Hogyan alakítsunk ki fejlesztői és tesztelői környezeteket Microsoft Azure-ban

Nagy Gáspár *független konzulens, olyan csapatokat oktat és segít, akik fejlesztői munkájuk minőségét és hatékonyságát agilis tesztelési módszerekkel, azon belül is viselkedésalapú szoftverfejlesztéssel (behavior driven development - BDD) szeretnék növelni. Gáspár a megalkotója és jelenleg is vezető fejlesztője a nyílt forráskódú SpecFlow tesztelési eszköznek, amely meghatározó BDD eszköz a .NET platformon.*

Több mint 15 éve foglalkozik szoftverfejlesztéssel, korábban szoftvertervezőként és fejlesztői coachként is dolgozott. Rendszeres előadó nemzetközi konferenciákon és a Scrum Alliance akkreditált oktatója a Certified Scrum Developer programban. Microsoft Certified Professional az Application Lifecycle Management témakörben.

Gáspárt megtalálhatod a twitteren (@gasparnagy), a blogján (http://gasparnagy.com) vagy vállalkozásának (Spec Solutions) weboldalán (http://www.specsolutions.eu).

TARTALOMJEGYZÉK

BEVEZETŐ

Nem olyan régen az egyik fórumon valaki arról kérdezett, hogyan tudna a lokális tesztfuttatásai eredményéről emailt küldeni. Mivel elég furcsa volt a kérdés, visszakérdeztem, hogy miért is szeretne ilyet csinálni. Miért nem a build szervert használják az ilyen értesítésekre. Erre a következő meglepő választ kaptam.

> *„Jelenleg nincs build szerverünk. A rendszergazdánk jelenleg is azon dolgozik, hogy munkába állítson egy build szervert. Amíg ez megtörténik, a főnököm kérésének megfelelően, minden tesztet kézzel kell futtatnom, és az tesztek futási eredményeit emailben elküldenem a megfelelő személyeknek."*

Persze mindez csak elsőre meglepő. Ha végiggondolom a projekteket, amelyeken korábban dolgoztam, nem nehéz hasonló példákat találni. Elvesztegetett idő, felesleges pluszmunka, csúszások, csökkenő lelkesedés: várunk az infrastruktúrára.

A felhőalapú szolgáltatások napjainkra szerencsére olyan szintre fejlődtek, hogy már nem csak az alkalmazások skálázható erőforrásaként használhatók, hanem a fejlesztési folyamatot is támogathatják. Aki még nem fejlesztett a felhőben, annak furcsának tűnhet, de egyre több cég helyezi át teljesen vagy részben fejlesztési infrastruktúráját a felhőbe.

A téma annál is inkább fontos, mert az informatikában az infrastrukturális költségek aránya a humán erőforrás költségeihez (azaz a fejlesztők fizetéséhez) képest alacsony. Egy átlagos fejlesztő egy havi fizetéséből elég jó fejlesztői gépet lehet vásárolni, míg más szakmákban az alapvető eszközök csak hónapok múlva térülnek meg. Persze vannak speciális, költséges eszközigényű fejlesztési projektek is, de általánosságban a hatékonyságnövelés alapja a fejlesztők idejének jobb kihasználása kell, hogy legyen.

Ebben a cikkben kis betekintést szeretnék nyújtani abba, hogyan lehetséges a Microsoft Azure mint felhőszolgáltatás segítségével a fejlesztés hatékonyabbá tétele.

MIRE VAN SZÜKSÉG?

A fejlesztési folyamat során nem is gondolunk arra, milyen infrastruktúrát használunk a saját munkaállomásunkon kívül. Ha jól működik a rendszer, minden ott van a helyén, és végzi a munkáját.

Ahhoz, hogy áttekinthessük, mely területeken tudnak a felhőalapú szolgáltatások és a Microsoft Azure a segítségünkre lenni, gondoljuk végig mégis, hogy mi mindenre lehet szükség.

- *ALM Infrastruktúra* – az ALM (Application Lifecycle Management, alkalmazás életciklus menedzsment) infrastruktúra felel az alkalmazásfejlesztés során felmerülő adatok (pl. követelmények, feladatok vagy épp a forráskód), valamint tevékenységek tárolásáért és kezelésért.
- *Build szerver* – A csapatmunka és a minőségbiztosítás legfontosabb eszköze. A build szerver az alkalmazás kódjának fordításán és az automatizált tesztek futtatásán keresztül biztosítja, hogy az elkészült feladatok eleget tegyenek a követelményeknek, és megfeleljenek a minőségi elvárásoknak.
- *Build/teszt ágens* – A build szervereknél általában elkülönül a buildek koordinálásának és az eredmények megjelenítésének szerepe attól a szolgáltatástól, amely az alkalmazás fordítását és a tesztek futtatását végzi. Ez utóbbinak, amit build illetve teszt ágensnek (build/test agent) is neveznek, alkalmazkodnia kell a projekt speciális igényeihez (pl. a fordításhoz speciális SDK-kra lehet szükség).
- *Tesztszerverek/-rendszerek* – Ahhoz, hogy az alkalmazás elkészült részeiről visszajelzést szerezhessünk (akár a tesztelőktől, akár az ügyfél képviselőitől), szükség van arra, hogy az alkalmazás aktuális állapotát valahova telepítsük. Ehhez általában több tesztszervert is használunk, amelyekre az alkalmazás meghatározott szabály szerint kerül telepítésre (pl. a fejlesztői tesztrendszerre automatikusan telepítésre kerül az alkalmazás, de pl. az ügyfél által elérhető „demókörnyezetbe” csak akkor, ha a tesztelők bizonyos minőségi követelményeket ellenőriztek).
- *Teljesítménytesztrendszer* – A teljesítmény ellenőrzéséhez az alkalmazás használatát kell szimulálni. Ehhez olyan bonyolult infrastruktúra kiépítése szükséges, amely képes megfelelő számú felhasználói interakciót generálni, a rendszer működését monitorozni és az eredményeket összesíteni.
- *Megosztva használt fejlesztői rendszerek* – Bizonyos speciális alkalmazások esetén a fejlesztői környezet kialakítása nehéz, mert a fejlesztés speciális hardvereket vagy szoftvereket igényel. Ilyenkor – különösen az alkalmazás karbantartási szakaszában, mikor már ritkán van szükség fejlesztésre – egy megosztva használt fejlesztői gép hasznos lehet.

Természetesen egy adott projektben nincs feltétlenül szükség az összes fent említett infrastruktúrára, de ALM-, build- és tesztrendszerekre a legtöbb projektnek szüksége van.

INFRASTRUKTURÁLIS KORTÖRTÉNET

Amikor dolgozni kezdtem, a szerverek minden esetben önálló fizikai gépeket jelentettek. Egy ilyen szerver beszerzése és beüzemelése hosszadalmas folyamat volt, amit az ár-érték-arány pontos feltérképezése és a megtérülés kiszámítása kellett, hogy megelőzzön. Ez volt az infrastrukturális támogatottság középkora, az ózonszagú szerverszobák világa, ahol mi, fejlesztők is besegítettünk a túlterhelt rendszergazdának, aki egyszerre csavarozta a gépeket, konfigurálta a hálózatot, és valahol a sokadik ablakban néha arra is jutott ideje, hogy a tesztrendszerünket telepítse. A „hány programozó kell egy villanykörte cseréjéhez” vicc realitássá vált, amikor egy komplett projektteam állta körbe és látta el jó tanácsokkal azt a fejlesztőt, aki nagy nehezen elvállalta, hogy kicserélje a szerver hibás meghajtóját.

Sok évvel később aztán egyszer a főnököm boldogan jelentette be: virtualizált szerverekre térünk át, és ezennel minden gondunk megoldódik. Vettünk két nagyobb teljesítményű szervert, és ezzel a fejlesztési infrastruktúra újkorába kerültünk. Aki már átélte ezt a váltást, vagy jelenleg is ilyen környezetben dolgozik, tudja, hogy sajnos az élet még virtualizált szerverekkel sem habostorta. A végtelennek tűnő szerverkapacitás

hamar felaprózódik, ahogy egyre több virtuális gép kerül beüzemelésre, és egyszer csak azt vesszük észre, hogy a build csigalassú, a tesztrendszer olyan szinten erőforráshiányos, hogy ciki rajta demózni az ügyfélnek, és megint csak heteket kell várni, hogy egy új virtuális szerver munkába álljon.

Miért is nem jelent megoldást sok cégnél a virtualizált szerverek bevezetése? Az ilyen rendszerek felállítása és üzemeltetése speciális tudás megszerzését igényli a rendszergazdák részéről. Érteni kell a speciális licensz-kezeléshez, a monitorozáshoz, és meg kell tudni tervezni az igények és erőforrások megfelelő összerendelését. Sokan alábecsülik ezt a tanulási folyamatot, virtuális szervereiket „hagyományos" szerverüzemeltetési módszerekkel próbálják meg menedzselni, ezáltal a virtualizáció lényegét és előnyeit nem tudják kihasználni.

A másik tipikus probléma, hogy a fejlesztőknek valójában legtöbbször nem virtuális gépekre, hanem szolgáltatásokra van szükségük. Tegyük fel, hogy webes alkalmazást fejlesztünk, és szeretnénk felállítani egy olyan tesztrendszert, amelyen a tesztelők ellenőrizni tudják az eredményeket. Az alkalmazásunk jelenleg még csak egy egyszerű webalkalmazás, nem igazán tűnik szükségesnek egy saját virtuális gép beüzemelése. Ezért a rendszergazda egy megosztottan használt teszt-webszervert konfigurál be az alkalmazásunk számára, amelyen már más alkalmazások is futnak. Ezáltal az azonos szerveren futó alkalmazások között bizonyos függőségek alakulnak ki, nehezebbé válik az erőforráselosztás és annak eldöntése is, hogy melyik alkalmazás „húzza le" a többieket. A megosztottan használt szerver különböző, nehezen analizálható hibák forrása is lehet. Összességében ez a megoldás a fejlesztők számára nem jelent lényegi előrelépést.

A harmadik probléma nem is probléma igazán, hanem inkább lehetőség. Ha adottak a körülmények az infrastruktúra gyors és egyszerű bővítésére, ez pozitívan hat az innovációra és az új ötletek megjelenésére. Ha fél óra alatt fel lehet állítani egy új tesztrendszert, a fejlesztők bátrabban mernek új ötleteket kipróbálni. Ezek az ötletek sokszor sokkal fontosabbak az üzleti siker szempontjából, mint a nehezen keresztülvitt, pár százalékos hatékonyságnövelő intézkedések. Az innovatív környezet kialakítása és az ehhez szükséges infrastrukturális lehetőségek fontos szerepet kell, hogy játszanak a fejlesztési stratégiában.

Persze mindamellett, hogy az innováció fontos, az üzemeltetésnek is fel kell készülnie a nagyobb számú, de egyszerűbb, rövidebb élettartamú virtuális rendszerek kezelésére. A felkészületlen rendszergazda könnyen abban a helyzetben találhatja magát, hogy hirtelen sokkal több rendszert kell felügyelnie, amelyekről sokkal kevesebb háttérinformációval rendelkezik. „Használja valaki még a prj123-test01.corp.hu virtuális szervert, vagy leállíthatom?" – gondolom ismerős az ilyen kör-email.

A fejlesztési infrastruktúra legújabb korát a felhőalapú fejlesztés képviseli, ami az fejlesztési rendszer üzemeltetésének alacsonyabb szintjét (pl. fizikai infrastruktúra, virtuális szerver beüzemelése, operációs rendszer) leválasztja a fejlesztés szempontjából releváns szolgáltatásokról. Ez lehetőséget ad arra, hogy a csapat, beleértve a fejlesztőket és a rendszergazdákat is, a projekt megoldására fókuszáljon.

Ahogy ezt még a későbbiekben részletesebben kifejtem, a fejlesztés felhőalapú támogatásával a fizikai, illetve belső virtualizált rendszerek üzemeltetésének problémáira megoldást kaphatunk.

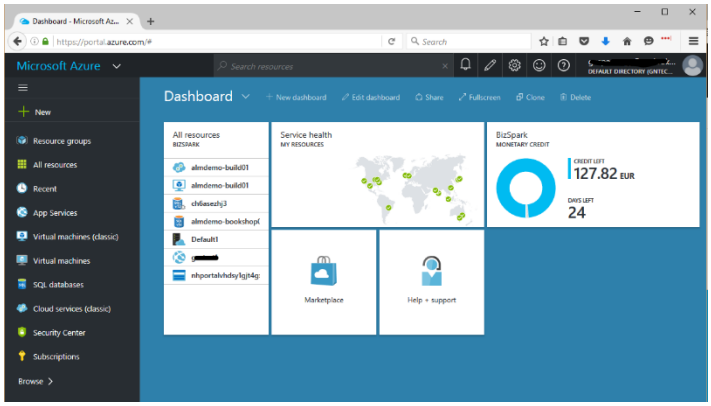
A felhőalapú fejlesztés bevezetése általában lépésenként és folyamatosan történik. A legtöbb esetben a végeredmény valamilyen vegyes rendszer, ahol bizonyos folyamatok a cégen belül üzemeltetett szervereken, bizonyos folyamatok a felhőben, a Microsoft Azure szolgáltatásainak segítségével folynak.

MIÉRT AZURE?

Bár mint ahogy említettem, a fejlesztés felhőalapú támogatása általában lépésenként történik, azért a stratégia kialakításához fontos ismerni, milyen előnyökkel jár egy ilyen infrastrukturális modell, illetve konkrétan a Microsoft Azure bevezetése. Ebben a fejezetben néhány ilyen szempontot vizsgálok meg részletesebben.

SKÁLÁZHATÓ

Amikor a fejlesztés felhőalapú támogatásának előnyeiről esik szó, a skálázhatóság mindenképpen az első helyen kell, hogy szerepeljen. Viszont ebben a környezetben a skálázhatóság másképp értelmezendő, mint a klasszikus felhőalapú szolgáltatásoknál. Általában a skálázhatóság azt jelenti, hogy a szolgáltatást kiszolgáló erőforrásokat (például egy webes alkalmazást kiszolgáló webszerverek számát) a terhelésnek megfelelően dinamikusan növelni illetve csökkenteni lehet.



Amikor a fejlesztés támogatására az Azure-t használjuk, az ilyen fajta skálázhatóságra ritkán van szükség. A fejlesztés során az erőforrásigények változását sokkal inkább a projekt életciklusa határozza meg. Ez jelenthet pár hónapos vagy akár pár órás aktív időszakot, amikor több, illetve nagyobb teljesítményű erőforrásokra (build ágens, tesztszerver, stb.) lehet szükség. Az Azure, illetve a rá épülő fejlesztői szolgáltatások mindegyike könnyen, bizonyos esetekben automatikusan skálázható. Amikor új szervert vagy szolgáltatást kell beindítanunk, a skálázási feladatok az Azure Portalon keresztül (https://portal.azure.com) rövid időn belül elvégezhetők. A legtöbb művelet, beleértve például egy új virtuális gép létrehozását is, scriptelhető, és így a gyakran előforduló feladatokat automatizálni is lehet.

EGYSZERŰ

Az Azure Portal és a többi Azure-ra épülő fejlesztői szolgáltatás úgy lett kialakítva, hogy a műveleteket egyszerűen és könnyen el lehessen végezni. Különösen igaz ez a leggyakrabban használt műveletekre. Ezáltal az Azure használata kis idő alatt elsajátítható, a betanulási költség elfogadható keretek között tartható.

GYORS

Természetesen megfelelő célhardver megvásárlásával céges kereteken belül is megfelelő teljesítményű rendszereket lehet kialakítani. Viszont ezen hardvereket (a viszonylag nagy egyszeri bekerülési költségük miatt) általában hosszabb ideig, több évig használják. A hardverek fejlődését látva könnyű belátni, hogy a ma erős szerver két év múlva már legfeljebb közepes kategóriába kerül.

Mivel a fejlesztői infrastruktúra nem termel közvetlenül bevételt, ezért az idővel „csökkenő” teljesítmény hatását nehéz számítani. Nehéz azt a pontot megtalálni, amikor a teljesítmény már nem elégséges, és új

szervert kell beszerezni. Általában a „még jó lesz” meggondolás érvényesül, és a fejlesztés hatékonysága észrevétlenül romlik le.

Az Azure-ban elérhető erőforrások (CPU, memória, stb.) ugyanazok, mint amiket a céges szerverekben használnak, viszont ezek folyamatosan cserére kerülnek, és a modernebb vagy nagyobb teljesítményű virtuális szerverek és más erőforrások rendre megjelennek az Azure-ban. Ezek követése és a használt szolgáltatások megfelelő teljesítményszinten tartása sokkal könnyebb, különösen, hogy az Azure-ban nem fizikai szerver szinten, hanem szolgáltatás vagy virtuális gép szinten lehet a megfelelő erőforrásokat megadni.

Mivel ezek a beállítások könnyen, percek alatt módosíthatók az Azure Portalon keresztül, könnyen kipróbálható, hogy az egyes beállítások milyen hatással vannak a fejlesztésre, például mennyi idő alatt fordul le a projekt, vagy mennyi ideig tart a tesztfuttatás.

Modern

Az Azure, mint felhőalapú szolgáltatás leginkább az infrastruktúra-mint-szolgáltatás (Infrastructure as a service, IaaS) kategóriába tartozik. Ahogy azt az előző szakaszban említettük, a szolgáltatás alapjául szolgáló fizikai infrastruktúrát (szerverfarmokat, hálózati elemeket) folyamatosan fejlesztik. Ezzel az infrastruktúra modernitása folyamatosan biztosítva van.

Bizonyos Azure-ra épülő komponensek azonban, például a Visual Studio Team Services, vagy maga az Azure Portal, szoftveres szolgáltatásokat nyújtanak. Ilyen értelemben ezek a szoftver-mint-szolgáltatás (Software as a service, SaaS) kategóriát képviselik. Az ilyen szolgáltatások általános jellemzője, hogy sokkal dinamikusabban követik a kor és a felhasználók igényeit, és így biztosítható, hogy a fejlesztés hosszú távon is modern környezetben folyjon.

Segíti az innovatív, önmenedzselő csapatok kialakulását

A fejlesztés támogató infrastruktúrájának megtervezését a fejlesztők és a rendszeradminisztrátorok közösen végzik.

Az Azure és a rá épülő fejlesztői szolgáltatások (pl. VSTS) úgy lettek kialakítva, hogy azokat fejlesztők is könnyen menedzselni tudják. Ez azt jelenti, hogy segítségükkel könnyen felállíthatók olyan önmenedzselő csapatok, amik a projekten belüli adminisztrációs és konfigurálási feladatokat maguk el tudják végezni, és kevésbé függnek a rendszeradminisztrátorok időbeosztásától. Ez hatékonyabb napi munkamenetet eredményez, ugyanakkor a fejlesztői ötletek kibontakozását is segíti.

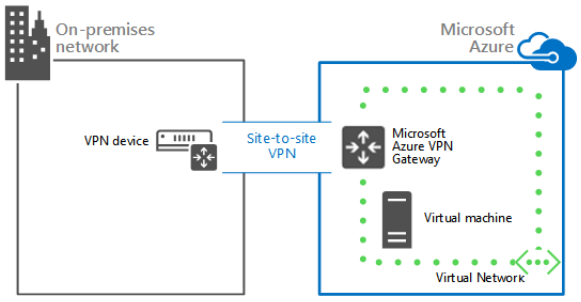
A fejlesztők valamilyen szintű bevonása a rendszerek menedzselésébe a DevOps szemlélet kialakításában is szerepet játszhat.

Kombinálható cégen belül üzemeltetett infrastruktúrával

Amikor az Azure-t szoftverszolgáltatások (SaaS) üzemeltetéséhez használjuk, például webes alkalmazáshoz, legtöbbször a teljes infrastruktúra (pl. webszerverek, adatbázisszerverek, stb.) az Azure-ban üzemeltethető.

Amikor a fejlesztés támogatására használunk felhőalapú szolgáltatásokat, legtöbbször valamilyen vegyes infrastruktúra kialakítása szükséges, azaz az infrastruktúra bizonyos elemei (amelyek nem igényelnek speciális

hardvert, és nagyobb szükség van a skálázhatóságukra) az Azure-ban kapnak helyet, másokat pedig a cégen belül tárolt szerverek szolgáltathatják.



Az Azure szolgáltatásait biztonságos VPN kapcsolattal köthetjük össze a céges hálózattal. Ehhez az Azure szolgáltatásokat olyan virtuális hálózatba (VNet) kell szerveznünk, amelyet egy site-to-site VPN kapcsolattal lehet a céges hálózathoz kapcsolni. Megfelelő routing beállításokkal a két hálózat elemei kölcsönösen kommunikálni tudnak egymással, azaz az Azure szolgáltatások úgy kezelhetők, mintha azok a cég belső hálózatán lennének. Természetesen ez esetben letiltható, hogy az Azure szolgáltatásokat közvetlenül az internetről el lehessen érni.

A KÖLTSÉGEK A HASZNÁLATTAL ARÁNYOSAN JELENTKEZNEK (NINCS EGYSZERI BEFEKTETÉSI KÖLTSÉG)

Az Azure esetében a szolgáltatások árazása használatalapú, azaz nincs szükség egyszeri nagyobb befektetésre. A szolgáltatások kis költséggel vagy épp ingyen kipróbálhatók, ami az infrastruktúra-tervezést és -bevezetést egyszerűsíti, a kockázatokat csökkenti.

Az Azure bizonyos fejlesztői szolgáltatásai (pl. VSTS) havi/éves előfizetésalapúak, mások a használat idejétől és a felhasznált erőforrások idejétől függenek. Például a virtuális gépeket, amennyiben nincsenek használatban, le lehet állítani, így gyakorlatilag csak a használat idejére kell díjat fizetni (a kikapcsolt virtuális gép adatainak tárolása minimális költséggel jár).

Az árak minden esetben tartalmazzák a szükséges szoftverlicenszek díját is, azaz például egy Windows-alapú virtuális gép beállításakor nem kell külön Windows-licenszt beszerezni.

Bár a magyarországi cégek ritkábban veszik figyelembe, de mindenképpen megemlítendő, hogy a felhőalapú infrastruktúrák kevésbé terhelik a környezetet, mivel a szerverfarmokon a nagy mennyiségű számítógép áramellátása, hűtése és hulladékkezelése hatékonyabban oldható meg, mint az egyes cégek szerverszobáiban.

Az költségek becsléséhez „Pricing calculator” használható, amely elérhető a https://azure.microsoft.com/en-us/pricing/calculator/ címen.

AZ MSDN-ELŐFIZETÉSEK HAVONTA FELHASZNÁLHATÓ KREDITPONTOKAT TARTALMAZNAK

Amennyiben a fejlesztők rendelkeznek MSDN-előfizetéssel, ezekhez – előfizetéstől függően – különböző mennyiségű, havonta megújuló kreditpontot kapnak. Ez a kredit fejlesztésre és tesztelésre szabadon felhasználható. Még a legkisebb MSDN-előfizetés kreditjei is elegendőek ahhoz, hogy például a fejlesztő bizonyos feladatokat időszakosan bekapcsolt virtuális gépeken végezhessen el.

Alapesetben, amennyiben a kredit a hónapforduló előtt elfogy, a szolgáltatások a hónapforduló végéig leállnak, és nem generálnak fizetendő költséget, azaz az Azure szolgáltatások biztonsággal kipróbálhatók költségek nélkül. Ki szeretne passziánszozni egy 20 magos 140 GB RAM-mal rendelkező szerveren?

FEJLESZTŐI INFRASTRUKTÚRA AZ AZURE-BAN

Ebben a fejezetben néhány konkrét példát mutatok be arra, az Azure-alapú szolgáltatások milyen szituációkban tudják a fejlesztést támogatni.

PROJEKTTÁMOGATÁS VISUAL STUDIO TEAM SYSTEM SEGÍTSÉGÉVEL

Az ALM (Application Lifecycle Management, alkalmazás életciklus menedzsment) olyan átfogó fogalom, amin az alkalmazásfejlesztés során felmerülő adatok és tevékenységek tárolását és kezelését értjük. Az ALM eszközök legtöbbször támogatják az alkalmazás követelményeinek (pl. user story), a munka menedzselésének (pl. task) és karbantartásának (pl. bug) leírásait, valamint lehetőséget adnak a forráskód verziókezelésére.

A Microsoft Team Foundation Server (TFS), illetve annak felhőalapú változata, a Visual Studio Team Services (VSTS) a Microsoft ALM-rendszere, amit sok .NET-alapú projektben alkalmaznak.

A Team Foundation Server a teljes fejlesztési folyamatot lefedően nyújt támogatást a projekt tervezésétől a fejlesztésen át a karbantartási periódusig. Ezáltal megkönnyíti a projektek támogatását, mert nem kell különböző, az egyes részfolyamatokat támogató eszközöket integrálni és összehangolni. Bár egy új TFS projekt létrehozása és bekonfigurálása az elérhető sablonoknak köszönhetően pár perces folyamat, lehetőség van a projekt folyamatainak testre szabására és a vállalat speciális igényeinek lekövetésére. A Team Foundation Server sokrétű termék, részletes leírása meghaladja e cikk kereteit. Ezekről bővebben a https://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx címen található részletes leírás. Ebben a fejezetben a felhőalapú változat, a Visual Studio Team Services specialitásait tárgyalom részletesebben.

A Visual Studio Team Services egy a Microsoft Azure infrastruktúrán futó TFS rendszer, amiben a felhasználók vagy cégek elkészíthetik saját regisztrációjukat, amelyhez egyedi URL (pl. https://azéncégem.visualstudio.com) tartozik. Ez az URL reprezentálja a vállalat vagy csapat saját ALM-portálját, amelynek saját felhasználói, jogosultságai és beállításai vannak.

A VSTS az első öt felhasználóig ingyenesen használható, így egy most induló projektcsapat költségek nélkül ki tudja próbálni a felhőalapú ALM előnyeit. Több felhasználó esetén a költségek a felhasználók számától függnek, de azok a fejlesztők, akik MSDN-előfizetéssel rendelkeznek, költségek nélkül használhatják a szolgáltatást. A VSTS árazásáról az https://www.visualstudio.com/en-us/products/visual-studio-team-services-pricing-vs.aspx címen lehet részletes információt találni.

Az ALM infrastruktúra – mérettől függően – komplex, több szerverből álló rendszer. Az alkalmazásszerver és az adatokat tároló adatbázisszerver legtöbbször külön gépre kerül, de sokszor van szükség a szolgáltatások további skálázására. Ennek beállítása és üzemeltetése komplex feladat. A VSTS használatával erre nincs szükség, hiszen a szolgáltatás menedzselése központilag történik.

A VSTS fejlesztése rövid, háromhetes ciklusokban történik. Ez a gyakorlatban azt jelenti, hogy az újítások és javítások az alkalmazásban rövid időn belül megjelennek, és használatba vehetők. Bár a telepített TFS is relatív sűrű, negyedéves release-ciklusokat használ, a cégek a frissítések költségeinek és az esetlegesen keletkező időkiesések csökkentésének érdekében általában ennél jóval ritkábban, 1-2 évente frissítik a rendszerüket.

A VSTS előnyei között a már említett egyszerűbb karbantartás és az új funkciók hamarabbi megjelenésén túl még az integrált felhőalapú buildrendszert (lásd lejjebb) és a könnyebb, megbízhatóbb és biztonságos elérhetőséget lehet megemlíteni. Ez utóbbi segítségével az ALM-rendszerünket bárhonnan, biztonságos

kapcsolaton keresztül elérhetjük mind webböngészőn, mind Visual Studion keresztül. Így az elosztott munkavégzés könnyebbé válhat, és egyszerűbben bevonhatjuk az ügyfelet is a projekt folyamatába.

FEJLESZTŐI MUNKAÁLLOMÁS AZ AZURE-BAN

A konkrét fejlesztés leghatékonyabban saját munkaállomáson végezhető. Ezen a felhőalapú szolgáltatások megjelenése sem változtatott (legalábbis eddig).

Viszont van néhány speciális eset, amikor jól jön egy másodlagos munkaállomás. Ezt természetesen meg lehet oldani a fejlesztő gépén futó virtuális gépekkel, de ez két problémát vet fel. Egyrészt a fejlesztői gépeket megfelelő teljesítményűre kell méretezni. Másrészt – ami valószínűleg a költségek szempontjából fontosabb – ez a gyakorlatban azt okozza, hogy a fejlesztők maguk kezdik konfigurálni és telepíteni ezeket a virtuális gépeket, rejtett, de jelentős projektidő-kiesést okozva.

A Microsoft Azure portálon keresztül percek alatt létre lehet hozni egy virtuális gépet. Ehhez többfajta virtuális gépsablon közül választhatunk, amelyek között megtalálhatunk Windows Server, Client és különböző Linux disztribúciókat is. Ezek mindegyike „ready-to-run”, azaz létrehozás után azonnal használatba vehető, további telepítésre vagy konfigurálásra nincs szükség. Amennyiben több, hasonló speciális konfiguráció létrehozására van szükség, a beállításokhoz tartozó merevlemezkép elmenthető, és a virtuális gép ilyen elmentett állapotok alapján is létrehozható.

Nézzük meg kicsit részletesebben, hogy milyen esetekben lehet hasznos, rövid idő alatt új virtuális munkaállomás létrehozása.

Fejlesztés Linux rendszeren

Az egyik ilyen eset az, amikor egy speciális projekt vagy komponens fejlesztéséhez olyan konfigurációra van szükség, amely nem egyezik az alap fejlesztői környezettel. Például egy .NET fejlesztőnek egy kis modult kell fejlesztenie Linux alatt. Ehhez – valószínűleg csak korlátozott időre – szükség lehet egy Linuxos gépre. Bár a fejlesztő valószínűleg rendelkezik valamennyi Linux ismerettel, valószínűleg nem gyakorlott Linux rendszergazda. Az Azure virtuális gépsablonjai között számos Linux disztribúció és verzió szerepel. Ezek alapján a virtuális gép létrehozása pár perc alatt elvégezhető, és a fejlesztéshez szükséges csomagok telepítése után gyakorlatilag egy órán belül a munkaállomáson hatékony fejlesztés kezdhető.

Régi alkalmazások karbantartása

Hasonló a helyzet, amikor nem feltétlenül más operációs rendszerre, ám olyan valamilyen környezetre van szükség, amely nem kompatibilis a fejlesztői géppel vagy túl bonyolult a beállítása. Nekem két ilyen projektem is volt, az egyiknek Visual Studio 2008-ra, a másiknak a Delphi egy régebbi verziójára volt szüksége. Mindkét projektet sokkal kényelmesebb volt egy-egy önálló virtuális gépen fejleszteni.

Az ilyen speciális beállítások általában a régebbi, már karbantartási szakaszban levő projektekre jellemzők. Az ilyen projekteken általában már csak igény szerinti, időszakos munka folyik. Bár a projekt általában nem egy ember, hanem egy kisebb csapat felelősségi körében marad, egyszerre csak egy fejlesztő dolgozik rajta. Ilyen esetekben érdemes lehet elgondolkodni egy megosztva használt virtuális munkaállomás beállításán, amin mindig az a fejlesztő dolgozik, aki a projekt feladatait ellátja. Így elkerülhető az, hogy a projektkörnyezet beállításával felesleges időt kelljen eltölteni. Az sem nagy probléma, ha esetleg a projekt újra aktívabb szakaszba kerül, ilyenkor ugyanis a virtuális gép klónozásával viszonylag rövid idő alatt további projektkörnyezeteket lehet kialakítani.

Visual Studio új verziójának kipróbálása

Talán kissé speciális eset, de mindenképpen említésre méltó, hogy a Visual Studio fejlesztői csapata az új verziók kiadás előtti verzióit elérhetővé szokták tenni Azure virtuális gépsablonok formájában. Ez azt jelenti, hogy percek alatt létrehozhatunk egy olyan új fejlesztői gépet, amelyen az új Visual Studio anélkül kipróbálható, hogy a saját fejlesztői környezetünket a korai verzió esetleges mellékhatásaival veszélyeztetnénk, illetve hogy értékes időt veszítenénk el a telepítéssel.

FELHŐALAPÚ BUILD

Egyre több projekt használ build szervereket az alkalmazás minőségének ellenőrzésére. A build szerver az alkalmazás kódjának fordításán és az automatizált tesztek futtatásán keresztül biztosítja, hogy az elkészült feladatok megfeleljenek a követelményeknek és eleget tegyenek a minőségi elvárásoknak.

A projektcsapatok általában többféle, különböző célokat szolgáló buildeket definiálnak. A folyamatos integrációs (continous integration, CI) build minden elvégzett és a verziókövető rendszerbe bekerült fejlesztési feladat után automatikusan lefut, és ellenőrzi, hogy a megoldás jól illeszkedik, integrálódik-e az alkalmazás aktuális állapotába. Más buildek hosszabb, alaposabb ellenőrzéseket végezhetnek (nightly build), vagy épp a termék kiadásához szükséges telepítő csomagot állítják össze, esetleg telepítik is (release build).

A projekthez tartozó buildek felhőalapú definiálására és menedzselésére a Visual Studio Team Services-ben (VSTS) nyílik lehetőség (lásd Projekttámogatás Visual Studio Team System segítségével). Itt a projekthez tetszőleges számú buildet vehetünk fel, egyebek között megadhatjuk, milyen lépéseket kell végrehajtani egy build során (pl. fordítás, tesztfuttatás), hogy mikor induljon el automatikusan a build (trigger), illetve hova kerüljön a build eredménye.

Azt, hogy milyen lépéseket kell a build során végrehajtani, egy listából választhatjuk ki, amely tartalmazza az összes tipikus build lépést, de lehetőséget ad egyéni lépések megadására is (pl. PowerShell scripteken keresztül). A legtöbb build tartalmaz valamilyen fordítási és tesztfuttatási lépéseket. Ezen lépések lefuttatása, különösen a tesztfuttatás, általában idő- és erőforrásigényes, ezért a jobb skálázhatóság érdekében ezeket nem a build szerver, mint a buildek koordináló szolgáltatása, hanem egy külön szolgáltatás, a build ágens (build agent) végzi. Egy build szerver általában több független build ágenssel dolgozik együtt.

Hosztolt build ágensek

A VSTS azon túl, hogy koordinálja a buildeket, egy Azure-alapú build-ágens-rendszert is nyújt. (Saját magunk által menedzselt ágenst is hozzá lehet adni természetesen, lásd később.) Ezeket a virtuális ágenseket, amelyeket „Hosted Agent”-nek neveznek, úgy kell elképzelni, hogy megfelelő számú Azure virtuális gép van tesztfuttatásra előkészítve, és amikor olyan új VSTS build indul el, ami hosztolt build ágensek használatára van bekonfigurálva, az egyik szabad build ágenshez rendelődik, és a build lépései azon a virtuális gépen futnak le. Ha épp nincs szabad ágens, a build-kérelem bekerül egy sorba, és megvárja, amíg egy virtuális build ágens felszabadul.

A hosztolt build ágensek használatát a VSTS felügyeli, és szükség esetén újabb ágenseket indítanak (vagy állítanak le) annak érdekében, hogy a várakozás ne legyen túl hosszú.

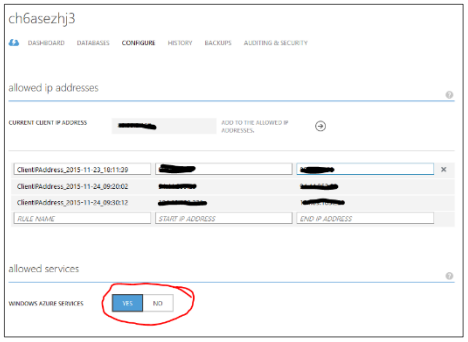
Mivel rengeteg projekt használja a VSTS-t, a build igényeik kiegyenlítik egymást, és ez hatékonyabb üzemeltetést eredményez. Persze ami számunkra fontosabb, hogy nem kell aggódnunk, ha egyszerre akár öt

projektünk is akar buildeket indítani, valamint azon sem kell gondolkodnunk, mit kezdjünk az éjszakai kihasználatlansággal.

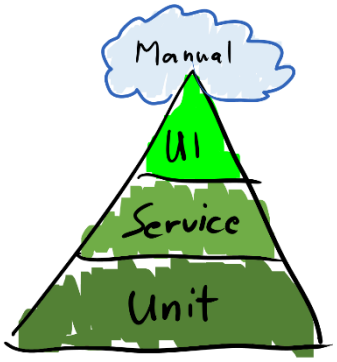
A VSTS által definiált hosztolt build ágensek abban különböznek a magunk által definiáltaktól, hogy itt nincs lehetőségünk a virtuális géphez közvetlenül hozzáférni, például hogy valamit előre telepítsünk rá. Ez persze a gyakorlatban nem akkora probléma, mert az ágensek eleve tartalmaznak sok, leggyakrabban használt komponenst, beleértve a .NET Framework-öt, de Java és node.js is telepítve van az ágenseken. Az ágensek elérhető szolgáltatásainak részletes listáját a https://msdn.microsoft.com/library/vs/alm/build/services/hosted-agent-pool#software oldalon lehet megtekinteni.

Ahhoz hogy a projektünk „kompatibilis" legyen a virtuális ágensekkel, a projektek függőségeit érdemes úgy definiálni és beállítani, hogy a fordíthatóság ne függjön a gépre telepített szoftverektől vagy szoftverfejlesztési készletektől (software development kit, SDK). Ezt legegyszerűbben úgy tehetjük meg, ha a függőségeket valamilyen függőségmenedzselő rendszeren keresztül definiáljuk. Ez .NET projektek esetében a NuGet. A NuGet használata biztosítja, hogy a függőségeket csak meghatározott, pontos verzióval rendelkező NuGet csomagokból használja a fordító. A NuGet csomagok az azonosítójuk és a verziójuk által egyértelműen beazonosíthatók. Ezeket a csomagokat nem is kell a verziókezelőbe betenni, mert ezek (bármelyik build ágensen) a megadott azonosító alapján a nuget.org oldalról automatikusan letöltődnek.

Persze nem csak a fordítás, hanem a tesztek futtatása is függhet különböző külső szolgáltatásoktól (pl. a teszt adatbázisszervert használ a működés ellenőrzéséhez). Ez különösen integrációs és rendszertesztek esetén van így. Az Azure-alapú hosztolt build ágens használata önmagában nem zárja ki az ilyen tesztek futtatását, de valahogy el kell érnünk, hogy a tesztek futtatásához szükséges külső szolgáltatások a hosztolt build ágensről is elérhetők legyenek. Például, ha a tesztünknek a futtatáshoz SQL Serverre van szüksége, akkor beállíthatunk egy Azure-ban futó SQL Servert, és ezen szerver elérését (connection string) adhatjuk meg a tesztünknek. Ahhoz, hogy az SQL Server a build ágensről is elérhető legyen, ezt külön engedélyezni kell. Az engedélyezéshez az Azure Portal-on a létrehozott SQL Server beállításainál a „Configure" fülön engedélyezzük a „Windows Azure Services" beállítást az „allowed services" szakaszban.



Bár a hosztolt build ágens kapacitás elsőre végtelennek tűnik, azért vannak korlátai. Egyrészt az alap VSTS előfizetés (az ingyenes is) csak meghatározott mennyiségű build futtatását teszi lehetővé (CPU időben mérve). Ami viszont ennél is fontosabb, hogy az integrációs tesztek számának növekedésével a tesztek lefutási ideje is nő, amelyet skálázással és párhuzamos futtatással csak ideig-óráig lehet kordában tartani. Minél hosszabb a tesztek lefutási ideje, annál később tud visszajelzést nyújtani a fejlesztőknek, és annál kisebb a valószínűsége, hogy a fejlesztők maguk is lefuttassák ezeket a teszteket a munka befejezése előtt. Erre a problémára a Mike Cohn által a Succeeding with Agile című könyvben leírt tesztelési piramis koncepció ad meg lehetséges megoldást. A piramis azt jelképezi, hogyan érdemes a különböző tesztek mennyiségét a projekten belül elosztani. A legnagyobb mennyiségben független és izolált unit teszteket érdemes definiálni, és csak a nagyobb összefüggéseket érdemes integrációs és rendszertesztekkel ellenőrizni. Mivel az integrációs tesztek elsőre



hasznosabbnak tűnhetnek, sok projektben a tesztelési piramis szintjei megfordulnak, és sok rendszer-/integrációs teszt mellett csak kevés unittesztet tartalmaznak. Az ilyen csúcsán álló tesztelési piramist szokták viccesen tesztelési jégkrémnek is nevezni (valószínűleg azok, akik épp nem tudnak mit csinálni, mert épp a tesztlefutásra várnak).

Amennyiben sikerül a tesztelési piramisnak a megfelelő arányokat megtartani, lehetőségünk nyílik arra is, hogy a hosztolt build ágenseken csak a unitteszteket futtassuk.

Felhőalapú buildelés saját teszt ágensekkel

Persze vannak olyan esetek, amikor sem a NuGet csomagok, sem a unittesztek, sem az Azure-ban tárolt adatbázisok nem segítenek, és szükség van egy saját, speciálisan beállított build ágensre. Szerencsére ez nem jelenti azt, hogy búcsút kellene intenünk az Azure-nak. A VSTS pár egyszerű lépéssel bármilyen Windows-os vagy Linux alapú gépet build ágenssé alakít. Ehhez az operációs rendszernek megfelelő csomagot kell letölteni, és a VSTS projekt elérésének, valamint egy megfelelő VSTS projekt jogokkal rendelkező felhasználó adatainak megadásával a VSTS-be beregisztrálni (részletes leírást a https://msdn.microsoft.com/en-us/library/vs/alm/build/agents/windows illetve a https://github.com/Microsoft/vso-agent/blob/master/docs/vsts.md címen lehet találni).

A VSTS build ágensek maguk kapcsolódnak a build szerverhez (HTTPS protokollon keresztül), ezért beállításukhoz nincs szükség bejövő hálózati kapcsolat engedélyezésére.

Mivel a saját build ágens konfigurálásának és regisztrálásának nincs speciális infrastrukturális feltétele, a saját build ágenseinket futtathatjuk egy Azure-ban futó virtuális gépen, a cégen belül futó virtuális gépen, de akár a saját laptopunkat is build ágenssé alakíthatjuk, ha épp úgy tartja kedvünk.

Fontos megjegyezni, hogy az ingyenes VSTS-regisztrációk az Azure hosztolt build ágenseken kívül maximum egy saját ágens regisztrálását teszik lehetővé. Természetesen a regisztráció bővíthető, amennyiben több ágensre lenne szükség.

Interaktív tesztek futtatása build során

Ahogy korábban említettem, a build közben futtatandó tesztek akkor tudnak gyors visszajelzést adni az alkalmazás állapotáról, ha a legtöbb teszt unitteszt és csak kevesebb, átfogó ellenőrzéseket végzünk integrációs tesztekkel. Az ilyen integrációs tesztek speciális esete, a piramis csúcsa, amikor a tesztek magát az alkalmazást indítják el, és a felhasználói felületen (user interface, UI) keresztül automatizálják az alkalmazást. Ez például egy webes alkalmazás esetén azt jelenti, hogy a tesztek elindítanak egy webböngészőt, és valamilyen webautomatizáló eszköz (pl. Selenium WebDriver) segítségével a böngészőbe betöltött alkalmazást vezérlik. Az ilyen tesztek úgy tesztelik az alkalmazást, ahogy a felhasználó is, ezért hasznos visszajelzést jelenthetnek. Sajnos a felhasználói felületen történő tesztelés azzal is jár, hogy a tesztek nagyságrendekkel lassabban futnak le, mint a unit- vagy az alkalmazás valamilyen technikai felületét használó integrációs tesztek. A lassabb futás mellett ezek a tesztek jobban ki vannak téve a tesztelői környezet változásainak (pl. a böngésző frissítést igényel), illetve az alkalmazás változásainak (tipikusan a felhasználói felület változik leggyakrabban). Mindezek miatt – a tesztelési piramisnak megfelelően – csak kevés folyamatot érdemes így tesztelni.

Persze a fentiek nem jelentik azt, hogy nincs szükség az alkalmazás felhasználói felületét meghajtó tesztekre. Nézzük meg tehát, hogy az ilyen tesztek milyen speciális technikai feltételeket igényelnek. Ahhoz, hogy ezt megértsük, vizsgáljuk meg, hogy is történik a tesztek futtatása a build ágenseken általában.

A build ágensek általában service-ként futnak a gazdagép operációs rendszerén. Ez azért praktikus, mert az ilyen service-eket az operációs rendszer anélkül is képes futtatni, hogy bárki be lenne jelentkezve a gazdagépre. Ez egyrészt az üzemeltetés stabilitása miatt fontos (pl. a gép képes újraindulás után folytatni szerepét, mint build ágens), másrészt azzal, hogy nincs szükség bejelentkezett felhasználóra, értékes erőforrásokat (pl. memória) spórolunk meg.

A build ágensek ilyen módon történő indítása általában jó megoldás, hiszen a legtöbb unit- és integrációs teszt alapvetően számításokat és I/O műveleteket végez. A felhasználói felületet meghajtó tesztek azonban – ahogy a nevük is jelzi – felhasználói felületet (pl. egy browser ablakot) igényelnek. Erre a service-ként futtatott build ágenseknek nincs lehetősége általában. Mindez igaz a VSTS Hosted ágensre, de ugyanúgy az általunk telepített ágensekre is, hacsak másképp nem rendelkezünk.

A felhasználói felületet meghajtó tesztek futtatásához két lehetőség közül tudunk választani. Az egyik, hogy az ágens által létrehozott Windows Service-t úgy állítjuk be, hogy az rendelkezzen az „interact with desktop” beállítással. Ezt a Windows service beállításainál tudjuk megtenni. Ez sok esetben kielégítő megoldás lehet, de néhány alkalmazás esetében nem vagy nem megbízhatóan működik. Ilyenkor megoldást jelenthet, ha a build ágensünket nem mint service-t, hanem mint interaktív alkalmazást futtatjuk egy előzetesen bejelentkezett felhasználó környezetében. Ehhez magát az ágenst (VsoAgent.exe) kell parancssorból futtatnunk.. Ilyen esetekben sem kell feltétlenül manuálisan beavatkoznunk, mert a Windows „AutoAdminLogon” szolgáltatását igénybe tudjuk venni (lásd https://support.microsoft.com/en-us/kb/324737).

Bármelyik megoldást is használjuk, mindenképpen saját build ágenst kell beüzemelnünk, hiszen a VSTS Hosted ágens nem tud interaktív teszteket futtatni.

Tesztrendszerek az Azure-ban

Ahogy a bevezetőben már említettük, ahhoz, hogy minél hamarabb visszajelzést kaphassunk a fejlesztéseink eredményéről, olyan környezetet kell kialakítani, ahol az eredményeket bemutatásra és kipróbálásra gyorsan elő lehet készíteni. Ennek egyik fontos eleme, hogy könnyen és gyorsan új tesztrendszereket tudjunk létrehozni.

A tesztrendszerek létrehozásához az Azure nagy segítséget nyújthat, hiszen az ilyen – gyakran rövid életű – rendszerek klasszikus módon történő beüzemelése nehezen kivitelezhető és olyan időigényes, hogy az ötlet, amelyről visszajelzést szerettünk volna beszerezni, kifullad, mire a rendszer feláll.

Egy ötlet kipróbálásához szükséges befektetés megtérülése nehezen számítható, az Azure használatával azonban csak a tényleges használat után kell díjat fizetni (illetve ha a fejlesztők MSDN-előfizetéssel rendelkeznek, akkor a legtöbb esetben még azt sem).

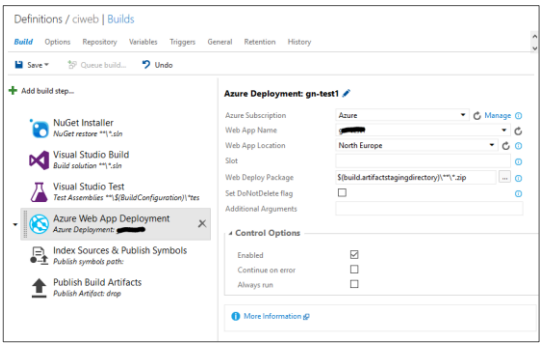
Az alábbiakban néhány tipikus esetet mutatok be Azure-alapú tesztrendszerek használatára. Bár külön nem részletezem, de mindegyik esetre igaz, hogy amellett, hogy az alkalmazásunkat manuálisan („kézzel”) telepítjük ezekre a rendszerekre, a telepítés automatizálható is. Sok esetben a telepítést megfelelően beállított buildek végzik.

Azure WebApp (App Service) tesztrendszerek

Webes alkalmazások esetében legtöbbször egy új tesztrendszer felállítása nem igényli, hogy új (virtuális) gépet üzemeljünk be. Ilyenkor a legegyszerűbb egy meglévő gép web szerverére (IIS) új webalkalmazást bekonfigurálni. Ez persze hagyományos környezetben csak látszólag ilyen egyszerű, mert nem könnyű

megtalálni azt a megfelelő, már meglévő gépet, amelyen a feltelepített alkalmazás nem okoz teljesítmény- és kompatibilitási problémákat a többi alkalmazásnak.

Az Azure WebApp funkcionalitása segítségével egyszerűen telepíthetünk webes alkalmazásokat, anélkül, hogy a gazda virtuális gép telepítésével és az erőforrásainak menedzselésével foglalkoznunk kellene. Ezt az Azure megoldja helyettünk.



A WebAppok segítségével nagyfokú flexibilitást is kapunk, hiszen az alkalmazáshoz rendelt erőforrásokat (CPU, memória) igény szerint bármikor módosíthatjuk.

Ahhoz hogy az alkalmazásunkat egy Azure WebAppra telepítsük, többfajta lehetőség közül választhatunk (közvetlenül verziókezelőből, FTP-vel, stb.) A Visual Studio-ból elérhető webes projektek és a VSTS build lépései eleve fel vannak készítve arra, hogy az alkalmazásunkat egy WebApp-ba telepítsük. Amennyiben egy buildből szeretnénk telepíteni, egy „Azure Web App Deployment” lépést kell a buildhez adnunk és a létrehozott WebAppot beállítanunk.

Bár a WebAppok elsődlegesen publikus webes szolgáltatásokra használhatóak, természetesen lehetőség van arra is, hogy csak a céges hálózatból érhessük el őket. Ennek beállításához a https://azure.microsoft.com/en-us/documentation/articles/web-sites-integrate-with-vnet/ címen találunk információt.

Azure VM tesztrendszerek

Bár a WebAppok sok esetben optimális megoldást jelentenek, bizonyos esetekben szükség lehet arra, hogy a teljes virtuális gép beállításait kontrollálni tudjuk. Erre például akkor lehet szükség, ha az alkalmazásunk nem webes szolgáltatást nyújt (pl. asztali alkalmazás), vagy ha webes az alkalmazás, de speciális telepített szoftvereket vagy egyéb szolgáltatásokat igényel.

Ilyen esetekben létrehozhatunk virtuális gépeket is, amelyeket tesztrendszerként használhatunk. A virtuális gépek esetén különböző előre megadott sablon alapján, használatra kész rendszert kaphatunk percek alatt. Természetesen lehetőség van virtuális gép létrehozására, korábban elmentett lemezkép alapján is.

Hasonlóan a WebAppokhoz, a virtuális gépekhez rendelt erőforrásokat (CPU, memória, merevelemez) igény szerint dinamikusan módosíthatjuk.

A virtuális gépek alapértelmezésben Remote Desktopon keresztül érhetők el, de lehetőségünk van arra is, hogy virtuális gépünket vagy gépeinket a céges belső hálózatba integráljuk. Ezzel megoldható az, hogy a virtuális gépek és a belső hálózat gépei kölcsönösen használhassák egymás szolgáltatásait, anélkül, hogy ezeket az interneten elérhetővé tennék. Ehhez a virtuális gépeinket úgynevezett virtuális hálózatba kell szerveznünk (VNet) az Azure portálon keresztül, és létre kell hoznunk egy site-to-site VPN kapcsolatot a virtuális hálózat és a céges hálózatunk között. A beállításról részletesebben a https://technet.microsoft.com/en-us/library/dn786406.aspx?f=255&MSPPError=-2147217396 címen lehet információt találni.

Alkalmazások operációs rendszer kompatibilitási tesztelése

A virtuális gépeket használhatjuk annak tesztelésére is, hogy az asztali alkalmazásunk megfelelően telepíthető-e és működik-e különböző operációs rendszereken.

Ehhez általában egy „tiszta” gépre van szükségünk, azaz olyan gépre, amelyre az alkalmazásunkat nem telepítették még korábban.

Az Azure virtuális gépsablonok több különböző operációs rendszerrel rendelkező (pl. Windows 8.1) gép létrehozását teszik lehetővé percek alatt, így az ilyen jellegű tesztelés gyorsan elvégezhető. (A Windows operációs rendszerek asztali verziói csak MSDN előfizetéssel érhetőek el.)

Egyéb Azure tesztrendszerek

A cikkben leginkább a klasszikus webes és asztali alkalmazásokkal foglalkoztam, de természetesen létezik sok más olyan alkalmazás is, ami egyéb tesztkörnyezetet igényelhet. Például egy adatbázisba ágyazott szolgáltatás teszteléséhez egy tesztadatbázisra lehet szükség.

Az ilyen esetek nagy többségében az Azure szolgáltatásai szintén felhasználhatók. Ehhez az Azure portálon kell böngészni, milyen egyéb szolgáltatásokat tudunk létrehozni. A lista folyamatosan bővül, úgyhogy érdemes a lehetőségeket időről időre áttekinteni.

Tesztrendszerek kezelése VSTS segítségével

Egy-egy ötlet kipróbálásához egy ad-hoc létrehozott tesztrendszer tud segítséget nyújtani. Nagyobb alkalmazás szisztematikus tesztelése azonban azt kívánja, hogy több, különböző célú tesztrendszert üzemeltessünk, amelyekre az alkalmazást meghatározott rend szerint, megfelelő ellenőrzési és elfogadási kritériumok alapján telepítjük. Például az ügyfél által is elérhető demórendszerre csak azután telepítjük az alkalmazást, ha már a belső QA rendszerre telepítve lett, és ott a tesztelés sikeresen lezajlott.

A tesztrendszerek és a hozzájuk tartozó feltételek az alkalmazás tesztelési stratégiájának kidolgozásakor kerülnek meghatározásra. Ezek megfelelő használata (legalábbis bonyolultabb rendszer esetén) komoly odafigyelést igénylő munka.

A VSTS új, Release Management rendszere ehhez nyújt segítséget. A release management használatával lehetőségünk nyílik arra, hogy definiáljuk a különböző tesztrendszereinket, virtuális gépeket rendeljünk az egyes szintekhez, szabályokat adjunk meg arra vonatkozóan, hogy az egyes rendszerekre mikor és hogyan kell az alkalmazást telepíteni, és hogy a telepítéseket áttekintsük és vezéreljük.

A release management részletes leírása meghaladja e cikk kereteit, részletesebb tanulmányozásához a https://www.visualstudio.com/en-us/features/release-management-vs.aspx címen található információ.

Megoldások teljesítménytesztelésre

Olyan alkalmazások fejlesztésekor, amelyek publikusan elérhetők, vagy nagyszámú felhasználó használja őket, érdemes teljesítménytesztelést végezni. Bár sokféle módja lehet a teljesítmény tesztelésének, leggyakrabban ez az alkalmazás nagyszámú felhasználóinak szimulálásával történik.

A teljesítmény tesztelésnek több, különböző célja lehet (pl. maximális terhelhetőség, skálázhatóság, hibatűrés, stb.), amelyek más és más beállításokat és szimulált felhasználói karakterisztikát igényelnek. Ezen különböző teljesítményteszt-típusok részleteivel ebben a cikkben nem foglalkozom.

Mivel a nagyszámú felhasználó szimulálása, illetve a teljesítményteszt alatt álló alkalmazás is jelentős erőforrást igényel, az ilyen jellegű tesztek futtatásához rövid időre jelentős infrastruktúrát kell lefoglalni. Ez hagyományos környezetben meglehetősen nehéz. Egy korábbi projektemen például a teljesítményteszt idejére a fejlesztők gépeit is be kellett vonni a tesztelésbe, ami nyilvánvalóan nem kényelmes megoldás.

A felhőalapú infrastruktúra és azon belül is az Azure lehetőséget nyújt arra, hogy a teljesítménytesztek erőforrásigényét kielégítse. Ez történhet úgy is, hogy virtuális gépeket állítunk szolgálatba, amelyeken a felhasználókat szimuláló ágenseket és a tesztelendő alkalmazást futtatjuk.

A VSTS segítségével tudunk azonban teljesítménytesztet végezni anélkül, hogy a felhasználókat szimuláló ágensek telepítésével, futtatásával és skálázásával foglalkoznunk kellene. (Természetesen a tesztelendő alkalmazást magunknak kell beállítani, de ehhez is használhatjuk az Azure különböző szolgáltatásait, mint például a WebAppot vagy a virtuális gépet.)

A VSTS teljesítménytesztelési szolgáltatása, hasonlóan a már korábban említett szolgáltatásokhoz, használatalapú (virtuális felhasználó percek – VUM, alapján). Minden hónapban az első 20 000 VUM ingyenes.

A VSTS teljesítménytesztjeinek használatához egy speciális „Load Test” projektet kell létrehoznunk, amelyben a teljesítményteszt lépéseit és egyéb paramétereit megadhatjuk. A beállításokat lokális környezetben tesztelhetjük, majd a VSTS tesztelői környezetében lefuttathatjuk. A teszt lefutása után részletes jelentést kapunk az eredményekről, az esetleges hibákról és az egyéb monitorozott értékekről. A tesztek konfigurálásáról és futtatásáról a https://www.visualstudio.com/en-us/features/vso-cloud-load-testing-vs.aspx oldalon találunk részletesebb leírást.

Áttérés Azure alapú fejlesztésre

A korábbi fejezetekben megvizsgáltam jó néhány esetet, amelyben a fejlesztési folyamat hatékonyan támogatható Azure-alapú szolgáltatásokkal. Ebben a fejezetben egy pár ötletet szeretnék adni arra, hogyan érdemes a fejlesztés bizonyos aspektusainak felhőalapú szolgáltatásokra való helyezését elkezdeni.

Az Azure szolgáltatásainak összefoglalása

Az alábbiakban összefoglalom azokat a legfontosabb Azure-alapú szolgáltatásokat, amelyek egy ilyen áttéréskor felmerülhetnek.

Szolgáltatás	Mire használható	Árazás
Virtual Machine	• Fejlesztői környezet • Tesztrendszer • Installálási/kompatibilitási teszt • Visual Studio kipróbálás • Teljesítményteszt-futtatás	Percalapú, a több erőforrással rendelkező gépek percdíja magasabb
WebApp (App Service)	• Tesztrendszer	Percalapú, a több erőforrással rendelkező konfigurációk percdíja magasabb, ingyenes konfigurációk elérhetőek
SQL Server	• Adatbázis teszteléshez	Percalapú, a több adatot kezelő konfigurációk percdíja magasabb

VSTS	• Projekttámogatás • Verziókezelés • Build • Release menedzselés	Felhasználók száma alapján, az első 5 felhasználó és az MSDN előfizetéssel rendelkező felhasználók ingyenesen használhatják
VSTS Hosted Build Agent	• Build	Az ágensek száma alapján, havonta 240 perc ingyenes,
VSTS Load Test	• Teljesítménytesztelés	Virtuális felhasználó percek (VUM) alapján, minden hónap első 20 000 VUM-je ingyenes

Tervezés

Amennyiben nem csak ad-hoc igényről van szó, hanem bizonyos folyamatokat és rendszereket egy az egyben az Azure-ra kívánunk áthelyezni, érdemes az új infrastruktúrát megtervezni. A tervezésnél az alábbi szempontokat veendők figyelembe:

- Hogy fog az Azure infrastruktúra kapcsolódni a cég belső hálózatához?
- Melyik Azure adatcentrumban érdemes az infrastruktúrát üzemeltetni?
- Várhatóan milyen ritmus szerint és mennyi erőforrásra lesz szükségünk?
- Az üzemeltetés mely részét érdemes automatizálni (pl. virtuális gép létrehozása megadott sablon szerint)?
- Rendelkeznek-e a fejlesztők MSDN-előfizetéssel?

Bármilyen alapos tervezés is történik, a megvalósítandó tervet érdemes egy kisebb környezetben kipróbálni. Ez lehet egy kisebb projekt vagy egy projekt kisebb funkciójának fejlesztése.

Rutin

Sok fejlesztő rendelkezik MSDN-előfizetéssel, így a munkájához ingyen használhatna egy sor Azure funkciót, mégsem teszi. Ennek oka lehet, hogy a fejlesztők idegenkednek az Azure használatától, mert azt gondolják, hogy bonyolult, és rendszergazda-ismereteket igényel a beállítása.

Ezeket a gátakat úgy tudjuk legjobban lebontani, ha megmutatjuk a fejlesztőknek, hogy sok – elsőre bonyolultnak tűnő – feladatot néhány lépésben, egyszerűen meg tudunk oldani. Érdemes tehát a cégen belül tapasztalatmegosztó workshopokat tartani, ahol azok a fejlesztők, akik már használták az Azure funkcióit, megmutatják, hogyan lehet a legfontosabb feladatokat elvégezni. Érdemes a gyakorlatokat mindenkinek magának is kipróbálni, hiszen az a tudás rögzül legkönnyebben, aminek megszerzésében magunk is aktívan résztvettünk.

MIT CSINÁL A RENDSZERGAZDA?

A fejlesztés bizonyos részeinek Azure-ra való költöztetésével felmerülhet a kérdés, hogy mit is fog csinálni a rendszergazda ezután, hogy a feladatok egy részének menedzselését az Azure maga látja el.

Pánikra semmi ok. A kihasználatlan rendszergazda igen ritka. Bár az Azure kétségtelenül átvesz valamennyi feladatot, amik korábban a rendszeradminisztrátorok felelősségi körébe tartoztak, de ezzel együtt új lehetőségeket is teremt, amelyeken keresztül a rendszergazdák „felszabaduló” ideje hasznossá válhat. Például az Azure bizonyos, a cégen belül gyakrabban használt funkcióinak automatizálása és ezen scriptek karbantartása a rendszergazdák feladata lehet.

JOGI ÉS BIZALMI PROBLÉMÁK

(A cikk szerzője nem jogász, így a cikk tartalma semmiképpen sem tekintendő hivatalos állásfoglalásnak.)

Sokan tartanak attól, hogy a fejlesztés felhőalapú támogatásával jogi problémákat sértenek. A legtöbb jogi korlátozás azonban az ügyfelek adatainak kezelésére vonatkozik. Amikor a fejlesztési folyamatokat támogatjuk Azure szolgáltatásokkal, csak fejlesztési és tesztadatokkal dolgozunk, tehát a személyes adatok tárolásának helyére vonatkozó korlátozások nem érvényesek. Ez alól egyedüli kivétel, ha a teszteléshez használt adatbázis az éles adatbázis egy másolata. Ilyenkor bizonyos személyes adatok kikerülhetnének távolabbi szerverekre. Az Azure alkalmazásától függetlenül is az ilyen helyzeteket kerülni kell. Az éles adatok tesztadatként történő használata számos problémát vethet fel, amelyek akár komoly következményekkel is járhatnak (pl. valamelyik fejlesztő elveszti a laptopját, rajta az adatokkal).

Néhányan attól tartanak, hogy a fejlesztés felhőbe kerülésével az alkalmazás forráskódja illetéktelen kezekbe kerülhet. A veszély elemzéséhez azt kell megvizsgálni, hogy milyen védelmi lehetőségek állnak rendelkezésre az Azure-ban, és milyenek a céges infrastruktúrán. Általában véve elmondható, hogy az Azure szigorú azonosítási és egyéb biztonsági lehetőségei nem gyengítik a legtöbb cég biztonsági követelményeit, sőt, sokszor a céges infrastruktúra nem megfelelő kezelése (pl. egy biztonsági frissítés figyelmen kívül hagyása) épp a céges hálózatot teszi sebezhetőbbé. Természetesen konkrét biztonsági kérdésben szakembereket bevonva lehet pontos elemzést készíteni.

A forráskód védelme természetesen fontos szempont, de nem érdemes túlértékelni. Amennyiben a védelem olyan szigorú gátakat okoz, amelyek az innovációt visszafogják, versenyhátrányba kerülhetünk. Érdemes tanulmányozni azokat az interpretált programnyelveket is, ahol a program telepítése egyben a forráskód átadását is jelenti, de ez mégsem jelenti azt, hogy ezek a cégek nagyobb károkat szenvednének el emiatt, mint azok, amelyek fordított nyelvekkel dolgoznak.

ÖSSZEFOGLALÁS

A cikk a felhőalapú szolgáltatások egy speciális felhasználási módját vizsgálta meg, amely során ezeket a fejlesztés támogatására használjuk fel. Ennek okai között részleteztem az infrastruktúra könnyebb menedzselhetőségét, a dinamikusan változó erőforrásigények hatékony kielégítését és az innováció elősegítését az által, hogy az ötlet kipróbálásához szükséges infrastruktúra az Azure segítségével rövid idő alatt üzembe állítható.

Részletesebben vizsgáltam néhány tipikus esetet, amikor a fejlesztéshez a felhőalapú szolgáltatások és az Azure segítséget nyújthatnak. Részletesen tárgyaltam a Visual Studio Team Services által nyújtott

lehetőségeket (projekttámogatás, build, teljesítménytesztelés), illetve az Azure virtuális gépek és a WebAppok által támogatott szituációkat (fejlesztői gépek, tesztrendszerek, build ágensek, teljesítménytesztelés).

Végül néhány ötletet említettem arra vonatkozóan, hogyan lehet a felhőalapú szolgáltatások használatához hozzákezdeni, mik a tervezési, tanulási szükségletei az áttérésnek, illetve milyen kihívásokkal kell számolnunk.

Többször említettem a cikkben az innováció fontosságát. Természetesen ez az Azure használatára is igaz. Bár jó pár ötlet és példa felmerült, amikor az Azure a segítségünkre lehet, bizonyára sok olyan lehetőség van még, amire nem is gondoltunk. Ezek felfedezéséhez sok sikert kívánok!