Windows Azure

LE CLOUD ET LE DÉVELOPPEUR

COMMENT LE CLOUD RÉVOLUTIONNE LE DÉVELOPPEMENT ?

Introduction

Le Cloud Computing est sur toutes les lèvres. On en parle partout, tout le temps. Plus rarement, on évoque l'impact du Cloud sur le développeur, qui est loin d'être négligeable. En France, le marché du Cloud se rapproche des deux milliards d'euros.

Deux approches sont à considérer :
- Comment le Cloud modifie le modèle de développement des applications, l'architecture et la manière de penser l'application ;
- Comment le Cloud change les habitudes de travail du développeur.

Ces changements s'inscrivent dans un contexte particulier. Celui d'un marché dans lequel les entreprises ne sont plus en situation de pouvoir livrer des applications à un rythme ne prenant pas en compte la nécessité de s'adapter à des besoins et des conditions fluctuantes ; il leur est devenu primordial de déterminer si les investissements réalisés dans le développement d'une application porteront rapidement leurs fruits ou s'il faut les reconsidérer.

Pour mener à bien ces changements et réduire les délais de livraison, il faut donc envisager de nouvelles pratiques, axées sur la mise en place d'itérations rapides, en mode agile, et ciblant le « continuous delivery », c'est-à-dire l'intégration, voire le déploiement continu des applications à livrer. Le « continuous delivery » se fonde sur l'approche historique de l'ALM (Application Lifecycle Management) en rationalisant, grâce au Cloud et à la mise en place de scénarios « DevTests », le processus mais aussi l'infrastructure de développement. Tandis que la démarche « DevOps » apparaît de plus en plus incontournable en permettant une accélération des déploiements tout en réduisant les frictions opérationnelles entre les équipes de développement et d'infrastructure.

Notre propos dans ce livre blanc est d'illustrer comment le Cloud modifie le métier, le quotidien et les outils du développeur. Les changements sont profonds et reprennent la philosophie du Cloud : flexibilité et élasticité à la demande.

Comme vous le verrez tout au long de ce livre blanc, nous pouvons aller très loin dans ces deux domaines, mais c'est à vous que revient la responsabilité de trouver le juste équilibre. Comme avec une méthode agile, il ne faut pas appliquer strictement l'intégralité des pratiques associées aux approches « DevOps » et « DevTests », mais choisir celles dont vous avez réellement besoin. ■

Sommaire

Stéphane Goudeau
Cloud Architect – Microsoft France

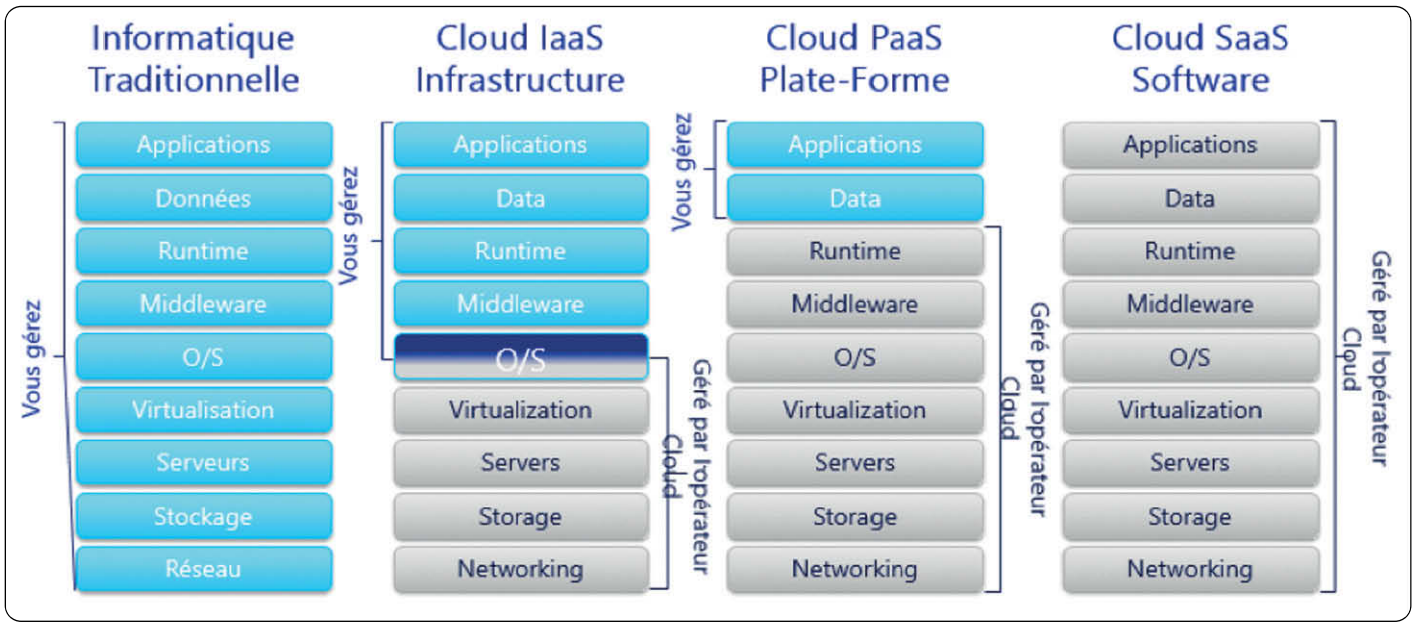
François Tonic
Rédacteur en chef de
cloudmagazine.fr / Programmez!

Imprimerie Jaures, 10, rue Vicq d'Azir, 75010 Paris
MICROSOFT France, RCS Nanterre 327 733 184, 39, quai du Président Roosevelt 92130 Issy Les Moulineaux

Coordination : Marwan Elfitesse (Microsoft France).

Cloud Computing : rappels

D'après la définition du NIST (National Institute of Standards and Technology), le Cloud Computing présente cinq caractéristiques essentielles, trois modèles de services et quatre modèles de déploiement (cf Definition of Cloud Computing v15) que sont la mutualisation des ressources, l'accès aux ressources en mode « libre-service », le réseau ubiquitaire, l'élasticité, la facturation à l'usage (« je paie ce que j'utilise »).



1. Modèles de services

Le Cloud Computing propose plusieurs modèles de services : l'infrastructure as a service (IaaS), la plateforme as a service (PaaS) et le software as a service (SaaS). Le schéma suivant montre les services automatisés (dits managés) en gris et les services gérés par l'entreprise et les administrateurs en bleu. Le PaaS permet aux développeurs de se concentrer sur les données et le code. Le fournisseur de Cloud va gérer toutes les couches « middleware » : moteurs de données, systèmes d'exploitations, runtimes, environnements de développeurs.

En IaaS, il est possible de personnaliser les environnements (machines virtuelles) et de créer des workloads dédiés.

En IaaS, l'utilisateur garde le contrôle et la gestion de la couche opérationnelle (système, logiciels, logiciels serveurs...).

2. Modèles de déploiement

Le Cloud Computing propose trois principaux modèles de déploiement : privé, public et hybride.

Chaque modèle peut avoir un impact sur le déploiement et l'utilisation de vos services.

Cloud privé

Le Cloud privé est souvent synonyme de IaaS. Il s'agit d'utiliser et de déployer des couches de virtualisation, de provisionnement, d'automatisation et de catalogues de services pour une utilisation interne à l'entreprise (sur un site ou sur l'ensemble des filiales). Le Cloud privé comprend des ressources à la demande, un catalogue de services. Il peut être déployé et exploité soit directement par l'entreprise, soit par un tiers. Il existe des clouds privés externalisés. Il est aussi possible de créer du PaaS privé (Private PaaS), par exemple les Web Sites du Windows Azure Pack.

Cloud communautaire

L'infrastructure Cloud est configurée pour une utilisation exclusive par une communauté d'utilisateurs, d'entreprises qui partagent les mêmes préoccupations (missions, exigences de sécurité, politiques et considérations de conformité). Le Cloud communautaire peut être, ou non la propriété de l'entreprise, être déployé ou non à demeure et être exploité soit directement par un ou plusieurs des organismes de la communauté, soit par un tiers (ou une combinaison des deux). Exemple : UnivCloud (Cloud communautaire pour les universités d'île de France).

Cloud public

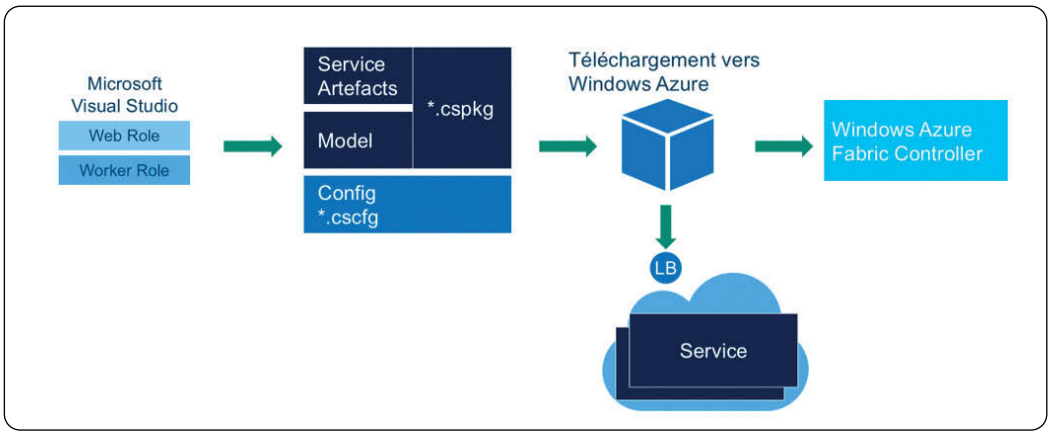
Un Cloud public est un service déployé et géré par le fournisseur (Microsoft déploie et administre Windows Azure) dans ses datacenters (ou des datacenters loués, ou via des partenaires). L'infrastructure Cloud y est mutualisée pour être utilisée par les clients. L'exploitation, dont le niveau varie selon le type de service (PaaS ou IaaS) est de la responsabilité de l'utilisateur. On paie à l'usage.

Cloud hybride

Le Cloud hybride se caractérise par la mise en œuvre d'infrastructures distinctes. Elles sont liées par des passerelles, des connexions pour faciliter le bon fonctionnement des applications et des données : typiquement utiliser un Cloud public et des ressources internes à l'entreprise. Un des scénarios les plus courants est le débordement de ressources sur un Cloud public (ou bursting) : une entreprise a besoin de ressources supplémentaires (serveurs, stockages, etc.) pour absorber un pic d'utilisation, elle utilise alors des ressources d'un Cloud public. Autre scénario : connecter des applications mobiles à des services de Cloud. Enfin, l'infrastructure d'un Cloud hybride peut résulter de la composition de multiples Clouds (cf http://nitaac.nih.gov/nitaac/cloud-computing).

Windows Azure : rappels

Windows Azure est l'offre de Cloud Computing de Microsoft pour héberger les services, tous types de données et les applications d'entreprise (quel que soit le langage). Windows Azure propose à la fois du PaaS (développement, déploiement d'applications, services managés) et du IaaS (pour exploiter des machines virtuelles Windows, Linux).



Déploiement d'une application sous Windows Azure

1. Les services PaaS

Le PaaS Windows Azure prend donc en charge un certain nombre de concepts de l'infrastructure :

- Modèles d'environnements prédéfinis ;
- Approvisionnement sur demande, en libre-service, et automatisé de ces modèles ;
- Surveillance des ressources utilisées ;
- Gestion élastique des ressources en fonction de leur utilisation ;
- Garantie de continuité de service pour les applicatifs hébergés, même en cas de panne matérielle, de mise à jour du système d'exploitation ou de mise à jour de l'applicatif lui-même.

Son objectif est donc de faciliter le développement, le déploiement et l'exécution d'applications et services dans un modèle de libre-service à la demande. Pour ce faire, Azure propose au développeur de bâtir son application en se fondant sur l'utilisation de modèles : les rôles.

Les différents rôles

Il existe deux types de rôles : le WebRole (modèle dédié à l'hébergement d'applicatif web) et le Worker Role (destiné à implémenter différents types de services). Les services et solutions sont construits avec une combinaison de Web Roles ou de Worker Roles, se répartissant les tâches à exécuter.

Les composants PaaS

La plateforme propose un ensemble de services destinés à être utilisés individuellement ou de manière combinée par les développeurs. Nous y trouvons : Service Bus, Media Services, un système de caching, Active Directory, SQL Database... et la liste s'allonge presque chaque mois.

Le packaging et déploiement des services PaaS

Pour implémenter une application Cloud, le développeur dispose d'un type de projet Cloud. Il va pouvoir développer avec les langages .net, mais aussi PHP, Java, Ruby, Python, etc.
Visual Studio permettra d'éditer les deux fichiers XML de définition : .csdef (la liste des rôles, la taille des machines virtuelles, les ports d'écoute...) et de configuration .cscfg (le nombre d'instances de chaque rôle, la configuration des certificats...) devant être utilisés pour la topologie du Cloud Service. Ces fichiers sont intégrés au package .cspkg (contenant les binaires) de déploiement du service hébergé.
Le déploiement du package peut être réalisé directement depuis Visual Studio.

2. Les services IaaS

Les services d'infrastructure Windows Azure permettent de créer, déployer, démarrer et gérer des machines virtuelles Windows ou Linux (format VHD). La virtualisation de l'infrastructure physique, le stockage et le réseau sont directement gérés par Windows Azure.
Différents modèles de création et de déploiement sont proposés :

- Création à partir d'une image proposée nativement dans la galerie Azure ;
- Création d'une machine virtuelle à partir d'un disque VHD persisté dans le stockage Azure. On récupère alors le dernier état de la VM ayant utilisé ce disque ;
- Création d'une image par capture et préparation d'une machine virtuelle existante en vue de la rendre générique ;
- Téléchargement de machines virtuelles configurées à demeure sur un serveur VHD.

Différentes tailles d'instances sont disponibles. Chaque instance correspond à une taille de ressources : processeur, mémoire vive, stockages, entrées/sorties, bande passante. Les machines virtuelles sont persistantes, grâce à l'utilisation d'objets blobs, hébergés dans les services de stockage Windows Azure. La fonction auto-scaling (montée en charge « automatique ») est disponible sur le IaaS et le PaaS.

3. Cloud Services et scénarios mixtes PaaS-IaaS

Sous Windows Azure, PaaS et IaaS ne sont pas isolés. Les machines virtuelles sont regroupées au sein de Cloud Services. Un Cloud Service est un conteneur logique permettant de gérer la supervision, la configuration, la sécurité, le réseau, voire la topologie d'une collection de machines virtuelles. Un Cloud Service peut contenir des instances de rôles PaaS ou des machines virtuelles IaaS, mais il ne peut pas héberger les deux types de machines virtuelles.

4. Sites Web : déployer directement votre site sur le Cloud

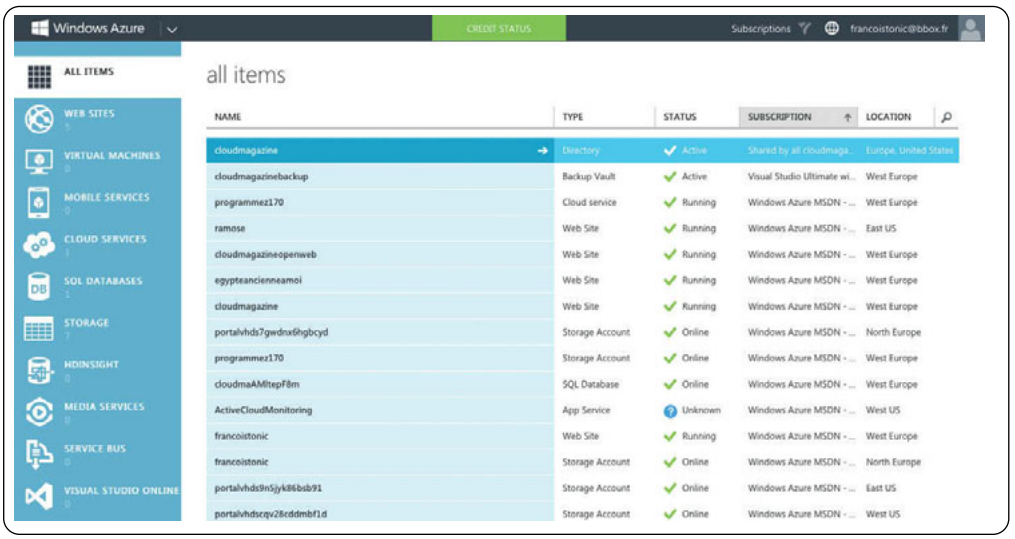
Les Azure Web Sites offrent un service d'hébergement permettant la construction rapide de serveurs Web avec la possibilité d'utiliser des gestionnaires de contenu populaires tels que Drupal, Umbraco, .NetNuke, Joomla, Mojo, phpBB, WordPress... Ce service est fondé sur l'utilisation du serveur internet IIS (Internet Information Server) et de son extension ARR (Application Request Routing). Ce service supporte ASP.Net, Node.JS, PHP, Git, WebDeploy, TFS, FTP...

5. Windows Azure et les terminaux mobiles

L'objectif est d'utiliser un ou plusieurs services Cloud en back-office couplés avec une application mobile. Ce service facilite l'écriture d'applications mobiles pour Windows 8, Windows Phone, HTML5 Client, iPhone/iPad, Android ou Xamarin en accélérant la mise en place des services de notification, de stockage de données et d'authentification des utilisateurs entre l'application et Windows Azure.

Impact du Cloud pour le développeur

Le PaaS et le IaaS impactent la manière de développer et de déployer aussi bien les applications que les sites web. Comment et pourquoi le Cloud révolutionne-t-il le développement ?



Le Cloud est un accélérateur de développement :

- Rapidité de mise à disposition d'environnement d'exécution (mutualisation et disponibilité des ressources) ;
- Facilité de l'évaluation des technologies via prototypage (Proof of Concept) sur des environnements disponibles à la demande ;
- Pré-installation d'environnements serveurs proposés par Microsoft (SQL Server, SharePoint Server, Biztalk Server, les solutions Oracle, Linux...) ;
- Partage et automatisation du déploiement de plateformes pour les tests, les développements ou l'intégration ;
- Activation de services applicatifs liés à la solution cible (Azure SQL Database, Azure Active Directory, Azure Service Bus, Azure Biztalk Services,...) ;
- Outillage Cloud avec Visual Studio Online ciblant la construction, l'observation du comportement et l'évolution des applications dans le Cloud alignée sur les bonnes pratiques du « continuous delivery » : agilité et industrialisation des développements, intégration continue, démarche devOps.

1. Evolution des applications vers le PaaS

Avec le PaaS, le développeur se concentre sur l'application et sur son code. Le PaaS expose des API, des services, des SDK. Il inclut des runtimes d'exécution. Windows Azure est ouvert à de multiples environnements de développement : .Net, Java, PHP, Ruby, Node.js, Python... L'ouverture est une de ses forces.

Adapter et migrer une application pour le Paas n'est pas toujours une tâche très aisée. Il faut souvent modifier l'architecture logicielle, changer les couches d'accès aux données, évaluer la sécurité.

Techniquement, cette migration n'est pas insurmontable, mais le coût et la durée du redéveloppement sont à considérer.

Pour le développeur, le Cloud impacte le développement et sa façon de travailler.

Pour bien utiliser Windows Azure, il doit connaître et comprendre les différents services, les SDK et API, choisir ceux qui lui seront utiles, et intégrer les bonnes pratiques.

Mais fondamentalement, développer une application en mode Cloud (PaaS ou IaaS), par exemple, pour faire un service SaaS ou utiliser des services Cloud, n'est pas plus compliqué qu'un développement « classique ».

Le développeur devra néanmoins être vigilant sur les points suivants :

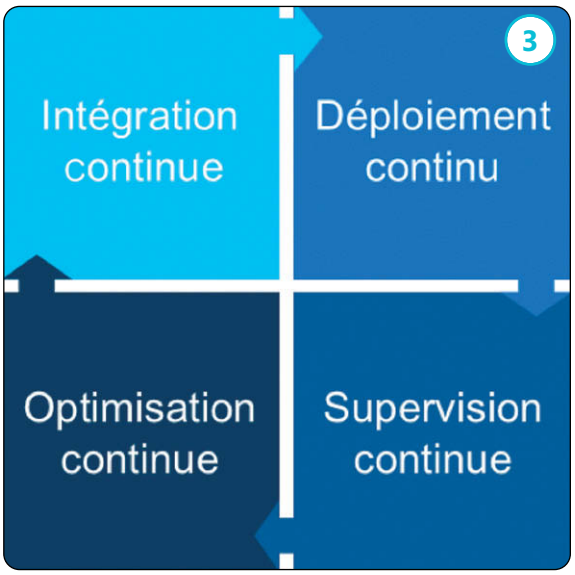
- Architecture logicielle (single-tenant ou multi-tenant, architecture FailSafe) ;
- Choix des API, frameworks et librairies du ou des services : chaque fournisseur propose ses propres librairies et bonnes pratiques ;
- Optimisation du code, des accès, des données : toute ressource consommée est facturée ;
- Prise en compte des critères de disponibilité, de performance, de sécurité et d'autres aspects critiques.

2. Gestion des montées de version du mode PaaS

Les services Cloud sont déployés et exécutés sur des machines virtuelles. Il s'agit d'un système d'exploitation Windows Server optimisé pour fonctionner dans cet environnement. Ce système d'exploitation évolue régulièrement et fait donc l'objet de mises à jour. Celles-ci sont nécessaires pour s'assurer que les solutions développées sur le PaaS Azure sont adossées à des systèmes fiables et sécurisés, intégrant les dernières innovations. Il est donc impératif pour le développeur comme pour le responsable système d'avoir une bonne compréhension de la façon dont les mises à jour sont appliquées, et comment elles peuvent impacter la disponibilité et la facilité de maintenance des applications dont ils sont responsables.

Ce modèle délègue à Microsoft la responsabilité de gérer l'infrastructure sous-jacente et notamment les mises à jour. Il serait inapproprié de supposer que cette délégation dispense les clients de se préoccuper des mises à niveau des systèmes d'exploitation hébergés. L'utilisateur d'Azure peut décider de contrôler la version de Windows Server déployée dans ses services Cloud, en utilisant l'attribut « osFamily » dans le fichier de Configuration du Service (.cscfg). A cet attribut « osFamily » s'ajoute l'attribut « osVersion ».

La configuration du service Cloud permet de choisir la mise à jour automatique des versions du système d'exploitation hébergé en spécifiant l'attribut « osVersion » avec un astérisque « * » (mais pas la mise à jour automatique de la « famille » du système d'exploitation, donc pas de mise à jour automatique de Windows Server 2012 à Windows Server 201X). Toutefois, sur le plan théorique, suivant la nature du service



Cloud, il faut envisager la possibilité qu'un simple changement de version (par exemple un correctif de sécurité) puisse être incompatible avec le code qui s'exécute dans l'application déployée sur la machine virtuelle.
Si l'utilisateur de ces services est habitué à déployer les patches de sécurité sur des systèmes de test ou de pré-production avant de les appliquer sur sa production, il préfèrera très certainement définir manuellement l'attribut « osVersion » en correspondance avec ses processus internes de validation de version.
Le développeur est également directement concerné par cette gestion des versions du système d'exploitation des machines virtuelles hébergées. En effet, une correspondance est établie entre le Windows Azure SDK et certaines des versions du système d'exploitation invité. C'est donc à lui qu'incombe la responsabilité de choisir un SDK compatible avec la cible de déploiement. Fort heureusement, la probabilité qu'une prochaine version de système d'exploitation hébergé (et potentiellement automatiquement mis à jour) ne supporte pas le SDK avec lequel le code a été compilé reste extrêmement faible.
Chaque nouveau système d'exploitation est testé, lors de sa publication, avec les deux dernières versions du SDK. Les compatibilités des familles de système d'exploitation et de SDK sont référencées dans la documentation en ligne : http://msdn.microsoft.com/en-us/library/windowsazure/ee924680.aspx. Lorsqu'une nouvelle version du SDK est publiée, les clients ont jusqu'à un an pour moderniser leurs solutions en la recompilant avec un SDK supporté.
Les montées de versions manuelles imposent le suivi et le test des versions de système d'exploitation hébergées qui doivent être « ciblées » avant d'être appliquées.
Les montées de versions automatiques ne permettent pas le contrôle avant publication, mais doivent être accompagnées de tests systématiques pour mieux les anticiper.
Dans les deux cas, cette démarche sera facilitée par la mise en place d'un environnement permettant de faire évoluer l'application à chaque nouvelle version de SDK, et de la tester pour chaque mise à niveau de système d'exploitation hébergé. D'où l'intérêt de l'automatisation des tests de non régression sur un environnement dont on maîtrise la configuration.
Le processus qui permet d'effectuer cette validation comprend les étapes suivantes :

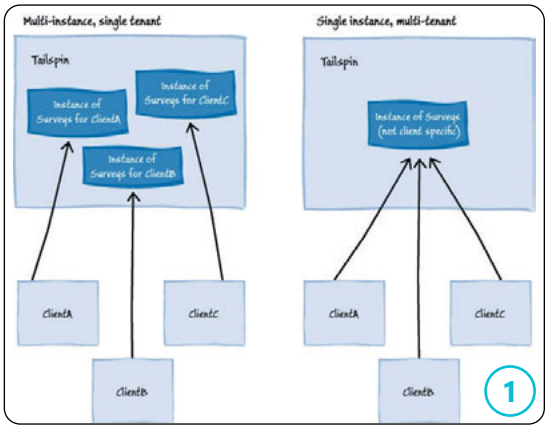
- Compilation de l'application avec la dernière version du SDK ;
- Exécution des tests unitaires (s'il y en a) ;
- Déploiement de l'application sur une machine virtuelle de test déployée dans Azure ;
- Exécution de tests supplémentaires (d'interfaces Web automatisées, par exemple).

Ce processus devra être déclenché à chaque nouvelle publication de version.
Une façon de se tenir à jour consiste à s'abonner au flux RSS d'information sur les versions de système d'exploitation : http://sxp.microsoft.com/feeds/3.0/ msdntn/WindowsAzureOSUpdates.
Pour la validation du fonctionnement du Cloud Service, il est préférable d'aller au-delà du simple démarrage de l'instance de chaque rôle. Par exemple, dans le cas d'un Cloud Service hébergeant un Web Role, il sera utile de mettre en place un test permettant de reproduire le scénario de navigation sur l'application ou le service Web.

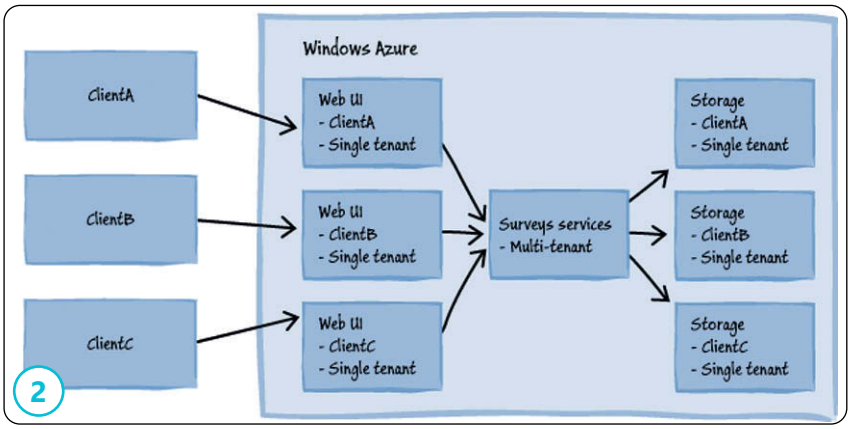
3. Architecture logicielle

Le multi-tenant, clé de voûte du Cloud Computing

Pour bénéficier de toute la puissance du Cloud, une application SaaS doit adopter une architecture spécifique : le multi-tenant (ou architecture multi-tenancy) **Fig.1 et 2**. Sans cette spécificité, l'application consommera inutilement des ressources et ne profitera donc pas des économies d'échelle associées au Cloud.
Le multi-tenant a pour objectif de mutualiser une même application (un même code) entre différents clients. Le multi-tenant s'applique aussi bien à l'application qu'aux données. Concrètement cela signifie pouvoir mutualiser 1 application / 1 base de données pour N clients. Il existe des modes de multi-tenancy intermédiaires (par exemple 1 application / N bases de données pour N clients : ce mode est fréquemment utilisé par les éditeurs logiciels qui utilisent Azure SQL Database et paient donc le service en fonction de la taille de la base).
Par opposition le single-tenant signifie : 1 application = 1 client.
En multi-tenant, l'isolation des instances et des données de chaque client est primordiale. Aucune fuite ne doit exister.
Le multi-tenant peut concerner les applications comme les services Cloud. Il permet d'utiliser la flexibilité des capacités du Cloud, d'assurer une meilleure montée en charge, tout en optimisant les ressources utilisées.
Il suppose la mise en place de mécanismes de gestion d'identité (contrôle d'accès, annuaire, service de jeton, éventuellement fédération des identités).



Vue logique d'une architecture single-tenant et d'une architecture multi-tenant



Exemple d'une architecture pour Windows Azure. Ce n'est pas l'unique design possible. Dans la pratique, une application Windows Azure se compose de nombreux services. Ces éléments peuvent être utilisés par des architectures single ou multi-tenant

Architecture FailSafe

Si l'on patiente suffisamment longtemps ou si l'on lui applique une pression suffisamment forte toute application ou service finit par rencontrer un dysfonctionnement. Il s'agit donc d'anticiper cette situation en définissant la façon dont l'application doit se comporter, et notamment en proposant une gestion souple des modes de défaillance afin qu'elle puisse continuer à offrir une valeur ajoutée.

Définir une architecture « FailSafe » requiert la catégorisation des types d'échecs :

- Transitoire - interruption de service temporaire, d'auto-guérison
- Enduring – intervention nécessaire
- Périmètre de la défaillance – Région, Service, Machine.

Cela suppose également la prise en compte des SLAs liés aux divers composants de la solution, la mise en place de mécanisme de surveillance et de télémétrie, la gestion des erreurs transitoires, l'utilisation d'une solution de mise en cache distribuée, et enfin un couplage faible entre les principaux composants de la solution.

Ce mode de conception d'une architecture dite « FailSafe » est décrit dans le whitepaper « Failsafe: Guidance for Resilient Cloud Architectures » de Marc Mercuri, Ulrich Homann, et Andrew Townhill http://msdn.microsoft.com/en-us/library/windowsazure/jj853352.aspx)

4. « Continous Delivery » : Intégration, déploiement, et suivi en continu des applications

Processus de « Continuous Delivery »

Le « continuous delivery » est un processus regroupant de multiples pratiques destinées à apporter de la valeur en continu, en déployant le plus fréquemment possible de nouvelles versions d'une application ou d'un service. L'un des ouvrages de référence est le livre de Jez Humble et David Farley : « Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation ».

Le « continuous delivery » englobe :

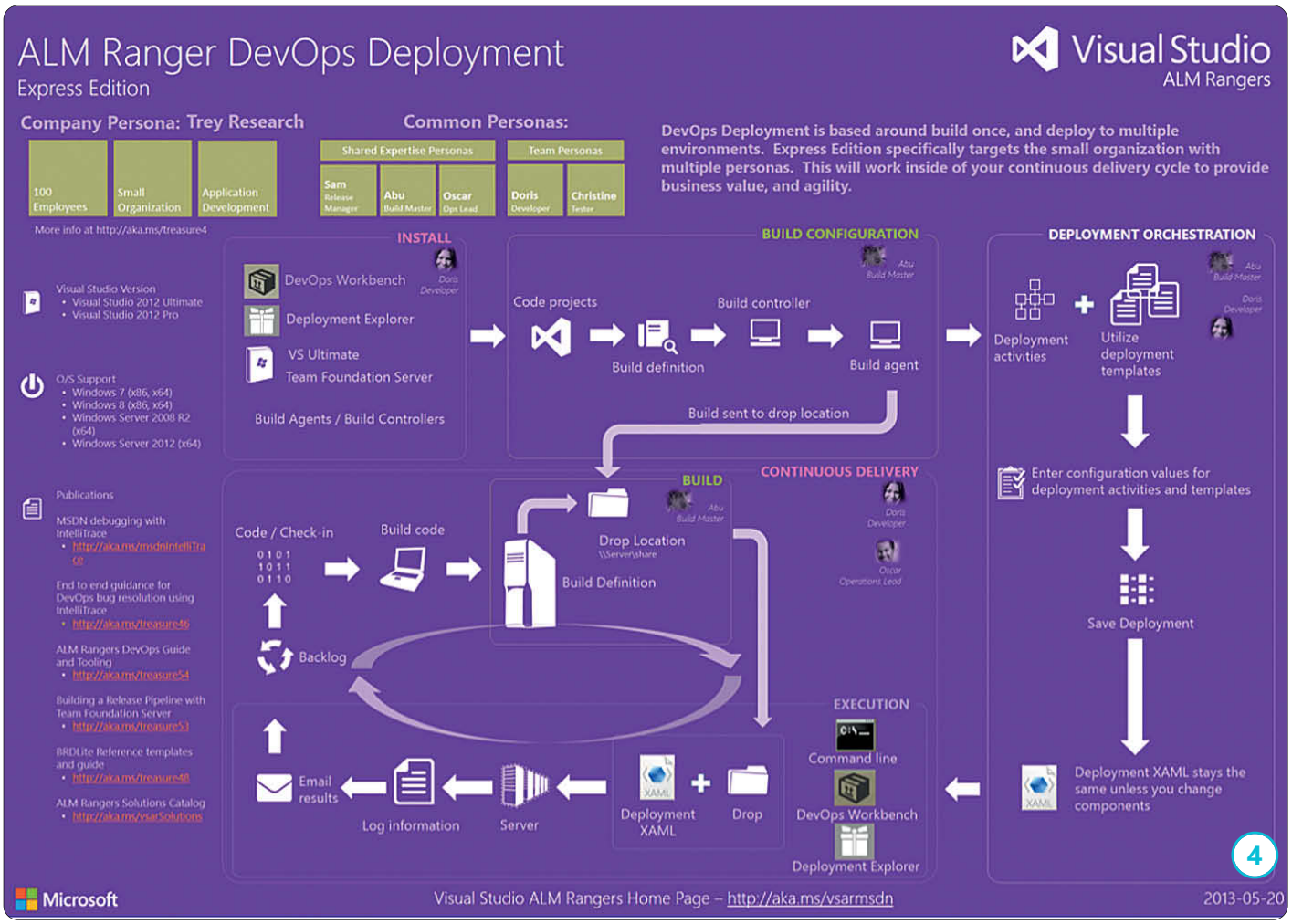
- L'intégration continue qui vise à réduire les efforts d'intégration en assurant automatiquement la génération, l'assemblage et les tests des composants de l'application ;
- Le déploiement continu qui consiste à automatiser le déploiement du logiciel construit sur la plateforme cible (test, intégration, pré-production voire production) à chaque nouvelle génération de livrable ;
- Le suivi en continu du comportement de l'application sur le plan technique mais également du point de vue de son usage afin d'améliorer l'efficacité de la solution **Fig.3**.

Intégration continue

Le processus de développement démarre avec un outil de contrôle de version du code source de l'application, et s'achève par le déploiement des binaires dans l'environnement de production. Dans l'intervalle, le code doit être compilé, les environnements doivent être configurés, de nombreux types de tests doivent s'exécuter...

En effet, dès qu'il y a plus d'un développeur qui sera appelé à modifier le code source de l'application, il est crucial de valider fréquemment l'intégration du travail de chacun avec celui des autres. Pour cela, une validation manuelle locale par le développeur lui-même ne suffit pas.

L'idéal est d'effectuer un ensemble de vérifications sur un environnement dont on maîtrise la configuration et indépendant des postes de l'équipe de développement. Chaque intégration



est vérifiée par une génération automatisée (y compris le test) pour détecter les erreurs d'intégration dès que possible. Le processus qui permet d'effectuer cette validation est couramment appelé « Chaîne d'intégration continue ».
Ce processus comprend en général les étapes suivantes :

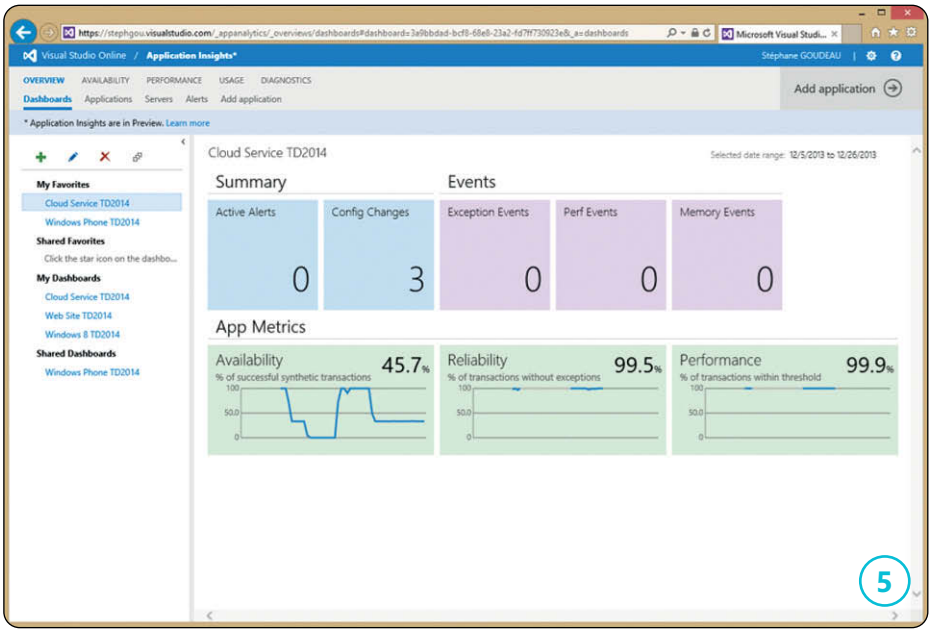
- Le développeur fait évoluer le code et procède à des tests localement sur son poste de travail. Cela inclut l'écriture de tests unitaires automatiques. Le code est archivé dans le contrôle de code source ;
- Un serveur de builds extrait la dernière version disponible du code depuis le contrôleur de code source. Il compile, exécute les tests unitaires pour s'assurer de la qualité du code produit et crée des packages de déploiement selon l'environnement de déploiement cible.
- Les packages sont déployés sur la plateforme cible installée sur Azure ;
- Des tests supplémentaires sont automatiquement lancés (tests d'interfaces graphiques automatisés, tests de vérification du bon déploiement,...).

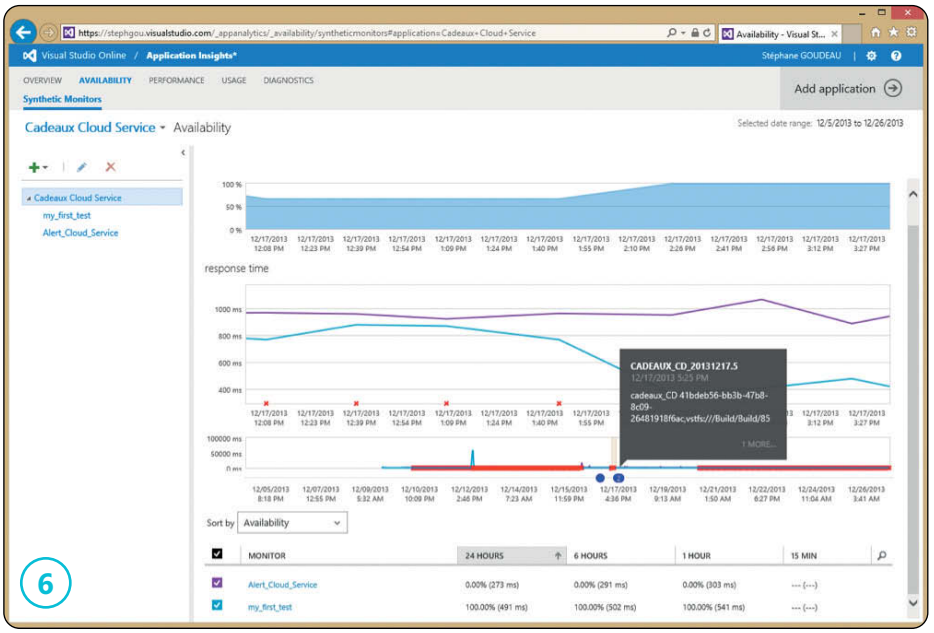
Généralement, ce processus, est déclenché le plus fréquemment possible : à chaque mise à jour du référentiel de code source. Il peut cependant être effectué régulièrement sans pour autant le faire à chaque archivage.... Idéalement, les phases d'intégration explicite deviennent optionnelles puisque le code est toujours intégré. L'automatisation du processus de génération des livrables est un facteur déterminant de la mise en place de ce workflow. Elle garantit le principe d'une opération reproductible, fiable et prévisible.

Déploiement continu

Il ne suffit pas d'automatiser le workflow lui-même, mais aussi la création des environnements, leur mise en service, et la maintenance de l'infrastructure. Cette automatisation permet de ne pas gaspiller du temps en tâches pouvant être directement réalisables par le système. Un des axes d'optimisation du développement sur le Cloud va donc consister à personnaliser le workflow de gestion des versions (« release pipeline ») en utilisant les modèles de génération par défaut fournis par Team Foundation Server (TFS) ou Visual Studio Online **Fig.4**. Windows Azure offre de nouveaux scénarios de déploiement par rapport à une infrastructure à demeure, avec la possibilité de tests dans un environnement de production, de déploiement continu des applications, et d'agilité accrue avec la facilité de déploiement. Dans le cas du PaaS, le scénario de déploiement dans Azure s'articule autour d'un certain nombre d'étapes qui peuvent s'orchestrer dans le processus de génération et de déploiement :

- Création du package pour le déploiement ;
- Téléchargement du package dans le stockage Azure ;
- Création des instances de l'application dans le nuage selon les directives de configuration ;





- Démarrage des instances déployées ;
- La solution peut être déployée directement dans une zone baptisée « Staging » dans le portail Azure avant d'être manuellement « basculée » en production. Il ne s'agit pas d'un environnement de pré-production, mais plutôt d'une zone de « smoke tests ».

Suivi en continu du comportement de l'application

L'application doit pouvoir s'adapter en permanence aux besoins de ses utilisateurs et il est nécessaire de s'assurer que les services proposés fonctionnent comme prévu. La mise en place de mécanismes de supervision technique et de télémétrie, permet de savoir comment les solutions sont utilisées et la façon dont elles fonctionnent. Application Insights est une composante de Visual Studio Online qui offre les fonctions suivantes :

- Définition de tableaux de bord pour avoir une vue d'ensemble de l'application **Fig.5** ;
- Affichage des informations de performance et de disponibilité des applications en considérant la connectivité à ces applications depuis des points d'observation localisés dans le monde entier et définition de seuils d'alerte **Fig.6**.

La collecte d'informations au moyen de mesures externes peut également être réalisée à l'aide de produits tels que System Center Global Service Monitor. Ces mécanismes sont cruciaux pour améliorer en permanence l'application. Ils permettent de garantir la qualité du service proposé tout en permettant de s'adapter à l'évolution du marché.

DevTests et DevOps : profitez des avantages du Cloud

Nous avons vu dans le chapitre précédent comment le Cloud, et particulièrement Windows Azure, impacte le développement et pas seulement l'architecture logicielle. Dans ce contexte, il est très important de comprendre les motivations et les perspectives qu'offre le scénario « DevTests » ainsi que les principes de la philosophie « DevOps ». Comment et pourquoi utiliser ces deux approches ?

Le scénario « DevTests »

En parallèle du « Continuous delivery », les clients ont aujourd'hui la possibilité de tirer parti des ressources dont ils disposent en interne ou dans le Cloud pour déployer leurs environnements de développement et de test. Windows Azure permet d'automatiser la mise à disposition de ces environnements sur des services Cloud (typiquement du IaaS et/ou sur des services à la demande) d'une manière fiable, sécurisée et rapide. Ce type de scénario est baptisé « DevTests ». Pourquoi avoir recours à Azure pour les scénarios « Développement et Tests » ?

Motivations

La première raison est de faire bénéficier les développeurs d'une infrastructure à la demande capable de rapidement monter en charge. En effet, avec ses services d'infrastructure , Windows Azure fournit une excellente plateforme pour vite déployer des infrastructures de développement et de test. Celle-ci intègre :

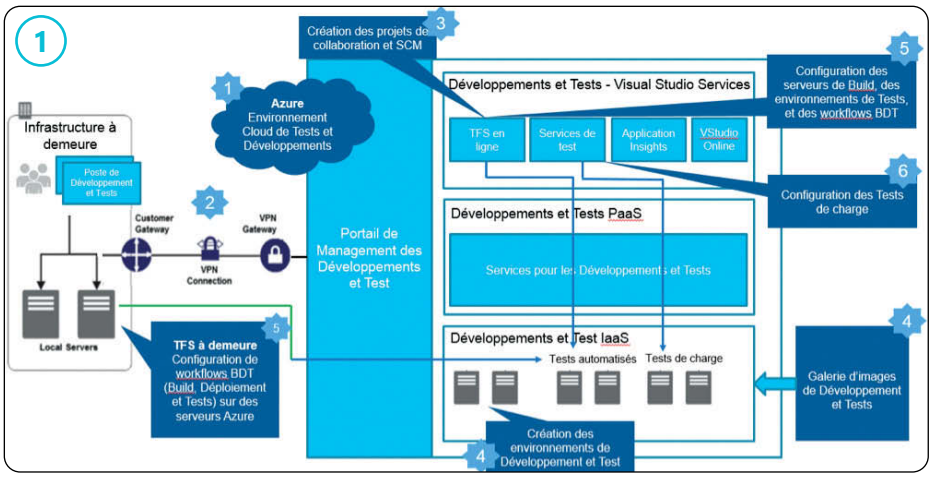
- Des services de stockage ;
- Des services de réseaux virtuels avec en option la possibilité d'établir des liens VPN pour offrir une connexion réseau sécurisée depuis l'infrastructure à demeure ;
- La possibilité de déployer des serveurs d'infrastructures dans le IaaS (gestion d'identité avec Active Directory, Hyper-V Recovery Manager, Windows Azure Backup,...) ;
- Des machines virtuelles persistantes dans le Cloud offrant un large choix de dimensionnement : jusqu'à 8 cœurs, 56GB de RAM et 16 disques (max 1TB, soit 16 TB max) ;
- Des images fournies avec la plateforme Azure, proposant une liste croissante de produits Microsoft et non Microsoft, nativement supportés dans Azure ;
- La possibilité de construire des images personnalisées ;
- La capacité à automatiser la création de VM via des commandes PowerShell ;
- L'intégration avec les services de la plateforme Azure.

Le scénario « DevTests » offre une solution adaptée aux équipes géographiquement dispersées et aux entreprises ne souhaitant pas déployer des infrastructures dédiées, parfois lourdes à mettre en oeuvre et à maintenir.

Perspectives

Le « DevTests » peut être envisagé selon plusieurs scenarii :

- Déploiement des environnements serveurs de test d'intégration et de pré-production ;
- Evaluation de logiciels ;
- Installation de l'infrastructure ALM, Application Lifecycle Management (contrôle de version, gestion des builds, release management, etc.) qui peut être déployée dans un modèle de Cloud, éventuellement hybride ;
- Fourniture de postes virtualisés pour le développeur ou le testeur.



Déploiement des environnements serveurs de test

Le « DevTests » permet de pré-packager des images virtuelles contenant les environnements cibles. Ceci évite d'immobiliser des ressources dédiées à demeure. Un autre avantage est la capacité de déployer directement dans l'environnement Windows Azure au cours du processus de génération, ce qui permet de tester les applications dans le même environnement que celui sur lequel tournera l'application une fois en production; cela garantissant ainsi la compatibilité entre les plateformes de test et de production. La notion de prêt à l'emploi est importante. Pour un développeur, un testeur, il s'agit de disposer des bons outils et environnements le plus rapidement possible, sans avoir de contrainte de ressources. Les environnements de tests et de Lab Management sont parfois très lourds à mettre en œuvre. La plateforme Windows Azure permet de mettre à disposition autant d'environnements cibles que nécessaire. Vous les montez et démontez.. Et surtout vous payez à la demande.

Evaluation de logiciels

Un service, ou une application, est fréquemment amené à évoluer, soit parce qu'il faut procéder à une montée de version du système d'exploitation cible (de Windows Server 2008 à Windows Server 2012 R2), soit parce qu'il intègre des dépendances par rapport à un composant système que l'on souhaite remplacer, soit encore parce que l'on souhaite associer un logiciel tiers à la solution. Pour bien gérer ces évolutions, il convient de mesurer leur impact. La plateforme Azure permet alors de réaliser les tests correspondants dans les plus brefs délais et à moindre coût.

Mise en production d'une infrastructure d'ALM

Le DevTests concerne aussi les outils. L'infrastructure requise pour le développement peut

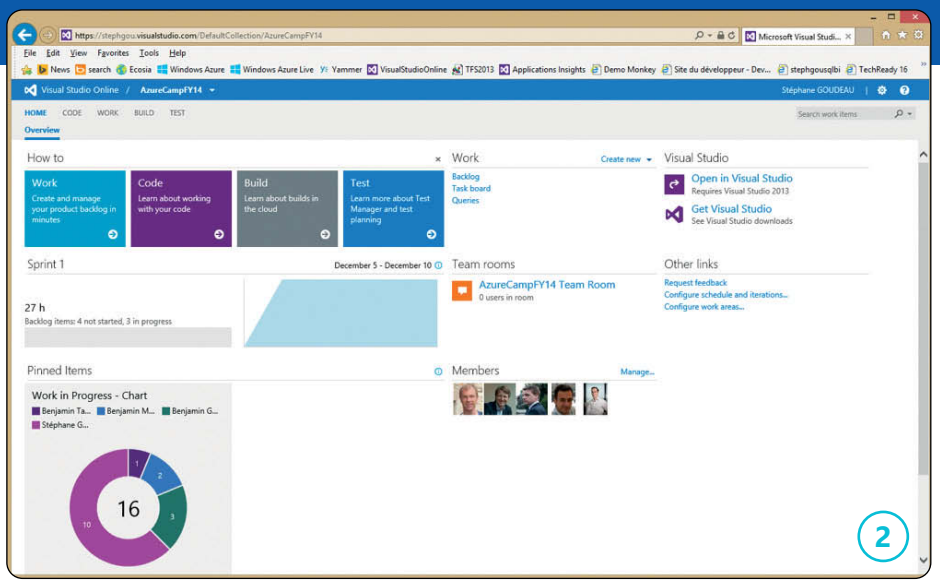
elle-même être déployée sur la plateforme Windows Azure, et proposer des services de gestion d'identité commune avec celle de l'infrastructure à demeure, de gestion des tests, de gestion du cycle de vie de l'application. Ceux qui ont déjà déployé une infrastructure ALM savent que ce n'est pas trivial, et que la maintenir ne l'est pas moins. Team Foundation Server (TFS) est serveur de l'offre ALM Visual Studio (démarche agile, backlog, tâches, automatisation des builds, tests, gestion de configuration, portail d'équipe, reporting, etc.). Il est également possible de déployer TFS sur des VM sur Windows Azure Machines Virtuelles ou sur ses serveurs dans une démarche hybride. Le 1er schéma illustre cette approche **Fig.1**. TFS est aussi disponible sous forme de services : Visual Studio Online. Vous pouvez aujourd'hui disposer rapidement d'un environnement ALM très complet, connecté à vos environnements de développement. Vous n'avez plus à déployer sur votre infrastructure : aucune maintenance, aucune mise à jour. Tout est géré par le service en ligne ! Ce service Windows Azure propose un portail d'accès aux différents projets de développement avec une page d'accueil personnalisable permettant d'avoir une vue complète sur les différents composants de son projet (sprint et backlog, contrôle de code source TFS ou GIT, tests, builds, membres de l'équipe, team rooms, tableaux de bord,...) **Fig.2**.

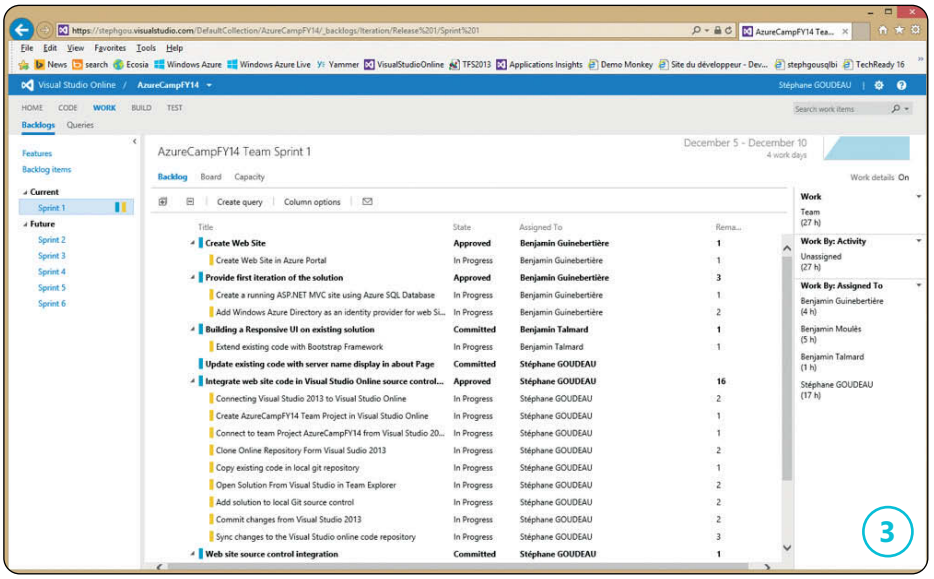
Visual Studio Online permet de choisir différents modèles associés à la méthodologie de développement (CMMI, MSF Agile ou Scrum). Les méthodes agiles décomposent les actions en petites étapes de planification (itération de 1 à 4 semaines). Chaque itération cible un travail d'équipe, à travers un cycle de développement logiciel complet. Cela permet de minimiser le risque global et de s'adapter rapidement aux changements. L'objectif est d'avoir une version disponible à la fin de chaque itération. Lorsque l'on développe une application, il est toujours nécessaire d'avoir une liste de fonctionnalités à développer. L'organisation de cette liste sera fonction de la méthode adoptée par l'équipe.

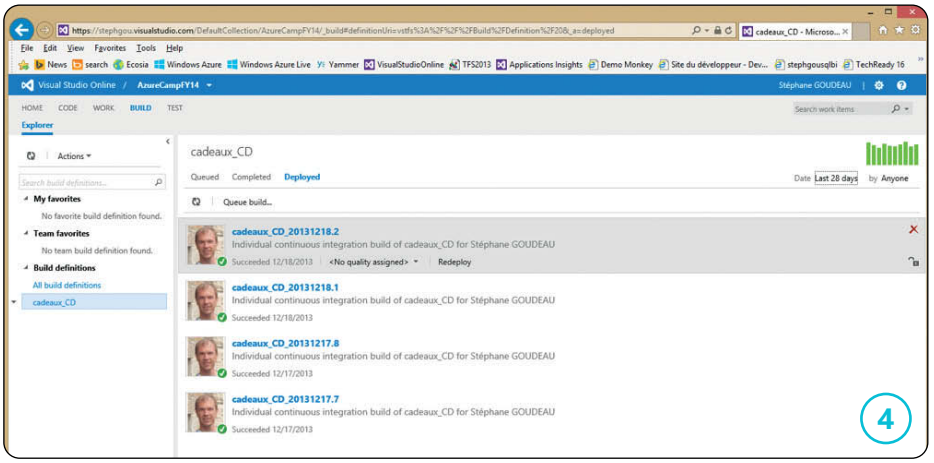
- Dans une méthode « classique », la liste des fonctionnalités est découpée selon une arborescence d'exigences ;
- Dans une méthode agile, on utilise une liste que l'on nomme « backlog ».

Le backlog est une pile des différentes fonctions proposées par l'application. Cette liste est susceptible d'être alimentée en continu. Elle évolue donc sans cesse et permet de définir les délais de mise à disposition d'une nouvelle fonctionnalité. Le Portail Visual Studio Online permet de définir et faire évoluer cette liste **Fig.3**.

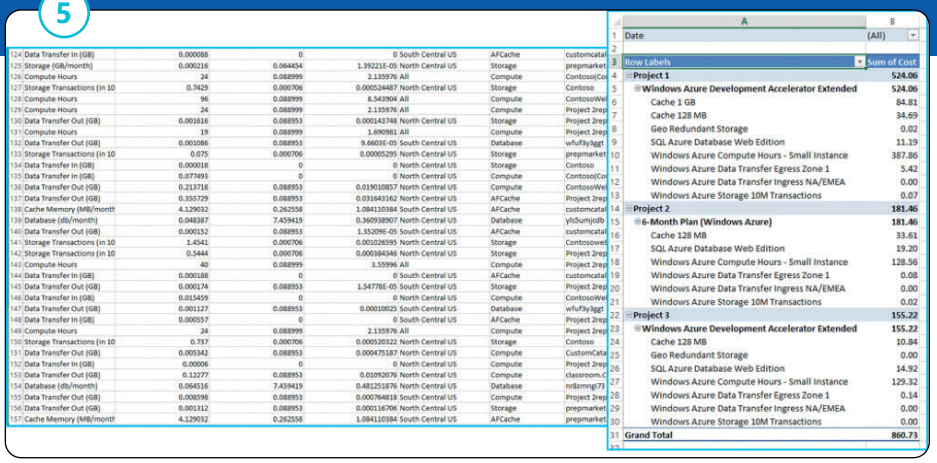
Visual Studio Online offre des services de Build, qui, couplés avec un mécanisme de déclenchement automatique de déploiement, assurent la mise en place du processus d'intégration continue **Fig.4**. Visual Studio Ultimate permet de générer des tests en simulant la charge utilisateur sur les applications Internet et les serveurs pour donner des informations précises sur le comportement de l'application ou du service Cloud dans un mode de fonctionnement proche de la réalité. Les résultats permettent de connaître les performances de l'application, de dimensionner les machines pour des performances optimales, en un mot d'anticiper. Visual Studio fonctionne avec toutes les technologies d'applications Web. Pour ce faire, il est nécessaire de mettre en place un « rig de test » constitué d'au moins un contrôleur de test et d'un agent. Un contrôleur de test est un service Windows qui a pour rôle de connaître un ensemble d'agents de tests et de les piloter. Les agents de tests sont également des services Windows qui, quant à eux, ont pour rôle d'héberger et de simuler les utilisateurs virtuels. Dans une telle configuration, Visual Studio n'est plus là que pour piloter le tout. Le test de charge réside sur le client Visual Stu-







5



Date	(All)
Row Labels	Sum of Cost
Project 1	524.06
Windows Azure Development Accelerator Extended	524.06
Cache 1 GB	84.81
Cache 128 MB	34.69
Geo Redundant Storage	0.02
SQL Azure Database Web Edition	11.19
Windows Azure Compute Hours - Small Instance	387.86
Windows Azure Data Transfer Egress Zone 1	5.42
Windows Azure Data Transfer Ingress NA/EMEA	0.00
Windows Azure Storage 10M Transactions	0.07
Project 2	181.46
6-Month Plan (Windows Azure)	181.46
Cache 128 MB	33.61
SQL Azure Database Web Edition	19.20
Windows Azure Compute Hours - Small Instance	128.56
Windows Azure Data Transfer Egress Zone 1	0.08
Windows Azure Data Transfer Ingress NA/EMEA	0.00
Windows Azure Storage 10M Transactions	0.02
Project 3	155.22
Windows Azure Development Accelerator Extended	155.22
Cache 128 MB	10.84
Geo Redundant Storage	0.00
SQL Azure Database Web Edition	14.92
Windows Azure Compute Hours - Small Instance	129.32
Windows Azure Data Transfer Egress Zone 1	0.14
Windows Azure Data Transfer Ingress NA/EMEA	0.00
Windows Azure Storage 10M Transactions	0.00
Grand Total	860.73

dio depuis lequel le testeur va configurer et lancer le test de telle sorte que les injecteurs (les agents de charge) génèrent la charge depuis différents emplacements sur l'application web. Avec Visual Studio Online, plus besoin de configurer et de déployer ce « rig de test ». Il suffit d'indiquer au niveau de Visual Studio que les tests de charge seront réalisés depuis des injecteurs s'exécutant sur la plateforme Windows Azure. C'est donc beaucoup plus simplement que peuvent s'exécuter les tests de charge aujourd'hui, et sans impact sur l'infrastructure interne : le Cloud offre une montée en charge quasi-illimitée…

Capacité à délivrer des environnements de travail pour les développeurs et testeurs

L'utilisation de système d'exploitation client n'est pas supportée aujourd'hui sur la plateforme Azure. Cette limitation contractuelle permet néanmoins d'envisager des scénarios VDI (Virtual Desktop Infrastructure) fondés sur la mise en oeuvre du protocole RDP sur une machine Windows Server 2012 R2.

En effet, l'accès au protocole RDP est activé par défaut sur toutes les images de Galerie de Windows et les paramètres RDP sont configurés automatiquement pendant la configuration de la VM afin que le port local 3389 soit ouvert sur la machine virtuelle ainsi qu'un port externe configuré de façon aléatoire dans la plage IANA 49152 à 65535. Le fichier RDP peut ensuite être téléchargé à partir du portail de gestion ou via les cmdlets PowerShell Azure ou par API REST. La connexion RDP doit être réalisée sur le FQDN du service Cloud associé à la VM de test/développement, ou sur son adresse IP publique (adresse VIP), en complétant l'un ou l'autre par le port externe configuré dans le portail. Pour les scénarios d'usage impliquant de multiples connexions sur la même machine, il est également possible de configurer RDS.

L'utilisation d'une souscription Azure associée à un compte MSDN permet ainsi de disposer d'une image pré-installée avec VisualStudio 2013 à destination des développeurs.

Implémentation d'une infrastructure de Développement et Test

L'implémentation d'une infrastructure de développement et tests dans un Cloud public comme celui proposé par la plateforme Windows Azure suppose le respect, la planification et la mise en œuvre de bonnes pratiques de gouvernance qui recouvrent différents sujets.

- Gestion des souscriptions ;
- Facturation des ressources ;
- Choix du type d'hébergement des services ;
- Sécurité et gestion d'identité ;
- Configuration des référentiels d'images.

Gestion des souscriptions

Lorsque l'on décide de mettre en place une infrastructure de développement et test à l'échelle d'une entreprise, il est nécessaire de définir les périmètres de responsabilité et l'organisation des environnements projet par souscription Azure. Il est donc préférable :

- De disposer de multiples souscriptions ;
- De définir leur niveau de délégation et de l'automatiser ;
- De leur appliquer des règles de nommages précises et systématiques ;
- Lorsque cela est possible, d'utiliser des comptes dédiés pour y accéder et d'établir des règles de création pour ces Microsoft Accounts (précédemment appelés Live IDs) ;
- De définir des groupes d'affinités pour optimiser les temps d'accès entre les ressources liées aux mêmes plateformes ;
- D'administrer les certificats liés au contrôle de la publication et à l'utilisation de nouvelles ressources.

Définition des unités de facturation (ChargeBack)

Avec l'avènement de la virtualisation et de la mutualisation des ressources (machines virtuelles, réseau, stockage,…) la Direction des Systèmes d'Information a tout intérêt à mettre en place les outils lui permettant de savoir qui a consommé quoi et le cas échéant, de lui refacturer. C'est également un moyen permettant de s'assurer que les ressources ne sont pas gaspillées. Il est fréquent de recevoir une demande de création de machines virtuelles; il l'est beaucoup moins de recevoir une demande de libération des ressources correspondantes. Avec le Cloud, et le portail en mode libre-service, cette bonne pratique devient une absolue nécessité.

La plateforme Azure permet de définir des quotas de ressources au niveau de la souscription, et d'obtenir une facturation détaillée à posteriori (certains outils complémentaires – comme Foglight - permettent de faire des projections sur la consommation à venir). Les données de consommation peuvent être téléchargées au format Excel, et manipulées avec des tables pivot **Fig.5**. Des règles pourront également être édictées et automatisées pour limiter la consommation (arrêt nocturnes des plateformes,…).

Choix du type d'hébergement des services

Le choix du type d'hébergement des services partagés (infrastructure ALM, annuaire, base de données…) est un pré-requis à la mise en place d'une infrastructure « DevTests ».

Différentes options sont à considérer : Services SaaS, Services déployés en IaaS ou Services à demeure ? Les Solutions SaaS offrent souvent moins de possibilités de personnalisation mais proposent un SLA, des plans de support intégrés, et sont supervisées (la gestion opérationnelle du service est incluse).

Sécurité et gestion d'identité

Une approche IaaS ou OnPremise pour les services partagés n'est pas neutre d'un point de vue sécurité et gestion d'identité. Les développeurs doivent être identifiés. En fonction du type de solution ALM utilisée (Team Foundation Server à demeure ou en IaaS, Visual Studio OnLine), ils peuvent l'être grâce à un compte Active Directory, ou avec un Microsoft Account.

Active Directory, ou une autre solution de gestion d'identité, peut être déployé en IaaS. S'il s'agit de s'appuyer sur une solution de gestion d'identité à demeure, il sera alors nécessaire de monter un VPN en mode site à site. Si les services correspondants sont déployés en IaaS dans Windows Azure, il faudra exposer des endpoints publics et les sécuriser par ACL.

Configuration des référentiels d'images

Comme nous l'avons déjà vu précédemment, Windows Azure propose une offre IaaS. Elle repose sur des machines virtuelles (VM). On dispose de trois types de VM : Windows Server, Linux, Workload pré-packagé. Par workload, il faut comprendre un environnement complet virtualisé avec le système serveur et l'ensemble des piles serveurs et applicatives nécessaires pour déployer et mettre en production le workload.

Pour un développeur attaché à la production, le workload est une notion importante. Il peut ainsi rapidement déployer les environnements cibles de l'application pour tester et débugger. Il peut aussi juger de la pertinence du système cible. C'est un élément clé pour la mise en œuvre de prototype comme pour la mise en place de l'application proprement dite.

La configuration du référentiel d'images suppose la mise en place d'un processus de versioning des images, de définition de la stratégie de

diffusion des images, et de déploiement des VM, de gestion du licensing des produits, de validation des environnements (manuelle ou automatisée), de choix des technologies et outils d'automatisation (PowerShell, System Center, Puppet, Chef) Fig.6.

La démarche DevOps

« DevOps » est un terme inventé par Patrick Desbois en 2009. John Allspaw et Jesse Robbins sont également à l'origine de ce mouvement qui est né d'un constat : la faible synergie des équipes de développement d'application et d'administration des systèmes. Dans une récente analyse du mouvement « Devops » Andrew Clay Shafer va même jusqu'à parler de sources de tension qu'il associe à un « Wall of Confusion ».

Le « Wall of Confusion »

Ce « mur » résulte des divergences d'objectifs entres développeurs et responsables opérationnels au sein d'une organisation. Souvent, les développeurs se concentrent sur la réponse aux exigences fonctionnelles par la production d'un livrable, mais pas du tout sur la maintenance de la solution en fonctionnement opérationnel. De leur côté, pour limiter les risques de problèmes inattendus, les administrateurs système tentent d'édicter des règles pour les déploiements qui s'avèrent être des contraintes architecturales pour le développement d'applications. Chaque groupe optimise ce qu'il considère être son périmètre, ce qui crée des conflits. Le développeur est à l'origine de changements, que le responsable des opérations souhaite minimiser pour réduire le risque de dysfonctionnement Fig.7.

DevOps : Une philosophie

DevOps est une philosophie. C'est une façon de penser. Dans ce modèle, développeurs et opérateurs doivent travailler ensemble pour s'assurer que tout nouveau déploiement d'application respecte les besoins opérationnels. Selon Gene Kim, CTO cofondateur de Tripwire, et auteur de l'ouvrage « The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win », la démarche « DevOps » repose sur trois principes fondamentaux :

- Acquérir une compréhension globale du système afin d'optimiser les performances de l'ensemble de ses processus, en se focalisant sur toutes les chaînes de valeur métier reposant sur des services IT ;
- Mettre en place des systèmes de mesure et des processus de remontée d'information systématique ;
- Favoriser le développement d'une culture fondée sur l'expérimentation et l'apprentissage en continu Fig.8.

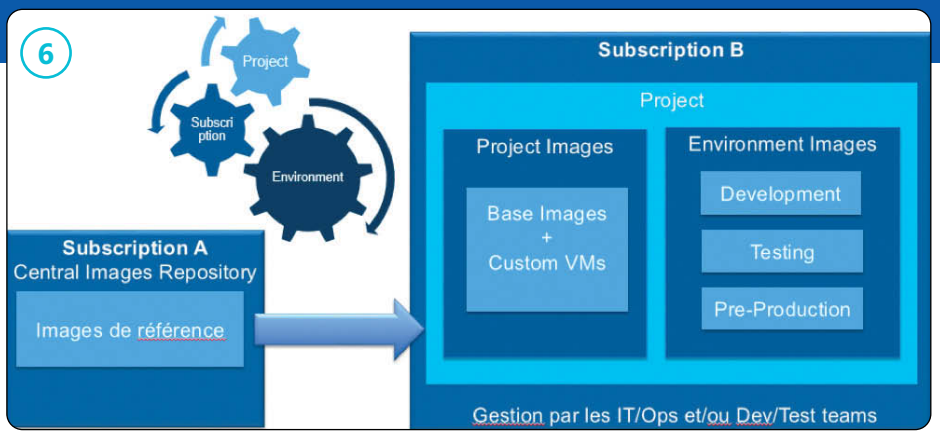
Acquérir une compréhension globale du système

L'objectif est de parvenir à optimiser l'intégralité des chaînes de valeur métier dépendant de services IT, en résolvant les problématiques au plus tôt afin de limiter leur impact sur le reste du système. Pour ce faire, chaque acteur du système se doit de penser globalement, au-delà de la tâche qui lui est assignée ou du département auquel il appartient, en vue de parvenir à une compréhension profonde du système dans son ensemble.

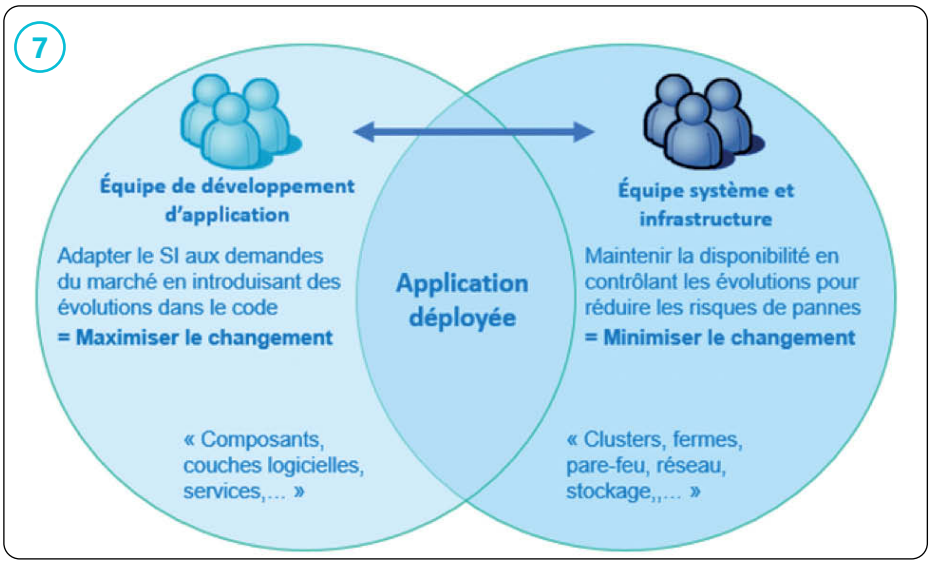
Concrètement cette démarche va se traduire par une évolution de l'organisation, de ses processus, du rôle et des périmètres de responsabilité de chacun, mais aussi en termes d'outillage et de technologie.

Etablir une collaboration plus étroite et plus efficace entre les équipes de développement et d'administration système requiert la mise en place de processus communs de déploiement (installation automatisée, configuration automatisée, vérification en mode « smoke-test » des composants ou services déployés sur l'ensemble des plateformes cibles), de supervision (détection et prévention d'incidents de performance, de sécurité, de disponibilité), de support et de

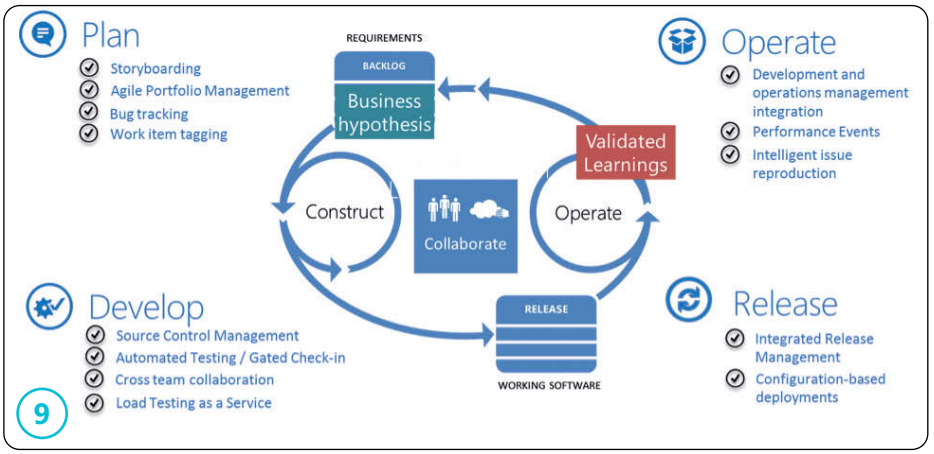
6



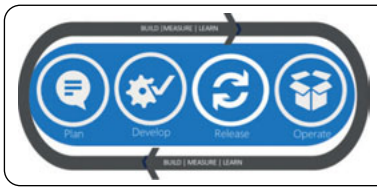
7

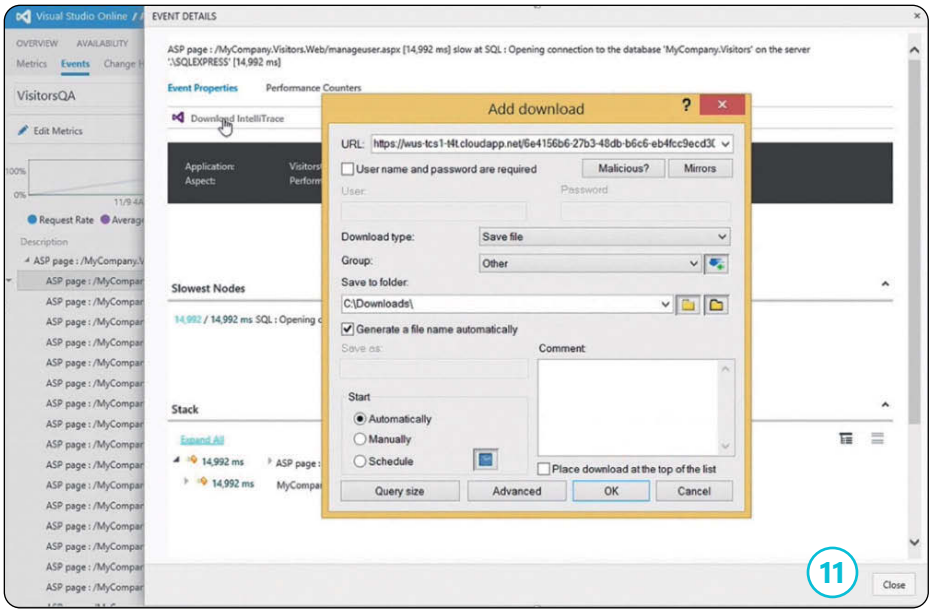


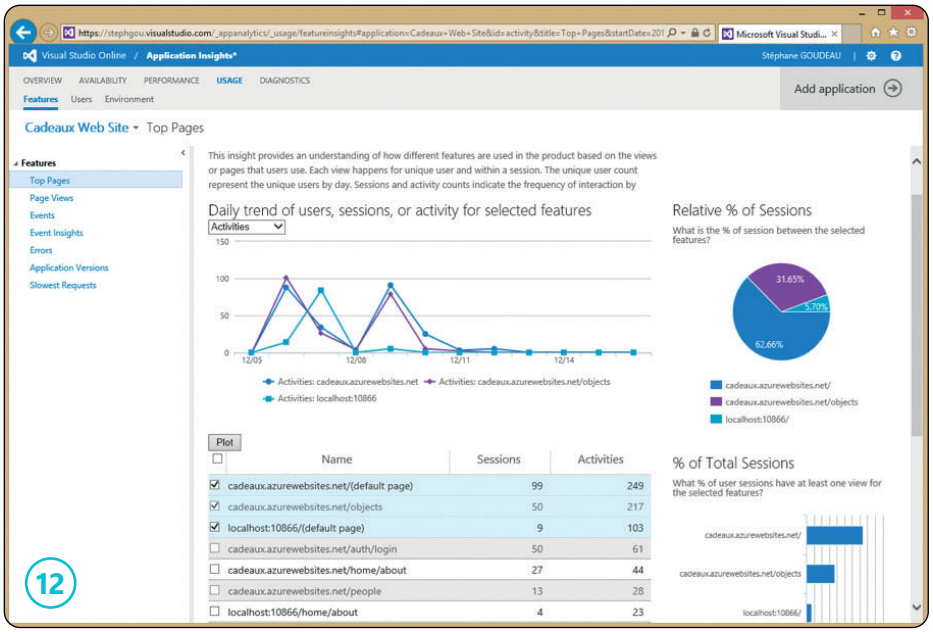
9



8







remédiation. Le « Continuous Delivery » présenté précédemment est un parfait exemple de ce type de processus.

- En effet, l'intégration continue, fondée sur une configuration centralisée du code source, s'inscrit dans une vision globale du système : l'intégrité de la totalité de la solution devient prioritaire vis-à-vis de ses composants ;
- Le déploiement continu, qui suppose la mise en place de modèles et de procédures automatisées, répétables et maîtrisées ;
- La nécessité de construire des tests de validation permet de prévenir les défaillances dans la production, ou à minima, de les détecter et de les corriger rapidement ;
- L'automatisation de la totalité de la chaîne de production logicielle offre une vue de bout en bout sur le cycle de vie de l'application Fig.9.

Mesures et processus de remontée d'information

La mise en place de ces mesures et processus de feedback cible deux objectifs :

- Il faut pouvoir corriger les dysfonctionnements au plus tôt, identifier les limites en termes de performance et de disponibilité ;
- En s'inspirant des principes d'une démarche Agile, elle vise à obtenir les informations permettant au métier de comprendre et répondre aux besoins fluctuants des clients (« Data-driven decision »).

Cela suppose la collaboration effective des équipes de développement et de gestion opérationnelle afin de déployer ces systèmes de mesure et de se les approprier :

- Métriques de performance : succès de déploiement, disponibilité du service, temps de réponse.
- Métriques de processus : nombre de nouvelles fonctions, délai de livraison d'une nouvelle version ;
- Métriques de disponibilité ;
- Métrique d'usage : loyauté des utilisateurs, pages les plus vues, types de devices utilisés.

Par exemple, comme nous l'avons vu précédemment, Visual Studio Online offre au développeur la possibilité de lancer un test de charge en s'appuyant sur des injecteurs directement montés dans Azure. L'application déployée aura donc fait l'objet de mesures permettant de garantir son bon comportement en charge avant même d'être supervisée par les équipes système et réseaux. Autre exemple, avec Visual Studio Online Application Insights, le développeur et le responsable système vont pouvoir observer le comportement des applications Azure déployées en test, mais aussi en production, ce qui permet de constater le dysfonctionnement ou la perte de disponibilité, et de faire le lien avec la version du Build correspondant Fig.10.

En cas de dysfonctionnement, Visual Studio Online permet grâce à Intellitrace d'identifier très précisément la ou les lignes à l'origine de l'erreur Fig.11.

Enfin il est possible d'accéder directement à la version du code utilisée pour déployer cette application.

Visual Studio Online Application Insights permet également d'obtenir des informations sur l'utilisation d'une application ou d'un Cloud Service. Par exemple, il est possible d'avoir une vue très fine sur les pages les plus vues (et éventuellement sur celles qui ne le sont jamais, ce qui pourrait amener à reconsidérer certains choix fonctionnels...) Fig.12. Ce portail permet aussi d'avoir des statistiques sur le nombre d'utilisateurs connectés Fig.13.

Tout un ensemble d'informations concernant le device, le système d'exploitation, la résolution d'écran,... sont également consolidées.

Culture « DevOps » : expérimentation et apprentissage en continu

La culture « DevOps » est un élément clé de la démarche. Dotée de valeurs fondamentales comme le respect mutuel, la confiance réciproque, ou la systématisation du partage de l'information et des outils, elle propose une vision positive de l'échec. En effet, les organisations doivent apprendre de leurs succès, mais également de leurs échecs. Elles doivent donc accepter de prendre des risques. L'adoption de ce type de démarche offre la possibilité d'anticiper et de définir de nouveaux besoins opérationnels résultant de cette prise de risque.

En outre, elle favorise le développement des compétences de l'ensemble des acteurs du système dans une recherche perpétuelle d'amélioration (« Kaizen »). La pratique systématique de cette approche est la condition sine qua non de cette progression. L'organisation devient ainsi

plus prompte à s'adapter, et recherche le changement plutôt que de le fuir. Un des exemples de mise en place de ce type d'évolution organisationnelle est la démarche d'introduction volontaire de défauts dans le système afin de constater au plus tôt la capacité du système à se remettre en service après un dysfonctionnement. Cela permet également de définir et partager les plans d'escalade et processus internes associés.

Automatisation de l'infrastructure

La démarche « DevOps » contribue à l'accélération des déploiements grâce à une automatisation de l'infrastructure. De multiples exemples illustrent l'intérêt de cette approche :

- Gestion de multiples environnements de déploiement comme dans le scénario DevTest précédemment présenté ;
- Contrôle du scale-out. Précisons que sur la plateforme Azure, cette gestion est maintenant nativement totalement automatisée (« autoscaling »)**Fig.14**.

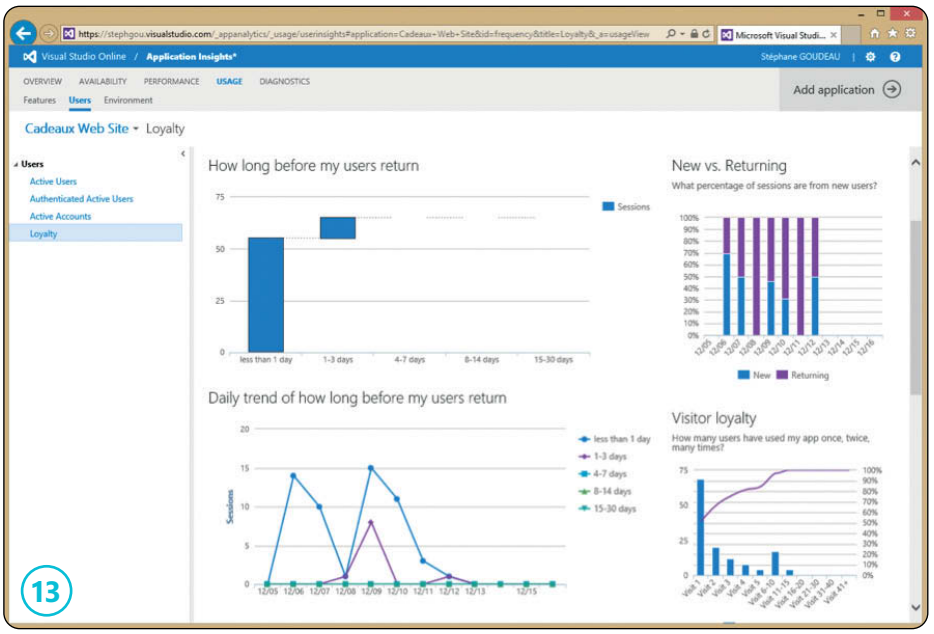
Dans le monde Microsoft, les objets permettant de faciliter l'automatisation d'une infrastructure sont les suivants :

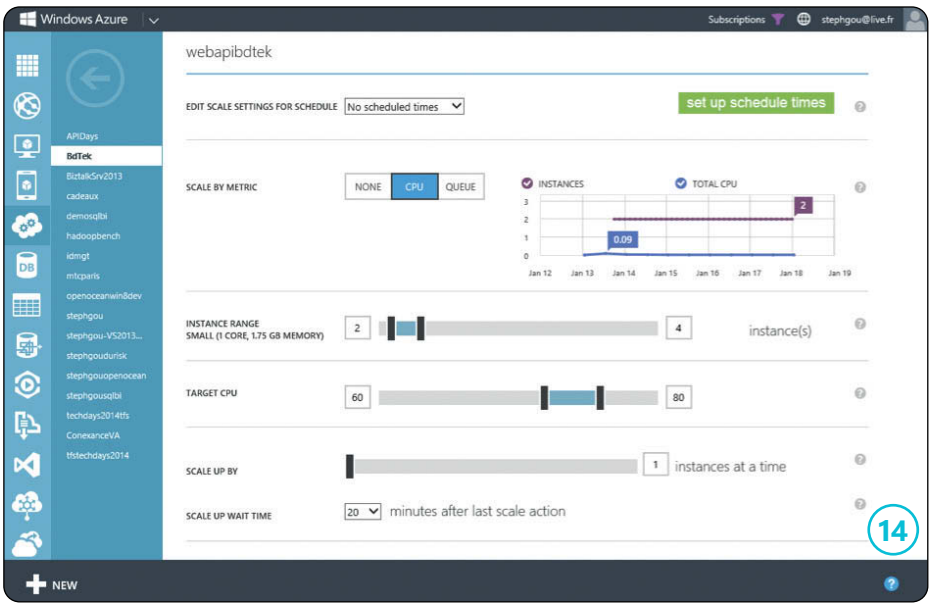
- Orchestration: Runbooks, PowerShell ;
- Format de déploiement de binaire : WebDeploy, DACPAC... ;
- Tests de vérification de déploiement: VS Web Test ;
- Modèles de configuration d'environnement: Service Template ;
- Agent de supervision : Management Pack ;
- Descriptifs d'incident : logs Intellitrace.

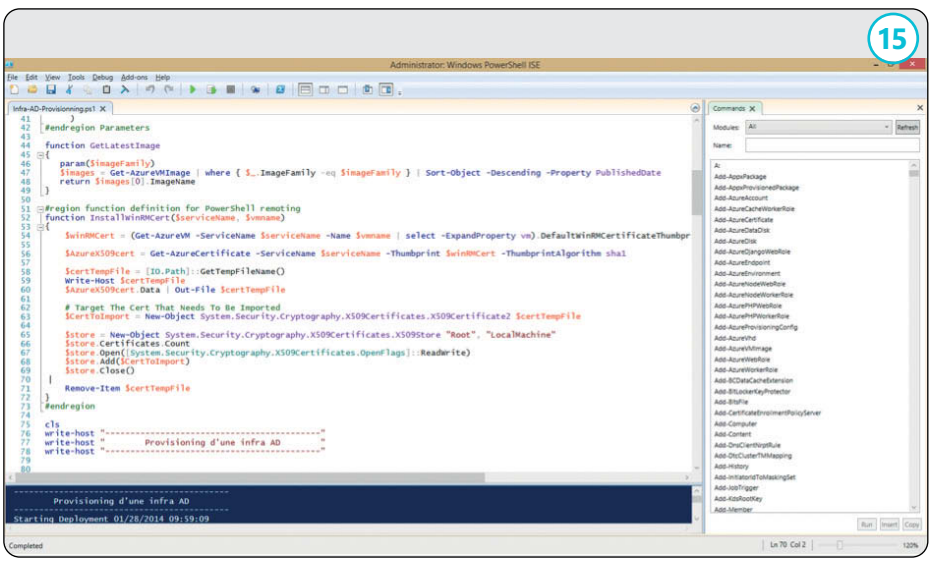
Windows PowerShell est un langage de script développé par Microsoft. Il est inclus dans Windows depuis la version 7.0 et propose un modèle de programmation orientée objet extensible par de nouvelles fonctions baptisées CmdLets. Dans le cas d'Azure, PowerShell peut être configuré pour importer ces CmdLets spécifiques avec la commande :

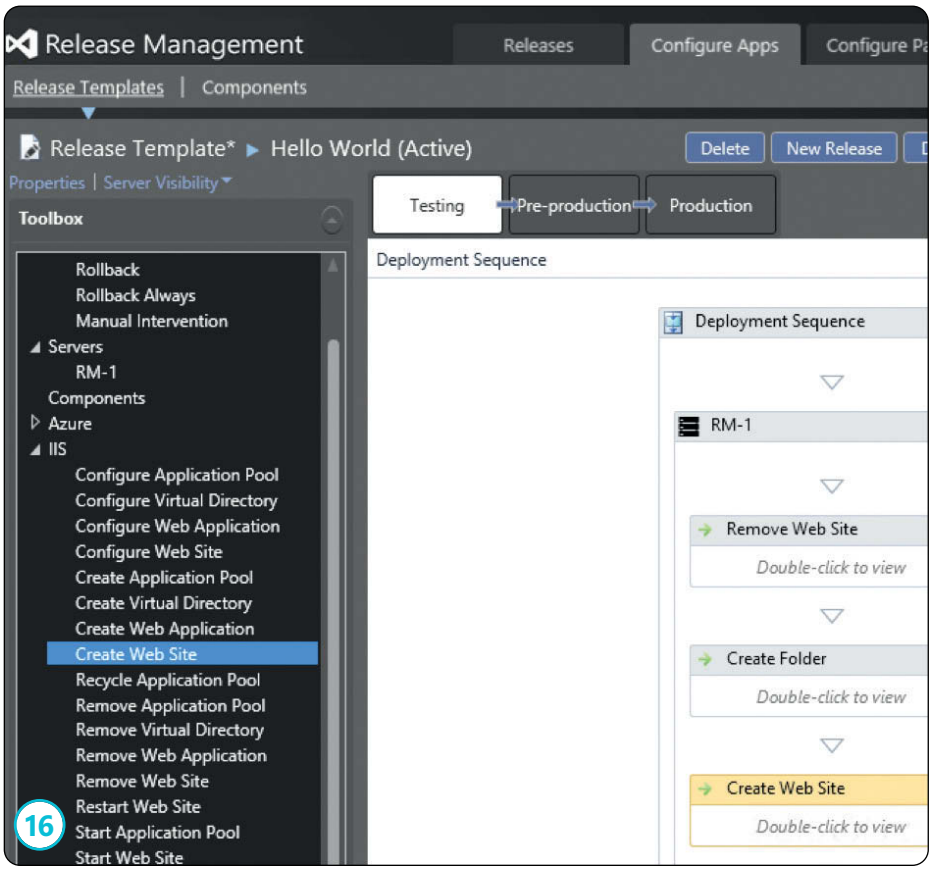
« Import-Module C:\Program Files (x86)\Microsoft SDKs\Windows Azure\PowerShell\Azure \Azure.psd1 ». Il est nécessaire d'obtenir les informations permettant d'authentifier les accès sur la souscription Azure (notamment la référence au certificat présent sur la machine cliente et déclaré dans le portail Azure) avec la commande « Import-AzurePublishSettingsFile "X:\[NomduRepertoire]\azure.publishsettings" ». Le fichier "X:\[NomduRepertoire]\azure.publishsettings" quant à lui, est obtenu directement par un appel REST sur l'URL « https://manage.windowsazure.com/publishsettings/index?client=vs&schemaversion=2.0 » PowerShell est aujourd'hui fourni avec un éditeur incluant des fonctions de coloration syntaxique et d'intellisense, Windows PowerShell ISE **Fig.15**.

De nombreux scripts Powershell sont disponibles pour automatiser le déploiement sur Azure d'infrastructures cibles fondées sur des modèles (notamment









http://gallery.technet.microsoft.com/scriptcenter et https://github.com/windowsazure/azure-sdk-tools-samples)

Ces scripts utilisent Windows Remote Management (WinRM) : implémentation Microsoft de WS-Management, protocole fondé sur SOAP et passant les firewalls http://msdn.microsoft.com/en-us/library/aa384470(v=vs.85).aspx

La délégation des accès depuis la machine cliente pour permettre l'automatisation de la configuration des serveurs distants (« multi-hop WinRM ») requiert l'activation de l'option CredSSP : « enable-wsmancredssp -role client -delegatecomputer "*.cloudapp.net" » ainsi que l'activation de la délégation d'identité dans l'éditeur de stratégie de sécurité du PC, dans le menu « Computer Configuration -> Administrative Templates -> System -> Credentials Delegation ». Enfin, il est nécessaire de sélectionner l'option « Allow Delegating Fresh Credentials with NTLM-only server authentication » et d'ajouter « AWSMAN/*.cloudapp.net » dans la section « Add Servers ».

L'automatisation du déploiement de l'application peut être assurée par un outil comme « Microsoft Release Management ».

Ce logiciel facilite la définition et la maintenance des processus de gestion des versions d'application dans le Cloud ou à demeure :

- Création des chemins d'accès de configuration ;
- Définition des orchestrations de déploiement (copie des fichiers, création de sites IIS, installation des fichiers msi…) ;
- Réutilisation sur les différents environnements de déploiement ;
- Planification de « builds » et déclenchement de la production d'une nouvelle version depuis Visual Studio ;
- Visualisation du pipeline des releases ;
- Modélisation du processus de gestion des versions d'application, **Fig.16**.

Autres exemples d'outils, les logiciels Chef ou Puppet, utilisés pour automatiser le déploiement d'une infrastructure sur la plateforme Azure en appliquant des « recettes » (recipes) qui se rapportent à des événements spécifiques de la gestion du cycle de vie de l'application (http://forge.puppe tlabs.com/msopentech/windowsazure).

Cette automatisation peut également être assurée par un développement spécifique visant à s'assurer que les composants du Cloud Service sont déployés avec une configuration appropriée. En effet, les infrastructures commencent à dépendre du code, au même titre que les applications qu'elles hébergent. Le Cloud ne fait qu'amplifier cette tendance.

Le principe est d'intégrer les spécifications d'exploitation dans celles de l'application, de sorte que l'automatisation du déploiement de l'application et de sa supervision puissent faire partie des livrables. Puisque ces spécifications sont généralement déclarées pour chaque application déployée, leur définition fait partie de la conception de l'application elle-même.

Cette approche de « provisioning d'infrastructure fondé sur un modèle » s'inscrit dans une démarche que l'on pourrait comparer à celle que propose le PaaS Azure avec les fichiers « Azure Service Definition Schema (.csdef) » et « Azure Service Configuration Schema (.cscfg) » ou les « Service templates » de System Center 2012. Afin d'éviter que le code d'une application soit étroitement couplé à un profil d'infrastructure spécifique, des «blueprints» sont associés aux applications précisant les « règles », descriptives ou prescriptives, ainsi que les paramètres d'entrée qui permettent aux services Cloud de connaître les actions à effectuer au nom de l'application. Ces stratégies sont délivrées aux services Cloud via des API bien définies et directement corrélées aux profils de services d'infrastructure que propose Azure afin de configurer un ensemble d'instances et les différentes ressources connexes. Enfin, les DSC (« Desired State Configuration ») de PowerShell 4.0 proposent un modèle de développement adapté au déploiement « continu » d'une application et de son infrastructure sous-jacente (à demeure ou dans le Cloud), car il empêche tout « écart » par rapport à la configuration cible. Ce modèle de « provisioning » utilise les extensions de langage PowerShell pour permettre des déploiements de services déclaratifs, autonomes et reproductibles.

Visual Studio Online : la révolution est déjà là

Comme nous l'avons vu précédemment, **Visual Studio Online** est un complément indispensable pour le développement et la supervision d'une application pour le Cloud, en cumulant de multiples fonctions ALM, des fonctions de gestion opérationnelles (performances, disponibilité, diagnostic) et de télémétrie (grâce à Application Insights).

Cette solution d'APM (Application Performance Management) va donc bien au-delà de ce qui était proposé jusqu'à présent, en donnant **réellement une vue de bout en bout sur le cycle de vie d'une application** (et pas uniquement sur son cycle de développement). Elle offre donc une vraie passerelle entre le monde des développeurs et celui des responsables opérationnels. **Cet outil s'intègre donc parfaitement dans une démarche DevOps appliquée à un processus de « Continuous Delivery ».**

Visual Studio Online est disponible en trois éditions : basic, avancée et professionnelle, avec des tarifs spécifiques. Il inclut Monaco. Il s'agit d'un environnement de développement doté d'un éditeur de code (pour JavaScript, HTML, CSS) accessible dans son navigateur web et dédié au développement de sites Web.

Il peut être utilisé pour éditer, coder, modifier un site web déployé sur Windows Azure Web Site, et permet d'établir des connexions avec les gestionnaires de source (Visual Studio Online ou Git).

Windows Azure