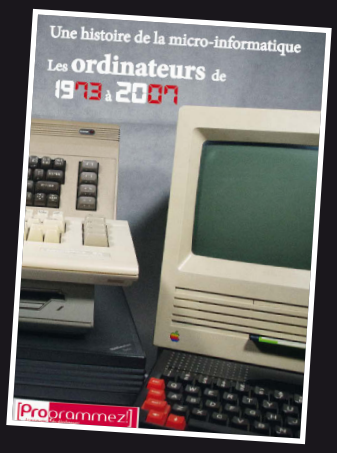
220

221

226

230

237



Histoire de la micro-informatique 1973-2007

12,99 €
(frais postaux inclus)

238

239

240

Technosaures

Prix unitaire : **7,66 €**
(frais postaux inclus)

n°1

n°2

n°3

tarif unitaire 6,5 € (frais postaux inclus)

☐ 220 : |__| ex ☐ 237 : |__| ex
☐ 221 : |__| ex ☐ 238 : |__| ex
☐ 226 : |__| ex ☐ 239 : |__| ex
☐ 230 : |__| ex ☐ 240 : |__| ex

soit |__|__| exemplaires x 6,50 € = |__|__|__|__| €

Technosaures ☐ N°1 ☐ N°2 ☐ N°3
soit |__|__| exemplaires x 7,66 € = |__|__|__|__| €
☐ Histoire de la Micro-informatique 12,99 €

..... |__|__|__|__| €

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

soit au **TOTAL** = |__|__|__|__|__| €

☐ 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 241

L'Injecteur de dépendance

Le DI (Dependency Injector) permet de créer des services utilisables partout dans l'application. Il est possible de créer deux types de services :
- simple : appel complet du code ;
- partagé : fonctionne comme un Singleton.

Ces services sont définis dans le fichier app/config/services.php.
Voici quelques exemples :

Un service simple

```
$di->set('fpdf', function () {
    $oPdf = new Fpdf();

    $oPdf->AddPage();
    $oPdf->Image('logo_entete.png',8,5,25);

    return $oPdf;
  }
);
```

Ce code permet de créer un nouveau PDF à chaque appel du service. Ce PDF est pré-paramétré avec un logo d'en-tête. Pour appeler ce service, depuis un contrôleur par exemple, il suffit d'écrire l'instruction suivante :

```
$oPDF = $this->di->get('fpdf');
```

Un service partagé

```
$di->setShared('config', function () {
  return include APP_PATH . "/config/config.php";
});
```

Pour y avoir accès, il suffit d'utiliser l'instruction suivante :

```
$oConfig = $this->di->get('config');
```

Moteur de vues Volt

Phalcon propose son propre moteur de rendu de vues. Il est bien entendu écrit en Zéphir / C, ce qui le rend très rapide. Volt embarque de nombreuses fonctions très utiles et permet par exemple de formater (supprimer des espaces, éviter des caractères qui provoquent des erreurs JS…) de la donnée ou même de la transformer (trier les valeurs d'un tableau, mettre en majuscule…).
Vous pouvez retrouver ces fonctions dans la documentation de Phalcon ainsi que des exemples dans mon livre sur Phalcon.
Voici un exemple écrit en Volt :

```
{% block produit %}
    {% for produit in produits %}
      * Nom : {{ produit.nom|e }}
      {% if produit.status == 'actif' %}
        produit : {{ produit.prix + produit.taxes/100 }}
      {% endif %}
    {% endfor %}
{% endblock %}
```

Gestion de formulaires

Avec Phalcon, il est possible de créer différents types de formulaires assez rapidement. Grâce à phalcon-devtools, cette boîte à outils vous permet de créer des formulaires d'ajout / modification / suppression en une seule commande. Il suffit d'indiquer sur quelle table se baser pour créer ces différents formulaires.
Exemple de commande :

```
phalcon scaffold --table-name=utilisateur --template-engine=volt
```

Vous pouvez aussi créer vos formulaires vous-même sans l'aide de phalcon-devtools. À la base, Phalcon propose différents types de champs avec des validateurs pré-paramétrés (Email, Fichier, Date, Zone de texte…).
Vous pouvez aussi y ajouter vos propres validateurs, et en une seule commande Phalcon contrôle les données saisies par l'utilisateur. S'il y a des erreurs, il vous suffit de récupérer les messages puis de les afficher.
Voici un exemple de formulaire :

```
class UtilisateurForm extends Form
{
  public function initialize()
  {
    $oEmail = new Email('email', [
      'placeholder' => 'Saisir un e-mail',
      'class'     => 'form-control',
    ]);
    $oEmail->setLabel('E-mail');
    $this->add($oEmail);

    $oMotDePasse = new Password('mot_de_passe', [
      'placeholder' => 'Saisir un mot de passe',
      'class'     => 'form-control',
    ]);
    $oMotDePasse->setLabel('Mot de passe');
    $this->add($oMotDePasse);
  }
}
```

Après avoir instancié le formulaire créé précédemment, il suffit de passer l'objet à la vue.
En Volt, il suffit d'utiliser l'instruction suivante pour générer le libellé du champ :

```
{{ form.label("email") }}
```

L'objet form génère la ligne suivante :

```
<label for="email">E-mail</label>
```

Pour générer l'input, il suffit d'utiliser la fonction render :

```
{{ form.render('email') }}
```

La fonction produit la ligne suivante :

```
<input type="email" id="email" name="email" class="form-control" placeholder=
"Saisir un e-mail">
```

Gestionnaire d'événement

Phalcon possède un catalogue assez conséquent d'événements sur lequel il est possible de se brancher. Ce système est identique aux Hooks de WordPress. Il est même possible de développer ses propres événements personnalisés.
Il faut savoir que ces événements sont particulièrement utiles : vous pouvez par exemple contrôler l'accès aux différentes pages de votre application.
Un autre exemple pratique : vous pouvez récupérer toutes les exceptions afin de les rediriger dans des logs.
Il y a de bonnes opportunités de conception à saisir avec Phalcon. De cette manière, vous dupliquez très peu de code et votre application est plus simple à maintenir.
Voici un exemple :

```
use Phalcon\Mvc\Dispatcher;
use Phalcon\Events\Manager;

$container->set(
  'dispatcher',
  function () {

    $oGestionnaireEvenement = new Manager();

    $oGestionnaireEvenement->attach(
      'dispatch:beforeExecuteRoute',
      new SecuritePlugin
    );

    $oGestionnaireEvenement->attach(
      'dispatch:beforeException',
      new ErreurPlugin
    );

    $oDispatcheur = new Dispatcher();

    $oDispatcheur
      ->setEventsManager($oGestionnaireEvenement);

    return $oDispatcheur;
  }
);
```

Dans cet exemple, j'associe un plugin (une classe) à l'événement *beforeExecuteRoute* dans le but de contrôler l'accès aux pages (utilisateur connecté / non connecté, niveau de droits…).
L'événement *beforeException* permet de réceptionner l'exception avant de faire peur à l'utilisateur qui parcourt l'application Web.
Si vous souhaitez découvrir d'autres astuces sur cet aspect, je vous invite à consulter mon livre sur Phalcon.

Gestion du cache

Si vous souhaitez aller encore plus loin dans la performance, la mise en cache est une solution très intéressante.
La mise en cache est utile dans différents cas :
- Lorsqu'un calcul complexe et long est effectué et que le résultat change très rarement.
- Lorsqu'une vue / page HTML générée ne change quasiment jamais (par exemple, dans le cas d'un site vitrine statique).
- Lorsque les données récupérées grâce à une requête SQL varient très rarement.

L'équipe de Phalcon conseille toutefois de comparer les performances, avant et après la mise en place de cache.
Le service de cache s'interface avec différents moteurs :
- Apcu,
- Libmemcached,
- Memory (stockage en mémoire),
- Redis,
- Stream (stockage fichier).

Après avoir initialisé le service de cache avec l'un des moteurs précédents, vous pouvez ajouter / récupérer des données en cache depuis n'importe où.
Voici un exemple :

```
$oCache = $this->di->get('cache');
$sCleCache = 'loto.resultats';

$aResultats  = $oCache->get($sCleCache);

if ($aResultats  === null) {

    $aResultats  = Loto::getResultats();

    $oCache->set($sCleCache, $aResultats , 86400);
}
```

Dans cet exemple, je tente de récupérer en cache les résultats du loto. Il y a deux raisons pour lesquelles le cache me renvoie la valeur null : soit je n'ai pas encore initialisé le cache ; soit le cache a expiré. Si le résultat est null, je fais une requête vers un serveur tiers afin de récupérer les derniers résultats du loto. Pour finir, j'enregistre ces nouveaux résultats en cache pour une durée de 86400 secondes (soit 24H).
Je ne suis pas obligé de préciser la durée d'expiration, la durée par défaut est paramétrée à l'initialisation du service de cache.

Le cache de requête SQL

Phalcon va encore plus loin en ce qui concerne les requêtes SQL. Vous pouvez indiquer directement dans une requête PHQL la mise en cache du résultat.
Par exemple :

```
$oRequete = $this->modelsManager->createQuery('SELECT *
          FROM NovaMooc\Models\Utilisateurs
            WHERE prenom LIKE :prenom:');
$oRequete->cache(
  [
    'key'     => 'utilisateurs_phql',
```

```
    'lifetime' => 14400,
  ]
);
```

Grâce à la fonction cache, plus besoin de vérifier si le cache a expiré, s'il faut relancer la requête… Tout est géré par le système.
Vous pouvez trouver d'autres exemples et plus d'explications dans mon livre sur Phalcon.

PSR - PHP Standard Recommendations

Fort de ces expériences précédentes, le créateur de Phalcon tenait à ce que le framework respecte au maximum les PSR (PHP Standard Recommendations). Ces recommandations sont disponibles sur le site https://www.php-fig.org/, et éditées par une liste de membres reconnus dans le domaine des frameworks.
Lors d'un vote de la communauté, Phalcon et son créateur ont été ajoutés à cette table en 2013. Depuis ce jour, Phalcon y est présent, comme d'autres (PrestaShop, Yii, CackePHP) tandis que d'autres sont partis (Laravel, Symfony…).
Le terme FIG signifie Framework Interoperability Group. L'objectif final : que les frameworks utilisent la même interface pour leurs composants. Par exemple la PSR 3 – Logger Interface recommande des noms de fonctions et des usages spécifiques :
Extrait de la PSR 3 :

```
<?php

namespace Psr\Log;

/**
 * Describes a logger instance.
 *
 * The message MUST be a string or object implementing __toString().
 *
 * The message MAY contain placeholders in the form: {foo} where foo
 * will be replaced by the context data in key "foo".
 *
 * The context array can contain arbitrary data, the only assumption that
 * can be made by implementors is that if an Exception instance is given
 * to produce a stack trace, it MUST be in a key named "exception".
 *
 * See https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-3-logger-interface.md
 * for the full interface specification.
 */
interface LoggerInterface
{
    /**
     * System is unusable.
     *
     * @param string $message
     * @param array $context
     * @return void
     */
    public function emergency($message, array $context = array());

    /**
     * Action must be taken immediately.
     *
     * Example: Entire website down, database unavailable, etc. This should
     * trigger the SMS alerts and wake you up.
     *
     * @param string $message
     * @param array $context
     * @return void
     */
    public function alert($message, array $context = array());
```

Les frameworks qui appliquent ces PSR rendent leur utilisation plus accessible au développeur. De plus, les outils tiers ont moins de code à écrire pour s'interconnecter à ces frameworks, ils doivent simplement utiliser les noms normés.
Phalcon s'est fixé de respecter un maximum ces PSR. Pour renforcer le respect de ces normes, la version 4 de Phalcon requiert une extension supplémentaire nommée php-psr.
Cette extension permet à n'importe quel framework d'utiliser les interfaces recommandées par PHP-Fig. Désormais, Phalcon utilise directement ces interfaces et prouve encore plus sa détermination à les appliquer.
À l'heure actuelle, Phalcon gère les PSR-3, PSR-0, PSR-4, PSR-7, PSR-11, PSR-11, PSR-13 et PSR-17. Ces PSR sont détaillés dans la documentation en ligne de Phalcon.

Les inconvénients

L'installation

Phalcon possède beaucoup d'avantages, mais aussi quelques points faibles.
La première chose à savoir : pour l'utiliser en production, il faut avoir la main sur le serveur ou contacter votre hébergeur.
Comme vous le savez désormais, ce qui rend puissant Phalcon, c'est qu'il soit écrit comme une extension. Mais c'est aussi un point faible. Il faut pouvoir l'installer et l'ajouter directement à PHP. Si votre serveur de production est mutualisé, cela risque d'être compliqué.
Dans ce cas, il faut contacter votre hébergeur pour lui demander s'il est possible d'ajouter Phalcon. Ce qui peut peser dans la balance, c'est le fait que Phalcon soit désormais installable grâce à PECL. Cela lui confère une certaine crédibilité auprès des hébergeurs et des administrateurs systèmes. De cette façon Phalcon n'est plus qu'un simple framework Open Source non validé par la communauté. Il est reconnu grâce à PECL.

3

La communauté 3

La communauté de Phalcon s'est concentrée sur ce qu'elle sait faire de mieux, c'est-à-dire de la tech et de l'ingénierie. Laissant de côté, jusqu'ici, la partie communication (meetups, événements, articles…). Ce qui a dû certainement entraîner un manque de visibilité auprès des développeurs.

Conséquence : peu de personnes connaissent Phalcon, même de nom. J'espère qu'en écrivant cet article, je contribuerai à le rendre plus connu – parmi les nombreux langages et frameworks existant sur le marché.

Depuis que j'ai intégré l'équipe de Phalcon, j'ai remarqué ceci : dès qu'un nouveau développeur commence à s'y intéresser, il vient poser quelques questions sur Discord ou sur le forum dédié. En revenant vers eux quelques semaines après, ils sont ravis et ont complètement adopté Phalcon.

Aujourd'hui, Phalcon a bientôt 10 ans d'expérience et sa communauté est petite, mais fidèle.

Dans cette configuration, difficile d'écrire de nouvelles fonctionnalités en grand volume. Mais il y a quelques effets positifs à cela :

- Nous priorisons les tâches les plus importantes.
- Nous élaborons une vraie réflexion quant à la manière de penser et d'écrire les fonctionnalités.
- Nous pouvons facilement prendre en compte les contributions. Les administrateurs prennent le temps de lire toutes les suggestions et y répondent rapidement.

Je ne sais pas si vous avez déjà essayé d'intégrer un projet Open Source tendance. Souvent, les administrateurs sont surchargés par les propositions de code. Ce qui leur demande du temps avant de pouvoir lire et valider votre code.

Avec Phalcon, je n'ai pas rencontré ce problème. Les administrateurs sont accueillants et vous aident à contribuer.

Depuis que je fais partie de l'équipe de Phalcon, je peux même vous accueillir et vous guider dans vos premiers pas.

La peur de faire une grosse erreur vous empêche de vous lancer ? Je vous rassure, le projet est bien verrouillé et de nombreuses tâches sont automatisées. Grâce aux tests et à GitHub Action, chaque code envoyé est passé au peigne fin.

Le code est testé avec Linux, macOS, Windows et si une erreur survient, il est nécessaire de la corriger. Sans quoi, le code ne sera pas intégré dans Phalcon.

De plus, la validation de deux développeurs confirmés est requise. Pas de stress !

Comme en témoigne Franck :

« Phalcon est le meilleur des frameworks customisables, avec une communauté à l'écoute des développeurs. »

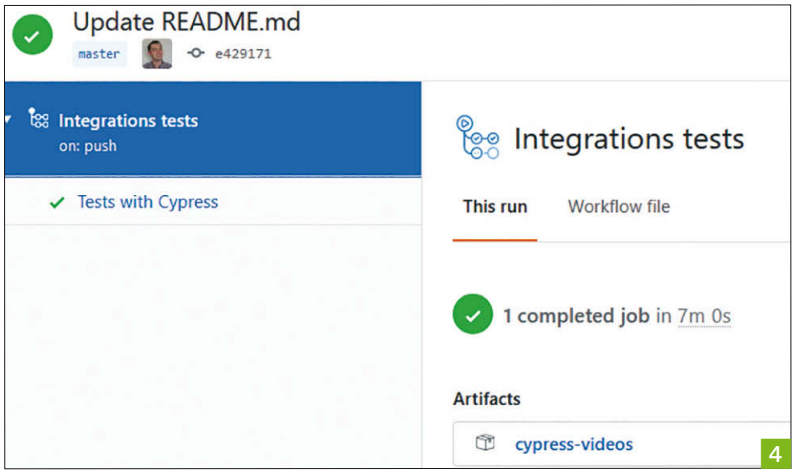
Conclusion

J'espère que cet article vous a donné envie de creuser ce sujet. Si vous souhaitez vous lancer avec Phalcon, je ne peux que vous conseiller le premier livre français paru récemment et disponible en librairie physique et en ligne. Cet ouvrage détaille aussi les nouveautés apportées par la version 4 de Phalcon.

Si vous avez envie de tester un projet Phalcon concrètement, je vous conseille ce projet GitHub :
https://github.com/les-enovateurs/phalcon-nova-mooc
Un GitHub Action est inclus dans le projet : il permet de tester le projet avec Cypress et génère une vidéo de test. Pour la consulter, il suffit de se rendre dans l'onglet Action, puis de cliquer sur le dernier test. Dans la partie Artifacts se trouve un zip téléchargeable contenant une vidéo. 4

Pour plus d'informations :

Interview d'Andres Gutierrez – 2016 : https://link.medium.com/vfycm0Eff5
Interview d'Andrés Gutiérrez – 2015
https://blog.phalcon.io/post/interview-with-phalconphp-creator-andres-gutierrez
Première version de Phalcon : https://github.com/phalcon/cphalcon/releases/tag/phalcon-v0.4.5
Membre du PHP-fig : https://www.php-fig.org/personnel/
Liste des PSR : https://www.php-fig.org/psr/
Exemple PSR 3 : https://www.php-fig.org/psr/psr-3/
Documentation PSR-3 Phalcon : https://docs.phalcon.io/4.0/fr-fr/logger
Livre Phalcon :
https://www.editions-eni.fr/livre/phalcon-3-developpez-des-applications-web-complexes-et-performantes-en-php-9782409022746
Case study de Phalcon par la RTM – Régie des transports métropolitains :
https://www.rtm.fr/sites/default/files/2018-05/phalcon.pdf
D'autres témoignages (en anglais) :
https://forum.phalcon.io/discussion/20477/whats-your-opinion-about-phalcon-php-framework



4

Thomas Moyse
Senior Software Engineer chez **LUCCA**
Son Twitter : https://twitter.com/t8g

Jason De Oliveira
CTO chez **LUCCA**
Son Blog: http://www.jasondeoliveira.com

Un formulaire vraiment réactif avec Angular et RxJS

Dans une application de gestion, un formulaire peut rapidement devenir complexe. Ce champ doit s'afficher ou pas en fonction de la valeur de son voisin, celui-ci devient obligatoire si l'utilisateur a coché cet élément. Il existe déjà des outils pour créer dynamiquement des formulaires (formly par ex.), mais aucun à ma connaissance pour gérer simplement la complexité d'un formulaire dont les champs sont interdépendants.

Le problème à résoudre

Définissons le formulaire que nous nous proposons de rendre réactif. Tout d'abord, nous avons besoin d'un modèle de données pour l'objet que va exposer notre formulaire.

```
export interface Kamoulox {
 canard: string;
 beaujolais: number;
 mitterrand: string;
 pressing: string;
}
```

Maintenant que nous avons cet objet, nous pouvons construire un formulaire classique avec Angular.

```
import { Component, OnInit } from "@angular/core";
import { FormGroup, FormControl } from "@angular/forms";

@Component({
 selector: "kmx-form",
 template: `
  <form [formGroup]="form">
   <div class="form-group">
    <label>canard</label>
    <input formControlName="canard" />
   </div>
   <div class="form-group">
    <label>beaujolais</label>
    <input formControlName="beaujolais" type="number" />
   </div>
   <div class="form-group">
    <label>mitterrand</label>
    <select formControlName="mitterrand">
     <option>parapluie</option>
     <option>homard</option>
    </select>
   </div>
   <div class="form-group">
    <label>pressing</label>
    <input formControlName="pressing" />
   </div>
   <button>Kamoulox !</button>
  </form>
 `
})
export class FormComponent {
 form = new FormGroup({
  canard: new FormControl(),
  beaujolais: new FormControl(),
  mitterrand: new FormControl(),
  pressing: new FormControl()
 });
}
```

Notre formulaire est en place. Maintenant un peu de challenge. Faisons entrer la complexité. Nous allons définir des règles métiers que notre formulaire devra respecter.

- **Mitterrand n'est visible que si beaujolais est supérieur à 100.**
- **Pressing devient youpi si beaujolais est égal à 101 et si Mitterrand est un homard.**
- **Si canard est égal à Lucca, alors beaujolais vaut 101.**

Simple, non ? Face à de telles règles, nous pouvons opter pour ajouter des conditions et du code à la fois dans le composant et dans sa vue. Le problème est que ces règles vont avoir tendance à varier dans le temps. De plus cela risque vite de rendre le formulaire complexe à tester et à maintenir.

Découplage métier / technique

Donc ce que nous voulons, c'est un code testable et maintenable. Nous souhaitons séparer les parties techniques de notre application de ses parties métiers. Nous souhaitons isoler ces règles métiers et les rendre faciles à faire évoluer et faciles à tester.
Pour cela rien de mieux que des fonctions pures. Une fonction pure est testable et donc simple à faire évoluer et à maintenir.

Définition : une fonction pure est une fonction dont la valeur de retour est toujours la même pour les mêmes arguments et dont l'évaluation n'a aucun effet indirect (pas d'effet de bord).

Nous allons donc créer une fonction pour chaque champ dynamique. Pour nous aider à garder des fonctions simples, chacune de ces fonctions sera responsable de l'affichage d'un champ et d'un seul.
Avant de transformer nos règles métiers en fonctions, définissons-en

les entrées et les sorties. Les champs dynamiques dépendent des valeurs de certains autres champs. Ces valeurs seront nos entrées. Une fonction règle doit nous aider à déterminer l'affichage d'un champ : sa visibilité et son libellé. Représentons cela par un modèle de configuration.

```
export type FieldConfig = {
 label?: string;
 visible?: boolean;
};
```

Accompagné de la valeur modifiée du champ, ce sera la sortie de nos fonctions.
Et voici nos règles :

```
export function mitterrandRule(beaujolais) {
  const visible = beaujolais > 100;
  return { config: { visible } };
}
export function pressingRule(beaujolais, mitterrand) {
  const label =
   beaujolais === 101 && mitterrand === "homard"
   ? "youpi"
   : "pressing";
  return { config: { label } };
}
export function beaujolaisRule(canard) {
  if (canard === "lucca") return { beaujolais: 101, config: { disabled: true } };
  return {};
}
```

Mise en place de la réactivité

L'état du formulaire détermine à travers les règles son affichage. À chaque changement de cet état, les règles sont rejouées pour mettre à jour l'affichage.
On va observer l'état de l'objet pour mettre à jour l'affichage. On a un outil pour cela : les observables !

Définition : *Observable*, *Subject*, *BehaviorSubjet* : La librairie RxJS met en oeuvre le design pattern Observer. Un *Observable* est un objet qui encapsule une source de données et en distribue les valeurs à ses abonnés appelés *Observer*. La source ne fait pas directement partie de l'Observable en général. Par exemple la méthode *get* du service *HttpClient* expose un observable créé à partir d'une requête http.
Pour RxJS, un *Subject* est un *Observable* sans source de données, les données seront fournies au fur et à mesure via la méthode *next*. Un abonné reçoit les données à ce moment-là.
Un *BehaviorSubject* est un *Subject* avec une mémoire. Si *Observer* est ajouté à un *Subject* après un *next*, il ne recevra pas l'information. Il n'en va pas de même avec un *BehaviorSubject*. Dans les mêmes conditions, l'*Observer* reçoit la dernière valeur émise. 1
Nous avons donc besoin de deux flux : un flux pour la configuration des champs et un flux pour les données. Pour cela nous allons tout simplement étendre *BehaviorSubject* afin de contrôler le plus finement possible ces flux.
KamouloxConfigSubject est notre flux qui stocke la configuration et délivre chacun de ses changements.

```
// Kamoulox défini plus tôt
import { Kamoulox } from "./kamoulox.service";
import { BehaviorSubject, Observable } from "rxjs";
import { map } from "rxjs/operators";

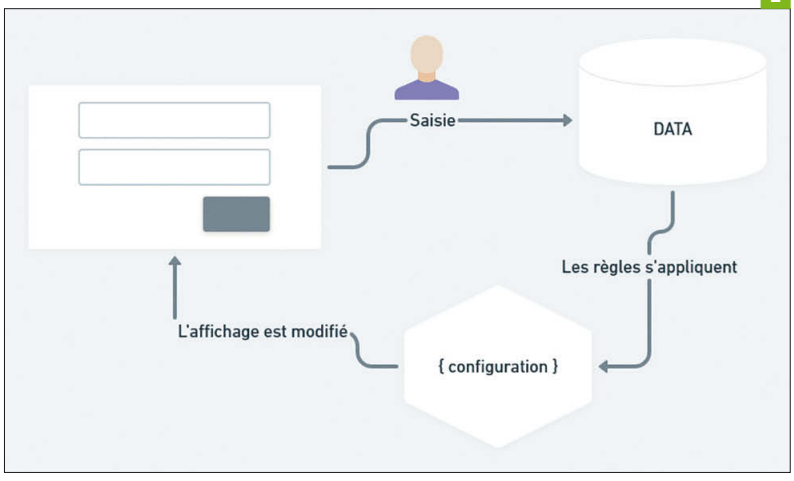
export type FieldConfig = {
 label?: string;
 visible?: boolean;
};
export type FieldConfig$ = {
 [key in keyof FieldConfig]: Observable<FieldConfig[key]>;
};
type Config<T> = {
 [key in keyof T]: FieldConfig;
};
export type KamouloxConfig = Config<Kamoulox>;
export class KamouloxConfigSubject extends BehaviorSubject<KamouloxConfig> {
 // Helpers pour accéder aux configs spécifiques d'un champ
 canard: FieldConfig$ = {
  label: this.pipe(map((c) => c.canard.label)),
  visible: this.pipe(map((c) => c.canard.visible)),
 };
 // ... (idem pour chaque champ)
 // Note : pourrait être géré dans un proxy
 constructor(config: KamouloxConfig) {
  super(config);
 }
 // Mise à jour d'une partie de la configuration
 update<K extends keyof Kamoulox>(key: K, value: Partial<FieldConfig>) {
  this.next({ ...this.value, [key]: { ...this.value[key], ...value } });
 }
}
```

De la même manière, *KamouloxSubject* représente le flux des données modifiées par le formulaire.

```
import { BehaviorSubject } from "rxjs";
import { Kamoulox } from "./kamoulox.service";

export class KamouloxSubject extends BehaviorSubject<Kamoulox> {
 // Mise à jour d'une propriété de l'objet
```

1



```
  change<K extends keyof Kamoulox>(key: K, value: Kamoulox[K]) {
    this.next({ ...this.value, [key]: value });
  }
}
```

Utilisation des flux dans le formulaire

Maintenant que nous avons nos deux flux (données et configuration), il nous reste à faire deux choses : lier ces flux au formulaire et lier ces flux entre eux via les règles.

Intéressons-nous au formulaire. Celui-ci intervient de plusieurs façons.

- D'une part, il est la principale source des données.
- D'autre part, il doit être alimenté par les données.
- Et enfin il doit réagir aux changements de configuration pour adapter son affichage.

Lier le formulaire aux données se fait très simplement :

```
// les données sont écoutées pour remplir le formulaire
this.kamoulox$.subscribe(kamoulox =>
  this.form.patchValue(kamoulox, { emitEvent: false })
);
// le formulaire est écouté pour mettre à jour les données
this.form.valueChanges.subscribe(kamoulox => this.kamoulox$.next(kamoulox));
```

Notez-le *{emitEvent: false }* qui évite une boucle infinie : on ne veut pas que *this.form.valueChanges* se déclenche pour les changements autres que la saisie directe.

Faire réagir le formulaire aux changements de la configuration est assez simple avec les helpers que nous avons créés sur *KamouloxConfigSubject*.

Par exemple, cacher le champ mitterrand en fonction de sa configuration :

```
<form [formGroup]="form">
  <div class="form-group" *ngIf="config$.mitterrand.visible | async">
    <label>mitterrand</label>
    <select formControlName="mitterrand">
      <option>parapluie</option>
      <option>homard</option>
    </select>
  </div>
  <!-- Autres champs ... -->
  <button class="button">Kamoulox !</button>
</form>
```

La pierre angulaire : ré-introduction des règles.

La mise en place des flux dans le formulaire est finie : le formulaire reçoit les données, la saisie utilisateur modifie les données et la configuration met à jour l'affichage.

Il nous reste une dernière étape pour la construction de notre formulaire réactif : faire évoluer la configuration et les données en fonction des règles métiers.

Pour cela, nous allons créer un service dédié *KamouloxService* dont le rôle sera de regrouper les règles métier et les appeler à bon escient.

Ce service sera aussi en charge de créer et d'initialiser les flux de données et de configuration :

```
@Injectable({ providedIn: "root" })
export class KamouloxService {
  // initialisation de la config avec des valeurs par défaut
  config$ = new KamouloxConfigSubject({
    canard: { visible: true },
    beaujolais: { visible: true },
    mitterrand: { visible: true },
    pressing: { visible: true, label: "pressing" },
  });
  // initialisation des données (pourrait être issu d'une api)
  kamoulox$ = new KamouloxSubject({
    canard: "",
    beaujolais: 0,
    mitterrand: "parapluie",
    pressing: "",
  });

  constructor() {}
// ...
}
```

Maintenant, nous allons écouter chaque règle métier en la "nourrissant" avec ce dont elle a besoin. Puis mettre à jour nos deux flux en récupérant le retour de chaque règle métier.

Code sur Github / programmez.com

La méthode *feedRule* est simple. Elle transforme le flux de données en filtrant sur les clés demandées puis applique la règle demandée avant de restituer les résultats.

```
private feedRule<K extends keyof Kamoulox>({ keys, rule }: {
  keys: Array<K>;
  rule: RuleFunction;
}): Observable<
  { config?: Partial<FieldConfig>; } & Partial<Kamoulox>
> {
  return this.kamoulox$.pipe(
    map((kamoulox) => keys.map((key) => kamoulox[key])),
    distinctUntilChanged((x, y) => JSON.stringify(x) === JSON.stringify(y)),
    map(kamouloxValues => rule(...kamouloxValues)),
  );
}
```

Notez l'utilisation de *distinctUntilChanged* qui évite d'appeler la règle si les données d'entrées n'ont pas changé.

Et voilà ! Notre formulaire fonctionne ! Et chaque champ s'adapte bien aux valeurs des autres champs selon les règles établies. Celles-ci sont complètement isolées du reste du code dans des fonctions pures, faciles à tester, à maintenir et à faire évoluer.

Beaucoup d'optimisations peuvent être apportées à ce simple exemple. J'en ai cité quelques-unes dans les commentaires des exemples de code. D'autres challenges restent à relever : avoir des règles asynchrones, regrouper les règles, si beaucoup de règles sont jouées en cascade n'appliquer les changements au formulaire qu'une fois la configuration stabilisée …

N'hésitez pas à donner votre avis via son GitHub (https://github.com/t8g/kamoulox). Et pourquoi pas une librairie complète ?

Christophe Nasarre
En plus de développer dans les technologies Microsoft depuis près de 30 ans, Christophe a travaillé comme relecteur technique pour un grand nombre de maisons d'édition depuis 1996 sur des ouvrages comme "CLR via C#" et "Windows Internals". Vous pouvez retrouver ses articles sur https://codenasarre.wordpress.com, http://labs.criteo.com/category/dotnet et https://medium.com/@chnasarre. Il partage ses outils et du code sur son GitHub https://github.com/chrisnas. Suivez-le sur Twitter avec @chnasarre ! **Criteo**

WSL + Visual Studio = attacher/démarrer une application .NET Core Linux sous Windows 10

Cet article décrit comment s'attacher à un process .NET Core tournant sous Linux avec WSL, mais aussi comment démarrer un process Linux pour une séance de débogage avec Visual Studio.

Venant du monde Windows, je ne trouve pas si facile de développer les applications .NET Core pour Linux. J'ai l'habitude de coder et déboguer dans Visual Studio. Maintenant, j'ai besoin de générer les binaires sous Windows (à cause de l'intégration continue de Criteo), de déployer un artefact vers Marathon afin de pouvoir faire tourner une application dans un container Mesos. Chez Criteo, il est nécessaire de générer tout un tas de services afin de permettre un débogage distant ou l'analyse d'un fichier de dump mémoire.

Mais que faire si je voulais simplement tester et déboguer un petit scénario sur ma machine Windows bien aimée ? Le Windows Subsystem for Linux (ou WSL) est parfait pour faire tourner une application .NET Core Linux mais sous Windows. Cependant, comment s'attacher ou même démarrer une séance de débogage depuis Visual Studio ?

Je vais passer le reste de cet article à décrire comment préparer votre machine Windows 10 pour obtenir ces petits miracles.

Linux sous Windows: bienvenue à WSL !

Je ne vais bien évidemment pas rentrer dans les détails de WSL ici. Il est juste nécessaire de savoir qu'une fois installé, il permet d'ouvrir un shell Linux sur votre machine Windows sans aucune technologie de machine virtuelle. Il est aussi possible de partager des répertoires entre Windows (où vous voulez générer vos applications) et Linux (où il faudra exécuter les assemblies ainsi générées).

La première étape est de faire démarrer la fonctionnalité WSL sur votre Windows: 1

Après un redémarrage, vous devriez pouvoir ouvrir un prompt WSL: 2

Vous pouvez aussi installer votre distribution Linux préférée si vous le désirez. L'étape suivante consiste à installer le runtime .NET Core ou le SDK afin d'être capable de faire fonctionner votre application sous Linux.

Il est maintenant temps de regarder votre disque dur, mais avec une perspective linuxienne: 3

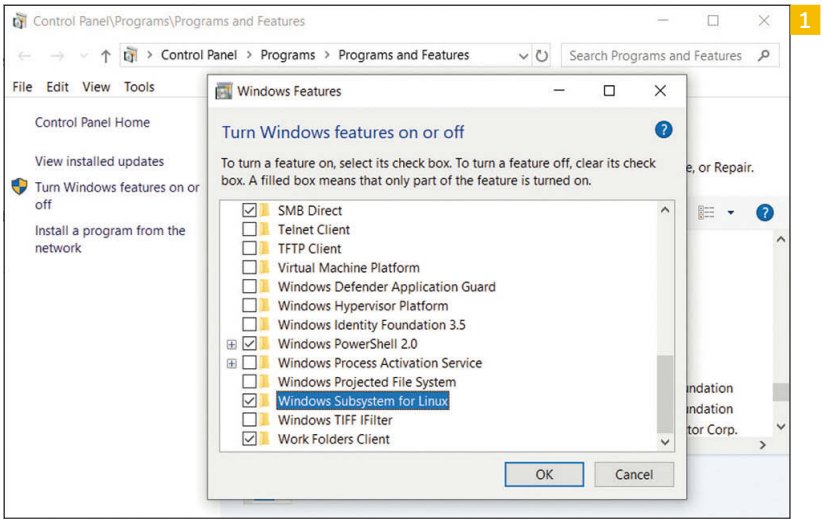
Vos disques sont mappés sous /mnt sans les ":" habituels. Dans mon cas, j'ai créé un répertoire wsl sous mon disque d: afin d'y effectuer mes expériences. Avec Visual Studio, j'ai généré une application TestConsole avec le code suivant :

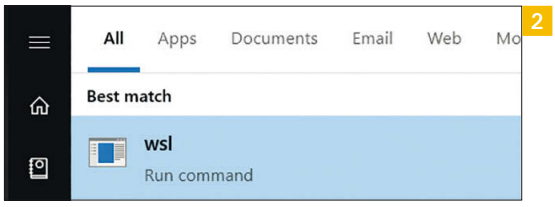
```
using System;

namespace TestConsole
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Enter x to EXIT...");
      while(true)
      {
        var cmd = Console.ReadLine();
        if (cmd.ToLower() == "x")
          return;

        Console.WriteLine($"> {cmd}");
      }
    }
  }
}
```

Je publie l'application afin d'obtenir toutes les assemblies nécessaires 4





Dans un prompt WSL, tapez dotnet suivi du nom de l'assembly de votre application et voilà ! 5
Une application Linux .NET Core tourne sur votre machine Windows.

Comment s'attacher à une application tournant sous Linux

Disons que j'ai détecté un souci et que je veux déboguer l'application Linux avec mon Visual Studio. Lorsque vous demandez à attacher le débogueur de Visual Studio à un process, plusieurs types de connexions sont disponibles : 6
Le type de connexion SSH sera utilisé pour WSL avec le type d'architecture de communication suivant : 7
Le débogueur de Visual Studio envoie des commandes au débogueur Linux vsdbg distant à travers un canal de communication SSH. Voici les différentes étapes à suivre pour installer les composants manquants :

- Par défaut, un serveur SSH est installé avec WSL. Cependant, je n'ai pas été capable de faire fonctionner ces communications avec et j'ai donc dû le désinstaller:

```
sudo apt-get remove openssh-server
sudo apt-get install openssh-server
```

- La configuration SSH nécessite d'être changée afin de pouvoir passer par la sécurité basée sur un nom d'utilisateur/mot de passe nécessaire pour Visual Studio. Si vous préférez une sécurité basée sur des clés, allez à la fin de l'article pour trouver les ressources expliquant comment faire. Si, comme moi, vous ne savez pas utiliser vi en toute efficacité juste pour éditer un fichier, installez nano (merci à mon collègue Kevin Gosse/@kookiz pour ce truc :-)

```
sudo apt-get install nano
```

- Dans le répertoire /etc/ssh/sshd_config, changez le paramètre PasswordAuthentication

```
sudo nano /etc/ssh/sshd_config
PasswordAuthentication yes
```

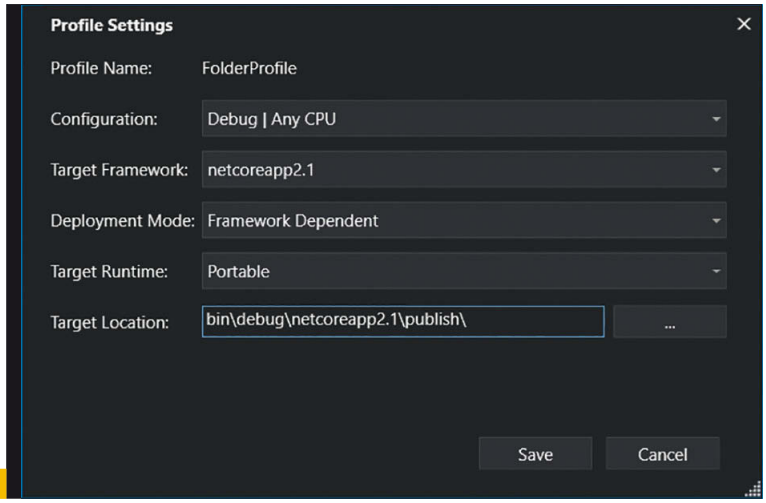
- Redémarrez le serveur SSH

```
sudo service ssh start
```

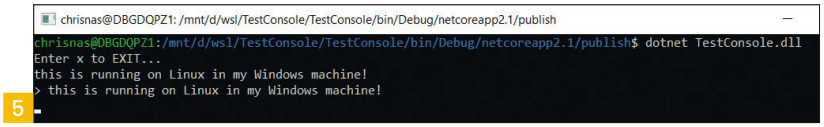
- Vous aurez besoin d'installer unzip pour récupérer vsdbg et le décompresser

```
sudo apt-get install unzip
curl -sSL https://aka.ms/getvsdbgsh | bash /dev/stdin -v latest -l ~/vsdbg
```

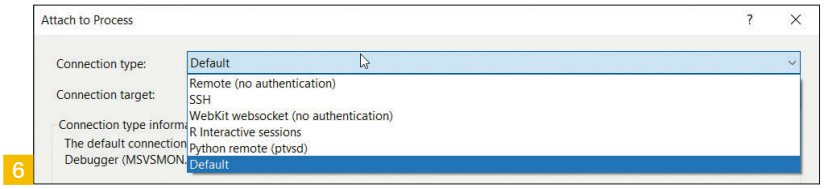
Vous êtres maintenant en mesure de sélectionner SSH comme type de connexion et d'entrer votre nom de machine avant de cliquer sur le bouton Refresh. Ensuite, une nouvelle boîte de dialogue devrait apparaître pour vous demander votre nom d'utilisateur et mode de passe de WSL : 8
Après avoir cliqué le bouton Refresh, la liste du bas devrait contenir les processes Linux tournants dans WSL 9
Sélectionnez votre application .NET Core et cliquez sur le bouton Attach afin de sélectionner le débogueur managé 10
Maintenant, si vous définissez un point d'arrêt dans le code et que



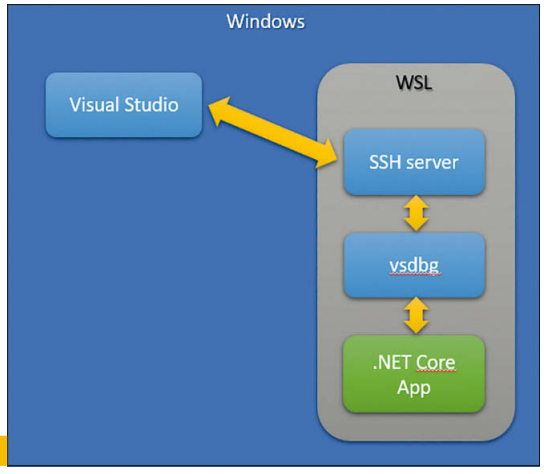
4



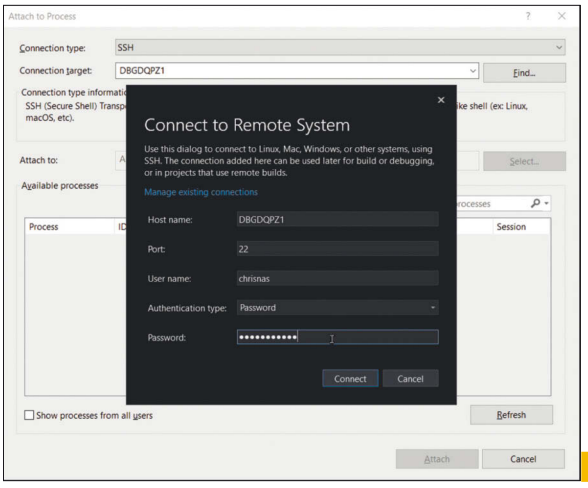
5



6



7



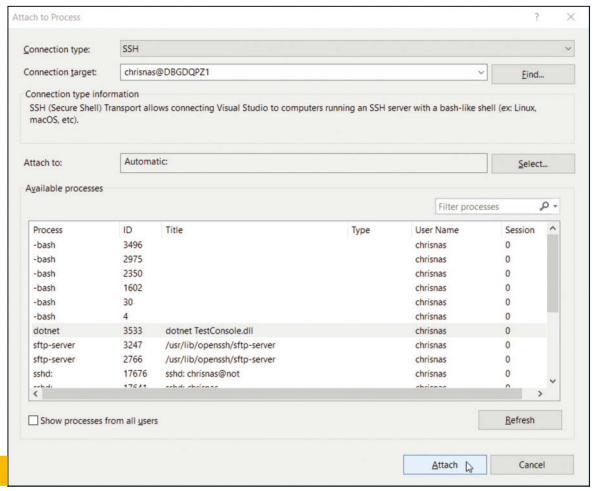
8

vous le déclenchez avec une action appropriée dans votre prompt WSL 11
Le débogueur devrait s'arrêter comme attendu ! 12
Notez que lorsque vous arrêtez votre session de débogage, l'application Linux ne sera pas stoppée; simplement détachée du débogueur et continuera donc à tourner.

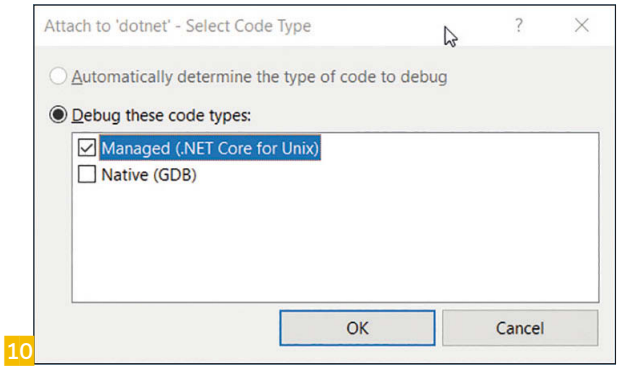
En avant pour F5 !

C'est bien beau de s'attacher à une application Linux, mais il serait encore mieux de la démarrer depuis le débogueur de Visual Studio. Pour atteindre ce but, vous devrez ajouter une autre pièce à l'architecture : 13
Comme détaillé dans cette page WIKI (https://github.com/Microsoft/MIEngine/wiki/Offroad-Debugging-of-.NET-Core-on-Linux---OSX-from-Visual-Studio), il est possible de dire à Visual Studio d'exécuter des actions de débogage grâce au fichier launch.json suivant :

```
{
 "version": "0.2.0",
 "adapter": "c:\\tools\\plink.exe",
 "adapterArgs": "-ssh -pw <password> chrisnas@DBGDQPZ1 -batch -T ~/vsdbg/vsdbg
--interpreter=vscode",
 "configurations": [
  {
   "name": ".NET Core Launch",
   "type": "coreclr",
   "cwd": "/mnt/d/wsl/TestConsole/TestConsole/bin/Debug/netcoreapp2.1/publish",
   "program": "TestConsole.dll",
   "request": "launch"
  }
 ]
}
```



9

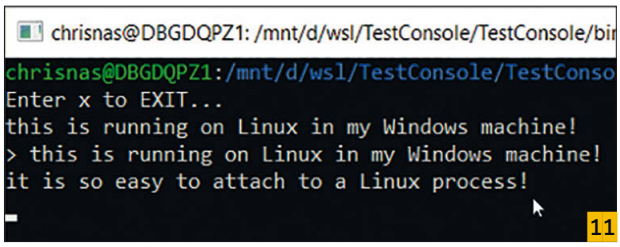


10

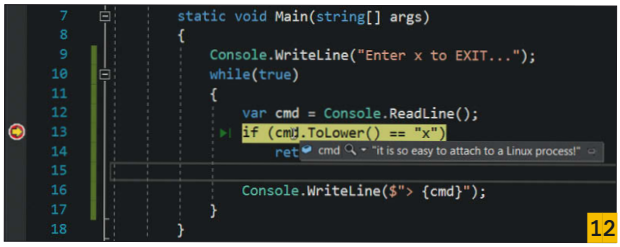
L'outil plink (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.htm) de putty sera utilisé comme adaptateur par Visual Studio pour communiquer avec vsdbg tournant dans WSL. La propriété **adapterArgs** donne les mêmes informations SSH/machine/utilisateur/mot de passe que vous fourniriez à l'interface graphique de Visual Studio UI dans le scénario d'attachement. La section **configuration** définit quelle requête ("launch" au lieu de "attach" et quel répertoire/assembly doit être démarré) sera envoyée à vsdbg.
Une fois le fichier créé, (dans mon cas sous d:\wsl\vs folder), vous n'avez qu'à taper la commande suivante dans le panneau Immediate de Visual Studio :

```
DebugAdapterHost.Launch /LaunchJson:d:\wsl\vs\launch.json
```

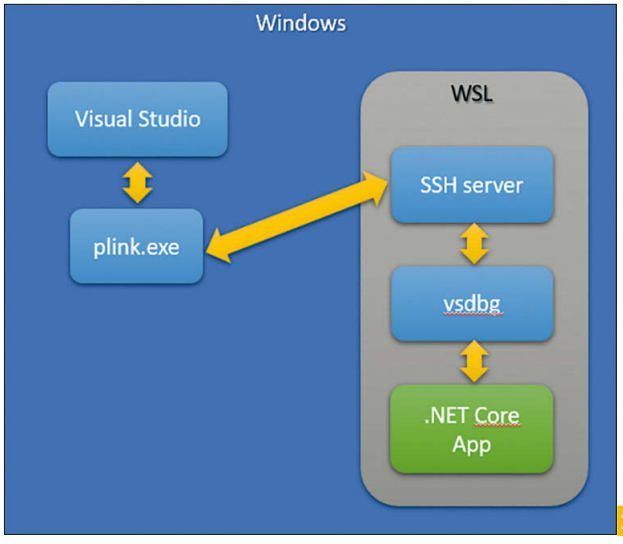
et si vous avez un point d'arrêt en place sur la première ligne de votre application, le débogueur devrait s'y arrêter : 14
Dans un prompt WSL, vous pouvez voir les 2 nouveaux process démarrés 15



11



12



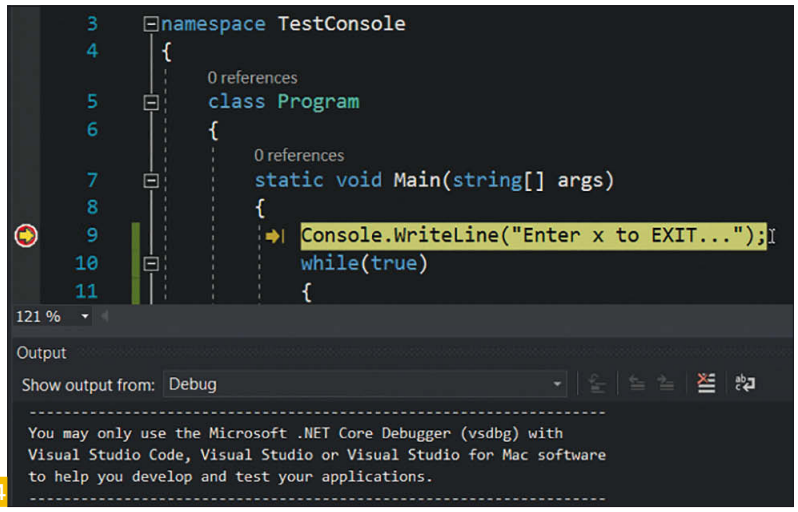
13

Mais attendez !

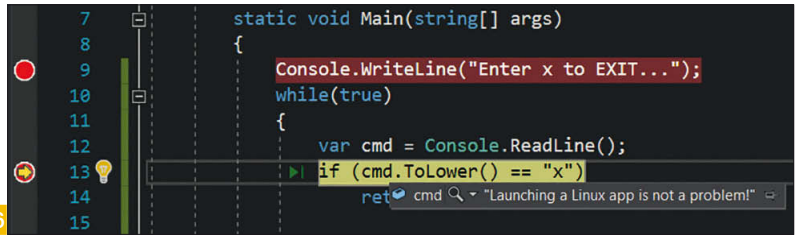
J'ai un grave problème maintenant : je n'ai pas de prompt dans lequel taper mes entrées pour mon application console… Cependant, comme toujours sous Linux, vous avez simplement à écrire dans un fichier pour résoudre le problème. Le flux stdin de votre application est accessible depuis /proc/<pid>/fd/0.
Ainsi, lorsque je tape la commande suivante :

```
echo "Launching a Linux app is not a problem!" > /proc/18341/fd/0
```

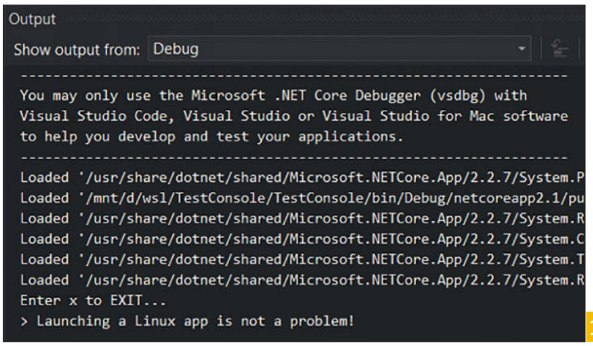
Mon point d'arrêt est déclenché dans Visual Studio : 16
Vous pouvez noter que tout ce qui est envoyé vers la console apparaît aussi dans le panneau Output de Visual Studio 17



14



16



17

15

```
chrisnas 18329  0.1  0.1 3511300 50680 ?       Ssl  12:04   0:00 /home/chrisnas/vsdbg/vsdbg --interpreter=vscode
chrisnas 18341  0.0  0.0 2709616 19204 ?       Sl   12:04   0:00 /usr/bin/dotnet /mnt/d/wsl/TestConsole/TestConsole/bin/Debug/netcoreapp2.1/publish/TestConsole.dll
```

Ressources

Durant mes investigations, j'ai trouvé un certain nombre de ressources qui pourraient vous être utiles (surtout si vous voulez configurer SSL avec des clés au lieu de passer un nom d'utilisateur et son mot de passe en clair).

- VS + WSL basique - https://devblogs.microsoft.com/devops/debugging-net-core-on-unix-over-ssh/
- Bonne description de comment utiliser Visual Studio 2017 pour s'attacher à une application Linux .NET Core tournant dans WSL - http://parsstudent.com/debug-netcore-running-in-wsl-with-vs2017/
- Déboguer .NET Core depuis VS 2017 et WSL - https://www.dotnet-catch.com/2017/04/23/debugging-net-core-from-vs2017-on-windows-subsystem-for-linux/
- VS/C++ avec WSL (décrit comment installer WSL et configurer un serveur open ssh server) - https://www.planetanalog.com/using-visual-studio-and-wsl-to-develop-linux-applications-for-the-iot/
- Configurer SSH sous WSL - https://www.illuminiastudios.com/dev-diaries/ssh-on-windows-subsystem-for-linux/
- Wiki pour VSDBG - https://github.com/Microsoft/MIEngine/wiki/Offroad-Debugging-of-.NET-Core-on-Linux---OSX-from-Visual-Studio
- Excellente description de comment configurer votre environnement de développement avec VS Code et WSL - https://devblogs.microsoft.com/commandline/an-in-depth-tutorial-on-linux-development-on-windows-with-wsl-and-visual-studio-code/

À la différence du scénario d'attachement, si vous arrêtez votre session de débogage, l'application Linux et vsdbg seront tués.

Dorra Dhouid
Consultante senior Python
chez **INVIVOO**
dorra.dhouib@invivoo.com

Introduction au module OS de Python

Ce mois-ci, nos reptiliens abordent un sujet passionnant mais finalement assez peu abordé : le module OS de Python. Il permet d'utiliser des fonctions du système et d'interagir avec lui. Il s'agit d'un des modules les plus importants du langage. Dora nous présente quelques bases. **La rédaction.**

Le module OS est un module python built-in qui nous permet d'interagir avec le système d'exploitation de plusieurs manières différentes :

- Naviguer dans le système de fichiers
- Obtenir des informations sur des fichiers/dossiers
- Rechercher et modifier les variables d'environnement
- Déplacer des fichiers
- Créer/Supprimer des fichiers
- …

Passons maintenant à la pratique, dans la suite de ce tuto nous allons voir les commandes les plus communes du module OS.

Navigation dans le système de fichiers

Pour commencer, on importe le module os. Comme c'est mentionné plus haut, il s'agit d'un module built-in, aucun module tiers ne doit être installé.

```
import os
```

On peut commencer par regarder notre emplacement actuel dans le système de fichiers grâce à la commande **getcwd()** (cwd pour Current Working Directory).

```
# Get current working directory
os.getcwd()
```

Pour notre premier exercice, imaginons que nous voulons lister tous les fichiers contenus dans un dossier spécifique C:/TutoOS/MonDossier, récupérer le fichier « a_renommer.txt » et le renommer en « mon_fichier.txt » 1

Première étape est de nous placer dans ce répertoire, pour cela on utilise la fonction **chdir()** (CHange DIRectory) qui prend en paramètre le chemin du dossier :

```
os.chdir("C:/TutoOS")
```

Si le dossier n'existe pas, nous obtenons une exception :

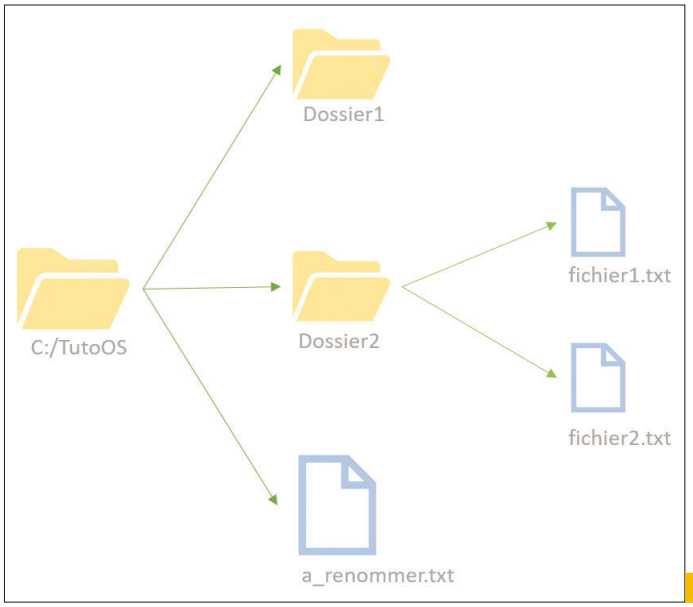
```
FileNotFoundError: [WinError 3] Le chemin d'accès spécifié est introuvable: 'C:/TutoOS'
```

Pour lister le contenu de mon dossier, le module OS fournit la fonction **listdir(path= « . »)**

listdir() renvoie une liste contenant les noms des entrées du répertoire donnés par le paramètre d'entrée path. Cette liste n'inclut pas les entrées spéciales "." et '..' même s'ils sont présents dans le répertoire.

Vous pouvez passer en paramètre le chemin du répertoire que vous voulez parcourir, sinon, par défaut, ce sera le répertoire actuel.

Depuis Python 3.5, une nouvelle fonction **scandir()** permet également de parcourir le contenu d'un répertoire pour obtenir les noms des fichiers dans le chemin d'accès donné, mais elle diffère de **listdir()** de deux manières :



1

- Elle renvoie des objets DirEntry légers qui contiennent la chaîne de nom de fichier et fournissent des méthodes simples qui permettent d'accéder aux données supplémentaires que le système d'exploitation peut avoir renvoyées.
- Elle renvoie un générateur au lieu d'une liste, ainsi **scandir()** agit comme un véritable itérateur au lieu de renvoyer immédiatement la liste complète

Etant donné qu'elle est plus performante que **listdir()**, il est conseillé d'utiliser la fonction **scandir()** pour itérer sur les éléments d'un répertoire.

Pour chacun des éléments retournés par **scandir(),** nous vérifions s'il a le nom du fichier qu'on recherche (pour rappel c'est le fichier « a_renommer.txt ») et si c'est le cas nous le renommons.

Pour renommer un fichier, OS fournit la fonction **rename(src, dest)** qui premier en paramètres le nom du fichier à renommer et le nouveau nom.

Cette fonction permet également de renommer les dossiers.

```
for f in os.scandir():
  if f.name == 'a_renommer.txt':
    os.rename(f.path, 'mon_fichier.txt')
```

Nous avons donc vu une première façon de renommer un fichier, mais cette façon peut être bien simplifiée :

La fonction **os.rename(src, dest)** peut prendre en paramètre les chemins du fichier source, et destination. Et comme nous connaissons le chemin du fichier à renommer, nous pouvons directement utiliser cette fonction sans parcourir le répertoire.

```
os.rename("C:/TutoOS/a_renommer.txt", "C:/TutoOS/mon_fichier.txt")
```

Supposons que nous ne connaissons pas l'emplacement exact du fichier que nous recherchons ; on sait juste qu'il est dans le répertoire « Mon Dossier », mais on ne sait pas dans quel sous répertoire il se trouve. OS fournit une fonction **os.walk()** qui va nous aider à localiser notre fichier.
La fonction **os.walk()** permet de lister récursivement tous les fichiers et les dossiers à partir d'un point dans l'arborescence.
Cette fonction est un générateur qui retourne des tuples de dossiers, sous-dossiers et fichiers :

```
for dirpath, dirnames, filenames in os.walk("C:\TutoOS"):
    print("Current Path:", dirpath)
    print("Directories:", dirnames)
    print("Files:", filenames)
```

Nous obtenons le résultat suivant :

```
Current Path: C:\TutoOS
Directories: ['Dossier1', 'Dossier2']
Files: ['mon_fichier.txt']

Current Path: C:\TutoOS\Dossier1
Directories: []
Files: []

Current Path: C:\TutoOS\Dossier2
Directories: []
Files: ['fichier_1.txt', 'fichier_2.txt']
```

Accès aux variables d'environnement

Autre fonctionnalité clé du module OS est l'accès aux variables d'environnement.
Imaginons que nous avons une variable d'environnement VAR_1, il suffit d'appeler la fonction get() de os.environ pour récupérer la valeur de ma variable.
La fonction get() prend en paramètre le nom de ma variable.

```
os.environ.get("VAR_1")
```

Exécution des commandes

Le module OS fournit deux fonctions qui permettent à un script Python d'exécuter n'importe quelle commande du système d'exploitation (que vous pourriez taper dans une fenêtre de console):
os.system: Exécute une commande shell. La sortie de la commande apparaît normalement dans le flux de sortie standard de la session Python ou du programme.
Cette fonction reçoit la commande en chaine de caractère.
Exemple :

```
os.system("dir")

...
20/02/2020 13:22    <DIR>          .
20/02/2020 13:22    <DIR>          ..
20/02/2020 13:22    <DIR>          SousDossier_1
               0 fichier(s)              0 octets
....
```

os.popen: Exécute une commande shell et retourne un objet de type file que vous pourrez utiliser pour accéder au flux d'entrée et de sortie de la commande exécutée.

Creation/Suppression de fichiers

Maintenant qu'on a fini avec notre premier exercice, passons à une autre fonctionnalité du module OS : la création/suppression de fichiers et/ou répertoires.
Supposons que nous voulons créer un nouveau répertoire dans «C:/TutoOS » que nous allons appeler SousDossier_1.
Il est facile de créer un répertoire grâce à la fonction **makedirs()**.
Cette fonction se charge de créer les répertoires intermédiaires s'ils n'existent pas.
Cependant, il est nécessaire de vérifier au préalable s'il n'existe pas déjà et de résoudre les problématiques liées à une situation de compétition.
À partir de la version 3.4, Python gère nativement ce cas. La fonction **makedirs()** accepte un paramètre supplémentaire **exist_ok**.
Si **existe_ok** est égal à true, la méthode ne renverra pas d'exception si le répertoire existe déjà.

```
os.makedirs("C:TutoOS/SousDossier_1", exist_ok=True)
```

Pour supprimer un fichier il suffit d'appeler la fonction **os.remove(path)** avec en paramètre le chemin du fichier.
Pour supprimer un répertoire, c'est la fonction **os.rmdir(path)** avec en paramètre le chemin du répertoire. Attention, le dossier doit être vide pour être supprimé.
Pour supprimer un répertoire non vide, on utilise la fonction ***shutil.rmtree(path)***

Introduction à os.path

Je profite de ce petit pour parler du module **os.path** qui apporte des fonctionnalités bien utiles à la manipulation des fichiers notamment vérifier qu'un fichier existe, récupérer le chemin d'un fichier/répertoire, vérifier qu'un élément est un répertoire/fichier ….
Voici quelques exemples bien pratiques :

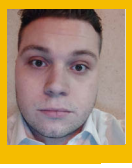
- **os.path.dirname(file_path)** : retourne le chemin du répertoire parent de « file_path »
- **os.path.exists(file_path)** : retourne True si le chemin existe sinon False. A utiliser avant de supprimer un fichier par exemple
- **os.path.isdir(dir_path)** : retourne True si l'élément est un répertoire, False sinon. On retrouve son équivalent pour les fichiers : **os.path.isfile(file_path)**
- **os.path.join()** : Construit un nom de chemin à partir d'un ou de plusieurs noms de chemins partiels reçus en paramètres. Ainsi, il se charge d'ajouter et adapter le séparateur à l'OS courant.

Conclusion

Ce tuto était un aperçu de tout ce qu'on peut faire avec le module OS, je n'ai présenté que les fonctions que j'utilise le plus souvent. Le module fournit d'autres fonctionnalités qui sauront répondre à vos besoins de matière de manipulation de fichiers.

Christophe Pichaud
Lead Architect
Microsoft chez Infeeny
christophep@cpixxi.com |
www.windowscpp.com

Julie Lacognata
Infeeny
julie.lacognata@infeeny.com



Kevin Ansard
Infeeny
kevin.ansard@infeeny.com

infeeny | econocom

Déploiement d'une application API REST sur Docker / Kubernetes sous Linux

Partie 2

Dans cet article, nous allons réaliser une application Web en .NET Core 3.1 avec un déploiement Kubernetes sous un environnement Linux. Le but de cette application est de vous montrer comment utiliser et déployer avec Kubernetes. Mais surtout de vous expliquer le concept même de Kubernetes et pourquoi ce sont, avec Docker, des technologies phares du moment.

Passons maintenant à Kubernetes via Microk8s.

Kubernetes, qu'est-ce que c'est ?

Pour rappel, Kubernetes n'est ni plus ni moins qu'un orchestrateur. C'est un ensemble de services réseaux qui permet de lancer des pods Docker en cluster avec une gestion de load-balancing. Pour résumer, il gère les conteneurs et les pods. On va utiliser Kubernetes qui lui-même utilise des objets pour décrire l'état souhaité de votre cluster (c'est là où s'exécute le code). Il va en outre, nous permettre de définir les applications souhaitées, les processus, le nombre de replicas, le volume et plein d'autres paramètres. Voici un petit lexique Kubernetes :

Objets

Une fois les états de vos objets déclarés, Kubernetes fait appel au Control Plan, qui lui-même va changer les états de vos objets de base en état souhaité. Plus simplement, le Control Plan est un ensemble de processus dans votre cluster. Comme nous l'avons dit précédemment, Kubernetes fonctionne avec des objets et va définir un ensemble de choses. Néanmoins, il y a des objets qui sont inclus de base :

Pods

Les pods correspondent au processus en cours d'exécution et encapsulent un ou des conteneurs applicatifs. Ce sont des instances uniques, cela signifie qu'ils possèdent :

- Une IP unique ;
- Un fichier qui indique comment le conteneur doit être exécuté ;
- Des ressources de stockage.

Ce qu'il faut retenir, si nous devions résumer un peu tout ça, c'est que les pods peuvent correspondre à une application ayant sa propre mémoire, sa propre IP. Néanmoins, un pod a une durée de vie définie et ne sera pas en mesure de se relancer automatiquement de lui-même.

Replica

Correspond au nombre de répliques d'un même pod. Il ne faut pas oublier que chaque pod a une IP unique, une mémoire unique, etc. Même si, ils partent de la même encapsulation de conteneurs.

Services / Micro-services

Ils vont définir la logique entre les pods (entre eux) et la politique d'accès à ceux-ci. À quoi ça sert ? Comme nous l'avons énoncé auparavant, les pods ont des IP uniques, mais imaginons une application avec plusieurs instances (replicas de pods) de backend et qui ont chacune une IP, cela peut poser problème. La principale question étant : comment peut-on y accéder si les IP changent ? C'est pour cette raison que les services sont existants, car ils vont simplifier l'accès entre les différents pods et leur politique d'accès.

Volume

Maintenant que nous avons expliqué ce que sont les pods et les services, il reste un problème majeur : nos pods peuvent mourir et donc perdre leurs données. Ou tout simplement, nous avons également besoin, si un pod possède plusieurs conteneurs de partager la mémoire entre eux. Donc, comment allons-nous partager cette mémoire sachant que chaque instance possède sa propre mémoire indépendante des autres ? C'est pourquoi le Volume est une sorte de mémoire partagée entre les conteneurs d'un même pod. Concrètement, le Volume, initialement, n'est qu'un dossier accessible aux différents conteneurs. Un Volume doit être défini, ainsi que sa politique d'accès.

Objets : Contrôleurs

Nous allons maintenant vous présenter les différents contrôleurs que l'on retrouve sur Kubernetes. Un contrôleur n'est ni plus ni moins qu'un objet Kubernetes, mais qui a la particularité d'ajouter une fonctionnalité. On peut résumer ça en disant que ce sont des « super objets » !

ReplicaSet

Un ReplicaSet a pour but de définir et de maintenir un ensemble de pods. Pour ce faire, nous utilisons le sélecteur pour identifier les pods, que le contrôleur va posséder. Imaginons que nous demanderons à un pod de se répliquer 3 fois. Si un de ces pods meurt, le ReplicaSet va en recréer un. Si nous ne voulons que 2 répliques finalement, le contrôleur ReplicaSet va donc supprimer un des pods. Il a donc pour but de rendre stable cet ensemble de pods.

Deployments

Le Deployment est une sorte de gestionnaire qui va nous permettre de changer l'état de nos pods ou de nos ReplicaSet. Autrement dit, le nombre de replicas voulu, ou autre statut de ceux-ci (comme par exemple l'arrêt des ReplicaSet, le stop du Deployment, l'augmentation de la charge traitable, etc.). En outre, ce gestionnaire vous permet de modifier les statuts et parfois même de revenir à des versions antérieures de déploiement.

DaemonSet

Le DaemonSet permet de gérer les nœuds (ou n'odes), plus précisément les pods qui doivent être exécutés dans ces nœuds. Si des nœuds sont ajoutés, le contrôleur va alors ajouter les pods demandés. En opposition, s'ils sont supprimés (les nœuds), les pods qui étaient utilisés vont être récupérés par le DaemonSet. La suppression de ce contrôleur va nettoyer les pods qui avaient été ajoutés.

StatefulSet

Le contrôleur StatefulSet gère le déploiement et la mise à l'échelle d'un ensemble de pods. Le StatefulSet est assez similaire au Deployment (le contrôleur), car il y a cette notion de gestion des pods, mais contrairement au Deployment il y a le terme de « sticky identity » (en français « identité collante ») pour chacun de ses pods. Ce qui signifie que les pods sont créés à partir de la même spécification, mais qu'ils ne sont en aucun cas interchangeables : chacun d'entre eux possède un ID unique et permanent. Le StatefulSet est une sorte de Deployment statique et que par conséquent il possède des limites.

Jobs

Les Jobs permettent de définir un cycle de vie, on va donc leur définir un nombre de réussites voulues. Ce qu'on veut dire par là, c'est que le contrôleur va créer des pods et va « regarder » s'ils se terminent avec succès. Une fois le nombre de succès atteint, la tâche est terminée. Le Job peut donc être supprimé, ce qui nettoiera les pods créés.

Notions d'architectures

Nœuds

Les nœuds peuvent être assimilés à des machines virtuelles ou physiques. Chacun de ces nœuds possède un ou plusieurs services qui sont nécessaires à la gestion des pods. Les nœuds sont gérés par le composant **master**, plus précisément par le contrôleur Node (Node Controller en anglais, NDLR.). Ce contrôleur a plusieurs rôles, nous allons les lister ci-dessous :

- Il tient à jour la liste interne de ses nœuds ;
- La surveillance de la santé des nœuds (qui le surveille toutes les ***n*** secondes, ***n*** étant définit au préalable par nos soins) ;
- Affectation d'un bloc CITR (d'un **master**).

Mapping de port

Le mapping de port, comme son nom l'indique, est censé « mapper » les ports. Ce que l'on entend par « mapper » est la possibilité d'attribuer un port à un objet spécifique ainsi que son adresse IP. Dans le cas de Kubernetes, les Pods (objet) peuvent rechercher les services à l'aide de leur namespace. Chacun de ces services se charge quant à lui d'attribuer les IP et les ports à chaque pods. Un service peut avoir défini plusieurs ports.

```
spec:
    selector:
        app: MyApp
    ports:
        - name: http
          protocol; TCP
          port: 80
          targetPort: 9376
        - name: https
          protocol: TCP
          port: 443
          targetPort: 9377
```

Comme nous pouvons le voir ci-dessus, nous définissons plusieurs ports. Nous pouvons définir manuellement les IP de nos pods, en sachant qu'il vous faudra donc toujours définir un service (ou DNS) qui se chargera de l'attribution des ports et des IP. Pour ce faire, les pods auront besoin des namespaces pour accéder au service DNS (ou à d'autres services).

Mise en place de Kubernetes

(cf. Installation Kubernetes)

La commande :

```
sudo microk8s.enable dns dashboard registry
```

Permet de créer un service avec le nom donné en local. Plus précisément, un registry est une sorte de « repository » d'images. Grossièrement on peut dire que c'est un espace de stockage d'images.

Nous allons, à partir de maintenant configurer le portail Kubernetes. Celui-ci est utile notamment pour vérifier les « statuts » de vos services :

```
sudo microk8s.enable dns dashboard registry ingress
```

Cette commande ci-dessous, va nous permettre de créer une sorte de contrôler pour gérer les différents services.

```
sudo microk8s.kubectl proxy --accept-hosts=.* --adress=0.0.0.0 &
```

Cette commande nous permet d'accepter n'importe quel port et adresse.

```
sudo microk8s.kubectl -n kube-system edit deploy kubernetes-dashboard -o yaml
```

La dernière commande nous permet d'ouvrir VIM pour éditer un fichier de configuration du portail.

Dans la section spec / container / args nous allons ajouter la ligne : --enable-skip-login

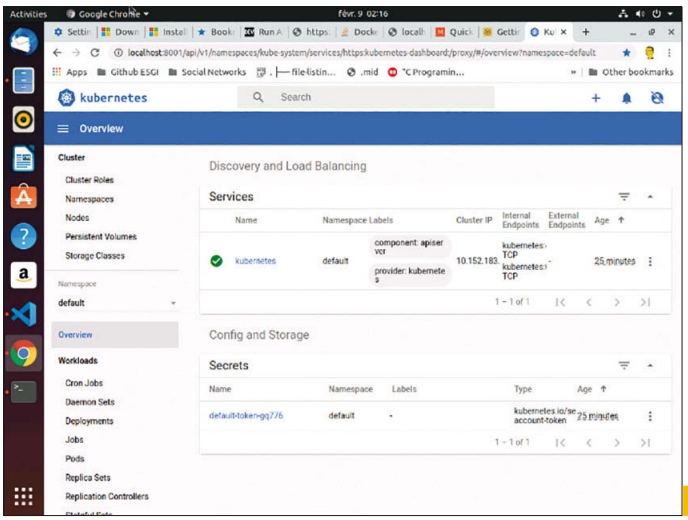
La séquence de touche est la suivante :

- ESC
- I
- --enable-skip-login
- ESC
- :w
- :q

Va vous permettre de générer ça :

```
spec:
    containers:
    - args:
      - --auto-generate-certificates
      - --namespace=kube-system
      - --enable-skip-login
```

Nous pouvons enfin accéder au portail via l'URL suivante : http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/ 6



6

Nous allons pouvoir utiliser Kubernetes à partir de là. Ce qu'il faut savoir c'est que pour déployer une application sur Kubernetes, nous allons utiliser Helm. Helm, pour rappel, est un gestionnaire de paquets pour Kubernetes comme annoncé plus haut lors de l'installation. Au préalable nous allons lancer la commande suivante :

```
sudo microk8s.kubectl config view --raw >~/.kube/config
```

La procédure pour faire tourner une application sous Kubernetes est la suivante : Kubernetes a besoin d'une image qu'il peut extraire lors de son utilisation. Il nous faudra donc :

- Créer notre image et l'envoyer

Nous allons d'abord builder en local sous Docker et tagger l'application (création de l'image) :

```
sudo docker build . -t localhost:32000/app1:latest
```

- Nous allons pousser l'application dans un registry locale :

```
sudo docker push localhost:32000/app1
```

- Nous allons déployer l'application via Helm (pour information, Helm va chercher une image, il est donc nécessaire de charger notre image au préalable, pour rappel) !

À la fin de toutes ces étapes, voilà l'architecture de vos fichiers que vous devez retrouver dans le dossier kubernetes_project : 7

7

Déploiement sous Helm

Pour réaliser le déploiement, nous avons besoin des 4 fichiers suivants dans le dossier **chart** :

- ./chart
  - charts
  - Chart.yaml
  - templates
    - deployment.yaml
    - service.yaml
  - values.yaml

Fichier Chart.yaml

```
name: aspnet3-demo
version: 1.0.0
```

Fichier values.yaml

```
environment: development

apphost: k8s

label:
 name: aspnet3core

container:
 name: aspnet3
 pullPolicy: IfNotPresent
 image: localhost:32000/aspnet3k8s
 tag: latest
 port: 80
replicas: 3

service:
 port: 8888
 #type: ClusterIP
 type: NodePort
```

Fichier templates/services.yaml

```
apiVersion: v1
kind: Service
metadata:
 name: {{ .Release.Name }}-service
 labels:
  app: {{ .Values.label.name }}
spec:
 ports:
 - port: {{ .Values.service.port}}
  protocol: TCP
  targetPort: {{ .Values.container.port }}
 selector:
  app: {{ .Values.label.name }}
 type: {{ .Values.service.type }}
```

Fichier templates/deployment.yaml

Code complet sur programmez.com & github

Comme vous pouvez le constater, c'est le fichier values.yaml qui compte le plus :

- Champs container / image : nom de l'image docker dans la registry locale ;
- Champs replicas : c'est le nombre de pods Docker à déployer ;
- Champs service / type : NodePort indique que le service est accessible depuis l'extérieur du Cluster.

Une fois que les fichiers sont prêts, on peut déployer à l'aide de la commande :

```
helm install aspnet3release ./chart
```

La commande ***install*** permet de déployer l'image et a3 correspond au nom donné au déploiement. Maintenant, afin de vérifier que les services se sont correctement déployés, nous pouvons exécuter la commande suivante :

```
sudo microk8s.kubectl get all
```

La réponse à cette commande est la suivante :

```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/aspnet3release-deployment-9b94ff7c8-f7xfh   1/1     Running   17         38d
pod/aspnet3release-deployment-9b94ff7c8-fjwsq   1/1     Running   18         38d
pod/aspnet3release-deployment-9b94ff7c8-xn6sx   1/1     Running   20         38d
pod/kubernetes-bootcamp-69fbc6f4cf-zzc77        1/1     Running   22         39d

NAME                     TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/aspnet3release-service   NodePort   10.152.183.30   <none>        8888:31404/TCP   38d
service/kubernetes-bootcamp      NodePort   10.152.183.26   <none>        8080:32667/TCP   39d

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/aspnet3release-deployment   3/3     3            3           38d
deployment.apps/kubernetes-bootcamp         1/1     1            1           39d

NAME                                           DESIRED   CURRENT   READY   AGE
```

```
replicaset.apps/aspnet3release-deployment-9b94ff7c8  3     3     3     38d
replicaset.apps/kubernetes-bootcamp-69fbc6f4cf       1     1     1     39d
```

Si vous souhaitez affiner votre recherche, vous pouvez modifier la commande ci-dessus en indiquant le sélecteur (sudo microk8s.kubectl get all –selector app=aspnet3release). Avec la commande « ip a », on récupère l'adresse IP de la machine Linux : 172.23.58.135.

Pour vérifier que vos pods sont bien exécutés (état RUNNING), tapez la commande :

```
sudo microk8s.kubectl get pods
```

La réponse à cette commande est la suivante :

```
NAME                                    READY   STATUS    RESTARTS   AGE
aspnet3release-deployment-9b94ff7c8-f7xfh   1/1     Running   17         38d
aspnet3release-deployment-9b94ff7c8-fjwsq   1/1     Running   18         38d
aspnet3release-deployment-9b94ff7c8-xn6sx   1/1     Running   20         38d
kubernetes-bootcamp-69fbc6f4cf-zzc77        1/1     Running   22         39d
```

Il est important que vos pods soient en état RUNNING comme les 3 premières lignes, qui correspondent à nos trois réplicas. Nous allons maintenant vérifier que nos services sont correctement exécutés :

```
sudo microk8s.kubectl get services
```

La réponse à cette commande est la suivante :

```
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
aspnet3release-service   NodePort    10.152.183.30    <none>        8888:31404/TCP   38d
kubernetes-bootcamp      NodePort    10.152.183.26    <none>        8080:32667/TCP   39d
```

8

En ce qui concerne le mapping de port, nous vous invitons à retourner voir la section Kubernetes / Notions d'architectures / Mapping de port.

On consulte le portail sur une autre machine :

http://172.23.58.135:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#/workloads?namespace=default

Le portail permet de visualiser les services, les pods et les autres ressources Kubernetes.

Sur la ligne service on repère que le port est mappé comme suit : 8888 :31404 : c'est l'adresse externe du cluster :

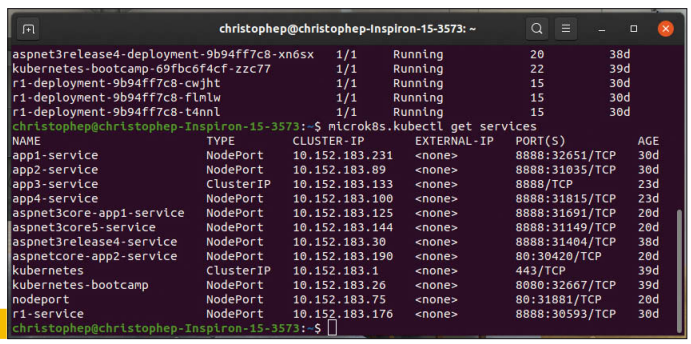
Lançons un browser depuis une autre machine allons sur

http://christophep-inspiron-15-3573:31404/WeatherForecast 9

Petites commandes bien utiles

Si vous souhaitez savoir à quels nœuds appartiennent vos différents services ou vos pods, nous vous conseillons d'exécuter la commande :

```
sudo microk8s.kubectl get nodes
```



8

La réponse à cette commande est la suivante :

```
NAME                  STATUS   ROLES    AGE   VERSION
christophep-inspiron-15-3573   Ready    <none>   39d   v1.17.3
```

Ici nous pouvons voir que nous n'avons qu'un seul node et qu'il correspond à notre machine virtuelle.

Utilisation du Linux depuis un PC Windows

Avec SSH, on se connecte à distance au PC Linux via Windows Subsystem for Linux v2 (WSL v2) : c'est une solution. Le confort ultime, c'est d'utiliser VS Code en mode SSH : on se connecte au PC Linux en donnant un user/password et on dispose d'un terminal en bas de fenêtre. Voilà à quoi cela ressemble : 10

Ce n'est pas forcément trivial de prime abord pour un développeur habitué à l'environnement Windows. Si vous souhaitez continuer de coder sous Linux, nous vous conseillons d'utiliser les méthodes indiquées plus haut.

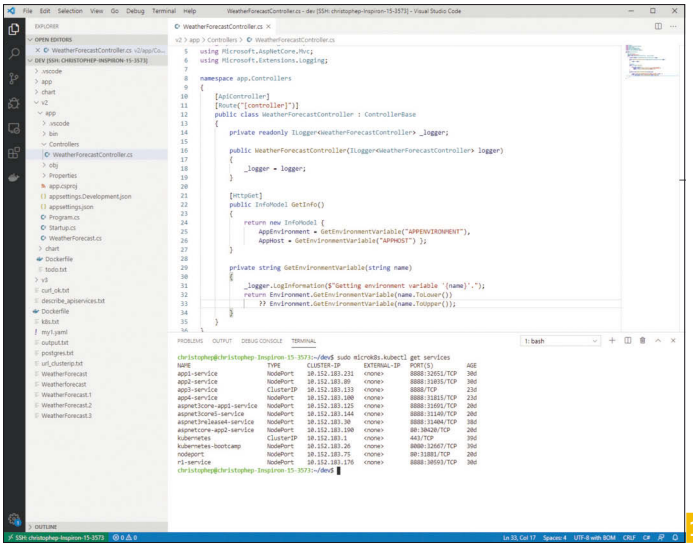
Trucs et astuces pas chers

Un PC pour Linux pas cher ? Allez chez Dell, Cdiscount ou Amazon… Cherchez une machine Pentium avec 8 Go de RAM et un disque SATA de 1 To avec une distribution Linux. Vous économisez la licence Windows. Vous en aurez pour 350 € pour une belle machine. Vous achetez un disque dur SSD 250 Go à 40€ et vous l'insérez dans le portable. C'est ce que j'ai fait. Je dispose d'une machine de course pour faire tous mes développements. 8 Go sous Linux, c'est la fête ! Les processeurs Pentium sont puissants et bien moins cher que les Core iX.

Conclusion

Si on résume le concept de Kubernetes et de Docker : imaginez que Kubernetes soit le chef d'orchestre et un container Docker, le musicien. En outre, Kubernetes ne sert à gérer que l'organisation de vos micro-services / containers Docker. Et Docker quant à lui, vous permet de les créer. Kubernetes n'est pas très compliqué. Le problème c'est que les opérations se font via la ligne de commande, comme Docker. Il y a bien le portail qui permet de visualiser les informations, mais le tooling est faible en quantité.

9



10

Jon Mikel Inza
Référent IoT et Azure chez Exakis-Nelite
De formation technique, mais avec un parcours varié, je travaille aujourd'hui en tant que Référent IoT et Azure. Mon rôle principal est d'aider nos clients à construire des solutions cohérentes et optimisées par rapport à leurs besoins.

IoT : comment créer son IoT, le configurer et l'exploiter

Partie 1 : le décor

L'IoT est un sujet très à la mode ces dernières années. Il est généralement connu du grand public par le CIoT — Consumer IoT — (domotique, wearables, etc.) ce qui ne représente qu'une petite partie de l'univers IoT. Ce dernier intègre différents domaines avec des latitudes très importantes (business, technologie, fonctionnel) et devient le vecteur d'une vraie transformation digitale (industrie, santé, énergie/environnement, agriculture, automobile, enseignement, smart cities, smart buildings, transports, logistique, etc.).

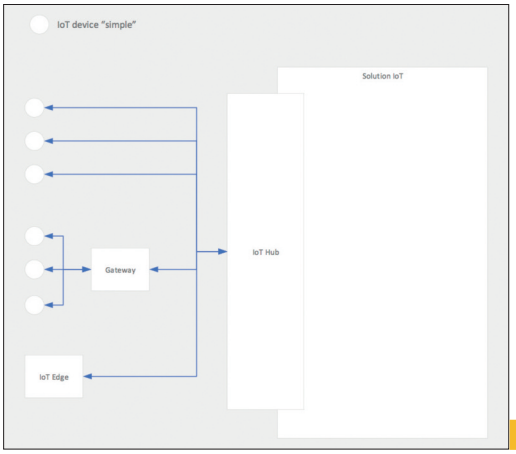
Avec Programmez!, nous souhaitions prendre un peu de recul sur cet effet de mode et donner un premier aperçu de l'univers IoT dans sa globalité (B2B, B2C).
Pour cela nous envisageons de publier une série d'articles qui démarrent avec la présentation de quelques généralités. La seconde partie de l'article illustre un cas concret de mise en place d'une solution IoT.

L'univers IoT

L'IoT d'aujourd'hui est l'évolution naturelle de disciplines qui existaient déjà dans des domaines comme l'industrie (M2M, Machine To Machine). Le cloud et les évolutions technologiques ont permis un changement de décor radical.
En très synthétique, l'IoT regroupe un ensemble de disciplines reposant sur :

- Un objet (device) capable de :
  - Récolter des informations (capteurs, cameras, etc.) ;
  - Transmettre des informations D2C — Device To Cloud - (protocoles divers, via des passerelles ou en direct, données brutes ou partiellement traitées) ;
  - Selon les capacités de l'objet :
    - Agir en fonction des informations collectées,
    - Agir en fonction de commandes ou messages reçus depuis le cloud ou des applications (peut dépendre aussi des protocoles et réseaux de communication choisis).
- Une solution capable de :
  - Gérer les objets (enregistrement, identification, sécurité, désactivation/activation, commandes C2D – Cloud To Device — messages C2D, mises à jour, etc.)
  - Gérer et traiter les flux de données (du point de vue IoT) ;
  - Stocker les données en fonction de leur type d'utilisation ;
  - Exploiter les données ;
  - Construire et intégrer des modèles intelligents sur la base de la télémétrie récoltée par la solution (Intelligent Cloud et Intelligent Edge).

1



1 - *IoT Flows*

Quelques-uns des enjeux les plus communs auxquels est confronté l'IoT sont :

- La sécurité ;
- La gestion des devices (mises à jour, commandes, intégration de nouveaux devices, etc.) de manière unitaire ou par flottes ;
- La gestion des flux d'information de manière optimisée ;
- Le traitement des données pour générer de la valeur (business, métier, etc.).

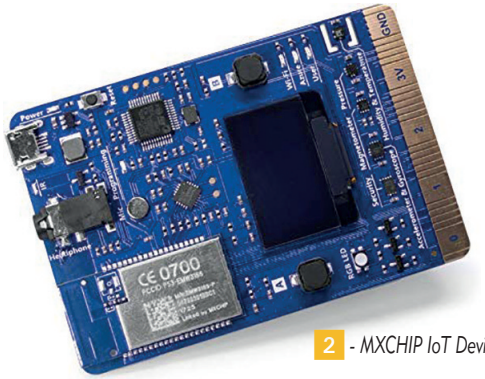
À ceci se rajoute le besoin de compétences transversales permettant de visionner et comprendre l'ensemble avec la bonne perspective et connaissance. Des éditeurs travaillent pour qu'une grande partie de ces enjeux soient absorbés par de nouvelles solutions technologiques (ex : la famille de services Azure IoT). Certains de ces services seront mentionnés et utilisés plus loin dans l'article.
Pour les lecteurs qui connaissent déjà l'IoT dans son ensemble, cette première partie peut paraître réductrice. Vous avez raison. L'idée est de définir un point de départ compréhensible, sans trop de complexité.

Les objets connectés

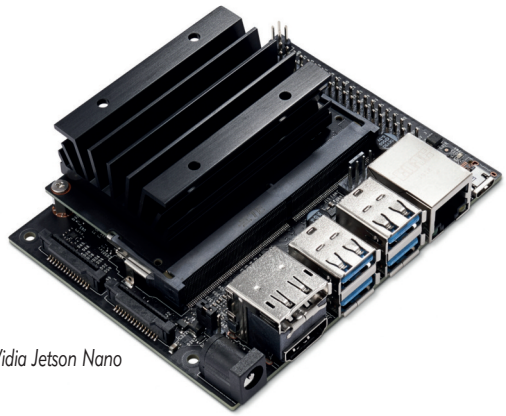
La verticalisation de chaque domaine donne lieu à des catalogues d'objets connectés très riches et dédiés à chaque domaine.
Il existe différentes familles d'un point de vue hardware.
Sans rentrer dans les détails du monde du hardware, ci-dessous les principales familles dans l'IoT :

- MCU (Microcontroller Unit) :
  - Mémoire et capacité de traitement réduite.
  - Très peu consommateur (donc potentiellement utilisable avec des batteries, piles), économe, robuste, capacité de traitement limitée (pas forcément un problème, tout dépend du projet).
  - Idéal pour des tâches simples.
  - S'adapte très bien à l'IoT pour les capteurs simples avec des besoins de processing limités.
  - Ex : capteur de température/humidité/pollution, machine à café, bouton IoT.
- SoC (System On a Chip) :
  - Plus puissant qu'un MCU, moins puissant qu'un MPU.
  - Architecture compacte par rapport à un MPU.
  - Plus consommateur qu'un MCU, moins consommateur qu'un MPU.

Les MPU étant souvent trop puissants et

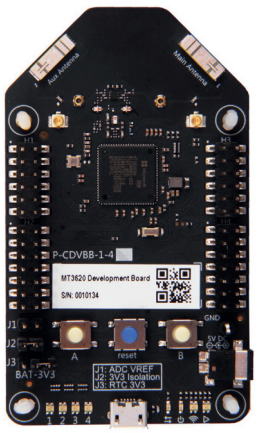


2 - *MXCHIP IoT Device Kit*

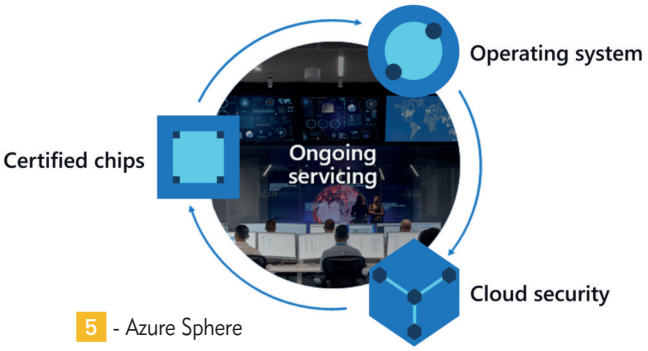


3 - *NVidia Jetson Nano*

4 - Arrow Vision AI Dev kit



6 - *Azure Sphere MT3620 Development Kit, de Seeed*



5 - Azure Sphere

consommateurs pour certains types d'IoT, les SoC sont une alternative intermédiaire. Ex : télévision, caméras IoT intégrant de l'intelligence, gateways, devices Edge, passerelles.

- MPU (Microprocessor Unit) :
  - Moins fréquent dans l'IoT.
  - Mémoire et capacité de traitement importante.
  - Consommateur et solution la plus onéreuse.
  - Capacité de traitement très importante.

Si vous voulez commencer à expérimenter avec la partie device, ci-dessous quelques références abordables :

- Solutions à base d'ESP32 (MCU) :
  - Connectique Wifi et Bluetooth,
  - Très faible consommation,
  - Robuste,
  - Utilisé dans de nombreux devices « light ».
- MXChip IOT Device kit (MCU): 2
  - Kit avec de nombreux capteurs,
  - Wifi,
  - Écran,
  - Boutons,
  - Complet pour du développement, mais peut-être moins adapté pour des projets réels.
- Solutions à base d'Arduino (MCU)
- Raspberry PI (SoC) :
  - Plus puissant que les MCU précédents,
  - Plus consommateur,
  - Cas d'utilisation différents des MCU (device Edge, passerelle).
- NVidia Jetson Nano Developer kit (SoC): 3
  - Même esprit que le Raspberry PI,
  - Capacité GPU importante pour un device de ce type (128 cœurs Maxwell) ce qui rend le device intéressant pour des cas d'usage Edge et/ou IA,
  - Encodage/décodage vidéo hardware.

Note : *le choix des solutions hardware pour une production est vital. Les critères à prendre en compte sont nombreux (besoins, fonctionnalités, capacités, consommation électrique, coûts, chauffe, stockage, sécurité, etc.).*
La liste précédente est un point de départ, souvent utilisée pour du POC (Proof Of Concept) ou POV (Proof Of Value).

Devices packagés :

- Arrow Vision AI Dev Kit : 4
  - Device complet (inputs, outputs, hardware capable d'intégrer des scénarios IA),
  - Compatible IoT Edge,
  - Bien packagé.

Devices sécurisés :

- Tout device intégrant Azure Sphere 5
  - MCU Sécurisé (architecture hardware conçue pour maximiser la sécurité),
  - OS (basé sur Linux),
  - Service Cloud.

6

Si vous n'avez pas besoin de prendre en charge la partie device et que vous vous concentrez dans la création de solutions applicatives IoT, il existe des simulateurs de devices:

- IoT Central avec des devices simulés ;
- Simulateurs dans Visual Studio ou Visual Studio Code (outils de développeur et pour du debug essentiellement) ;
- Simulateur standalone ;
- Ex : https://github.com/jonmikeli/azureiotdevicesimulator3 Simulateur pouvant être utilisé en tant qu'outil de développement ou plateforme de simulation (containerisable) ;
- Votre propre simulateur avec l'Azure IoT SDK (C, Python, Node.js, Java, .NET).

Normes, standards

Dans toute cette richesse et variété de contextes, on sent la nécessité d'aller vers des « standard » sur certains aspects. MQTT et AMQP sont les principaux protocoles de transmission de messages IoT. HTTP est également présent pour différentes raisons (moins orienté file de messages asynchrone).
Chaque domaine intègre ensuite ses propres spécificités pour des raisons diverses, par son historique ou les limitations au niveau des objets connectés.
Quelques exemples :

- IIoT (Industrial IoT) : ModBus, OPC UA ;
- CIoT (Consumer IoT) : IFTTT, Zigbee.

Réseau

Les types de réseaux utilisés sont également très variés et dépendent vraiment du contexte. Les plus répandus restent le réseau câblé, Sigfox, LoRa, protocoles radio, Wifi et 3G/4G.
Le bon choix dépend de nombreux paramètres (bande passante, environnement, portée des signaux, consommation, coûts d'utilisation, coûts du matériel, contraintes flux de communication, etc.).
NB-IoT et 5G sont très attendus, mais beaucoup d'interrogations gravitent autour de ces nouvelles technologies.

Architectures

Les architectures des solutions peuvent être très variées. Une solution domotique n'a clairement pas les mêmes besoins qu'un parc industriel de panneaux solaires. Dans l'IoT, il est plus important que jamais d'analyser correctement les problématiques à résoudre pour concevoir des solutions cohérentes (coûts, échelle, ROI – Return On Investment – TTM – Time To Market —, etc). Même s'il s'agit de projets techniques, la dimension business doit être très bien prise en compte dans les projets d'entreprise.

La complexité dans la maîtrise de cet univers pouvait parfois être perçue comme un frein. Les alternatives technologiques ont fait d'énormes progrès :

- Réduction des niveaux de complexité à maîtriser,
- Réduction très considérable des TTM.

Tout ne se fait pas en un seul click de bouton, mais les risques ont clairement été réduits et la proposition de valeur des solutions technologiques d'aujourd'hui est à prendre en considération.

Lambda architecture

La lambda architecture, dans des versions plus ou moins riches, est un type d'architecture très commun dans l'IoT.

Le principe de base repose sur les flux suivants : 7

Chaque parcours peut être plus ou moins riche, en fonction du projet. Il ne faut pas croire qu'il s'agisse uniquement de récupérer l'information et de les persister. 8

Data et IA

L'IoT, data et IA forment aujourd'hui un trinôme indissociable. La symbiose entre les trois est forte et sa richesse représente un potentiel unique.

La transformation est en train d'être radicale, mais fait « peu de bruit ». On parle même de « révolution silencieuse » dans des domaines comme l'industrie, l'énergie, la gestion environnementale, la santé, l'agriculture et l'automobile.

Intelligent Cloud et Intelligent Edge

L'Intelligent Cloud et l'Intelligent Edge découlent de cette symbiose.

L'Intelligent Cloud intègre des systèmes centralisés qui génèrent de l'intelligence au niveau du cloud alors que l'Intelligent Edge en fait de même, mais à proximité de la prise de mesure. Cette combinaison est extrêmement puissante et permet de couvrir des scénarios sans précédents.

Quelques exemples :

- Capteurs intelligents qui détectent des anomalies dans les données reçues. Ils sont capables d'agir sur la base d'Intelligences Artificielles embarquées dans des modules IoT Edge (Microsoft), sans attendre un retour de l'Intelligent Cloud.
- Orientation de panneaux solaires parce que le panneau lui-même « sait » comment s'orienter pour maximiser la production énergétique en fonction de la météo, heure et exposition.

Notre projet

Vu qu'il existe déjà de nombreux exemples autour d'objets qui transmettent des températures, taux d'humidité ou similaires, nous pensions qu'il serait utile de proposer un scénario différent implémentant des cas d'usages moins habituels. Nous avons donc imaginé un cas d'usage un peu plus interactif qui ne repose pas uniquement sur la récolte de data. Mêlons donc DevOps et IoT !

Cible :

Solution IoT permettant de :

- Déclencher des « builds surprise » depuis un objet connecté de type bouton.
- Suivre les déclenchements des builds.
- Gérer les boutons (activer, désactiver, statut, etc).

Un CI/CD reste un CI/CD (Continuous Integration/Continuous Delivery), mais pouvoir déclencher des pipelines de build à la demande, depuis un device totalement portable peut donner lieu à des cas d'usage intéressants.

Quelques usages pourraient être :

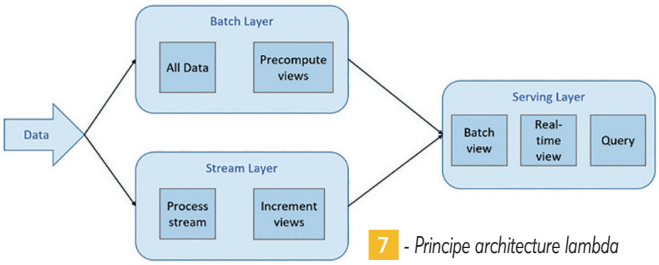
- Déclencher un build (avec les tests qui vont avec) chaque fois qu'un membre de l'équipe mentionne un « normalement, ça devrait fonctionner ».
- Un stakeholder du projet qui souhaiterait lancer des « builds surprise » pour s'assurer que la branche 'main' du projet est robuste.

On pourrait imaginer de faire un déploiement plutôt qu'un Build. Nous avons fini par retenir le build.

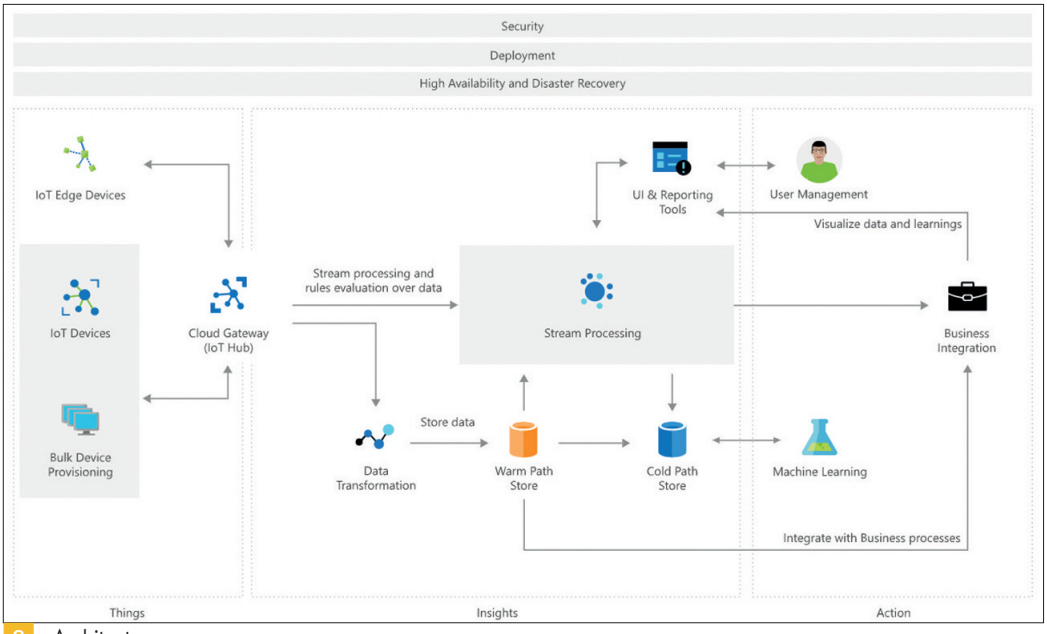
De manière générale, le projet peut être adapté pour lancer tout type d'action pouvant être déclenché par une API REST. Sentez-vous donc libre d'adapter le projet à votre contexte, besoin ou envie.

Les contraintes que nous nous sommes imposées sur ce projet sont :

- Rapide à mettre en place ;
- Minimiser les coûts ;
- Minimiser les développements.



7 - *Principe architecture lambda*



8 - Architecture macro

Architectures et solutions possibles

Il est temps de choisir le type d'architecture pour la solution.

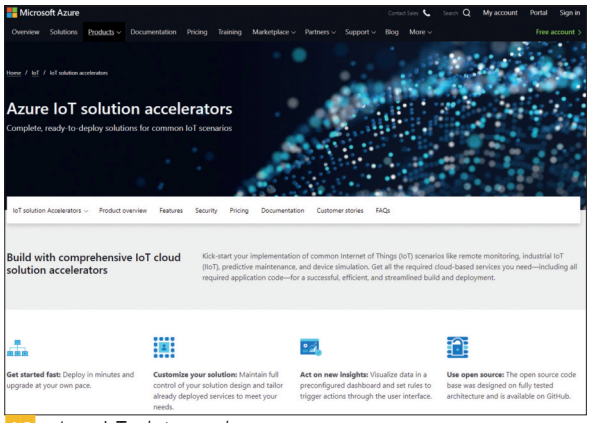
Deux grands axes sont possibles :

- PaaS, ensemble de services IoT qui, combinés, permettent de construire des solutions ultraperformantes à des coûts optimisés (parfois gratuits ou très réduits en phase de développement).
- SaaS, services totalement packagés, moins personnalisables et donc optimisables/adaptables, mais avec des temps de mise en place très courts (pouvant être gratuits en fonction de certaines conditions).

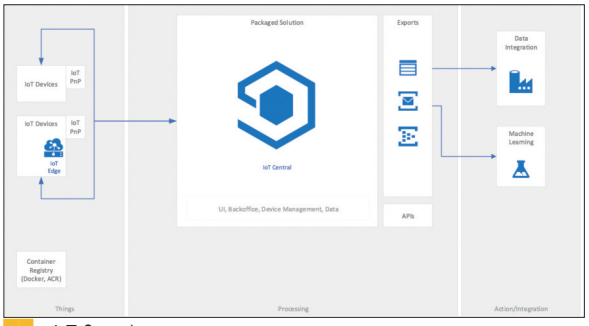
Si ceci ne vous parle pas, on pourrait le simplifier de la manière suivante :

- PaaS, version Légo de la solution, en combinant différents services.
- SaaS, version « presque clé en main » de la solution.

Ces métaphores sont très réductrices (en espérant ne pas vexer les connaisseurs) pour vous aider à comprendre la différence en termes d'approches.



10 - *Azure IoT solution accelerator*



11 - *IoT Central*

Plateforme technologique

Microsoft a subi une vraie transformation ces dernières années et les approches plus ouvertes, ciblant l'ouverture et l'intégration de leurs solutions ont donné lieu à la création de services à forte valeur ajoutée. De plus, Microsoft investit particulièrement dans le marché IoT et lui a permis d'occuper les places de tête dans ce secteur.

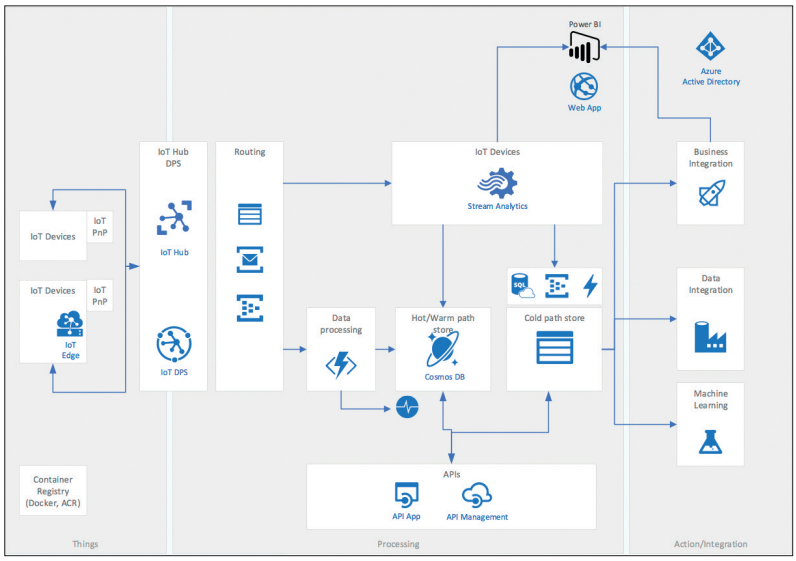
En effet, le catalogue de services, d'outils et de technologies IoT est très riche. Non seulement les différents services s'intègrent parfaitement entre eux, mais avec des solutions tierces ou Open Source.

Dans notre projet, nous partirons donc vers une solution Azure IoT.

PaaS, Platform as a Service

Il existe plusieurs plateformes de type PaaS IoT. Quelques solutions :

- Azure IoT Hub (avec Device Provisionning Service et le récent IoT Plug and Play, qui peut marquer un rôle important dans l'accélération de l'adoption de l'IoT).
- Azure Event Hub, qui permet de gérer la réception et le traitement de volumes très importants de messages.
- Azure Stream Analytics, qui permet de traiter des volumes importants de messages en streaming.
- Une bonne partie des services Azure applicatifs comme :
  - Azure Functions,
  - App Services,
  - Cosmos DB / Azure Database / Azure Storage / Time Series Insights (ou autres mécanismes de persistance),
  - API Management,
  - Logic Apps,
  - etc.



9 - *Exemple d'architecture globale IoT*

Vous trouverez ci-dessus le diagramme d'une architecture conventionnelle qui pourrait servir de point de démarrage. 9

Note : *Microsoft a publié quelques accélérateurs sur la base de ces technologies afin d'aider les architectes et développeurs dans le choix et l'intégration des différents composants. Ces accélérateurs absorbent une partie des questions qu'on peut se poser et implémentent des squelettes de solutions.*

URL : https://azure.microsoft.com/en-us/features/iot-accelerators/ 10

SaaS, Software as a Service

En ce qui concerne le SaaS, IoT Central est une solution « clé en main » qui couvre une partie intéressante des besoins des projets IoT « simples » :

- Gestion des devices (templating y compris) ;
- Création de visuels (dashboards, logs, etc.) ;
- Gestion des accès à la plateforme ;
- Exports de données ;
- Gestion d'événements/actions sur la base de messages reçus.

L'architecture précédente pourrait être remplacée par celle-ci contre : 11

Ce sera notre choix pour le projet de cet article.

Autres

En plus du PaaS et du SaaS, nous pouvons retrouver d'autres éléments comme :

- IoT Edge, solution technologique qui permet de déployer différents types d'intelligence au niveau des devices. Ceci donne lieu à des architectures hybrides et plus riches.
  La conception de la solution est particulièrement intéressante (repose sur de la conteneurisation).
- Azure Sphere, qui propose une solution à plusieurs axes (OS device, hardware, service) fortement sécurisée.

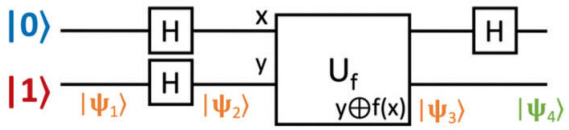
Partie 2 : à paraître dans Programmez! Hors-série n°1 le 3 juillet 2020

Jean-Michel Torres
IBM Quantum France
IBM Systems Center | Montpellier

Le détail du calcul de Deutsch

Cet article est un complément de celui paru dans le numéro 240 de Programmez!. Il détaille le calcul qui est derrière l'algorithme qui a été expliqué en pages 51 à 53. Il fait suite au dossier paru dans le numéro 239 de Programmez! qu'il est conseillé de lire au préalable.

Nous allons calculer les états de $|\psi_1\rangle$ à $|\psi_4\rangle$ pour le schéma suivant :



Notes préalables :

- ⊕ représente l'addition modulo deux, en clair : le « OU » exclusif de deux valeurs binaires (le résultat vaut 2 si l'une des deux valeurs et une seulement vaut 1).
- ⊗ représente le produit d'état pour deux qubits distincts. Cette opération (savamment désignée sous le nom de produit tensoriel) vérifie les propriétés habituelles d'associativité et de distributivité, mais n'est pas commutative : le membre de gauche concerne un qubit, le membre de droite concerne l'autre dans le calcul qui suit.

Commençons donc avec $|\Psi_1\rangle$:

$$|\psi_1\rangle = |0\rangle \otimes |1\rangle$$

Car le qubit de gauche est à l'état $|0\rangle$, et le qubit de droite est à l'état $|1\rangle$. Pour être un peu plus précis avec cette notation on devrait écrire :

$$|\psi_1\rangle = (1|0_{q0}\rangle + 0|1_{q0}\rangle) \otimes (0|0_{q1}\rangle + 1|1_{q1}\rangle)$$

Mais les notations étant déjà suffisamment lourdes, on allègera tout en veillant à bien garder à gauche ce qui concerne le qubit du haut sur le circuit, et à droite ce qui concerne le qubit du bas.

Pour aller à l'état chacun des deux qubits subit la transformation de Hadamard :
L'état $|0\rangle$ du premier qubit devient : $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
L'état $|1\rangle$ du second qubit devient : $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

Alors :

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

« Sortons les $\frac{1}{\sqrt{2}}$ et développons par rapport au terme de gauche :

$$|\psi_2\rangle = \frac{1}{2}(|0\rangle \otimes (|0\rangle - |1\rangle) + |1\rangle \otimes (|0\rangle - |1\rangle))$$

On peut alors réécrire cette addition en utilisant pour le qubit de gauche un état $|a\rangle$ qui vaut $|0\rangle$ puis $|1\rangle$ dans l'itération de 0 à 1 :

$$|\psi_2\rangle = \frac{1}{2}\sum_{a=0}^{a=1}(|a\rangle \otimes (|0\rangle - |1\rangle))$$

En développant par rapport au membre de droite à présent :

$$|\psi_2\rangle = \frac{1}{2}\sum_{a=0}^{a=1}(|a\rangle \otimes |0\rangle - |a\rangle \otimes |1\rangle)$$

Pour passer à l'état $|\psi_3\rangle$, l'état du second qubit (celui de droite) devient le « ou exclusif » de sa valeur avec la valeur de $f(a)$ par définition de notre circuit. Donc $|0\rangle$ devient $|0 \oplus f(a)\rangle$ et $|1\rangle$ devient $|1 \oplus f(a)\rangle$, soit :

$$|\psi_3\rangle = \frac{1}{2}\sum_{a=0}^{a=1}(|a\rangle \otimes |0 \oplus f(a)\rangle) - (|a\rangle \otimes |1 \oplus f(a)\rangle)$$

Avec, de manière évidente :
Si $f(a) = 0$ alors $0 + f(a) = 0$ et $1 + f(a) = 1$
Si $f(a) = 1$ alors $0 + f(a) = 1$ et $1 + f(a) = 0$

Donc :
Si $f(a) = 0$ alors $(|a\rangle \otimes |0 \oplus f(a)\rangle) - (|a\rangle \otimes |1 \oplus f(a)\rangle) = |a\rangle \otimes |0\rangle - |a\rangle \otimes |1\rangle$
Si $f(a) = 1$ alors $(|a\rangle \otimes |0 \oplus f(a)\rangle) - (|a\rangle \otimes |1 \oplus f(a)\rangle) = |a\rangle \otimes |1\rangle - |a\rangle \otimes |0\rangle$

C'est-à-dire :
Si $f(a) = 0$ alors $(|a\rangle \otimes |0 \oplus f(a)\rangle) - (|a\rangle \otimes |1 \oplus f(a)\rangle) = |a\rangle \otimes (|0\rangle - |1\rangle)$
Si $f(a) = 1$ alors $(|a\rangle \otimes |0 \oplus f(a)\rangle) - (|a\rangle \otimes |1 \oplus f(a)\rangle) = -|a\rangle \otimes (|0\rangle - |1\rangle)$

En utilisant le fait que : $(-1)^0 = 1$ et $(-1)^1 = -1$ on peut résumer les deux lignes ci-dessus :

$$|\psi_3\rangle = \frac{1}{2}\sum_{a=0}^{a=1}(-1)^{f(x)}(|a\rangle \otimes (|0\rangle - |1\rangle))$$

Soit encore :

$$|\psi_3\rangle = \frac{1}{2}(\sum_{a=0}^{a=1}(-1)^{f(x)}|a\rangle) \otimes (|0\rangle - |1\rangle)$$

$$|\psi_3\rangle = \frac{1}{2}((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle) \otimes (|0\rangle - |1\rangle)$$

Il reste à passer le quibt 0 (celui de gauche dans l'expression, celui du haut dans le schéma) par la porte H :

$$|\psi_4\rangle = \frac{1}{2}((-1)^{f(0)}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + (-1)^{f(1)}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) \otimes (|0\rangle - |1\rangle)$$

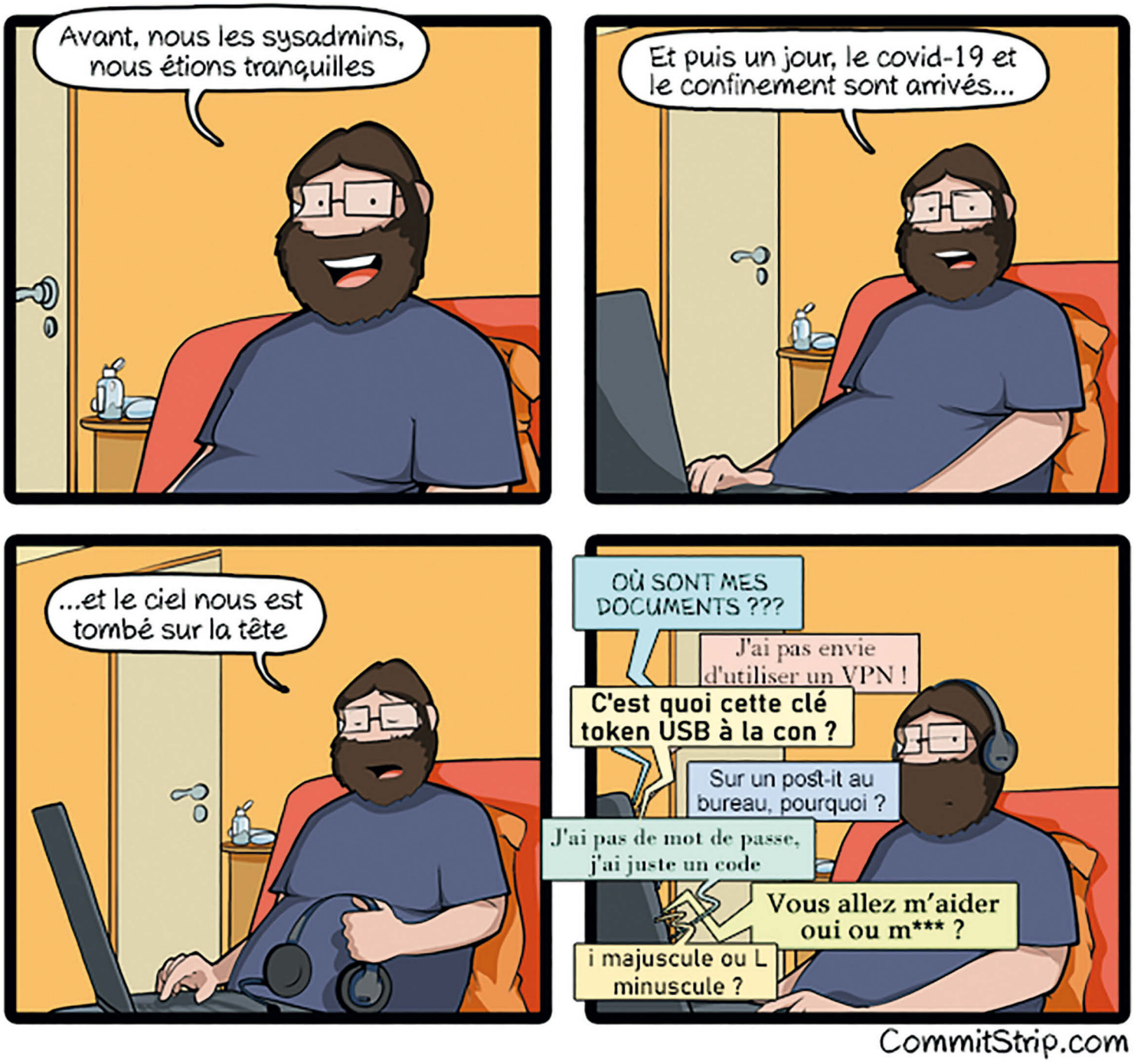
En regroupant les valeurs des états pour le quibt 0 :

$$|\psi_4\rangle = \frac{1}{2\sqrt{2}}(((-1)^{f(0)} + (-1)^{f(1)})|0\rangle + ((-1)^{f(0)} - (-1)^{f(1)})|1\rangle) \otimes (|0\rangle - |1\rangle)$$

Alors, on voit que :
Si $f(0) = f(1)$ (constante) alors une mesure sur le premier qubit donne $|0\rangle$,
Si $f(0) \neq f(1)$ (équilibrée) alors une mesure sur le premier qubit donne $|1\rangle$,

Si vous avez suivi jusque-là, vous méritez ceci : ⟨vo| ! (qui se lit bravo).

Sysadmin vs Covid-19



Une publication Nefer-IT, 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08 - Directeur de la publication : François Tonic
Rédacteurs en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Ont collaboré à ce numéro : ZDNet
Nos experts techniques : S. Almosni-Chiche, O. Roudier, R. Mnasri, T. Leriche, E. Pecoul, A. Payet, S. Blanc, J. Chraibi, L. Broudoux, J. Pastouret, T. Moyse, J. De Oliveira, C. Nasarre, D. Dhouid, C. Pichaud, J. Lacognata, K. Ansard, J. Mikel Inza, J-M Torres
Couverture : © Red Hat - **Maquette** : Pierre Sandré.
Publicité : François Tonic / Nefer-IT - Tél. : 09 86 73 61 08 - ftonic@programmez.com.
Imprimeur : SIB Imprimerie
Marketing et promotion des ventes : Agence BOCONSEIL - Analyse Media Etude - Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD Téléphone : 09 67 32 09 34

Contacts : Rédacteur en chef : ftonic@programmez.com - Rédaction : redaction@programmez.com - Webmaster : webmaster@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908 - © NEFER-IT / Programmez, juin 2020

Abonnement :
Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex. - Tél. : 01 55 56 70 55 - *abonnements.programmez @groupe-gli.com*
Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30.

Tarifs
Abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € - Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 € - Autres pays : nous consulter.

PDF
35 € (monde entier) souscription sur www.programmez.com

tangente
l'aventure mathématique

INTÉGRAL MATHS

LE NOUVEL abonnement numérique

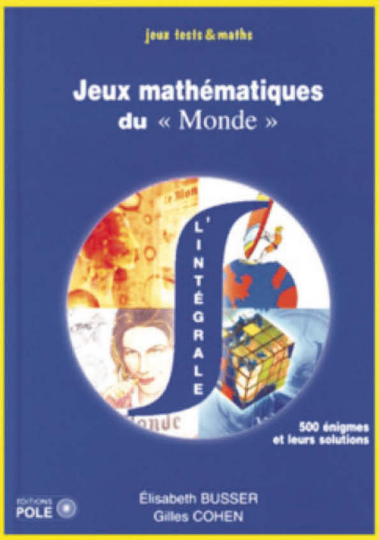
pour seulement **3,99 €** par mois

28 nos de Tangente

sur tangente-mag.com

17 hors séries

Et aussi... sur affairedelogique.com



400 problèmes du Monde et leurs solutions

www.infinimath.com/librairie

N°3
Disponible

Retour vers le passé :

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

Commandez directement sur **programmez.com**

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

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

NOUVEAU ! Abonnement 1 an : 30 €