TECNICO SUPERIORE PER LE APPLICAZIONI DISTRIBUITE

Figura nazionale di riferimento Tecnico superiore per i metodi e le tecnologie dello sviluppo di sistemi software

CORSO OR1118188001

Anno Formativo: 2012/2013

2000 ore

TESI

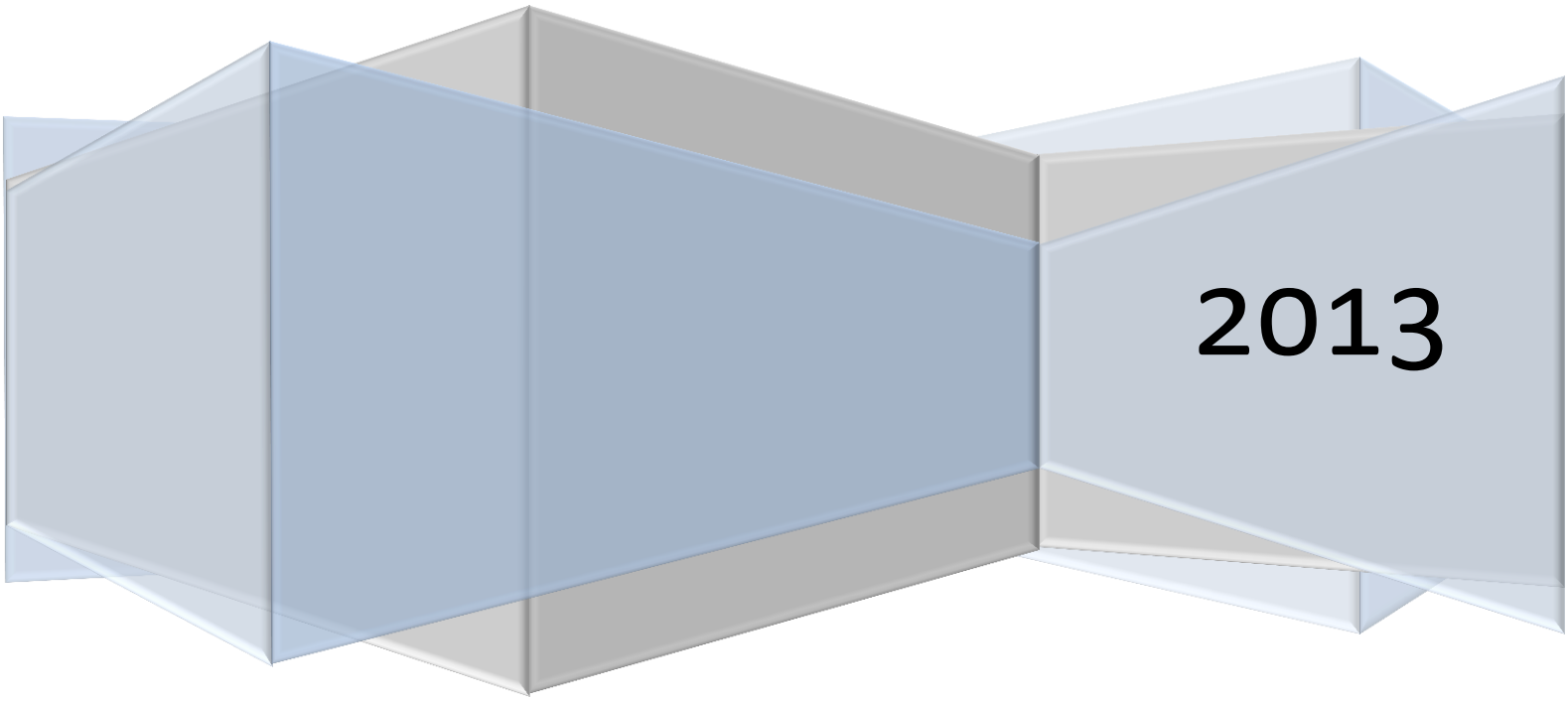
Titolo: "Cloud Computing e Windows Azure"

Monaco Nicola

Cloud Computing con Windows Azure

… dove stiamo andando

Nicola Monaco



Questo testo è dedicato a tutti i docenti del corso ITS “Tecnico Superiore per le Applicazioni Distribuite”

Grazie davvero…
Nicola

Sommario

Fonti:

1. Windows Azure Step by Step – Developing cluod-based applications with Microsoft Visual Studio 2010 (by Roberto Brunetti; publisher: Microsoft Press; released: May 2011)
2. Windows Azure Programmare per il Cloud Computing (di Fabio Cozzolino)
3. Portale di Windows Azure (www.windowsazure.com)
4. Developing Cloud Applications with Windows Azure Storage (by Paul Mehner; publisher: Microsoft Press)

Roberto Brunetti è un consulente, formatore e scrittore. È cofondatore di DevLeap, un'azienda concentrata sulla fornitura di contenuti ad alto valore e di servizi di consulenza a sviluppatori professionisti. È anche fondatore di ThinkMobile, la più grande comunità italiana sullo sviluppo mobile. Roberto è presente come speaker alle principali conferenze e lavora a stretto contatto con Microsoft Italia all'organizzazione di eventi e alla realizzazione di corsi di formazione.
Il suo libro: http://shop.oreilly.com/product/0790145309099.do?mybuyscid=12674760916&cmp=af-mybuy-0790145309099.IP
Il suo blog è raggiungibile all'indirizzo: http://blogs.devleap.com/rob/default.aspx

Fabio Cozzolino è Software Architet, Analyst e Developer in CompuGroup Medical Italia, società presso la quale svolge attività di sviluppo di piattaforma e di soluzioni di e-health. Partecipa costantemente come speaker a diversi eventi e meeting parlando di Windows Communication Foundation e di Windows Azure. È particolarmente attivo nelle community ed è presidente dello user group DotNetSide. Dal 2010 è MVP nella categoria Connected System Developer.
Il suo blog è raggiungibile all'indirizzo: http://dotnetside.org/blogs/fabio .
Potete inoltre seguire i post di Fabio su Twitter: http://twitter.com/fabiocozzolino .

Paul Mehner è stato un sviluppatore software, architetto, project manager, consulente, speaker, mentore, istruttore e imprenditore per più di trent'anni. Egli è cofondatore di South Sound .NET User Group, uno dei più vecchi user group al mondo ed è stato tra i primi membri del comitato dell'International .NET Association (INETA). Ora lavora anche per Wintellect come consulente senior e formatore.
Attualmente Paul è specializzato in cloud computing su piattaforma Windows Azure, architetture orientate ai servizi, Security Token[1] Servers, Windows Communication Foundation, Windows Identity Foundation e Windows Workflow Foundation. Prima di rinascere come protagonista in .NET nel 2000, le esperienze di Paul includono più di 20 anni di UNIX. Paul comincia la sua precoce carriera informatica nel 1977 su una basetta sperimentale (breadboard, utilizzata per creare prototipi di circuiti elettrici) con 256 byte di RAM, 12 interruttori basculanti (on-off), 9 diodi led ed un microprocessore RCA CDP1802.

[1] In generale un **token** è un oggetto che rappresenta qualcos'altro.
1. Nella programmazione, il codice sorgente si può suddividere in 5 classi di tokens (costanti, identificatori, operatori, parole riservate e separatori), in accordo con il linguaggio di programmazione.
2. Un token di sicurezza è un dispositivo fisico che, insieme a qualcosa in possesso dell'utente (come un codice), consente l'utilizzo del servizio: ad esempio l'uso del servizio di remote banking con l'accesso tramite token (http://it.wikipedia.org/wiki/Token_(sicurezza)).

Introduzione al Cloud Computing

Sull'homepage di Windows Azure (http://www.windowsazure.com/en-us/) è possibile leggere:

Windows Azure is an open and flexible cloud platform that enables you to quickly build, deploy and manage application across a global network of Microsoft-managed datacenters (*Windows Azure è una piattaforma cloud che consente di creare rapidamente, distribuire e gestire applicazioni attraverso una rete globale di data center gestiti da Microsoft*).

Ed ancora:
è possibile creare applicazioni utilizzando qualsiasi linguaggio, strumento o framework. È inoltre possibile integrare le applicazioni cloud pubbliche con l'ambiente IT esistente.

- Nota: dal momento che URL, nomi e procedure possono cambiare nel tempo, alcuni di questi potrebbero essere non più aggiornati nel momento della lettura di questo testo.

Servizi di infrastruttura

Windows Azure fornisce un'infrastruttura su richiesta che supporta la scalabilità[2] ed è in grado di adattarsi alle esigenze di business in continua evoluzione: che si tratti di creare nuove applicazioni o eseguire applicazioni esistenti, Windows Azure fornisce un valido rapporto prezzo-prestazioni della categoria ed un supporto end-to-end[3].

Sviluppo e test

Windows Azure consente di sviluppare e testare le applicazioni in minor tempo rispetto agli approcci tradizionali su PC, a costi ridotti e con la flessibilità necessaria per la distribuzione nel cloud o in locale.

Big Data

HDInsight e Windows Azure, una soluzione Big Data basata su Apache Hadoop , offre nuovi approfondimenti e scenari, inoltre consente di prendere opportune decisioni da come trattare i dati a quali

[2] **Scalabilità**: questa parola sarà molto usata in questo elaborato.
In generale: caratteristica di un dispositivo harware o software che consente la sua estensione con ulteriori capacità e funzionalità nel caso di necessità future. Un sistema si dice scalabile, quando è possibile aggiungere ulteriori funzionalità senza doverne modificare le caratteristiche fondamentali.
In particolare:

1) Possibilità di eseguire lo stesso software su computer di potenza diversa. La scalabilità di un software è una caratteristica di grande valore, in quanto consente alle aziende che lo acquistano di utilizzarlo inizialmente su macchine poco performanti e successivamente su macchine evolute.
2) La scalabilità è un fattore critico per le applicazioni distribuite e indica la capacità di adattarsi all'aumento degli utenti, all'incremento dei dati e alla diversificazione delle funzionalità richieste.

[3] **End to end**: da estremità a estremità. Caratteristica del protocollo quale il TCP (Transport Layer), per la rilevazione degli errori e del controllo di flusso. Si distingue dalla rilevazione effettuata nella modalità hop-by-hop (tratta per tratta) nel livello collegamento dati (Data Link) o nel livello rete (Network Layer). Il controllo TCP viene effettuato solo dalle due entità TCP poste negli host ai capi della connessione.

scelte di programmazione implementare. Gli approfondimenti accennati precedentemente provengono da tutti i tipi di dati e vengono forniti agli utenti aziendali tramite Microsoft Excel.

App mobili

Windows Azure rende semplice e veloce la compilazione di app mobili che supportano la scalabilità. È possibile archiviare dati nel cloud, autenticare utenti e inviare notifiche push (notifica di un evento in tempo reale) a milioni di dispositivi in pochi minuti.

Applicazioni Web

Windows Azure offre opzioni sicure e flessibili di sviluppo, distribuzione e scalabilità per applicazioni Web di qualsiasi dimensione. È possibile utilizzare gli strumenti esistenti per creare e distribuire applicazioni, senza il problema di gestire l'infrastruttura.

Contenuti multimediali

Con Servizi Multimediali di Windows Azure è possibile creare soluzioni di distribuzione multimediale scalabili, convenienti e complete in grado di trasmettere i contenuti multimediali ad Adobe Flash, Android, iOS, Windows ed altri dispositivi e piattaforme.

Archiviazione, backup e ripristino

Windows Azure fornisce soluzioni cloud di archiviazione, backup e ripristino scalabili e durevoli per tutti i dati. Funziona con l'infrastruttura esistente per migliorare in modo conveniente la strategia di continuità aziendale e fornisce le funzionalità di archiviazione richieste dalle applicazioni cloud.

Gestione delle identità e dell'accesso

Active Directory di Windows Azure fornisce un servizio di gestione delle identità del cloud di livello enterprise, per un'esperienza Single Sign-On[4] su applicazioni cloud e locali. Consente l'autenticazione a più fattori per una maggiore sicurezza e conformità.

[4] Il **Single Sign-On** è una speciale forma di autenticazione che permette ad un utente di autenticarsi una volta sola e ottenere accesso a diverse risorse.

Servizi di infrastruttura

Vantaggi

Scalabilità su richiesta, si paga solo per le risorse utilizzate

Con Windows Azure è possibile configurare nuove macchine virtuali Windows Server e Linux in pochi minuti e adattarne l'utilizzo in base alle proprie esigenze. Grazie all'approccio di pagamento a consumo, il cliente paga solo per risorse utilizzate e non è prevista alcuna penale per la modifica delle configurazioni delle macchine virtuali.

La miglior piattaforma per sviluppo e test

Se si dispone di un abbonamento MSDN, è possibile accedere ad una raccolta di immagini in cui sono disponibili più versioni del sistema operativo Windows Server e ad altro software lato server, quale SQL Server , SharePoint Server e BizTalk Server, che consente di sviluppare e testare rapidamente componenti a tariffe scontate. Sarà inoltre possibile utilizzare le proprie licenze software server MSDN per creare macchine virtuali personalizzate in base alle esigenze specifiche.

Estensione dell'ambiente

Con macchine virtuali di Windows Azure è possibile distribuire in pochi minuti le proprie immagini Windows Server o Linux personalizzate in un ambiente di produzione di più zone e coperto da contratto di servizio o iniziare a utilizzare un'immagine preconfigurata disponibile nella raccolta Microsoft. Grazie a Windows Azure e alla tecnologia di rete virtuale, l'ambiente cloud diventa una semplice estensione del data center sfruttando appieno Microsoft System Center, Active Directory e Visual Studio.

Scenari

Capacità raggiunta in pochi minuti e adattabile in base alle esigenze

È possibile utilizzare le proprie immagini o compilare in base a quelle preconfigurate fornite da Microsoft per creare rapidamente distribuzioni valide. Microsoft offre le prestazioni migliori in termini di prezzo sia per gli ambienti di produzione che per gli scenari di sviluppo e test, semplificando all'utente lo spostamento tra ambienti locali e cloud al fine di soddisfare le esigenze.

[Andreas Hogberg, IT Director TELENOR.
(http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?CaseStudyID=710000002349)]

Infrastruttura robusta e completamente supportata per SharePoint

È possibile configurare rapidamente farm di SharePoint Server senza investimenti iniziali. È possibile inoltre integrare codice con attendibilità totale per eseguire app avanzate e personalizzate, compilare siti con connessione Internet, nonché applicare la scalabilità su richiesta, eseguire test sulla versione che si prevede distribuire e pagare solo le risorse utilizzate.

[Scalabilità del sito TOYOTA, gazoo.com, con SharePoint 2013 e Windows Azure]

Infrastruttura robusta e completamente supportata per SQL Server

È possibile sviluppare, testare ed eseguire applicazioni con funzionalità SQL Server complete o eseguire la migrazione di applicazioni locali in ambiente cloud per ottenere vantaggi in termini di costi e prestazioni. È possibile sbloccare nuove strategie ibride di continuità aziendale, ad esempio distribuendo database primari SQL Server AlwaysOn in locale e database secondari basati su cloud in macchine virtuali.

[Adam Salvo, DevOps Manager, TREK. (http://www.microsoft.com/casestudies/Windows-Azure/Trek-Bicycle-Corporation/Bicycle-Firm-Moves-Retail-System-to-Cloud-Expects-to-Save-15-000-a-Month-in-IT-Costs/710000002640)]

Unica identità per Single Sign-On

È possibile compilare servizi ibridi e applicazioni cloud da distribuire agli utenti con modalità Single Sign-On, nonché associare le identità esistenti ad app in esecuzione in Macchine virtuali con i servizi Active Directory locali. Se si esegue Office 365, è sufficiente eseguire Active Directory Federation Services in Macchine Virtuali per effettuare la sincronizzazione con le identità locali per l'accesso Single Sign-On.

[Tim Fernandes, CaptiveLogix president: CaptiveLogix è un'organizzazione sanitaria utilizza Active Directory di Windows Azure per gestire le proprie informazioni. http://www.azurevideos.com/faces/captivelogix-2/]

Sviluppo e test

Vantaggi

Sviluppo e test delle applicazioni più rapidi

È possibile eseguire in modo autonomo il provisioning[5] del numero di macchine virtuali richiesto per lo sviluppo ed il test di applicazioni nel cloud senza che sia necessario attendere l'esecuzione di processi hardware, interni o appaltati. Rete virtuale, di Windows Azure, consente di connettersi globalmente alla propria rete locale e di applicare, con sicurezza, scalabilità orizzontale e verticale, nonché di generare carico per distribuire applicazioni in modo più rapido.

Riduzione dei costi con l'accesso di Windows Azure tramite MSDN

Se si attiva un abbonamento MSDN, è possibile sfruttare un credito di Euro 115,00 per Windows Azure con tariffe ridotte per Windows Server e utilizzare il proprio software MSDN, ad esempio SQL Server, senza alcun costo aggiuntivo.

Distribuzione in locale e nel cloud

Windows Azure consente di sviluppare e testare le applicazioni in modo più rapido e a costi ridotti, inoltre di scegliere successivamente le modalità di distribuzione. È possibile entrare nella fase di produzione con Windows Azure oppure esportare la Macchina virtuale e attivare l'applicazione in locale o con un Provider di hosting, in base alle esigenze.

Altri vantaggi

Gli sviluppatori che compilano applicazioni in locale o che ne seguono la migrazione nel cloud, beneficeranno immediatamente dei vantaggi descritti in precedenza tramite Macchine virtuali di Windows Azure. In Windows Azure sono inoltre disponibili numerosi servizi che possono essere utilizzati per sviluppare e testare un'ampia gamma di applicazioni. Alcuni di questi servizi includono Web Sites di Windows Azure, servizi Mobile, servizi Cloud, databases SQL e il servizio di Storage di Windows Azure.

Scenari

Creazione di un ambiente sandbox di sviluppo e test in pochi minuti

È possibile avviare Macchine virtuali da un unico core, fino a otto core, da meno di 1 Gb a 56 Gb in meno di 10 minuti e iniziare immediatamente la fase di sviluppo o test. Al nuovo ambiente di sviluppo e test è possibile accedere da qualsiasi computer, in ufficio, nella propria abitazione o in qualsiasi punto connesso a Internet nel mondo.

[Steve Novoselac, IT Manager TREK. (http://www.azurevideos.com/trek-standard/)]

Sviluppo di Windows Server o di applicazioni Linux in Machine virtuali

In Windows Azure è possibile sviluppare e testare un'unica macchina virtuale o una rete complessa di macchine virtuali connesse. Gli utenti MSDN possono eseguire software MSDN, ad esempio Visual Studio, SQL Server, SharePoint e Biz Talk Server, basandosi su Windows Server. È possibile inoltre creare una

[5] **Provisioning**: rifornimento, fornitura.

Macchina virtuale dalla raccolta, aggiungere tecnologia di integrazione continua, coem ad esempio Team Foundation Service o Git e iniziare immediatamente a creare codice.

[Timothy Khouri, Lead Architet JACKSON HEWITT. (http://www.azurevideos.com/jackson-hewitt/)]

Creazione di script per una distribuzione ancora più veloce

Gli script consentono di creare ambienti di sviluppo e test, reiterabili. La creazione di più configurazioni virtuali, ad esempio macchine e reti virtuali, può essere completamente automatizzata tramite script PowerShell che creano e collegano automaticamente tutte le risorse necessarie. Le macchine virtuali possono inoltre essere avviate e arrestate per ottimizzare i costi.

[Marius Pedersen, Associate Architect in Microsoft Services for TELENOR and other customers. (http://www.azurevideos.com/faces/telenor/)]

Test di carico e scalabilità

Un ambiente cloud consente di testare le applicazioni sotto carico senza influire sulle applicazioni di produzione, nonché di testare configurazioni in termini di scalabilità orizzontale e verticale per garantire che l'applicazione sia pronta per essere utilizzata dai clienti.

[Steve Whitby, Solution Delivery Center Director AVIVA. (http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?CaseStudyID=710000002635)]

Accelerazione dello sviluppo di SharePoint o SQL

È possibile creare un ambiente di test SharePoint senza la necessità di investire notevoli capitali, semplificando lo sviluppo successivo di applicazioni completamente attendibili o di siti con connessione a Internet avanzati, nonché configurare rapidamente un ambiente di test SQL senza alcuna attesa. Grazie a Windows Azure è possibile inoltre applicare la scalabilità in base alle proprie esigenze di sviluppo. Se si desidera effettuare l'aggiornamento alla versione migliore e più recente di SharePoint o SQL, è possibile eseguirne il test nel cloud senza alcun impatto sugli ambienti di produzione e implementarla in locale o nel cloud una volta pronta.

[(http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?CaseStudyID=710000002349)]

Connessione sicura praticamente ovunque

Con una rete virtuale da computer a sito, è possibile connettersi in modo sicuro da un computer protetto dal firewall aziendale o da un punto Internet pubblico direttamente alla rete che segue Windows Azure, per testare le proprie applicazioni senza che il personale IT debba apportare alcuna modifica.

Big Data

Vantaggi

Aperto e flessibile

Completamente basato su Apache Hadoop, HDInsight consente alle soluzioni Big Data di essere eseguite in Windows Azure oppure in ambiente locale in Windows Server o Linux. Questa flessibilità, consente di sfruttare i progetti Apache Hadoop esistenti, ad esempio Apache Pig, Hive, Sqoop e molti altri.

Hadoop senza problemi

È possibile creare rapidamente un cluster Hadoop in pochi minuti quando è necessarioe ridurne le dimensioni dopo avere eseguito i processi MapReduce, scegliendo quelle più adeguate per ottimizzare il tempo di analisi o i costi. Windows Azure PowerShell e l'interfaccia della riga di comando di Windows Azure consentono inoltre di integrare con semplicità HDInsight nei flussi di lavoro di analisi esistenti.

Approfondimento dei dati tramite Excel

Grazie all'analisi dei dati non strutturati da HDInsight in Microsoft Excel, gli utenti possono ottenere informazioni approfondite e unire i risultati restituiti da HDInsight con dati di origini interne, esterne, relazionali e non relazionali. L'utilizzo di tecnologie avanzate di Microsoft Excel , ad esempio PowerPivot e PowerView, consente inoltre di eseguire in tempo reale analisi dinamiche sui set di dati combinati.

Compilazione di app Big Data personalizzate

HDInsight offre agli sviluppatori la possibilità di programmare nel linguaggio scelto, ad esempio Java, .NET e altri linguaggi. Gli sviluppatori .NET possono utilizzare la potenza delle query integrate nel linguaggio grazie a LINQ to Hive, mentre gli sviluppatori di database possono sfruttare le competenze SQL esistenti per eseguire query e trasformare i dati tramite Hive.

Scenari

Elaborazione telemetrica avanzata

È possibile ottenere informazioni operative approfondite e comprendere meglio il comportamento dei clienti grazie all'elaborazione avanzata dei registri. I registri possono essere archiviati tramite il servizio di BLOB storage, mentre HDInsight consente di eseguirvi analisi che spaziano dall'aggregazione di base a tecniche sofisticate di apprendimento automatico e di data mining[6]. I risultati dell'analisi su logs (registri) e set di dati possono essere combinati e uniti ulteriormente con Data Explorer[7]e possono essere visualizzati con PowerPivot, PowerView e GeoFlow[8].

[6] Il **data mining** è l'insieme delle tecniche e metodologie che hanno per oggetto l'estrazione di una informazione a partire da grandi quantità di dati (attraverso metodi automatici o semi-automatici) e l'utilizzo scientifico, industriale o operativo di questa informazione. Ad esempio la Statistica, cioè il campo matematico dell'analisi dei dati, può essere definita come un modello di data mining.

[7] http://office.microsoft.com/en-us/excel/download-data-explorer-for-excel-FX104018616.aspx

[8] http://office.microsoft.com/en-us/download-geoflow-for-excel-FX104036784.aspx

[Halo 4 offre nuove possibilità di approfondimento dei Big Data nel *cloud*. (http://www.microsoft.com/casestudies/Windows-Azure/343-Industries/343-Industries-Gets-New-User-Insights-from-Big-Data-in-the-Cloud/710000002102)]

Preparazione per approfondimenti futuri

Grazie a Windows Azure è oggi possibile salvare a costo ridotto i dati nel relativo formato non elaborato senza alcun schema predefinito o senza alcuna pre-elaborazione. Il BLOB Storage rappresenta uno strumento flessibile per tutti i tipi di dati quali logs (registri), dati telemetrici e flussi di siti di social network. In futuro sarà possibile esaminare i dati storici utilizzando HDInsight.

[Val Fontana, Server and Tools Business Microsoft DATA LAKE. (http://www.azurevideos.com/faces/data-lake/)]

Sblocco del valore nascosto dei dati

Con HDInsight, gli algoritmi intelligenti di apprendimento automatico, disponibili nella libreria Mahout, consentono di scoprire schemi nascosti nei dati. È possibile applicare cluster ai dati per trovare segmenti naturali o per creare modelli predittivi per vedere tendenze nei dati.

[Ascribe Ltd. (http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?CaseStudyID=710000002092)]

Mobile apps (App-licazioni mobili)

Vantaggi

Aggiunta di un servizio back end[9] cloud in pochi minuti

Windows Azure consente di potenziare con flessibilità le proprie Mobile apps grazie a Macchine virtuali, Servizi cloud e Servizi mobili. Con il supporto di linguaggi noti come .NET e NodeJS, l'integrazione continua incorporata e la possibilità di accedere a qualsiasi API, il team di sviluppo può essere operativo in pochi minuti.

Utenti raggiungibili in qualsiasi piattaforma: Android, iOS e Windows

Con gli SDK nativi per Android, Apple iOS, Windows Phone e Windows Store, il componente Mobile Services di Windows Azure consente di raggiungere ogni utente in qualsiasi piattaforma. È possibile inoltre ottenere la massima flessibilità con l'API REST aperta, nonché indirizzare più app client allo stesso componente back end di servizi mobili per un'esperienza coerente in ogni dispositivo.

Notifiche push a milioni di dispositivi

Gli Hub di notifica consentono ad ogni app di inviare notifiche push a milioni di dispositivi Android, Apple iOS, Windows Phone e Windows Store con poche righe di codice. Gli Hub di notifica permettono di effettuare un servizio multicast[10] di routing pub/sub[11] basato su tag[12], che semplifica l'invio di push a un sottoinsieme particolare di utenti, ad esempio quelli che seguono una squadra specifica.

Creazione di app per i social network

Mobile Services consente di configurare in modo semplice l'autenticazione utente tramite gli account di Facebook, Google, Microsoft o Twitter. Dopo l'autenticazione degli utenti è possibile non solo personalizzare l'esperienza di questi ultimi ma anche aumentare il coinvolgimento e le possibilità di condivisione.

Connessione di sistemi locali

Mobile Services consente di sfruttare le risorse esistenti, sia in ambiente locale che nel cloud. È possibile utilizzare qualsiasi servizio di Windows Azure, ad esempio Service Bus, servizi multimediali (Media Services), Active Directory di Windows Azure ed il servizio di archiviazione di Windows Azure (Windows Azure Storage), oppure scegliere dai numerosi servizi di terze parti disponibili in Windows Azure Store. È possibile

[9] I termini **front end** e **back end** denotano, rispettivamente, lo stadio iniziale e lo stadio finale di un processo. Nel campo della progettazione software il front end è la parte di un sistema software che gestisce l'interazione con l'utente o con sistemi esterni che producono dati in ingresso; il back end è la parte che elabora i dati generati dal front end.

[10] Con il termine **multicast**, nelle reti di calcolatori, si indica la distribuzione simultanea di una informazione, verso un gruppo di destinatari.

[11] In informatica, l'espressione publish-er/subscribe-r, abbreviata in **pub/sub**, si riferisce ad un design pattern (o stile architetturale) utilizzato per la comunicazione asincrona fra diversi processi, oggetti o agenti. In questo schema, mittenti e destinatari di messaggi dialogano attraverso un tramite, chiamato dispatcher o broker.

[12] Un **tag** (o metadato) è una parola chiave associata ad un'informazione(un'immagine, una mappa geografica, un post, un video clip, ecc.), che descrive l'oggetto rendendo possibile la classificazione.

inoltre connettersi a database locali, a sistemi Active Directory e a Windows Intune[13] o, addirittura, a sistemi Oracle e SAP tramite adapter BizTalk[14].

Scalabilità su richiesta con disponibilità e portata globali

Con Windows Azure è possibile scalare le proprie app su richiesta e in base alla crescita del traffico senza che sia necessario eseguire il provisioning di alcuna infrastruttura aggiuntiva. Windows Azure offre un contratto di servizio a elevata disponibilità (SLA) per app compilate con Macchine virtuali, servizi cloud e Mobile Services e consente di accedere alla rete globale di Data Centers gestiti da Microsoft, permettendo di raggiungere utenti ovunque nel mondo.

Scenari

App per il marchio

Attualmente i titolari dei marchi più esclusivi creano le proprie app per raggiungere i clienti, non solo per pubblicare annunci pubblicitari. Le Mobile apps rappresentano un'opportunità unica per entrare a fare parte della vita quotidiana di ogni cliente e delle attività che più apprezza. Windows Azure semplifica tale possibilità grazie al supporto per le principali piattaforme, a un'ampia gamma di opzioni per l'archiviazione dei dati, a una soluzione innovativa di autenticazione utente e a notifiche push semplici come il caricamento di credenziali e la digitazione push.apns.send (http://msdn.microsoft.com/en-us/library/windowsazure/jj839711.aspx - http://www.windows-azure.net/success-callback-for-azure-mobile-services-push-apns-send/).

[Johan Brockfield, Strategy Consultant and Christer V. Aas Senior System Consultant MAKING WAVES. (http://www.azurevideos.com/making-waves/tech-making-waves/)]

App consumer

I clienti di tutto il mondo desiderano utilizzare giochi, riprodurre contenuto multimediale in streaming, gestire informazioni sugli account, pagare fatture e fare acquisti, tutto da un dispositivo mobile. Mentre il Web rimane un motore importante per la distribuzione di questi servizi consumer di base, le soluzioni mobili sono un fattore critico. Attualmente le principali aziende, dai giganti del settore dei media alle società di telecomunicazioni, fino alle società di taxi, nonché start-up e grandi organizzazioni, utilizzano Windows Azure per coinvolgere in modo significativo i propri clienti, fornendo prodotti su richiesta, ovunque i clienti si trovino.

[TechSmith – App iOS Coach's Eye per l'analisi istantanea di video. (http://www.techsmith.com/coachs-eye.html)]

App aziendali

È possibile utilizzare Windows Azure per compilare applicazioni aziendali o estendere applicazioni Web interne e favorire la produttività della propria forza lavoro, anche in ambiente Mobile. Utilizzando un

[13] Il servizio cloud **Windows Intune** consente di gestire e proteggere i computer in modo centralizzato tramite un'unica console basata su Web che consente ai computer, agli utenti e al personale IT di lavorare al massimo delle prestazioni senza compromettere il controllo, la sicurezza e le conformità (http://technet.microsoft.com/it-it/windows/intune.aspx)

[14] **Microsoft BizTalk Server**, spesso definito semplicemente come BizTalk, è un Business Process Management (BPM) e un middleware. Attraverso l'uso di "adattatori", che sono adeguati per comunicare con i diversi sistemi software utilizzati in una grande impresa, esso permette alle aziende di automatizzare e integrare i processi aziendali.

Service Bus Relay[15] con la propria app ospitata in Windows Azure, è possibile abilitare l'accesso mobile ai dati e alle app in locale, senza che sia necessario archiviare alcuna informazione sensibile al di fuori del firewall aziendale. Windows Azure consente inoltre di gestire identità e dispositivi con Active Directory di Windows Azure e Windows Intune.

[Gary Pretty, Deputy Head of Programming and Microsoft Technical Evangelist MANDO GROUP. (http://azurevideos.com/talk-talk/)]

[15] Il **Service Bus Relay** è un web service per la comunicazione sincrona e asincrona via rete. Si può anche affermare che è un meccanismo per sviluppo di applicazioni distribuite e ibride (ad esempio, esporre, in modo sicuro, un servizio ospitato su un cloud privato a client esterni).

Applicazioni Web

Vantaggi

Pronto per il lavoro

Windows Azure offre un'infrastruttura affidabile di livello Enterprise, per ospitare in modo sicuro, siti Web con disponibilità elevata per impostazione predefinita. Consente di proteggere i siti Web con il supporto sia di certificati SNI[16] che certificati SSL basati su IP, nonché di creare applicazioni aziendali affidabili, sicure e moderne, con accesso ai principali servizi di Windows Azure, come Active Directory, database SQL e Services Bus. Windows Azure mantiene l'infrastruttura sempre aggiornata, tramite l'applicazione automatica di patch sia verso il Server Web che al sistema operativo. Il sito Web può essere eseguito su una rete di data center globali, con un contratto di servizio garantito e opzioni di supporto 24 ore al giorno, 7 giorni a settimana.

Scalabilità World Wide Web

I siti Web di Windows Azure possono contare su una solida base che offre vantaggi ad alcune delle più grandi aziende al mondo, grazie alla sua elevata stabilità e al supporto di standard Internet leader nel settore. Progettato per gestire milioni di richieste al giorno, Windows Azure bilancia automaticamente il carico di tali richieste in base alle necessità, mantenendo l'infrastruttura sempre aggiornata. È possibile migliorare o ingrandire in pochi secondi, con facilità e senza modificare il codice, pagando solo per le risorse utilizzate.

Migliore esperienza con Visual Studio

Le potenzialità di ASP.NET e l'ecosistema avanzato di Visual Studio possono essere sfruttati appieno per spostare le applicazioni Web esistenti senza modificare il codice o apportando solo poche modifiche. Le informazioni sulla sottoscrizione ad Azure possono essere importate per garantire flessibilità di distribuzione e gestione. Da Visual Studio è possibile pubblicare, gestire e configurare rapidamente tutte le applicazioni Web. La connessione al progetto Team Foundation Service consente lo sviluppo in team e una distribuzione contollata.

Tempi di commercializzazione più rapidi

Il provisioning di un'applicazione Web di produzione, può essere eseguito da soli, in pochi minuti, senza dover aspettare l'approvvigionamento hardware o IT. È possibile distribuire con facilità i contenuti creati, utilizzando i propri strumenti di sviluppo preferiti o mantenere i siti sempre aggiornati con il supporto continuo all'integrazione. Le opzioni di distribuzione flessibile includono FTP, Git, GitHub, Bitbucket, CodePlex, TFS e DropBox.

Aperto e flessibile

È possibile creare applicazioni con un linguaggio a scelta (sono inclusi ASP:NET, PHP, Node.js, Python e Classic ASP) e connettere l'applicazione Web a una serie di database, tra cui database SQL, MySQL o soluzioni noSQL, a Windows Azure Store. In alternativa, si può iniziare con una raccolta di app che

[16] **Server Name Indication** (SNI) è una estensione del protocollo Transport Layer Security (TLS) che indica il nome dell'host verso il quale il client deve collegarsi all'inizio del processo di handshaking. Nei Server è usato per avere certificati multipli sullo stesso indirizzo IP e numero di porta.

includono applicazioni, framework e modelli open source comuni, come ad esempio WordPress, Umbraco, DotNetNuke, Drupal, Django, CakePHP e Express.

Scenari

Applicazioni di digital marketing

È possibile creare applicazioni Web per i clienti ovunque e su qualsiasi dispositivo, nonché progettare, sviluppare e offrire loro esperienze basate su rich media[17], campagne interattive e supporto per la connessione ai principali siti di social networking. Si può partire da zero con un moderno framework Web o da una raccolta di applicazioni Web open source e poi scalare su richiesta verso l'alto o il basso, per un'esperienza digitale ottimale.

[BMW Latin America. (http://www.microsoft.com/casestudies/Windows-Azure/BMW-Latin-America/BMW-Supports-Model-Launch-Develops-Prospects-with-Cloud-Based-Social-Marketing/710000001091)]

Presenza globale sul Web

Il business, il marchio e le comunicazioni con i clienti possono essere ottimizzate grazie all'uso di una piattaforma cloud sicura, scalabile e affidabile. È inoltre possibile spostare l'applicazione Web esistente o crearne una rapidamente da una raccolta di applicazioni Web, connettere l'applicazione Web ai servizi esistenti in Windows Azure o in locale e implementare e ridimensionare l'applicazione Web in qualsiasi area del mondo, su richiesta e pagando solo per le risorse utilizzate.

[Tamas Henning, Project Architect SKYPE]

Applicazioni aziendali

È possibile utilizzare tutte le potenzialità di .NET, di Visual Studio e dei servizi Windows Azure per creare applicazioni aziendali sicure, utilizzando Active Directory di Windows Azure per creare una federazione delle identità con l'ambiente locale oppure Office 365 e gestire l'accesso alle applicazioni da parte di dipendenti e partner, nonché connettersi in modo sicuro alle risorse locali, inclusi database, servizi Web e altre risorse.

[Johnny Halife. (Mural.ly http://channel9.msdn.com/Shows/Cloud+Cover/Episode-93-Real-World-Windows-Azure-with-Murally)]

[17] Il termine **rich media** e sinonimo di multimedia interattiva: prodotti o servizi su sistemi digitali, che rispondono alle azioni degli utilizzatori presentando contenuti come testo, grafica, animazioni, video, audio, giochi, ecc.

Contenuti multimediali

Vantaggi

Flusso di lavoro multimediale end-to-end integrato

L'integrazione di più tecnologie e prodotti risulta estremamente semplice con i Servizi Multimediali di Windows Azure, che consentono di creare flussi di lavoro end-to-end e di ridurre i costi associati all'integrazione di più prodotti e provider. È possibile scegliere i servizi Microsoft o dei suoi partner, per integrarli, pronti all'uso.

Espandere globalmente le attività aziendali

È possibile espandere le proprie attività aziendali e raggiungere destinatari in tutto il mondo, sfruttando le caratteristiche di globalità di Windows Azure e dei servizi multimediali, senza che sia richiesto di investire in infrastrutture. È possibile configurare risorse in tutto il mondo, tutte le volte in cui sono necessarie e disattivarle al termine delle operazioni, pagando solo le risorse necessarie nel momento in cui vengono utilizzate.

Transcodifica su richiesta

È possibile utilizzare il servizio cloud per eseguire la transcodifica dei propri video basandosi sui data center globali gestiti da Microsoft, senza pianificare le operazioni a seconda dei picchi di capacità, né preoccuparsi dell'inattività dei data center stessi. È possibile inoltre utilizzare transcodificatori Microsoft o uno dei codificatori integrati dei relativi partner.

Protezione delle risorse multimediali a livello avanzato

È possibile fornire contenuto di livello avanzato protetto da tecnologia DRM[18] e archiviato in modo sicuro. Le risorse vengono protette tramite crittografia dell'archiviazione durante il caricamento e tramite schemi differenti, quali PlayReady e AES Encryption durante la riproduzione.

Più dispositivi raggiunti con semplicità

Il modo più utile per raggiungere qualsiasi dispositivo ovunque: è sufficiente eseguire la transcodifica una volta e fornire il contenuto in più formati. Il servizio di streaming di Microsoft supporta la creazione dinamica di pacchetti in tempo reale, in formato HTTP Live Streaming (HLS[19]), Smooth Streaming[20] e MPEG-DASH[21]. Download progressivi consentono di fornire il formato migliore per i dispositivi utilizzati dai clienti,

[18] **Digital Rights Management** (DRM), il cui significato letterale è "gestione dei diritti digitali", è il sistema tecnologico che, applicato ad un'opera di ingegno, tutela il diritto d'autore e i diritti connessi rendendola protetta, identificabile e tracciabile.

[19] **HTTP Live Streaming** è un protocollo per lo streaming sviluppato da Apple e funzionante attraverso la classica connessione **HTTP** invece della più difficile da ottimizzare **RSTP**. Il video utilizza una compressione **H.264** e un audio in AAC o MP3 in uno stream MPEG-2.**QuickTime** sia su **Mac OS X** che su iOS è in grado di riprodurre il formato ed è anche l'unico modo per effettuare straning dal web su un **iPhone**.

[20] **Smooth Streaming** è un'estensione IIS che consente lo streaming multimediale verso client, utilizzando il protocollo HTTP.

[21] **Dynamic Adaptive Streaming over HTTP** (DASH) anche conosciuto come MPEG-DASH, consente streaming di alta qualità di contenuti multimediali su Internet, forniti da Web Servers convenzionali. È simile all'HLS di Apple cioè lavora frammentando il contenuto in una sequenza di piccoli segmenti di file basati su HTTP, ogni segmento contiene un breve intervallo di tempo di riproduzione di un contenuto multimediale molto lungo, in termini di tempo, come un film o la trasmissione in diretta di un evento sportivo.

mentre la creazione dinamica di pacchetti consente di risparmiare sui costi di archiviazione (storage), salvando un'unica copia del contenuto.

Monetizzazione del contenuto multimediale

Nei propri contenuti multimediali è possibile integrare annunci pubblicitari sovrapposti, precedenti, simultanei e successivi al contenuto stesso. I nostri servizi per i metodi di lavoro (workflow) e le strutture dei player (player frameworks) consentono di utilizzare qualsiasi server di annunci, di eseguire in modo semplice l'integrazione tra i dispositivi e di fornire report sull'utilizzo degli annunci pubblicitari che consentono di valutare i risultati in termini di investimento.

Scenari

Servizi di video di livello avanzato su richiesta (on demand)

Nei Servizi Multimediali di Windows Azure sono disponibili tutti i componenti necessari per far funzionare il proprio servizio di video su richiesta (Video on Demand) perché include tutti gli strumenti e i servizi per gestire l'elaborazione, la distribuzione e l'utilizzo del contenuto multimediale.

[Ciaran Quinn, Company Director DELTATRE
Mark Silver, Head of Digital CANADA'S OLYMPIC BROADCAST MEDIA CONSORTIUM
David Botbol, Deputy Managing Editor Sports FRANCE TELEVISION
Carlo De Marchis, CTO DELTATRE
Deltatre e Windows Azure trasmettono in streaming i giochi olimpici di Londra 2012
(http://azurevideos.com/deltatre/olympics/short-version/)]

Piattaforma di video online (Online Video Platform OVP)

Con l'integrazione dei Servizi Multimediali di Windows Azure è possibile creare o scalare[22] la propria piattaforma di video online e il proprio sistema di gestione del contenuto, utilizzando servizi precompilati e pronti per essere utilizzati, ad esempio codifica, creazione di pacchetti e streaming in qualsiasi formato per qualsiasi uso. I servizi integrati di prime o terze parti consentono di proteggere la propria piattaforma (…di servizi) anche nel futuro e di far crescere la propria azienda in tutto il mondo, sfruttando le caratteristiche di globalità dei Servizi Multimediali di Windows Azure, risparmiando in termini di costi e guadagnando in termini di scalabilità.

[Rick Cordella, Senior Vice-President Digital Media at NBC SPORTS GROUP]

Distribuzioni di soluzioni multimediali *end-to-end*

Estensioni Microsoft o di terze parti consentono di creare in modo semplice metodi di lavoro multimediali end-to-end in piena sicurezza, dall'utilizzo di contenuti fino alla codifica, la creazione di pacchetti e la protezione. Questo è l'obiettivo di più periferiche/piattaforme tipo Web, Windows, Android, iOS, TV, console di giochi, ecc.

[Collaborazione tra Microsoft e NBC Sports per trasmettere programmi sportivi in più piattaforme
(http://azurevideos.com/nbc-sports/)]

[22] Ricordo che **scalare** viene tradotto come la possibilità di estendere con ulteriori capacità e funzionalità il dispositivo, in virtù di necessità future.

Archiviazione, backup e ripristino

Vantaggi

Durevole

Disponibile in otto aree a livello mondiale, il servizio di archiviazione di Windows Azure è sicuro, affidabile e in grado di soddisfare ogni esigenza. La replica geo, progettata per garantire durata, offre ridondanza dei dati su più aree in modo da permettere l'accesso ai dati in caso di disastro locale.

Scalabile

Il servizio di archiviazione di Windows Azure si ridimensiona in base alle esigenze e supporta dati di qualsiasi dati di qualsiasi dimensione. Grazie alla disponibilità globale di Windows Azure, è possibile scegliere dove conservare i propri dati ed eseguire la scalabilità (espansione) nello stesso data center o tra data center.

Conveniente

Windows Azure offre archiviazione conveniente in tutto il mondo: si paga solo per le risorse utilizzate, disponibili ad un costo inferiore rispetto a molte altre soluzioni di archiviazione locali (on-premise), SAN[23] o NAS[24].

Scenari

Archiviazione ibrida

È possibile accedere in locale ai dati più utilizzati e immettere in cloud quelli meno utilizzati, usando servizi come StorSimple e Windows Azure. I dati vengono de-duplicati, compressi e crittografati prima dell'invio. È possibile ripristinare rapidamente i dati su un dispositivo StorSimple da qualsiasi postazione dotata di connessione ad Internet.

[Cynthia Weaver, AVP WALBRIDGE IT]

Backup dei dati sul cloud

I dati devono essere protetti da perdite e danneggiamenti. Con Windows Azure Backup è possibile eseguire il backup ed il ripristino dei dati da e su Sistemi Operativi Windows Server e Microsoft System Center, nonché archiviare e ripristinare files, snapshot di SQL Server Database e Macchine virtuali Hyper-V.

[AVIVA (http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?CaseStudyID=710000002635)]

Strategia di continuità SQL Server

È possibile attivare nuove strategie ibride di continuità aziendale, ad esempio distribuendo database primari SQL Server AlwaysOn in locale e database secondari basati sul cloud in Macchine virtuali. In caso di

[23] Una **Storage Area Network** (SAN) è una rete ad alta velocità di trasmissione (generalmente Gigabit/secondo) costituita esclusivamente da dispositivi di memorizzazione di massa, in alcuni casi anche di tipi e tecnologie differenti. Il suo scopo è quello di rendere tali risorse disponibili per qualsiasi computer connesso ad essa.

[24] Un **Network Attached Storage** (NAS) è un dispositivo collegato ad una rete di computer la cui funzione è quella di condividere tra gli utenti della rete una memoria di massa, costituita tra uno o più dischi rigidi.

errore locale, i database secondari basati su cloud possono essere utilizzati per ripristinare rapidamente le immagini, riducendo i tempi di inattività e contenendo la perdita di dati.

[Vesa Tikkanen, Chief Technology Architect SOLTEQ PLC]

Gestione delle identità e dell'accesso

Vantaggi

Active Directory e gestione delle identità in base alle proprie esigenze

L'utilizzo di Active Directory di Windows Azure (Windows Azure AD) come hub di identità (Identity Hub), consente di sfruttare un framework di identità flessibile per gli utenti e le applicazioni dell'organizzazione. È possibile creare nuove identità nel cloud o connettersi ad un'istanza locale di Active Directory esistente per gestire l'accesso a Microsoft Online Services, ad esempio Office 365 e altre applicazioni cloud.

Semplificazione dell'accesso da parte degli utenti

È possibile sincronizzare le identità in locale con Windows Azure AD e abilitare Single Sign-On per semplificare l'accesso alle applicazioni cloud da parte degli utenti. È possibile ridurre il numero di chiamate al supporto tecnico e di reimpostazioni di password con un unico set di credenziali per tutte le applicazioni a cui gli utenti accedono.

Protezione di dati e applicazioni sensibili

L'applicazione di criteri di autorizzazione e l'autenticazione avanzata degli utenti, consentono di proteggere l'accesso alle risorse cloud. È possibile applicare in modo coerente controlli di accesso basati su ruoli, gruppi e regole nonché aggiungere autenticazione a più fattori, per ottenere un livello ulteriore di sicurezza, soddisfare gli standard di sicurezza dell'organizzazione e rispettare la conformità alle normative relative alla protezione dei dati e alla sicurezza delle identità

Scenari

Active Directory per Office 365 e oltre

Active Directory di Windows Azure consente di applicare al cloud una gestione delle directory e delle identità di livello aziendale, tramite la quale è possibile amministrare centralmente l'accesso dei dipendenti ai diversi servizi, quali Windows Azure, Microsoft Office 365, Dynamics CRM Online, Windows Intune e altri servizi cloud di terze parti. È possibile abilitare l'accesso a servizi cloud per partner e clienti. Windows Azure AD è un servizio cloud di gestione delle identità a livello enterprise (elevato), caratterizzato da scalabilità globale ed elevata disponibilità.

La Georgia State University è passata al cloud per soluzioni Microsoft Office 365 risparmiando in costi di gestione per supportare al meglio i lavoratori mobili e remoti.

Sincronizzazione di identità locali (on-premise)

È possibile estendere la propria istanza locale di Active Directory di Windows Server per sincronizzare identità esistenti nel cloud. In Windows Azure AD è possibile altresì replicare aggiunte, eliminazioni e modifiche di utenti avviate in locale, semplificando in tal modo il processo di gestione degli utenti e proteggendo l'accesso a risorse presenti in locale, nel cloud o in entrambi gli ambienti.

[Nasos Kladakis, Product Marketing Manager Windows Azure AD
(http://www.azurevideos.com/faces/microsoft-nasos/)]

Abilitazione dell'accesso Single Sign-On

Per gli utenti è possibile abilitare facoltativamente l'accesso Single Sign-On per identità che si basano su Active Directory Federation Services o su altri servizi di token di sicurezza, per creare una federazione tra le directory cloud e quelle locali. Per le applicazioni cloud è possibile utilizzare la modalità Single Sign-On, tutte le volte in cui si desidera eliminare la necessità di più nomi utente e più password.

[Tim Fernandes, President CAPTIVELOGIX (http://www.azurevideos.com/faces/captivelogix-2/) Cloud Solutions for Health Care]

Il Cloud Computing

Differenti approcci al Cloud Computing

L'idea di base di qualunque piattaforma cloud è applicare un canone in base all'utilizzo effettivo delle risorse di questa piattaforma, scalando in base alle necessità del business.

I diversi fornitori di cloud computing, in genere, interpretano questo principio in modi diversi, fornendo differenti tipologie di servizi per raggiungere questo obiettivo.

Fondamentalmente esistono tre tipologie di approcci al cloud computing (Tabella 1):

1) Infrastructure as a Service (IaaS),
2) Software as s Service (SaaS),
3) Platform as a Service (PaaS).

Infrastructure as a Service

Alcuni fornitori mettono a disposizione dei clienti la propria infrastruttura per far realizzare le loro soluzioni, noleggiando server, load balancer[25], firewall e cablaggi. Ricade sull'utente il compito di configurare ed installare, da remoto, le proprie soluzioni. È possibile, quando si ha la necessità, di dimensionare l'infrastruttura a noleggio, potenziandola o depotenziarla, ricordando che si paga a utilizzo, appunto … necessità, e, quindi, bisogna saperla configurare.

Possiamo riassumere in tre punti la focalità di questo approccio:

1. i livelli più bassi dell'infrastruttura sono gestiti dal fornitore,
2. solo pochi fornitori metto a disposizione dei propri clienti un Sistema Operativo,
3. il cliente viene sollevato dagli aspetti fisici legati all'infrastruttura.

Software as a Service

Ci sono fornitori che noleggiano servizi. Il cliente configura il servizio attraverso un'interfaccia messa a disposizione dal fornitore medesimo, senza per questo dover conoscere i dettagli dell'infrastruttura che veicola il servizio.

Anche questo approccio si può riassumere nei seguenti punti:

1. l'utente (cliente) non ha la responsabilità ne il controllo dell'infrastruttura sulla quale il servizio è installato,
2. non ha il controllo sul sistema operativo sulla quale il servizio gira, ne qualunque altro elemento software se non quelli che l'interfaccia espone all'utente.

Platform as a Service

L'utente (cliente) noleggia una piattaforma sulla quale fare il deploy delle proprie applicazioni, senza preoccuparsi di configurare l'infrastruttura (IaaS) e senza le limitazioni dei servizi disponibili (Saas).

[25] Il **load balancing** (bilanciamento di carico) è una tecnica informatica utilizzata nell'ambito dei sistemi informatici che consiste nel distribuire il carico di elaborazione di uno specifico servizio, ad esempio la fornitura di un sito Web, tra più server, in maniera tale da avere larga scalabilità, ad esempio minori attese nella elaborazione di una pagina web, ovvero affidabilità dell'architettura nel suo complesso.

La piattaforma Windows Azure rientra nella definizione di PaaS (Tabella 1), dato che non fornisce l'accesso al sottostante ambiente di virtualizzazione o ai dettagli del sistema operativo quali l'interfaccia di rete, la configurazione degli IP o la gestione dei dischi.

I concetti chiave si possono riassumere nei seguenti punti:

1. Il fornitore della piattaforma è responsabile di tutto, dalla connettività di rete al runtime,
2. Le offerte PaaS forniscono sia il runtime della piattaforma che i relativi servizi applicativi,
3. Gli sviluppatori possono concentrarsi sulla logica di business[26] delle loro applicazioni.

Cloud service

La responsabilità dell'utente e del fornitore sono riassunte nella seguente immagine:

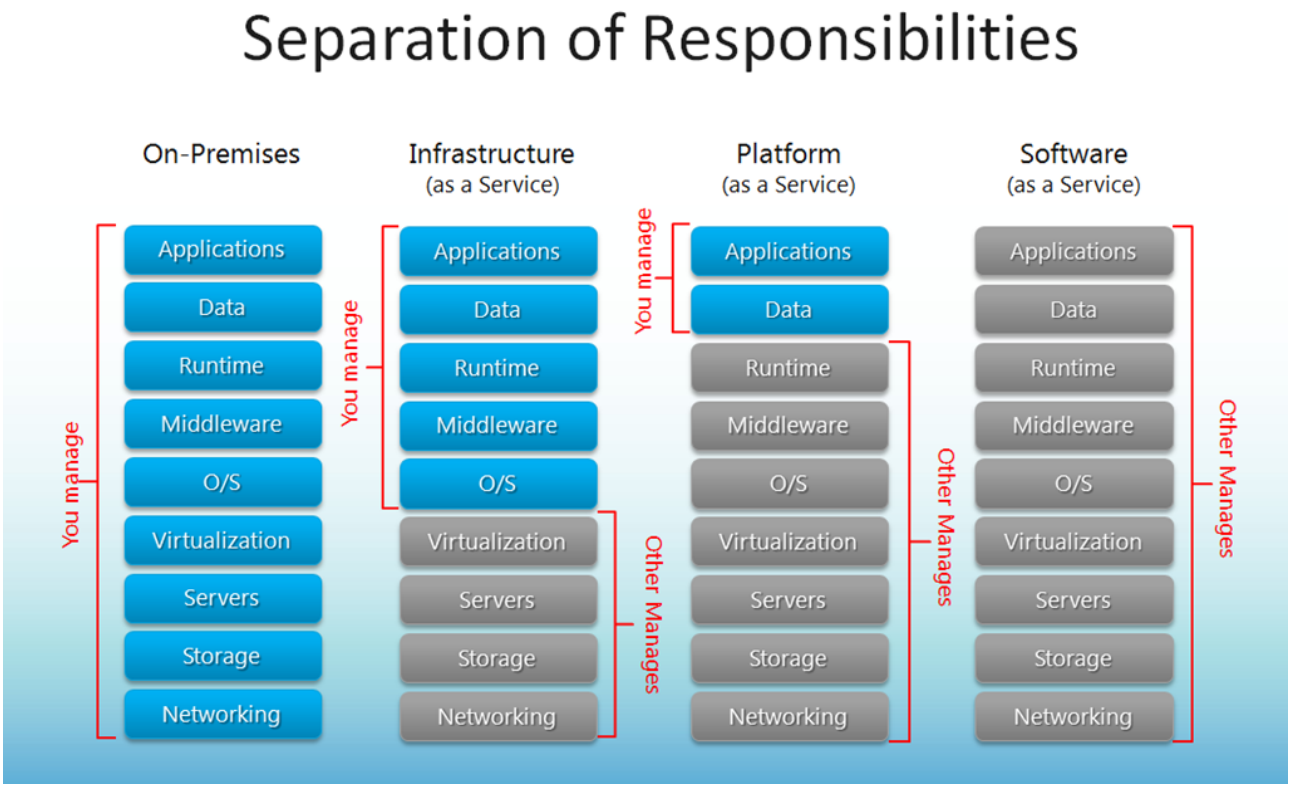


Figura 1 – Responsabilità dell'utente e del fornitore

Come si può vedere tutte le diverse offerte del settore del cloud prevedono la fornitura di un set di servizi che possono essere noleggiati, senza preoccuparsi dei servizi amministrati dal fornitore (riquadri in grigio).

La definizione di cloud computing offerta da Wikipedia (http://it.wikipedia.org/wiki/Cloud_computing) nel 2013 è: "In informatica con il termine cloud computing si indica un insieme di tecnologie che permettono, tipicamente sotto forma di servizio offerto da un provider ad un cliente, di memorizzare/archiviare e/o elaborare dati, grazie all'utilizzo di risorse hardware/software distribuite e virtualizzazione in Rete in un'architettura client-server".

[26] L'espressione **logica di business** o **business logic** si riferisce a tutta quella logica applicativa che rende operativa un'applicazione cioè l'elaborazione (il nucleo o core). Quindi ci si riferisce all'algoritmica che gestisce lo scambio di informazioni tra una sorgente dati (generalmente una base dati), l'interfaccia utente e le elaborazioni intermedie sui dati estratti.

In sostanza due sono gli importanti aspetti che ricorrono nelle offerte di servizi cluod:

1. L'uso di risorse distribuite (IaaS, SaaS e PaaS)
2. L'astrazione dello sviluppatore rispetto alle tecnologie sottostanti: cioè gestire risorse astratte come ad esempio lo storage distribuito, senza dover conoscere i dettagli tecnici sulla loro configurazione, messa in sicurezza e distribuzione.

Una prospettiva di lungo termine

Immaginiamo un futuro nel quale tutti gli aspetti fisici legati ai dati ed ai programmi siano completamente superflui per l'utente… ma abbiamo appena cominciato questo percorso: rimane ancora tutta la strada da percorrere.

Oggigiorno un termine che definisce un ecosistema di servizi interconnessi che possono scambiare dati e condividere processi, usando pattern e standard comuni, è SOA (*Service Oriented Architecture*).
Un servizio SOA può essere consumato da applicazioni installate su piattaforme eterogenee, che utilizzano sistemi operativi differenti e si basano su ambienti di programmazione differenti. L'architettura *service oriented* definisce concetti di interoperabilità che funzionano su sistemi e piattaforme diverse. Ciascun servizio può essere implementato con tecnologie e approcci differenti, dato che SOA si limita a definire il modo con cui questi servizi comunicano tra di loro e con le applicazioni client, dando agli sviluppatori la libertà di implementare la logica interna nel modo che desiderano.
Per esempio un servizio implementato in .NET può utilizzare solo altri componenti .NET e API Windows; questo servizio è completamente diverso rispetto ai suoi analoghi sviluppati in Java o Ruby ma deve poter "parlare" con altri servizi SOA o applicazioni client, per cui deve aderire agli standard citati precedentemente.

L'evoluzione dei linguaggi di sviluppo, dei sistemi operativi e dei framework già fornisce uno strato di astrazione rispetto alla piattaforma locale. Per esempio nella maggior parte dei linguaggi moderni, non c'è più bisogno di gestire la RAM in modo diretto. Al contrario negli attuali ambienti basati su garbage collector[27], una volta rilasciate le proprie istanze nel modo corretto, è il framework che si occupa di rilasciare memoria al sistema operativo.
I compilatori di oggi consentono di astrarre rispetto al codice macchina; i sistemi operativi aggiungono un livello di astrazione rispetto ai dettagli della memoria, dei dischi e delle schede grafiche, lasciando ai runtime[28], come il Common Language Runtime (CLR) o Java Virtual Machine (JVM), il compito di gestire i dettagli fisici al posto nostro.

---

[27] In informatica per **garbage collection** (letteralmente raccolta di rifiuti) si intende una modalità automatica di gestione della memoria, mediante la quale una sistema operativo, o un compilatore, o un modulo run-time, libera le porzioni di memoria non più utilizzate dalle applicazioni. Il **garbage collector** annoterà le aree di memoria non più referenziate, cioè allocate da un processo attivo, e le libererà automaticamente.

[28] **Run-time** o **runtime system** è il termine con cui in informatica si designa quel software che fornisce i servizi necessari all'esecuzione di un programma, pur non facendo parte in senso stretto del sistema operativo. Esempi di runtime system sono:

1. Il software generato dal compilatore per la gestione dello stack,
2. La libreria software per la gestione della memoria (es. malloc),
3. Il codice che gestisce il caricamento dinamico ed il linking,
4. Il codice di debugging generato in compilazione o in esecuzione,
5. Il codice di gestione dei thread a livello applicativo.

Anche gli interpreti di codice e le macchine virtuali possono essere considerati runtime system, mentre servizi attivi e processi concorrenti sono considerati come middleware.

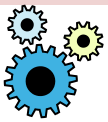
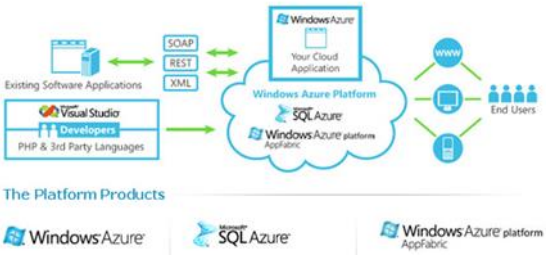
	Funzionalità	Servizi Offerti	Tecnologia utilizzata
 SaaS	**Industria del software** Hi-Tech, Internet, Finanza, Telecomunicazioni, Automobile, Mezzi di comunicazione, Produzione manifatturiera/industriale , Farmaceutica	Consulenza, Sviluppo e Test, Manutenzione, Supporto	Google Apps, Saleforce.com, Netsuite, Lotus, WebFilings, Zoho, Yahoo!mail, Hotmail, ecc.   
 PaaS	**Piattaforma unificata** SO – Prodotti Piattaforma di sviluppo Comunicazioni / Workflow - Metodi di lavoro / BI - Raccogliere dati e analizzare informazioni SCM - Gestione della catena di distribuzione / ERP -Pianificazione delle risorse d'impresa / CRM – Gestione delle relazioni coi clienti	Sviluppo e Test, Implementazione	  Google App Engine, Force.com, LongJump, RollBase, Amazon Elastic Beanstalk, VMware CloudFoundry, ecc.   
 IaaS	**Amministrazione dinamica** Server / Rete / Archiviazione / Sicurezza / Datacenter	Servizio di amministrazione dell'infrastruttura, Virtualizzazione	Amazon EC2, Rackspace, VMware, Joyent, Google Cloud Storage, ecc.   

Tabella 1

Windows Azure come soluzione PaaS

In una PaaS, non è necessario conoscere I dettagli tecnici di ciascun component, nè la differenza tra un hard disk RAID 0 o RAID 1; non occorre conoscere la velocità o la capienza di un hard disk, né se quest'ultimo è configurato come C oppure D. è sufficiente chiedere alla piattaforma dove salvare i propri dati e lasciare tutti i dettagli tecnici dell'operazione alla piattaforma stessa.La piattaforma Windows Azure nasconde questi dettali tecnici, mettendo a disposizione le API[29] necessarie per una gestione logica delle risorse.

È sufficiente creare un punto di archiviazione (*storage*), dargli un nome e usare l'endpoint[30] fornito dal sistema per gestire le risorse.

Continuando con lo storage.
Puoi usare standard quali REST e HTTP per interagire con questo tipo di storage.
Dove sono salvati i file? Semplicemente non hai bisogno di saperlo.
Sto usando dischi performanti? Idem come prima.

In un sistema come questo, ogni accesso a qualsiasi risorsa, consiste in un accesso al relativo servizio. Ciascuna API deve essere esposta come web service.

In pratica un sistema PaaS è una specie di "SOA per tutto" nel quale l'utente:

1. richiede un servizio storage dove salvare i propri file,
2. richiede al servizio storage di cercare uno specifico file,
3. richiede al servizio di gestione della piattaforma di aumentare o diminuire la scalabilità, sulla base delle proprie immediate esigenze,
4. richiede al servizio di storage di creare una nuova cartella,
5. il servizio replica alle richieste dell'utente (cliente) con una risposta che necessita di essere analizzata.

In particolare la piattaforma:

1. applica le patch non appena queste sono rese disponibili,
2. replica dati e runtime in modo da garantire fault-tolerance[31] e load balancing,
3. gestisce i dischi ed il resto dell'hardware,
4. se la quantità di dati aumenta, il sistema alloca automaticamente più dischi e riconfigura il load balancer senza alcun downtime ,
5. se l'applicazione smette di funzionare il sistema si riavvia automaticamente,
6. si può richiedere, ed è gestita dalla piattaforma, capacità computazionale,
7. se la macchina assegnata all'applicazione dell'utente (cliente), smette di rispondere, il servizio viene spostato in una nuova macchina, senza intervento alcuno da parte dell'utente.

[29] In informatica con il termine **Application Programming Interface** (**API**) o Interfaccia di programmazione di un'applicazione, si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un determinato programma. Spesso con tale termine si intendono le **librerie software** disponibili in un certo linguaggio di programmazione.
[30] Gli **endpoint** sono delle porte attraverso le quali le applicazioni comunicano con il mondo esterno.
[31] La **fault-tolerance** o **tolleranza ai guasti** è la capacità di un sistema di non subire interruzioni di servizio, anche in presenza di guasti. La tolleranza ai guasti è uno degli aspetti che costituiscono l'affidabilità.

Grandi opportunità per piccole aziende

È mia personale opinione che il cloud computing, e in particolare la piattaforma Windows Azure, rappresenta una grande opportunità per le piccole software house. Senza una infrastruttura di cloud computing, una piccola azienda o un freelance worker non potrebbero competere con società più grandi con notevoli capacità informatiche on-premise, quando si tratta di sviluppare applicazioni destinate a servire migliaia di utenti contemporaneamente, per il semplice motivo che le loro capacità d'investimento nell'acquisto di hardware sono limitate.

Ad esempio ipotizziamo di progettare un'applicazione finanziaria per il web (Figura 2).

Tra gli aspetti più importanti da considerare vanno annoverati non soltanto i **costi iniziali**, ma anche i costi necessari (di personale e non) per configurare e manutenere

1. Server web per la produzione e lo staging;
2. un database cluster;
3. Router e load balancer;
4. Firewall e sicurezza.

In aggiunta vanno considerati:

1. costi per banda richiesta dall'applicazione;
2. costi delle licenze del software;
3. costo di un numero di web server sufficiente a offrire una soluzione fault-tolerant per la propria applicazione, i danni collaterali conseguenti al fallimento, possono essere persino più costosi dell'investimento iniziale necessario ad evitare tali situazioni (global insurance);

Ammettiamo infine che l'azienda riesca a mettere in produzione l'applicazione e che il numero di clienti cresca rapidamente, l'azienda sarà costretta a comprare e configurare nuovo hardware, riconfigurando l'intero sistema (grazie al personale interno, opportunamente formato o a professionisti esterni). Questo può rappresentare un vero problema quando l'applicazione ha successo ed il numero di utenti cresce troppo in fretta.

Ricordiamoci che malfunzionamenti hardware e altri problemi sono dietro l'angolo, come insegna, del resto, la legge di Murphy: "Se qualcosa può andare storto, sicuramente andrà storto".

Dall'altro lato, però, se il numero di utenti non dovesse raggiungere le previsioni, l'azienda avrà già sprecato una considerevole somma per acquistare hardware in eccesso rispetto agli effettivi bisogni.

Se l'azienda dispone di un dipartimento marketing particolarmente efficace, potrebbe lanciare una campagna pubblicitaria per spingere la propria applicazione o servizio. In genere questo tipo di campagne prevedono un periodo di prova del prodotto per i nuovi utenti. Dal punto di vista del marketing, questa può essere una buona pratica, ma può condurre a drammatici picchi di traffico durante o immediatamente il lancio pubblicitario. Questo conduce a due ordini di problemi:

1. il rischio che i clienti già esistenti rivogliano i loro soldi indietro a causa di un ridotta qualità del servizio;
2. il rischio che gli utenti che stanno provando l'applicazione, non acquistino un servizio percepito come lento e inefficiente.

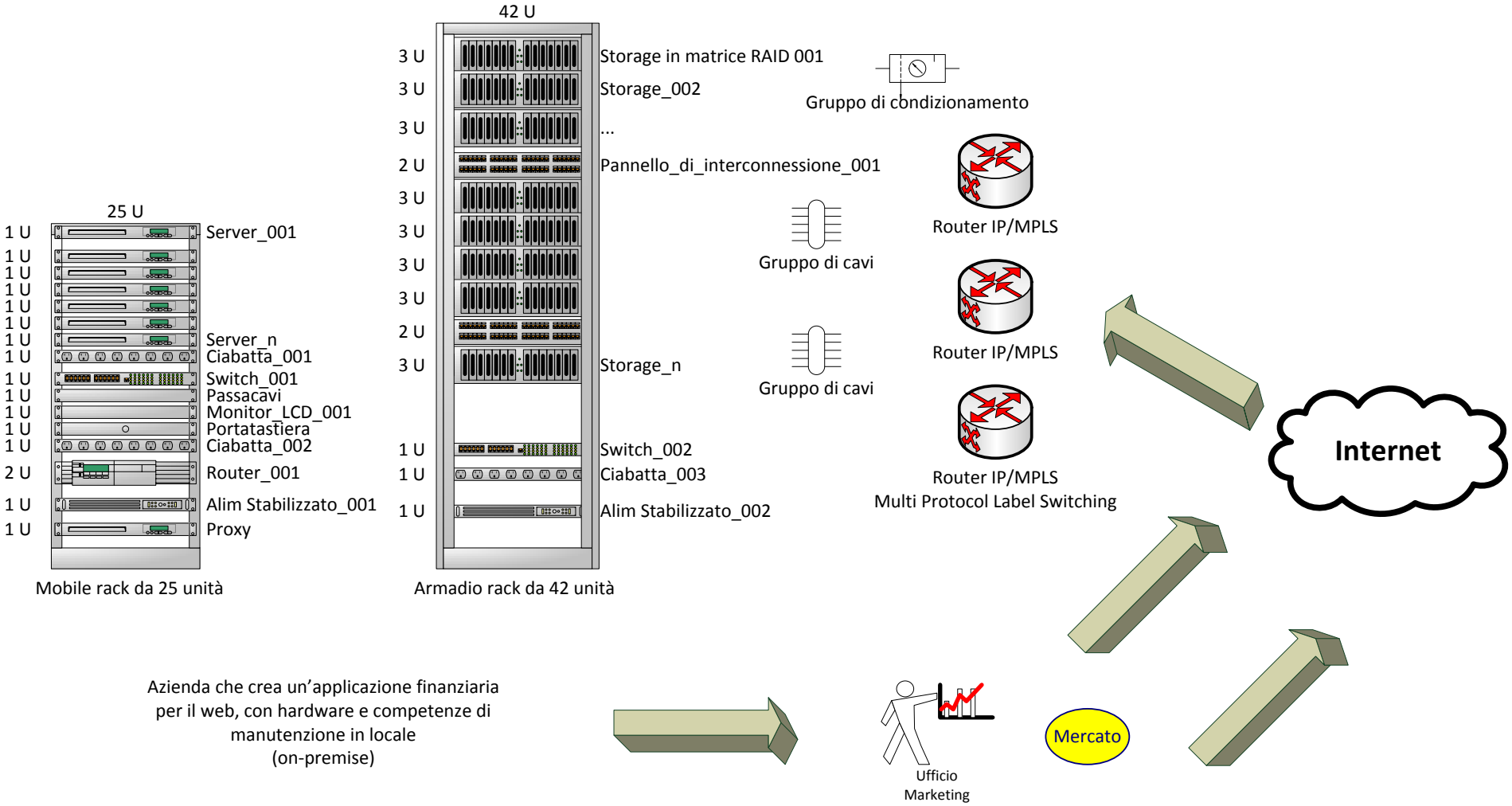


Figura 2 – Opportunità per piccole aziende

Se l'applicazione di esempio è un sito di e-commerce, potete immaginare i problemi che si avrebbero in una situazione simile. I clienti esistenti comincerebbero a sperimentare latenza durante le visite al sito, mentre i nuovi utenti vedrebbero un sito assai poco performante. La soluzione più semplice in questi casi, almeno dal punto di vista tecnico, sarebbe quella di incrementare il livello dell'hardware in concomitanza con il lancio della campagna pubblicitaria, ma in realtà i costi necessari rendono questo approccio poco praticabile.

Inoltre. Un gran numero di applicazioni devono fare i conti con quello che viene definito il "peak time". Per esempio, le visite ad un sito di e-commerce tendono ad aumentare durante le vacanze o prima della stagione estiva, oppure quando un nuovo prodotto viene lanciato sul mercato, o quando si apre una nuova stagione di moda. Analogamente, un'applicazione finanziaria genera normalmente più traffico verso la fine del mese o dell'anno fiscale, mentre un'applicazione dedicata al turismo e ai viaggi presenta dei picchi di traffico prima delle vacanze. Persino un'applicazione interna all'azienda presenta picchi di maggiore traffico verso la fine del mese, quando molteplici attività devono essere portate a termine (buste paga, consuntivo delle ferie, ecc.).

Se un'applicazione o un sito web di un'azienda sperimenta differenti livelli di carico nel corso del mese o dell'anno, ma questa ha optato per un modello a costi fissi , esiste un'intrinseca incongruità, dal momento che la maggior parte del tempo l'azienda è costretta a pagare per più di quanto effettivamente necessiti, solo per poter fronteggiare i picchi di traffico. Questo senza voler considerare il caso peggiore, in cui l'applicazione o il sito web della nostra ipotetica azienda si rivelino degli insuccessi, con conseguente perdita degli investimenti iniziali in hardware e licenze.

Al contrario, grazie all'impiego di soluzioni cloud, anche una piccola azienda è in grado di iniziare la propria attività con uno sforzo contenuto ed un costo minimo. Ecco quali sono i principali vantaggi derivanti dall'utilizzo di un'infrastruttura *cloud-based*:

1. nessun costo iniziale per i web server;
2. nessun costo fisso per la connettività di rete;
3. non è richiesta alcuna abilità particolare per l'installazione dei server;
4. nessun costo iniziale per i cluster di database;
5. le competenze richieste per la configurazione di un cluster di database non sono necessarie;
6. nessun router o load balancer da acquistare o configurare;
7. nessun firewall o software di sicurezza da acquistare o configurare;
8. non è richiesta alcuna competenza specifica per la messa in sicurezza sottostante;
9. nessun costo per l'ambiente di staging;
10. nessuna licenza da comprare.

Come questo elenco rivela, tutti gli sforzi iniziali solitamente richiesti da una soluzione in locale (on-premises) sono bypassati completamente. Si inizia a pagare solo nel momento in cui viene pubblicata la soluzione, e anche in quel caso è possibile aggiustare (aumentando o diminuendo) la capacità computazionale e lo spazio di archiviazione (storage), praticamente in tempo reale, per adeguarli alle effettive necessità.

Grandi opportunità per grandi aziende

I vantaggi del cloud computing si applicano ovviamente anche alle grandi aziende, ma con alcune differenze, quali ad esempio:

1. le grandi aziende dispongono di team appositamente dedicati all'IT e che probabilmente dispongono già delle competenze richieste per installare, configurare e manutenere un sistema di livello enterprise;
2. questi team sono in grado di gestire i differenti aspetti di una soluzione moderna, dai problemi di sicurezza alla performance della rete, sino alle politiche di installazione.
3. Lo stesso team può essere usato per diversi progetti, riducendo il costo marginale di una soluzione.
4. Team di grandi dimensioni dispongono in genere di procedure faul-tolerant e di persone appositamente preposte a rispondere ad eventuali avvisi di malfunzionamento.
5. Le grandi aziende solitamente avviano un nuovo progetto di business utilizzando macchine che già possiedono e possono soddisfare il carico iniziale.

È importante ricordare però che le soluzioni on-premises comunque incorrono in costi variabili nel corso del tempo: dai costi dell'energia elettrica, agli stipendi del personale IT, fino al noleggio della connettività.

Altre questioni riguardano tanto le grandi quanto le piccole aziende:

1. configurare uno stesso server affinché possa soddisfare le necessità di progetti completamente diversi può rivelarsi problematico per incompatibilità di una configurazione di un progetto con un'altra;
2. possono verificarsi problemi di scalabilità quando più applicazioni condividono gli stessi componenti.

Da un punto di vista puramente tecnico, usare server e infrastrutture differenti per ciascuna applicazione sarebbe la soluzione ideale, ma ciò avrebbe un impatto notevole sui costi complessivi. In un'infrastruttura di tipo cloud, i responsabili IT possono tenere separate le applicazioni e i servizi sia dal punto di vista logico che fisico, senza dover acquistare l'hardware necessario in anticipo.

Uno dei maggiori problemi che il cloud computing deve affrontare nelle grandi organizzazioni è dato dalla variabilità dei costi fatturati. Molte organizzazioni preferiscono allocare una somma fissa per ciascun progetto, piuttosto che assumersi il rischio di avere costi variabili.

Le grandi aziende dovrebbero modificare il loro approccio al cloud computing, anche in virtù del fatto che negli ultimi vent'anni sono stati già compiuti salti analoghi nella vita economica e personale. Ecco alcuni esempi:

1. Le compagni telefoniche hanno cambiato più volte il loro sistema di fatturazione, grazie a diverse offerte e piani tariffari, sia personali che business. Il sistema di tariffazione del cloud è molto simile, potendo passare da un costo fisso per funzionalità predeterminate, a una tariffa totalmente variabile.
2. In molti paesi sempre un maggior numero di persone vive in affitto. Molte compagnie non possiedono l'immobile dove hanno i loro uffici e quando hanno un problema possono chiedere assistenza al proprietario. Sul cloud avviene una cosa simile: problemi e patch ricadono sul proprietario dell'infrastruttura, non su colui che la usa.
3. Molte aziende noleggiano auto per i loro impiegati piuttosto che comprarle. Il noleggio ha tipicamente costi fissi, almeno entro limiti ragionevoli nell'utilizzo (relativi ad esempio al chilometraggio). Si paga di più solo al superamento di questi limiti. Il cloud computing è sostanzialmente la stessa cosa: si paga una tariffa fissa e si accettano delle limitazioni, ad esempio sulla banda e sullo storage, e solo se si eccedono queste limitazioni si pagano tariffe maggiorate.

4. Molte aziende già noleggiano hardware come personal computer, notebook e server; un po' come avviene nella piattaforma cloud, solo che questa include anche la banda, i router, i firewall e così via.

Il cloud computing incide anche sui sistemi "open source", dal momento che i consumatori non devono pagare alcuna licenza: il costo delle licenze è infatti incluso nelle tariffe di noleggio del cloud. In una soluzione on-premises i sostenitori dell'open source asseriscono che uno dei principali vantaggi nell'uso di sistemi operativi open source rispetto ad un sistema Windows consiste nel non dover pagare i costi delle relative licenze. Nel cloud questa affermazione semplicemente non trova applicazione, visto che gli utenti pagano per il sistema nel suo complesso.

Windows Azure e il Cloud Computing

Windows Azure è un sistema operativo per il cloud (e ospitato nel cloud) che opera una completa astrazione rispetto ai componenti fisici del sistema: l'utente deve soltanto scegliere le caratteristiche del servizio, i componenti e i livelli di Service Level Agreement (SLA), senza dover configurare alcun hardware o software. Per garantire la scalabilità e la fault tolerance , i dati immagazzinati in Windows Azure vengono replicati in tre nodi ed il load balancer lavora in modo trasparente.

Al momento in cui scrivo, l'utente può scegliere la capacità computazionale in base ad una varietà di macchine virtuali che si differenziano sulla base di:

1. CPU: è possibile variare da un singolo processore a 1 GHz fino a un processore a 8 core;
2. RAM: può variare da 768Mb fino a 8Gb (non è possibile scegliere il produttore, né la velocità né altre caratteristiche);
3. lo spazio disco locale: parte da 20Gb e può arrivare fino a 2Tb per ciascuna istanza. Non si sceglie la velocità, il controller o il tipo di ridondanza;
4. I/O performance: la scelta è elementare e include tre opzioni (bassa, moderata o elevata).

Se c'è bisogno di scalare verso l'alto, si può decidere di incrementare il numero di macchine utilizzate semplicemente da un file di configurazione e nel giro di cinque minuti, le macchine aggiuntive saranno perfettamente operative. Allo stesso modo se si vuole diminuire la capacità computazionale, è possibile ridurre il numero delle macchine e il sistema di fatturazione cesserà di conteggiare immediatamente le macchine dismesse.

È anche possibile cambiare le dimensioni delle macchine in qualunque momento; in questo caso però, occorrerà attendere un po' più a lungo prima di riavviare il servizio, perché il sistema ha bisogno di effettuare una nuova distribuzione dell'applicazione (deployment). Questo tipo di operazioni richiede in media cinque minuti, trascorsi i quali l'applicazione è di nuovo in esecuzione in nuove istanze su nuove macchine.

Tralascio di parlare che ogni aspetto tecnico del deployment ricade tra le responsabilità di Microsoft, come nessuno meglio di Microsoft conosce il funzionamento interno del .NET Framework, il kernel del sistema operativo, i componenti di Internet Information Services, SQL Server, e così via.

Per raggiungere una risorsa remota l'applicazione deve chiamare il servizio che espone la risorsa stessa. Dal punto di vista tecnico si usa l'URI[32] del servizio per inserire, modificare o cancellare una risorsa, mentre si

[32] L'**Uniform Resource Identifier** (URI) è una stringa che identifica univocamente una risorsa genericache può essere un indirizzo Web, un documento, un'immagine, un file, un servizio, un indirizzo di posta elettronica, ecc..

utilizza il pattern OData REST per le query al servizio aventi per oggetto le risorse esistenti (come liste, filtri, ordinamenti, paginazione, ecc.).

Questo di solito solleva una domanda: il codice che ho scritto girerà bene su Windows Azure?

La risposta è …. Dipende!

Se abbiamo sviluppato l'applicazione nel modo opportuno, disaccoppiandola dalle dipendenze e separando le funzionalità in layer appropriati, è probabile che girerà bene (se non meglio) sul cloud (se èp necessario salvare file in uno storage condiviso, alcuni adattamenti del codice saranno comunque necessari).

Al contrario, se il codice è monolitico ovvero non è stato fatto adeguato uso delle tecniche di programmazione orientate agli oggetti, o, peggio ancora, vi trovate di fronte ad un caso di "*spaghetti code*", che mescola elementi di user interface con business logic o di accesso ai dati, è verosimile che l'applicazione necessiti di un po' di lavoro per funzionare con un tipo di storage diverso. Comunque quando un'applicazione funziona con SQL Server, è probabile che possa essere portata su Windows Azure e SQL Azure con minime variazioni.

Se hai scelto di utilizzare il servizio storage di Windows Azure piuttosto che SQL Azure come servizio di storage per la tua applicazione, e hai ben strutturato la soluzione in layer, puoi sostituire il tuo attuale livello di accesso ai dati con uno nuovo (o adattare quello già esistente).
Il servizio di storage espone le risorse come servizio OData usando lo stesso pattern di Windows Communication Foundation (WCF) Data Services nel .NET Framework 4.0(in precedenza conosciuto nel .NET 3.5 come ADO.NET Data Services). Se hai scelto questo tipo di tecnica di accesso ai dati per la tua soluzione on-premises, hai soltanto bisogno di adattare il codice in modo da utilizzare il modello di sicurezza esposto da Windows Azure e il gioco è fatto.

Introduzione alla piattaforma Windows Azure

Il sistema operativo

Il componente più importante e fondamentale della piattaforma è rappresentato da Windows Azure, il sistema operativo creato da Microsoft appositamente per il cloud.

Al pari di un qualsiasi altro sistema operativo, il suo scopo è quello di fornire un livello di astrazione rispetto ai componenti fisici che compongono l'hardware della nostra macchina, oltre a una serie di servizi che possono essere utilizzati da qualunque applicazione.

Il sistema presenta svariate affinità con il tradizionale file system, anche le differenze non sono da meno. Windows Azure è un sistema operativo che poggia su un sistema di server interconnessi tra loro, in modo da offrire una infrastruttura comune e scalabile alle applicazioni.

Non mi dilungo sul livello di astrazione dei sistemi operativi di solito utilizzati in locale o del livello di astrazione di Windows Azure (peraltro già discusso in precedenza §Una..) rispetto ad un insieme di server, piuttosto sottolineo che, come i sistemi operativi tradizionali, Windows Azure espone un modo per salvare i dati, denominato storage locale.
Questo Storage non consiste in un hard disk fisico, né si traduce in una tradizionale directory condivisa in rete come ad esempio\\servername\sharename\. Ciò che invece Windows Azure fornisce è uno storage condiviso.

Allo stesso modo fornisce capacità computazionale.

Astraendo tutto ciò che è fisico, possiamo affermare semplicemente che carichiamo la nostra applicazione nell'ambiente e lasciamo a Windows Azure il compito di scegliere i server, i dischi e le strategie di load balancing migliori per la nostra soluzione.

Ne deriva che i bisogni della nostra applicazione possono essere descritti in modo logico e non fisico:

1. non puoi forzare Windows Azure a fare il deployment su uno specifico disco o server;
2. non possiamo configurare l'indirizzo IP nel nodo virtuale;
3. non possiamo installare una versione corrente di Windows Azure in locale (ma possiamo usare l'ambiente si sviluppo locale che simula la versione cloud di Windows Azure, in modo da poter testare le nostre applicazioni prima di pubblicarle);

Windows Azure è sul cloud: tutto è sul cloud.

Abbiamo visto i principali vantaggi dell'approccio cloud seguito da Microsoft (no hardware, ho driver, no IP, no router, no firewall, ecc.) e Windows Azure si fa carico di tutti questi aspetti per noi.

È sufficiente fornire un insieme di regole che il "cervello" di Windows Azure (chiamato Fabric), dovrà seguire al momento della pubblicazione della soluzione. Questo insieme di regole è detto modello (*model*) e consiste in una descrizione logica della configurazione di un'applicazione sul cloud.

Windows Azure usa il termine servizio (*service*) per identificare ciascun singolo pezzo di codice che può essere esposto e utilizzato. Per esempio, un'applicazione ASP.NET, un blocco *while* che elimina dalla coda alcuni messaggi, un servizio Windows Communication Foundation (WCF), ecc., sono tutti esempi di servizi.

Ciascun servizio può essere ospitato sulla piattaforma presso server differenti. Ciascun server, o più precisamente, ciascun nodo (*node*) è basato su Microsoft Windows Server 2012, per cui virtualmente qualunque codice che gira in una soluzione on-premises può girare anche sul cloud.

L'ambiente di hosting è distribuito su diversi nodi, ciascuno ospitato da server. Ciascun server fa infatti parte di una *server collection* che risiede in un *container*.

Un container è simile a quelli che puoi vedere stivati in una nave da carico o caricati su un treno merci, e viene connesso ad un datacenter semplicemente collegando un gigantesco cavo che fornisce energia elettrica e connettività di rete.

Da questo punto in poi Windows Azure si prende cura di tutto.

La creazione di un Windows Azure storage account

Tralascio di parlare della creazione di un account Windows Live ID e di come registrarsi nel portale di Windows Azure per la versione di valutazione (http://www.windowsazure.com/it-it/pricing/free-trial/ portale italiano) di 90 giorni.

Creiamo un storage account di **test**. Una volta entrati nel portale di Windows Azure cliccare su Archiviazione (Storage) e di seguito sul + di Nuovo

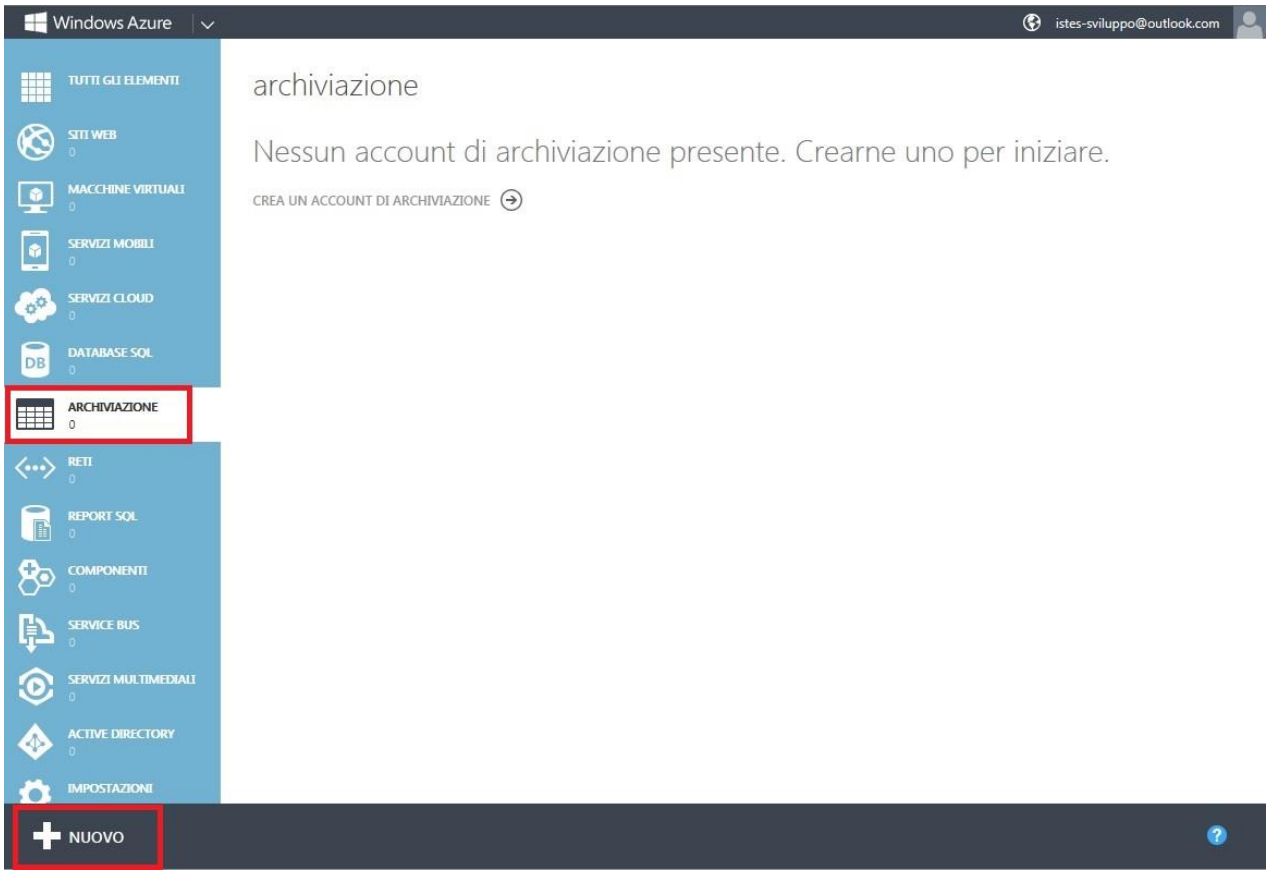


Figura 3 – Nuovo account di archiviazione (storage account) passo 1

Selezionare Creazione Rapida (Figura 4), nell'input box URL inserire un prefisso (unico, comunque controllato da Windows Azure) e nell'input box posizione/gruppo di affinità, selezionare un'area di distribuzione della tua applicazione. Disabilita la replica geo. Quando la replicazione geografica è abilitata per uno storage account, il contenuto archiviato viene replicato in una seconda locazione, questo per maggiori garanzie di recupero sdati, in caso di failover della locazione principale: ovviamente si può incorrere in costi addizionali.

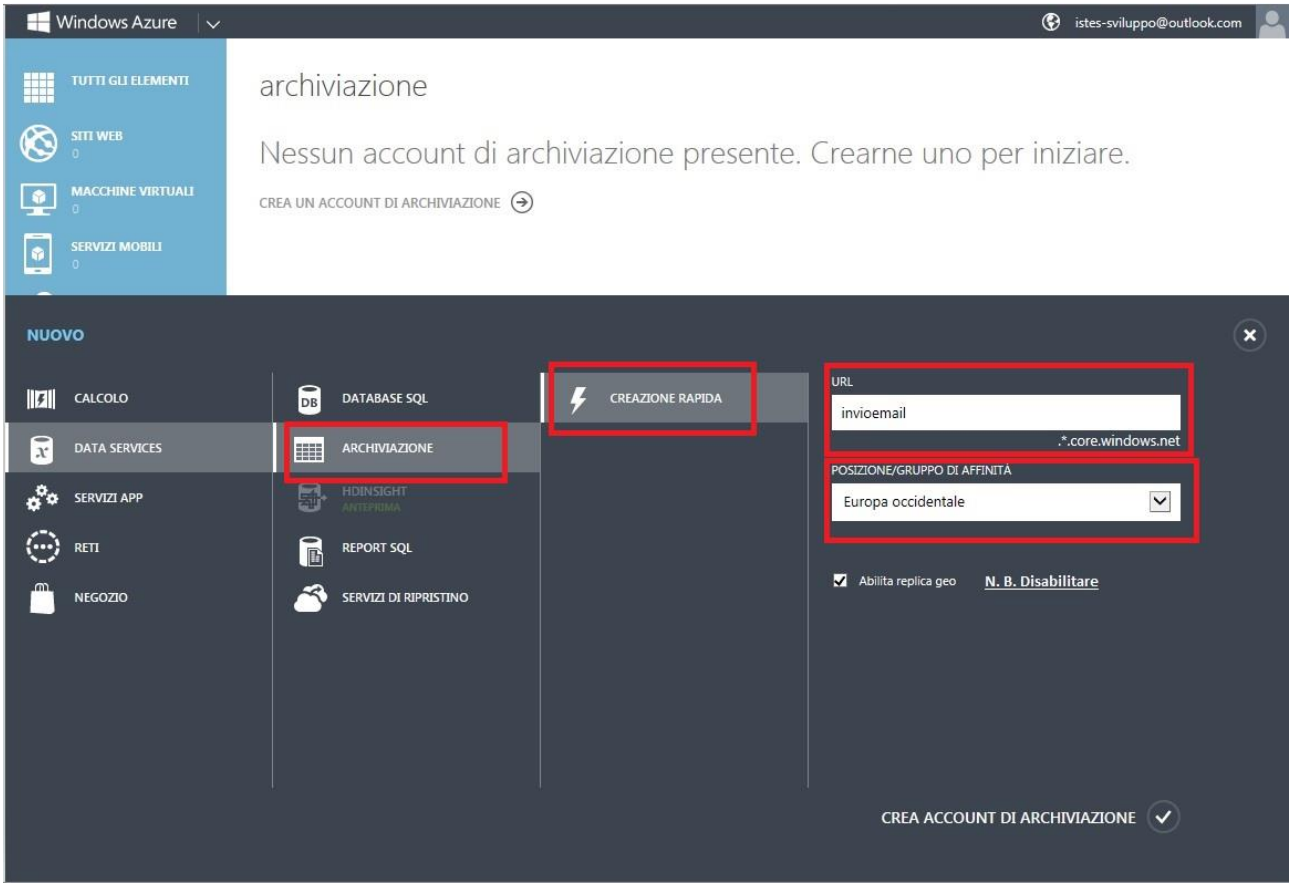


Figura 4 – Nuovo account di archiviazione (storage account) passo 2

Questo passo richiede alcuni minuti per il completamento. Mentre aspetti il completamento dell'operazione, puoi creare un account di **produzione**, diversificando la fase di test da quella di produzione: è spesso conveniente avere per i test in locale, un account di archiviazione locale, e su Windows Azure, un account di archiviazione per i test e un altro account per la fase di produzione.

A questo punto bisogna recuperare i valori delle chiavi di amministrazione dello storage account (Figura 5):

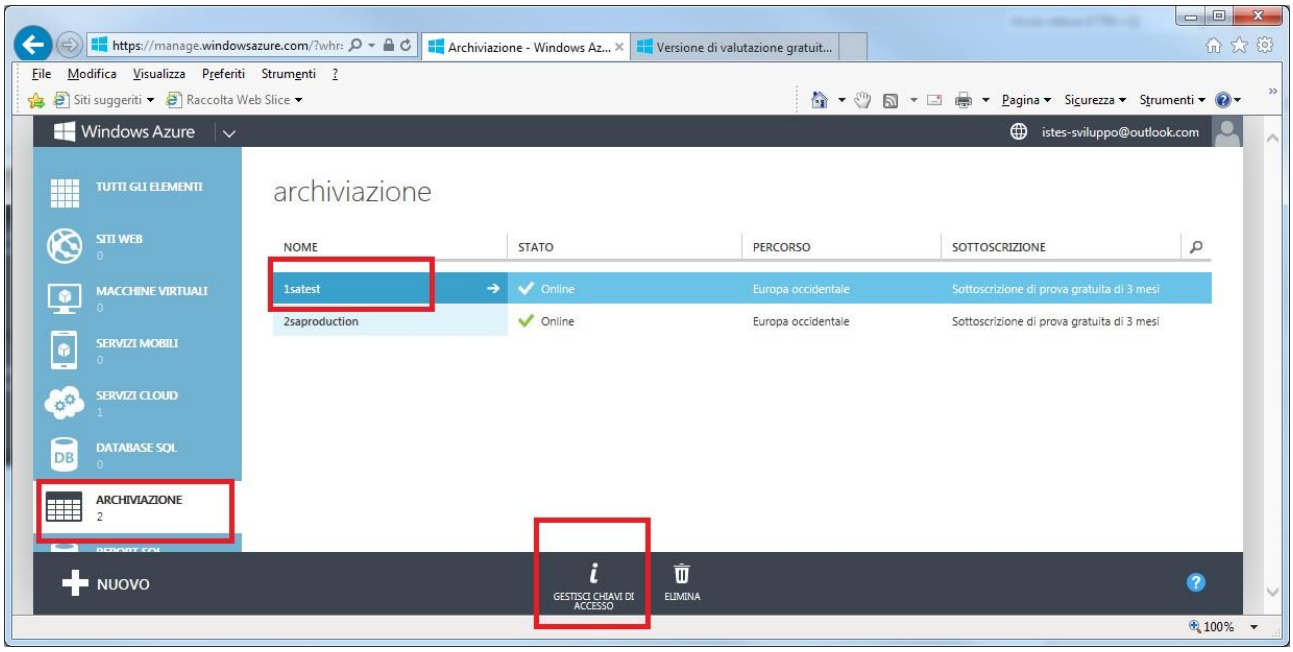


Figura 5 – Recupero dei valori delle chiavi di amministrazione dello storage account

Ci sono due chiavi (Figura 6) cosicché si può periodicamente cambiare la chiave che si usa senza causare una interruzione del servizio. Rigeneriamo la chiave che non si usa, di seguito cambiamo la connection string dell'applicazione facendola puntare sulla chiave rigenerata (Figura 7).

Se ci fosse stata una sola chiave, l'applicazione avrebbe perso la connettività allo storage account, finquando tu non avresti rigenerato la chiave e riconfigurata nella connection string dell'applicazione.

Figura 6 – Le chiavi per l'accesso

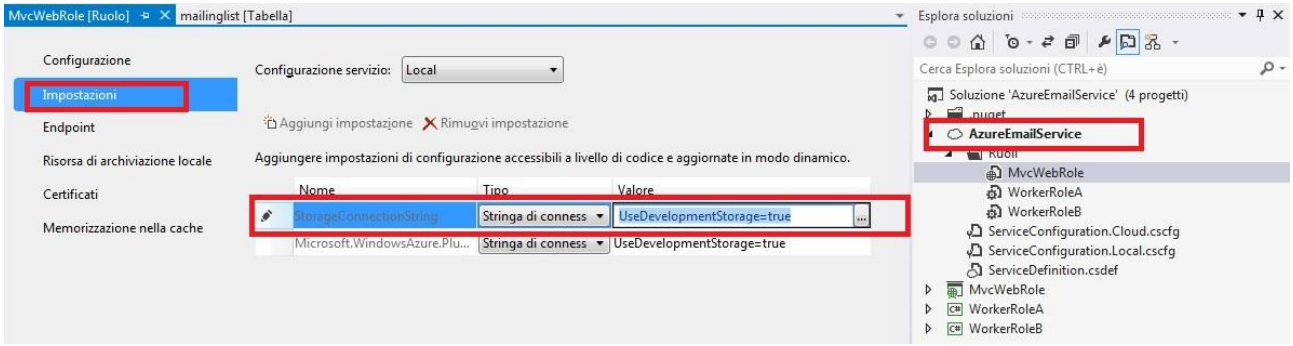


Figura 7 – Connection string che punta sulla chiave rigenerata

Installare e utilizzare Azure Storage Explorer

Azure Storage Explorer è uno strumento che puoi usare per richiedere e aggiornare tabelle, code e blob di Windows Azure (http://azurestorageexplorer.codeplex.com/) (Figura 8).

Lanciamo Azure Storage Explorer e inseriamo lo storage account (Figura 10) di test che abbiamo precedentemente creato, incolliamo la chiave che precedentemente abbiamo copiato (Figura 11)

Figura 8 – Installazione di Storage Explorer passo 1

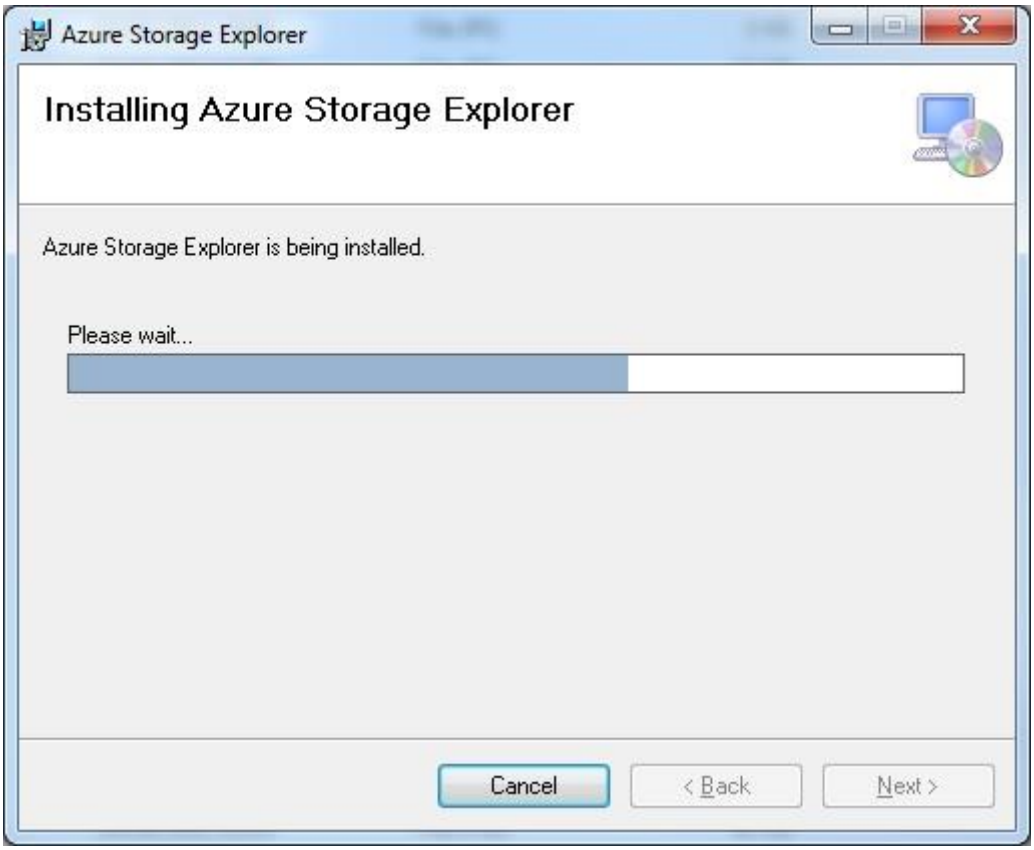


Figura 9 – Installazione di Storage Explorer passo 2

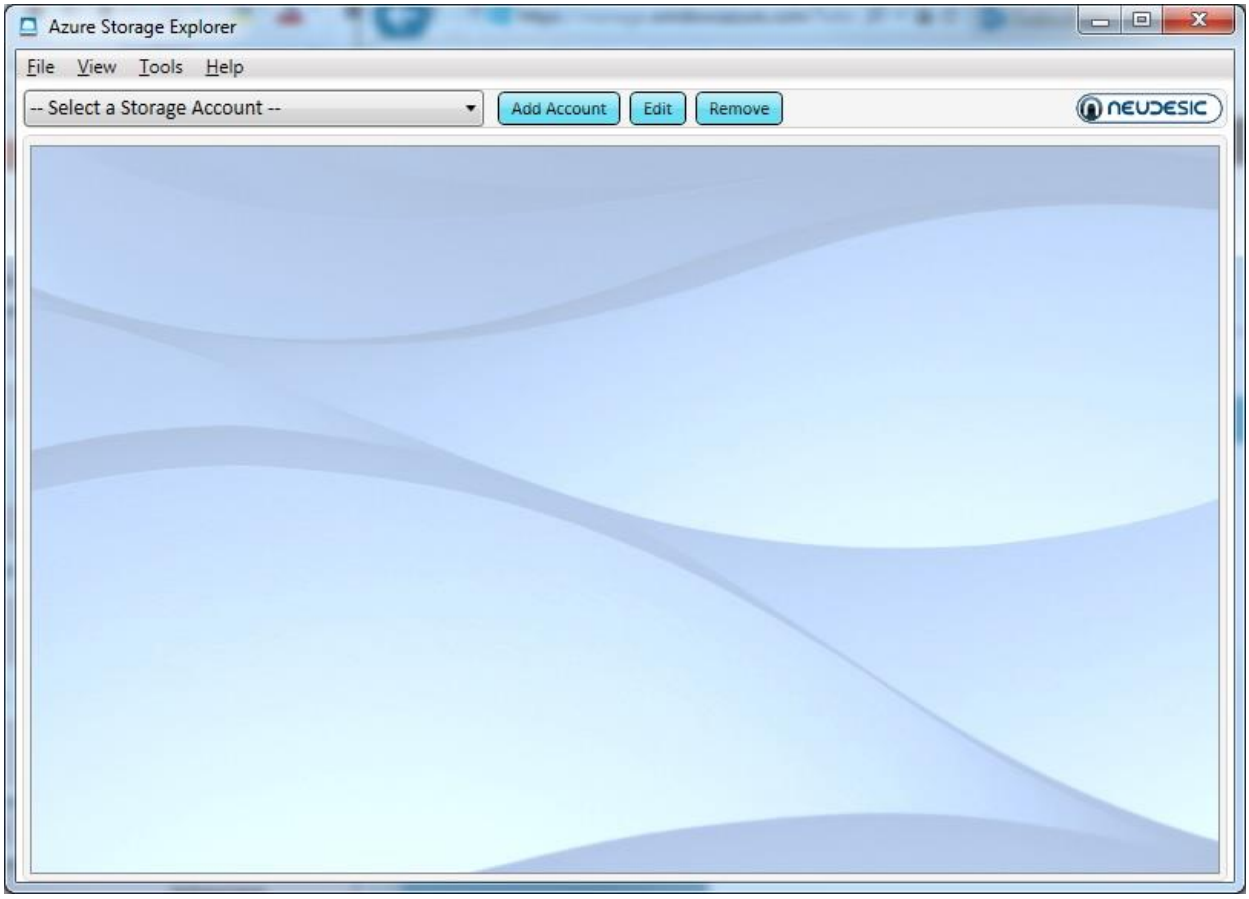


Figura 10 – Installazione di Storage Explorer passo 3

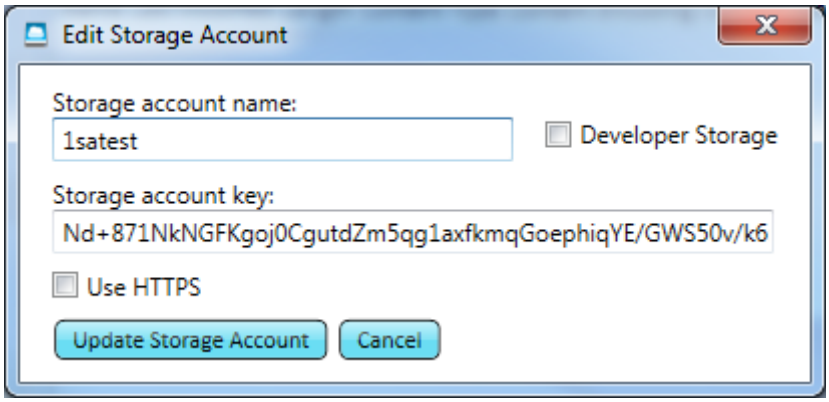


Figura 11 – Installazione di Storage Explorer passo 4 (inserimento delle credenziali di autenticazione)

Creiamo un servizio cloud

Nel browser, apri il Windows Azure Management Portal , clicca su Servizi cloud e di seguito sull'icona Nuovo (Figura 12)

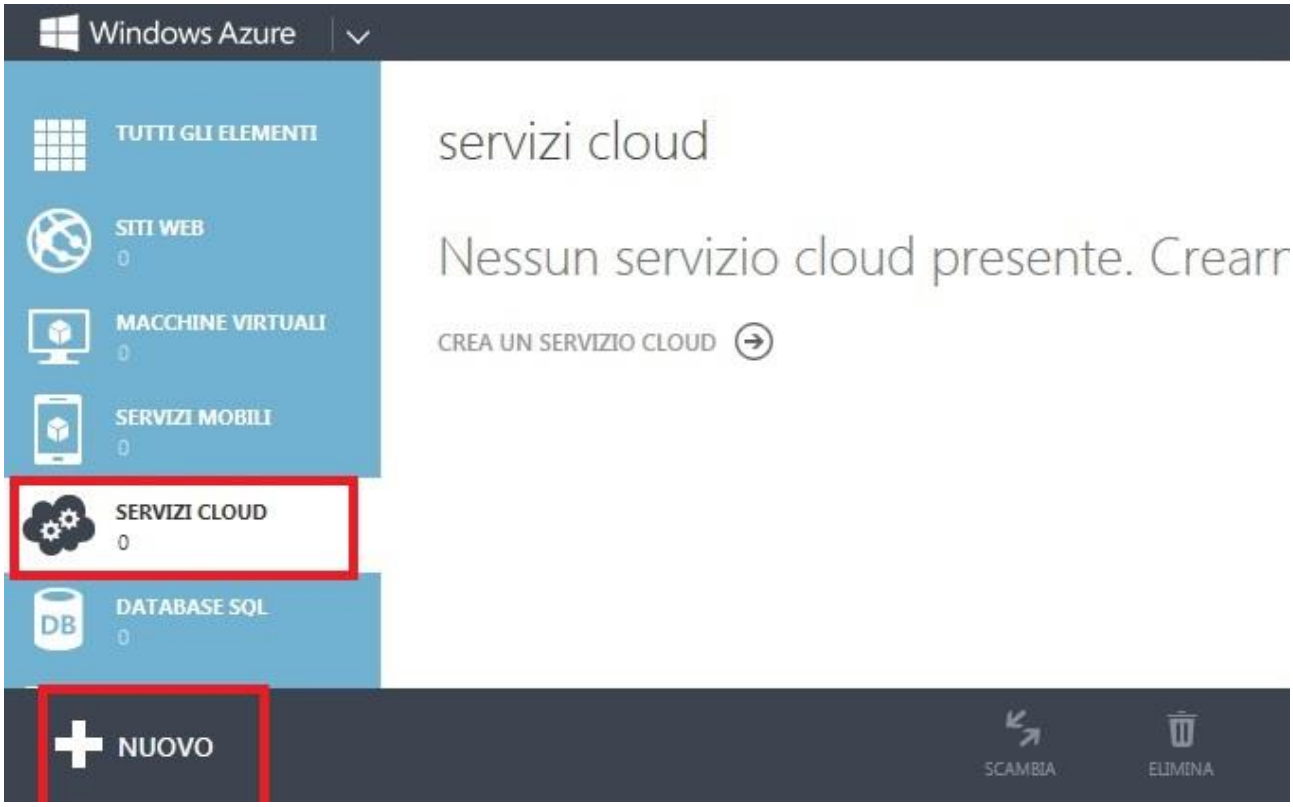


Figura 12 – Creazione di un servizio cloud passo 1

Cliccare sulla creazione rapida del servizio e nell'input box del prefisso dell'URL, inserire una descrizione che renda unico l'URL medesimo (come abbiamo visto già nella creazione di uno storage account) (Figura 13): l'URL che deriva dalla configurazione della figura sottostante è "*invioemailcloud.cloudapp.net*". L'URL rappresenta il nome pubblico per il tuo servizio (Public Service Name), dal momento che è necessario assegnare un nome univoco, il portale controllerà la disponibilità del nome mentre scrivi. Il nome non ha alcuna rilevanza pratica durante le operazioni di distribuzione (deployment) ma vale soltanto a distinguerlo dagli altri servizi. Di seguito inserisci la regione geografica dove ospitare il servizio (passaggio già visto nella

creazione degli storage account). Questa opzione permette di ospitare il tuo codice nella tua regione locale o in quella più vicina ai tuoi utenti/clienti.

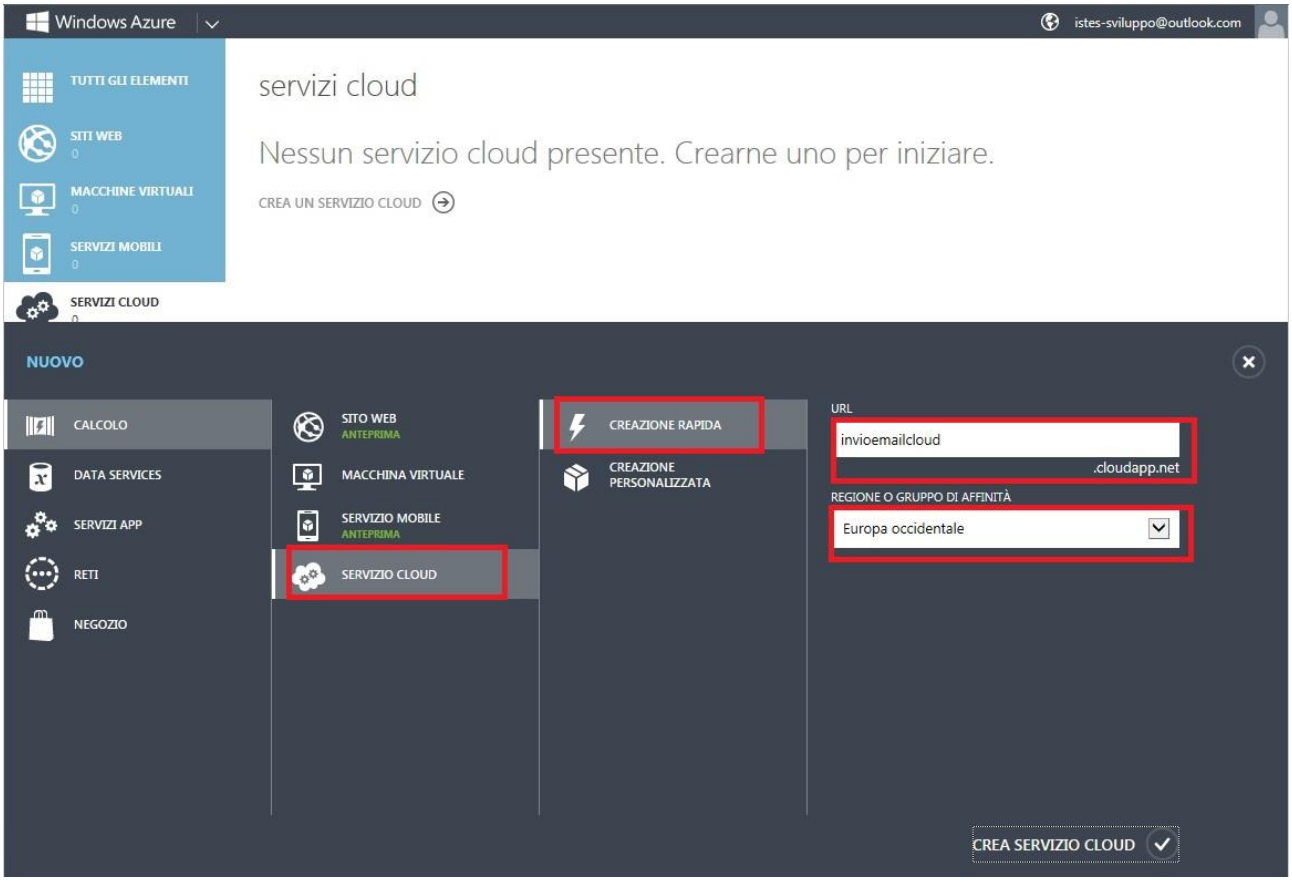


Figura 13 – Creazione di un servizio cloud passo 2

Abbiamo creato il nostro primo Hosted Service in cui effettuare il deployment della nostra soluzione!

Windows Azure espone due diversi ambienti per ciascun servizio ospitato: un ambiente di **produzione** e un'ambiente di **staging** (prova). I due ambienti utilizzano modelli e configurazioni differenti, così come un diverso numero di istanze. È possibile configurare l'ambiente di staging con un minor numero di istanze, sufficienti per testare l'applicazione, utilizzando differenti stringhe di connessione che puntano alle risorse di staging, invece che a quelle di produzione.

Quando un servizio viene caricato nell'ambiente di staging, viene fornita una URL temporanea composta da un GUID seguito dal nome del servizio. L'ambiente di staging è identico a quello di produzione in termini di tipologie di server, capacità di rete, tipo di processore e così via, per cui il sistema di tariffazione di Windows Azure comincerà a "funzionare" non appena viene effettuato il deployment del servizio, a prescindere dal fatto che il deployment sia stato effettuato in uno piuttosto che nell'altro ambiente.

È possibile invertire (*swappare*) i due ambienti ogni volta che vuoi. Di solito una prima versione dell'applicazione viene prima caricata nell'ambiente di staging e di seguito nell'ambiente di produzione. È anche possibile swappare in senso inverso, soprattutto quando l'applicazione, passata nell'ambiente di produzione, ha dei problemi e di conseguenza bisogna investigare per risolverli.

Windows Azure Storage

Il sistema operativo permette di immagazzinare tre differenti tipologie di risorse: blob, tabelle (*tables*) e code (*queues*). Uno storage blob è usato per salvare file, le tabelle rappresentano un tipo di contenitore per entità non strutturate, mentre le code sono utilizzate per disaccoppiare due applicazioni.

Per usare questo tipo di risorse, è necessario creare un progetto Storage Account tramite il portale. Gli Storage Account sono esposti pubblicamente via HTTP, in modo che sia possibile raggiungere, politiche si security permettendo, uno specifico storage virtualmente da qualunque luogo.

Gli Hosted Service possono usare lo Storage Account per salvare i dati dell'applicazione in modo permanente (si ricordi, al contrario, che lo storage locale è tale rispetto ad una singola macchina ed è soltanto temporaneo). Dal momento che lo storage è esposto via HTTP, una qualsiasi applicazione su un server on-premises o un'applicazione mobile, possono usare lo Storage Account di Windows Azure, a patto di disporre di una connessione internet.

Windows Azure espone i dati conservati nello Storage Account utilizzando standard internet come HTTP, REST [33]e OData[34] per cui questo risulta accessibile da qualsiasi piattaforma esistente.

Come puoi vedere in Figura 14, hai a disposizione tre differenti endpoint per il tuo storage, ciascuno dei quali è dedicato ad un diverso tipo di risorsa:

1. Blob. Questo servizio rappresenta il file system persistente per qualunque applicazione Windows Azure. Puoi usare il relativo endpoint per salvare immagini, documenti e altri file. Ciascun blob può essere organizzato in differenti *container*.
2. Table. Una table o tabella, è un contenitore di entità eterogenee che possono essere salvate e recuperate utilizzando una chiave. Una tabella non ha struttura predefinita e può essere organizzata in differenti partizioni, ciascuna delle quali può essere distribuita su server differenti per raggiungere il livello di scalabilità ottimale.
3. Queue. Una queue o coda, può essere usata per disaccoppiare applicazioni, abilitando la comunicazione asincrona. Windows Azure espone un semplice servizio che ciascuna applicazione, su qualunque piattaforma, può usare per inviare messaggi ad altre applicazioni.

Il tutto è esposto usando protocolli standard, per cui è possibile, ad esempio, effettuare una query REST tramite il metodo PUT da un'applicazione C# on-premises, per inserire un blob nello storage e quindi utilizzare il codice di un'applicazione Silverlight su Windows Phone 7 per compiere un'operazione GET per recuperare la lista di blob presenti nello storage.

---

[33] **Representational State Transfer** (**REST**) è un tipo di architettura software per i sistemi di ipertesto distribuiti come il World Wide Web. REST si riferisce ad un insieme di principi di architetture di rete, i quali delineano come le risorse sono definite e indirizzate. I sistemi che seguono i principi REST sono definiti spesso **RESTful**.

[34] Il protocollo **Open Data Protocol** (**OData**) nasce per standardizzare i meccanismi di accesso e di consumo dei dati attraverso l'utilizzo di tecnologie web ampiamente diffuse come l'HTTP ed il protocollo ATOM. Con delle semplici chiamate HTTP è possibile accedere alle basi dati di un numero sempre più vasto di portali web, come eBay, Flikr, Netflix, Stack Overflow e molti altri.
N.B. Con il termine **ATOM** ci si riferisce a due standard distinti. **Atom Syndication Format** è un formato di documento basato su XML per la sottoscrizione dei contenuti web, come blog o testate giornalistiche. **Atom Publishing Protocol** (AtomPub o APP) è un semplice protocollo basato su HTTP usato per la lettura, creazione e aggiornamento di risorse web.

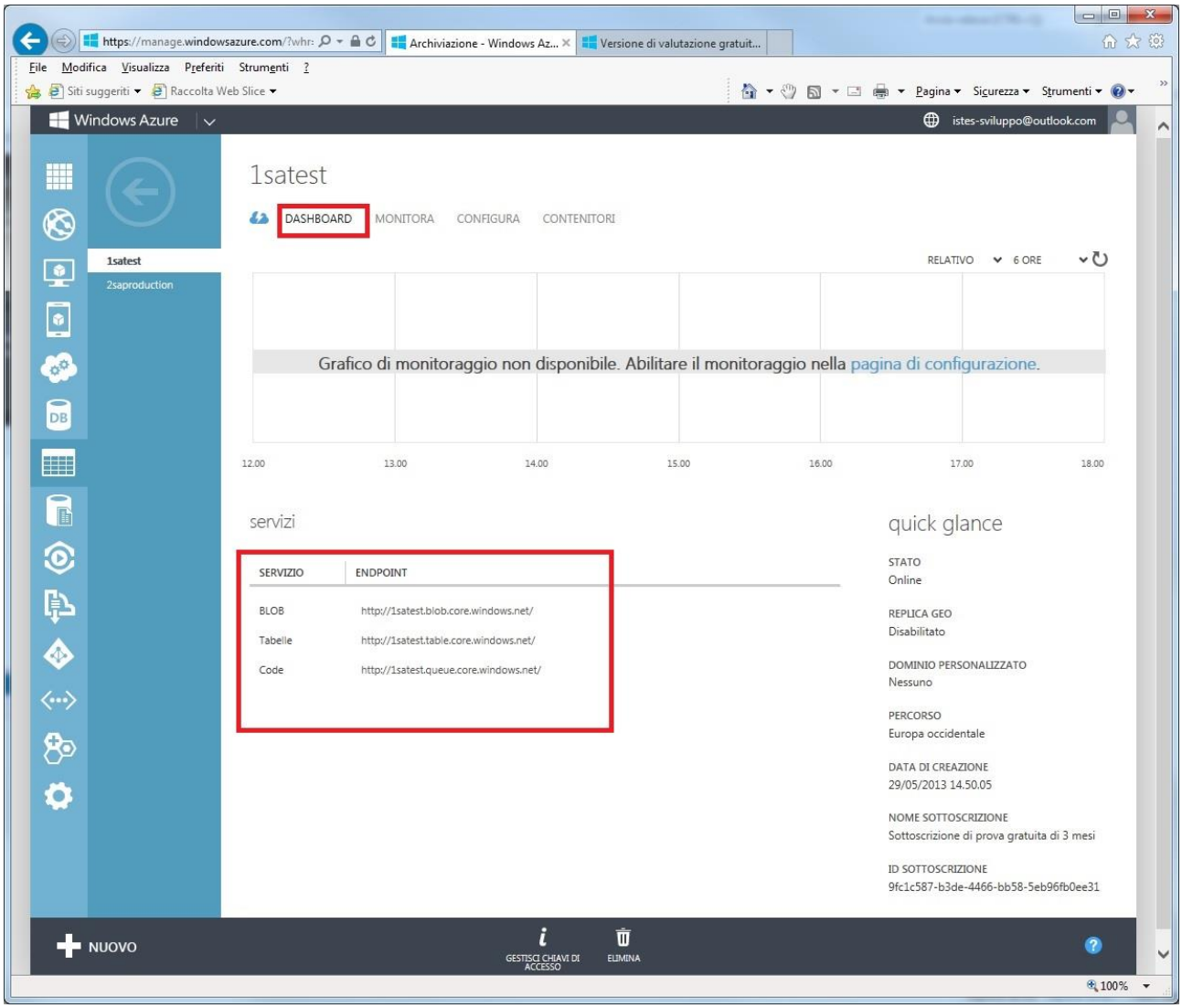


Figura 14 – Endpoint di blob, tabelle (Table) e code (Queue)

Worker Role

Lo Storage Account di Windows Azure espone un servizio che permette di disaccoppiare due applicazioni utilizzando un sistema di accodamento. Un classico esempio di questo tipo di soluzione consiste nel disaccoppiare il front end di un'applicazione web (il Web Role), rispetto al back end (il Worker Role).

Un *Worker Role* è un tipo di servizio che, per default, non è esposto all'esterno tramite endpoint, ma è dedicato allo svolgimento di determinate operazioni nel back end.

Un esempio di progetto Worker Role potrebbe essere un sistema di order-processing di ordini inseriti dal front end in una coda. Non appena inserito il messaggio in coda, il front end torna immediatamente libero per altre operazioni.

Le istanze del Web Role e del Worker Role possono essere modificate indipendentemente l'una dall'altra. Se vuoi accettare più ordini senza alcun problema e il thread[35] del front end che serve le richieste degli

[35] Un **thread** è una suddivisione di un processo in più sottoprocessi, che vengono eseguiti concorrentemente da un sistema di elaborazione mono o multiprocessore. In genere si può affermare che un thread è contenuto all'interno di

utenti, può essere collocato nel pool per gestire le nuove richieste in arrivo. Allo stesso modo se la coda comincia a diventare troppo lunga, puoi incrementare il numero di istanze del Worker Role.

Un Worker Role può aprire un endpoint verso l'esterno: in questo modo diviene raggiungibile in modo sincronoda un client remoto o da un Web Role sul cloud.

Virtual Machine Role

Una macchina virtuale, in Windows Azure, è un server nel cloud che è possibile controllare e gestire esattamente come un server o una macchina virtuale locale[36].

Per creare una nuova macchina virtuale è possibile partire da una delle immagini già predisposte e presenti nella Gallery di Windows Azure, oppure caricando un file VHD opportunamente preparato, con all'interno il sistema operativo che si vuole utilizzare (Figura 15).

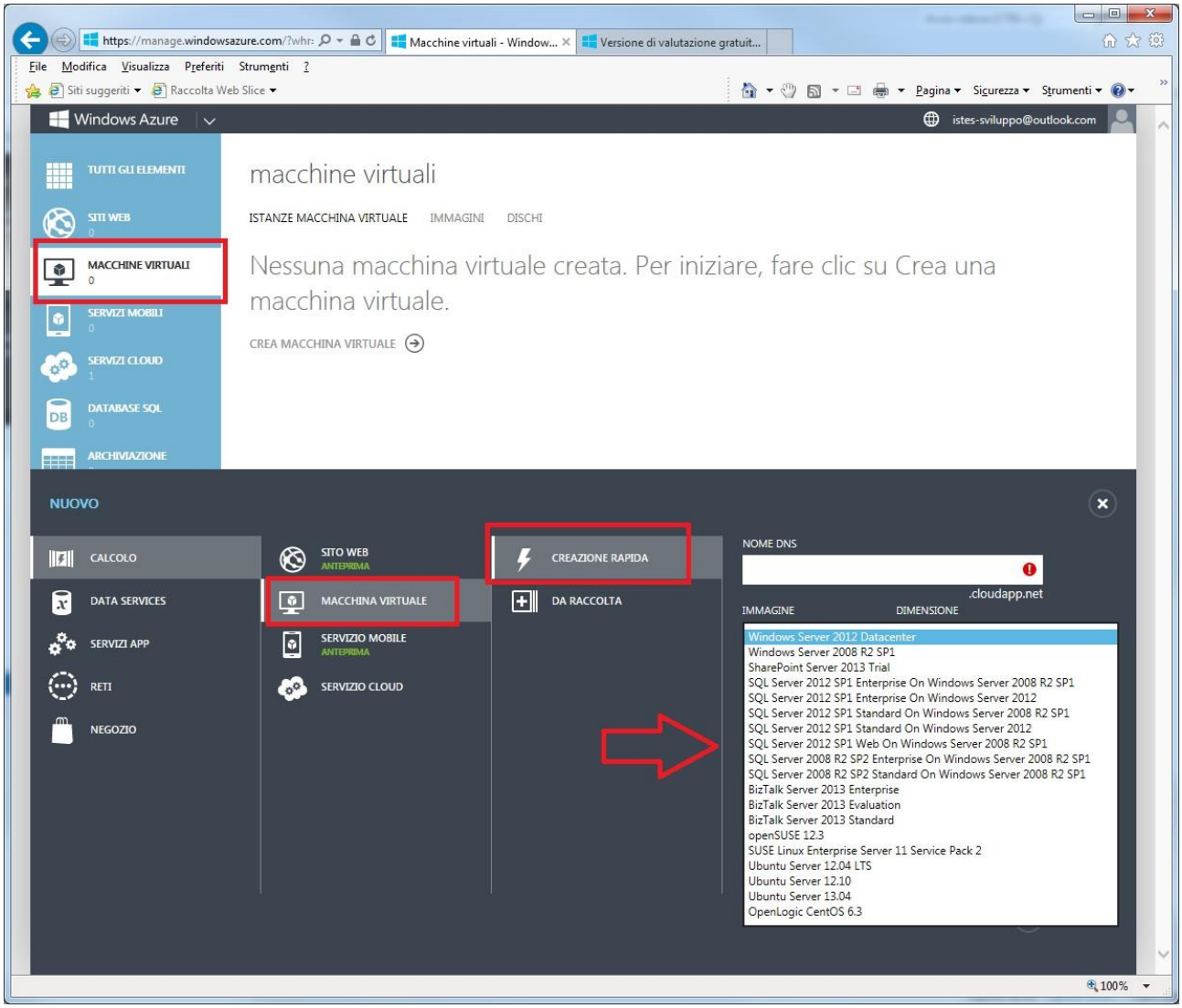


Figura 15 – Nuova macchina virtuale con dettaglio dell'mmagine da associare

---

un processo e che diversi thread contenuti nello stesso processo condividono alcune risorse (ad esempio, lo spazio di indirizzamento del processo), mentre processi diversi non condividono le loro risorse.

[36] http://social.technet.microsoft.com/wiki/contents/articles/14182.come-creare-una-macchina-virtuale-con-windows-server-2012-su-windows-azure-partendo-dalla-vm-gallery-it-it.aspx

È importante sottolineare che nella macchina virtuale, la sua configurazione, l'aggiornamento e l'applicazione di patch spetta ai sottoscrittori del portale Windows Azure. Al contrario Web Role e Worker Role costituiscono un ambiente automatizzato e più flessibile.

Windows Azure AppFabric (ex .NET Services fino a gennaio 2010)

Windows Azure espone i servizi base per l'intera piattaforma. Windows Azure AppFabric rappresenta un esempio di un set di servizi ospitati sopra il sistema operativo, in grado di fornire una piattaforma di middleware[37] completa.

Windows Azure AppFabric può essere usato per connettere tra di loro "pezzi" di un'applicazione, per gestire il controllo delle identità e degli accessi, per mettere in cache risorse remote e per creare applicazioni composite.

In generale Windows Azure AppFabric ci mette a disposizione un'infrastruttura di servizi cloud-based attraverso i quali gestire la connessione ad applicazioni distribuite di tipo cloud oppure on-premises.

I componenti fondamentali sono[38](Figura 16):

1. **Service Bus**. Lo scopo di questo componente è quello di suddividere un'applicazione (sia essa di tipo cloud o di tipo on-premises) in servizi accessibili su internet. Riesce in questo dando la possibilità al programmatore di suddividere l'applicazione in endpoint identificati da una URI. Questi endpoint sono quindi accessibili dai client attraverso il web.
2. **Access Control Service**. Consente ad applicazioni client di autenticarsi nel momento che richiedono un servizio di una applicazione ospitata nella piattaforma. Il fine è quello di capire cosa è permesso fare o non, all'applicazione client. Si tratta di una gestione dell'identità dell'applicazione da non confondere con la gestione dell'identità dell'utente che sta usando l'applicazione.
3. **Caching service**. Fornisce una distribuzione in memoria.
4. **Integration Service**. Fornisce le funzionalità di integrazione con BizTalk Server e patterns integrativi per facilitare lo sviluppo.
5. **Composite Application Service**. Questo servizio serve per facilitare lo sviluppo di applicazioni che usano servizi Windows Azure diversi. Contiene le estensioni .NET Framework e i componenti di integrazione di Visual Studio.

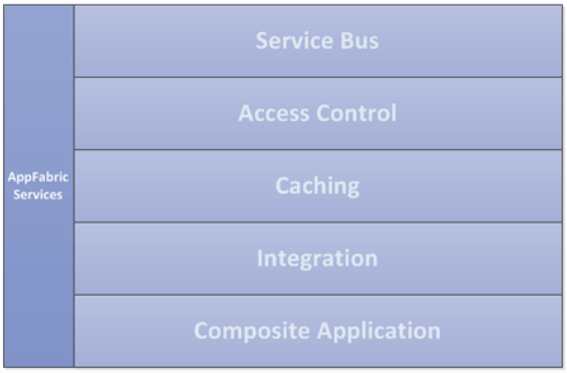


Figura 16 – Componenti di AppFabric

[37] Con **middleware** si intende un insieme di programmi informatici che fungono da intermediari tra diverse applicazioni e componenti doftware. Sono spesso utilizzati come supporto per sistemi distribuiti complessi.
[38] http://www.html.it/articoli/windows-azure-platform-appfabric/

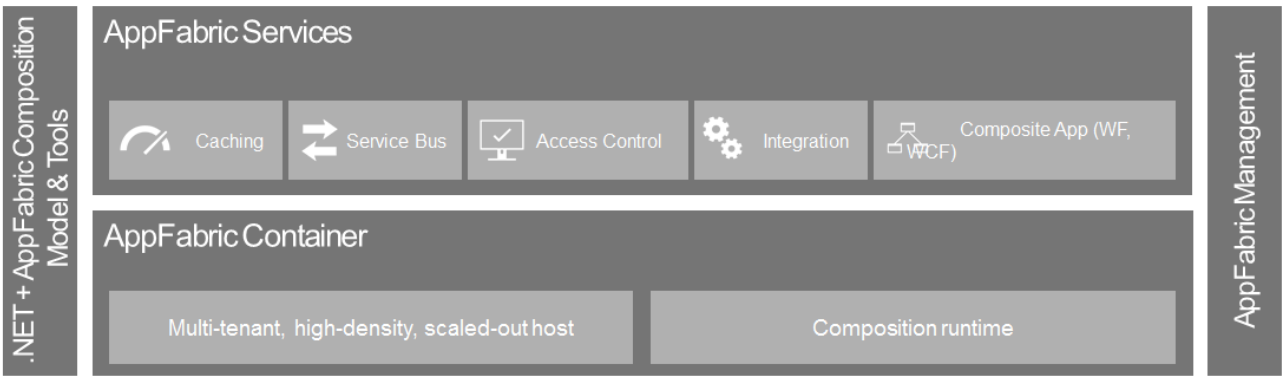


Figura 17 – Servizi di Windows Azure AppFabric

Service Bus

In particolare il servizio Service Bus fornisce connettività sicura e servizi di messaggistica, grazie ai quali, applicazioni distribuite e non interconnesse possono comunicare tra loro utilizzando il pattern del Service Bus. Ovviamente il Service Bus è ospitato sul cloud, per cui qualunque applicazione che dispone di una connessione a internet, può accedervi. Per cominciare a scambiare messaggi tra una qualunque applicazione e/o piattaforma usando il Service Bus, è necessario innanzitutto creare un nuovo namespace per il servizio tramite il portale. Il servizio usa standard internet come HTTP e REST per gestire servizi e scambiare messaggi.

La prossima Figura 18 illustra il tipico flusso di un messaggio inviato tramite Service Bus. In quest'immagine un'applicazione on-premisessituata dietro un firewall o un NAT si registra sul service bus mediante una semplice richiesta POST diretta all'URL pubblico, il service bus agisce come un relay, inoltrando i messaggi all'applicazione on-premises. Questa tecnica elimina il bisogno di esporre all'esterno l'applicazione on-premises, abilitando altresì nuove applicazioni a prendere parte al flusso di messaggi, semplicemente registrandosi sul namespace del servizio.

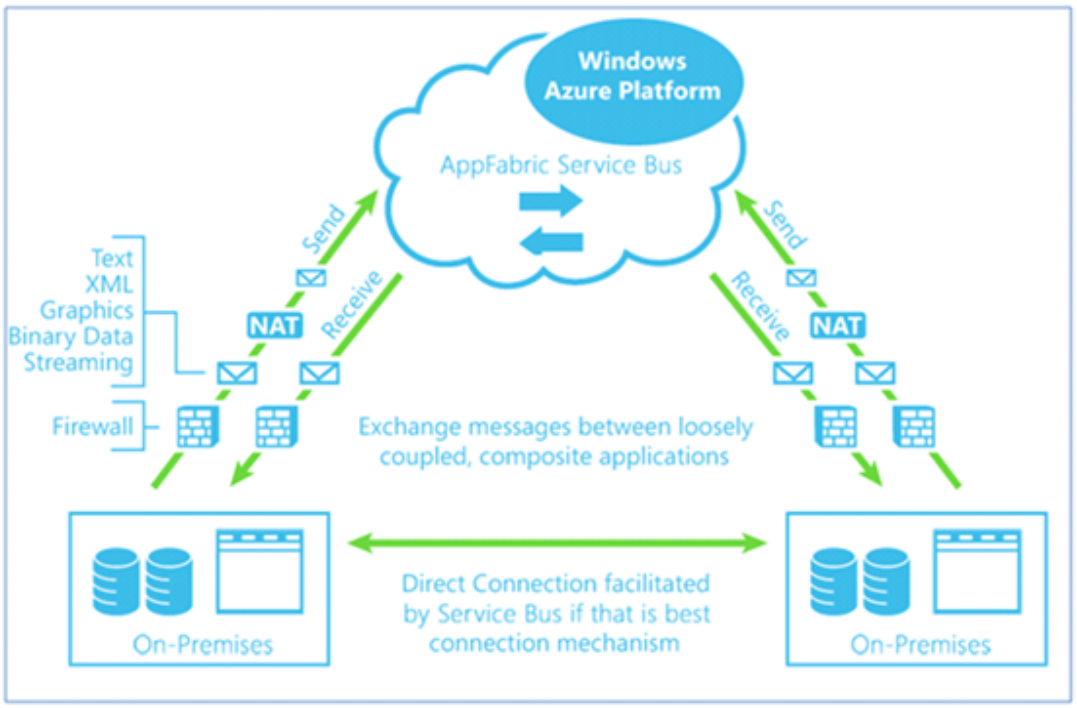


Figura 18 – AppFabric Service Bus

Access Control Service

Le regole di autorizzazione (*authorization rules*) possono essere esternalizzate da un'applicazione per essere inserite su Windows Azure AppFabric Acces Control Service. È possibile gestire i permessi e le claim[39], usando sia il portale che l'apposito strumento a riga di comando. Esistono anche strumenti di terze

[39] I Claims non sono ancora alle orecchie dei molti, mi sembrava il caso di parlarne con qualche esempio. Mi ispirerò ad un ottimo libro a riguardo che si chiama "Claims-based Identity and Access Control" di Microsoft, tra i cui autori c'è anche Vittorio Bertocci.
L'autenticazione Claim-based, o più generalmente la gestione dell'Identity tramite claims è un argomento insolitamente ostico ai molti, che probabilmente già hanno utilizzato o meglio implementato il pattern a loro insaputa.
Partiamo dalla necessità: autenticazione, autorizzazione, varie ed eventuali. Oggi funziona così:

- Il sistema custom chiede user-name e password (authentication)
- Autentica verso la sorgente dati
- Stabilisce i permessi (authorization)
- Permette all'utente di procedere nelle operazioni

Lo svantaggio dei sistemi di autenticazione custom è che sono appunto, custom e che isolano l'applicazione invece di integrarla con altri sistemi (a meno di procedure più o meno complesse per gestire il single sing-on. Inoltre supponiamo di utilizzare Kerberos per l'autenticazione: Kerberos durante il colloquio ci fornisce identità e gruppi, ma se avessi bisogno di altri dati? Per esempio una email?
Allora dovrei certamente andare a fare query su AD (in caso di autenticazione Windows) e comunque dovrei gestire la problematica puntualmente.

Il concetto di Claims è un pò diverso:

- Un soggetto T vuole accedere alla risorsa X
- Il gestore della risorsa X dice al soggetto T che ci vuole una certa autorizzazione per procedere
- Il soggetto T si rivolge a Y chiedendo di erogare un documento valido per X
- Y eroga il documento e T lo presenta al gestore di X per ottenere l'accesso (il gestore potrà così verificare che T abbia tutti i permessi per procedere)

Rispieghiamo tutto con l'esempio del volo aereo.
Una persona che deve prendere una aereo oggi non deve recarsi all'imbarco con una speciale smartcard che lo identifichi, erogata dalla compagnia aerea, ovvero con una coppia di credenziali "custom" per compagnia. Il processo attuale è abbastanza claim-based:

- Una persona si reca ai banchi di check-in dove viene fatta una sorta di identificazione (con ID, Patente, Passaporto)
- A quel punto il personale consegna una carta di imbarco che, oltre a rappresentare l'autorizzazione a salire a bordo contiene molte informazioni
- Il personale al gate può, leggendo sulla carta di imbarco, ottenere dati di interesse riguardo al passeggero

Questo processo è una mappatura quasi 1:1 di ciò che avviene nel modello Claims-based. Ora che sappiamo il meccanismo, ecco una immagine:

parti che possono facilitare la costruzione di queste regole (si possono trovare su http://www.codeplex.com e nel motore di ricerca digitare “Azure”).

Ecco un possibile scenario. Un’applicazione client (destra dell’immagine di Figura 19) invia i claim all’URL pubblico dell’Access Control per accedere a un’applicazione remota. Il servizio di Access Control controlla che i claim in ingresso soddisfino le regole, produce un insieme di claim in output e lo invia in un token sicuro che l’applicazione client può inoltrare a quella remota (parte sinistra dell’immagine).

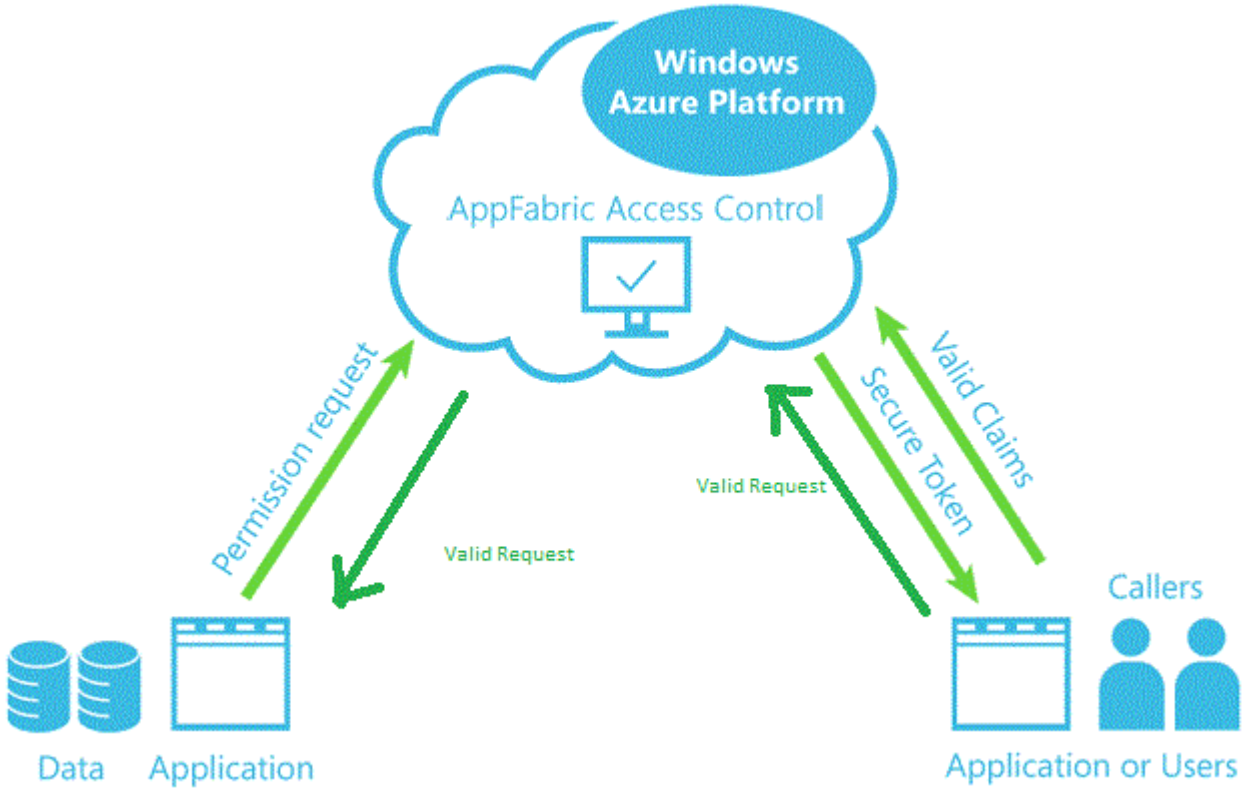
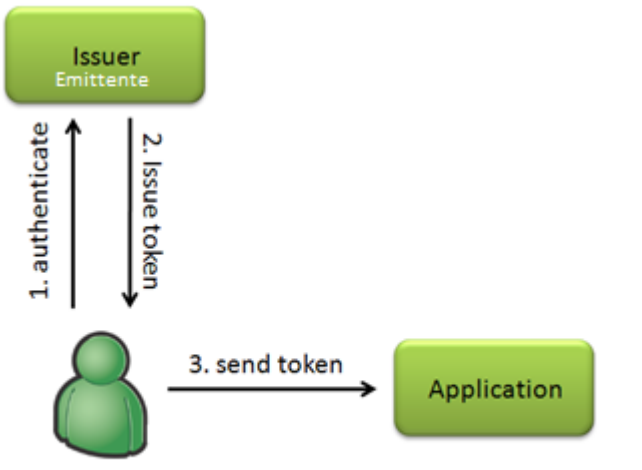


Figura 19 – AppFabric Acces Control Service



Issuers, security tokens, and applications

Il servizio di Acces Control utilizza REST e il Web Resources Acces Protocol (WRAP) per trasportare i claim avanti e indietro tra le applicazioni. Dal momento che questi protocolli sono pubblicamente esposti sul cloud, possono essere usati da qualunque posto e su qualunque piattaforma.

Caching Service

Il servizio di Caching fornisce un servizio di cache distribuito e in-memory per le applicazioni Windows Azure (Figura 20). Il codice di un Hosted Service può usare il servizio di caching per inserire nella cache locale risorse remote.

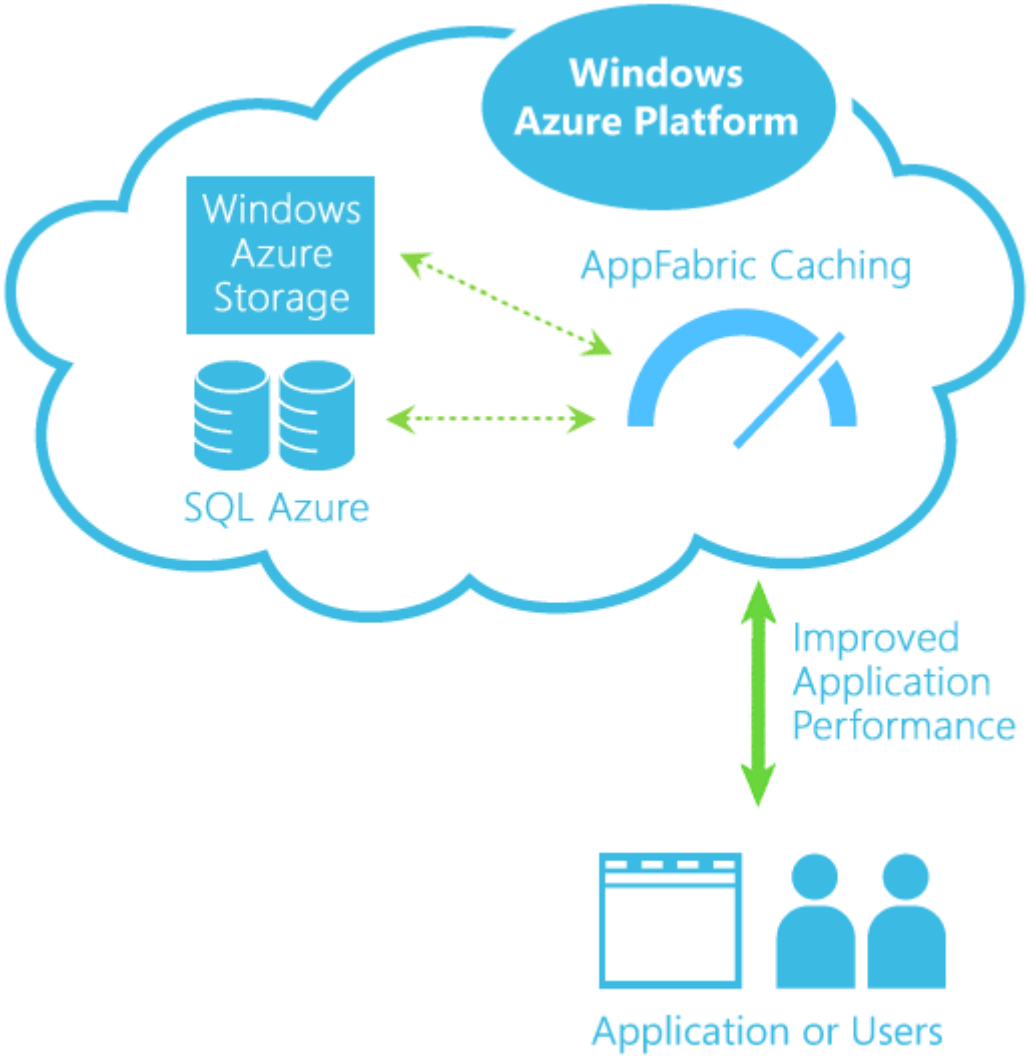


Figura 20 – AppFabric Caching Service

Integration Service

L'idea è quella di facilitare l'integrazione fra applicazioni ospitate sulla piattaforma Windows Azure e soluzioni SaaS di terze parti. L'Integration Service estende anche il servizio di Service Bus per agevolare l'integrazione con le applicazioni Line Of Business (*LOB*) esistenti (Figura 21).

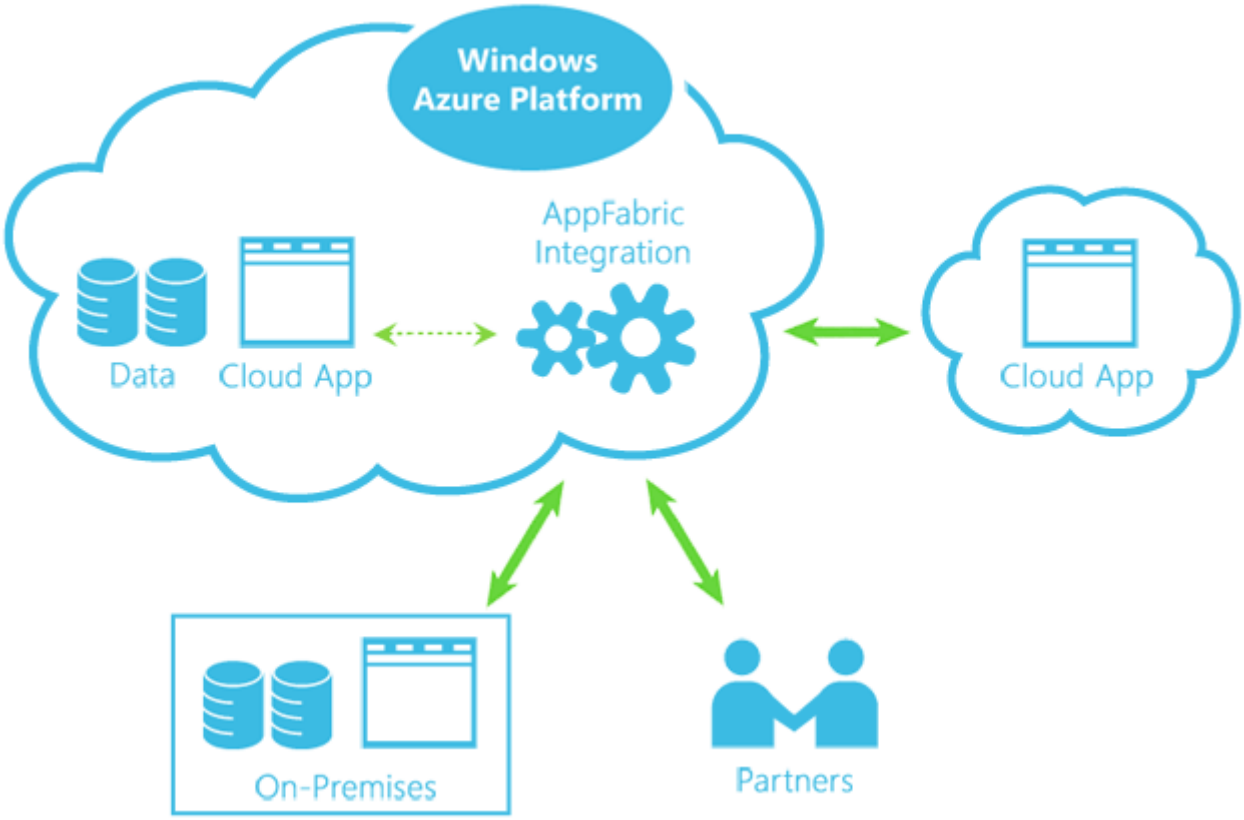


Figura 21 – AppFabric Integration Service

Composite Application Services

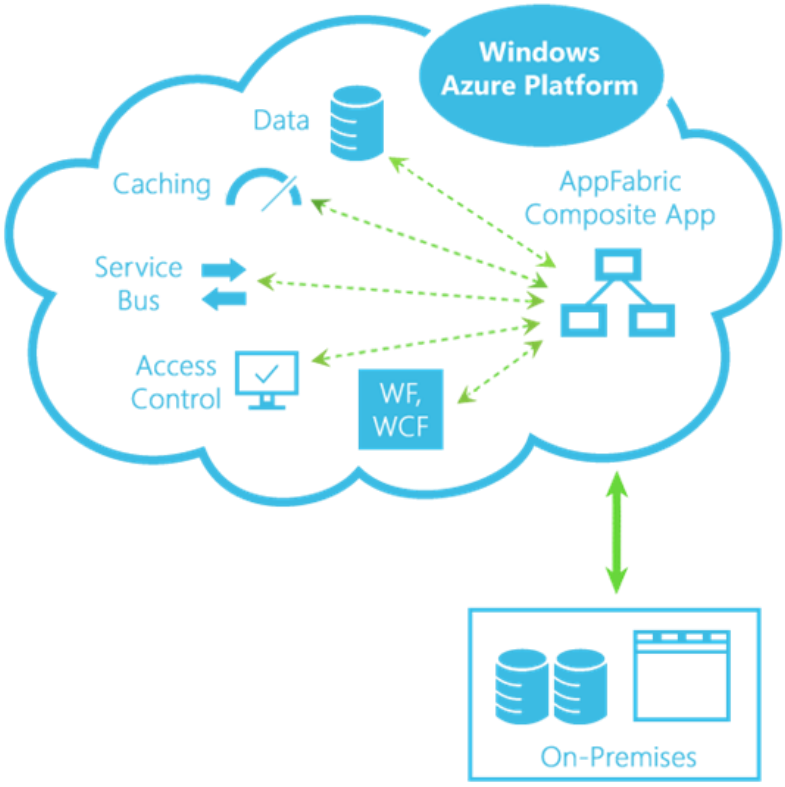


Figura 22 – AppFabric Application Service

Composite Application Service ha l'obiettivo di semplificare la distribuzione (*deployment*) di un'applicazione complessa che utilizza differenti servizi Windows Azure. Le applicazioni composite sono costruite assemblando componenti e servizi sviluppati in proprio o acquistati da terze parti, da distribuire su hardware on-premises e/o macchine nel cloud.

Il servizio consiste in un set di estensioni .NET per comporre applicazioni, un disigner virtuale integrato direttamente in Visual Studio per gestire la creazione e la distribuzione di tali applicazioni e un servizio per più affittuari (*multi-tenant*) che consuma la definizione del Composite Model e automatizza le operazioni di distribuzione.

Database SQL (precedentemente denominato SQL Azure)

L'ultimo, anche se non in ordine di importanza, componente della piattaforma è rappresentato dalla versione cloud di SQL Server, chiamato database SQL di Windows Azure (http://msdn.microsoft.com/it-it/library/windowsazure/ff602419.aspx). Il database è esposto tramite un nome di server virtuale che viene fornito al momento della creazione del server sul portale. È possibile accedere alle funzioni del database da un'applicazione cloud o da una tradizionale applicazione on-premises, semplicemente cambiando la stringa di connessione.

Creare un progetto Web Role

Questo capitolo mira a mettere in pratica il modello PaaS; in particolare cominceremo installando il Software Development Kit con Visual Studio 2012, passeremo in rassegna alcune delle API e degli strumenti di sviluppo in locale per cercare di capire l'insieme dei passaggi necessari per creare un progetto Azure e effettuare la distribuzione di un'applicazione nel *cloud*[40].

Software Development Kit

Per cominciare a sviluppare per Windows Azure, hai bisogno dell'SDK appropriato e dei relativi strumenti. Infatti, dato che Windows Azure è un ambiente multi-piattaforma, è necessario installare l'SDK specifico.

Per gli esempi illustrati in questo testo faremo uso di C# (C Sharp) e Visual Studio 2012 (ma gli stessi concetti si applicano a Visual Studio più datati come 2010 e 2008).

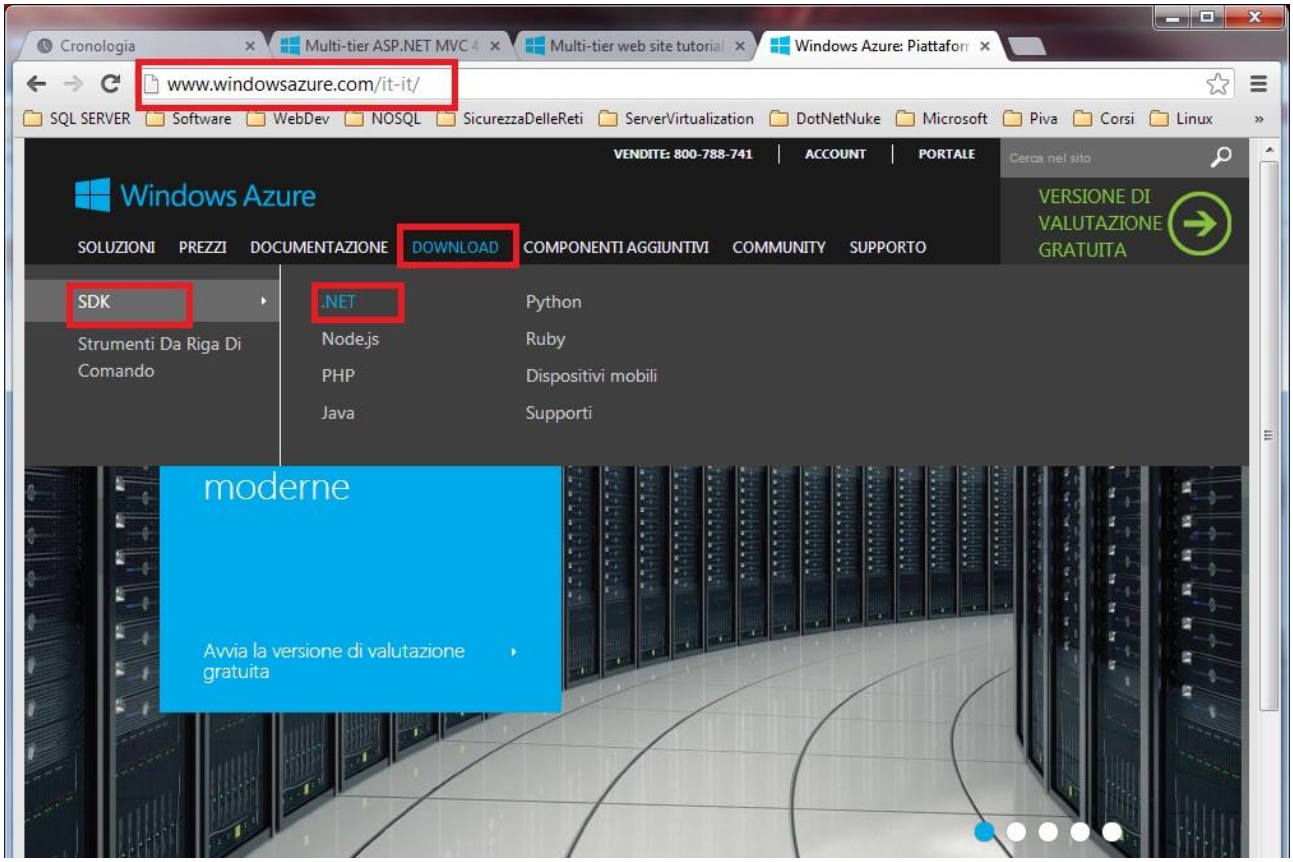


Figura 23 – Home Page Windows Azure (particolare del download dell'SDK)

Windows Azure mette a disposizione anche SDK specifici per altri linguaggi (Figura 23), così come strumenti per altre piattaforme di sviluppo (Figura 24).

[40] Dal momento che URL, nomi e procedure possono cambiare nel tempo, alcuni di questi potrebbero non essere più aggiornati nel momento che ti appresti a leggere questo testo. In questo caso, il mio consiglio è di cercare le informazioni relative al prodotto a partire dalla home page di Windows Azure (http://www.windowsazure.com).

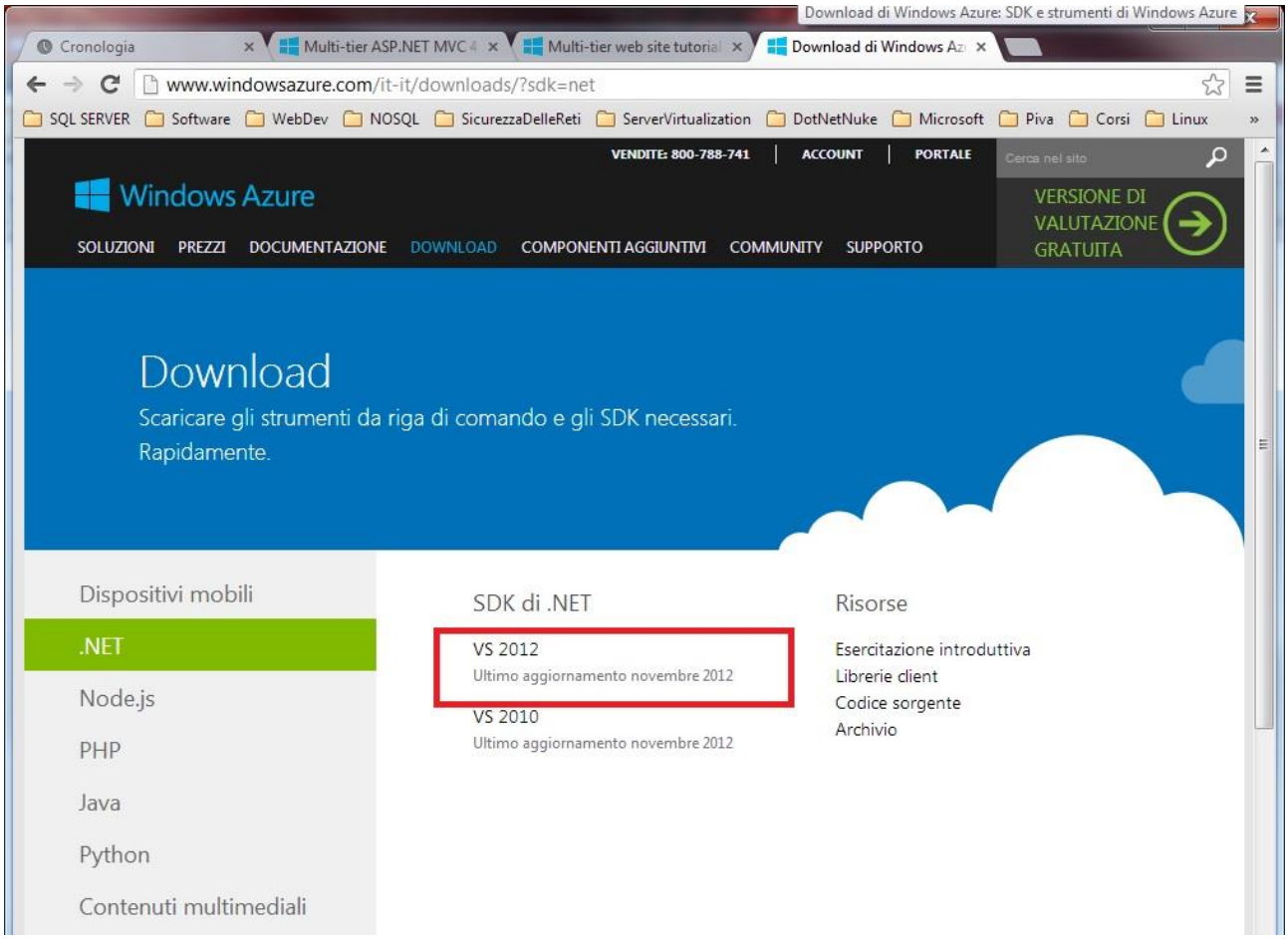


Figura 24 – Download dell'SDK di .NET per Visual Studio 2012

Stiamo scaricando il file *vwdorvs11azurepack.exe*, e si apre l'installazione guidata della piattaforma Web 4.5 (questa è la versione in questo momento… Figura 25)

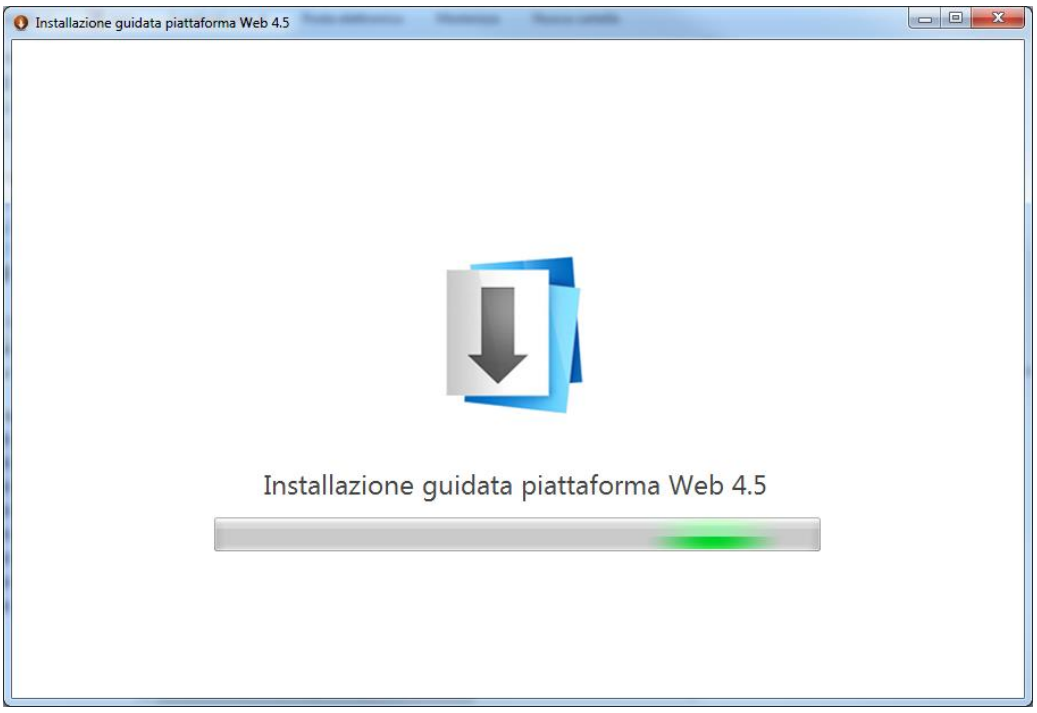


Figura 25 – Installazione del Platform Installer

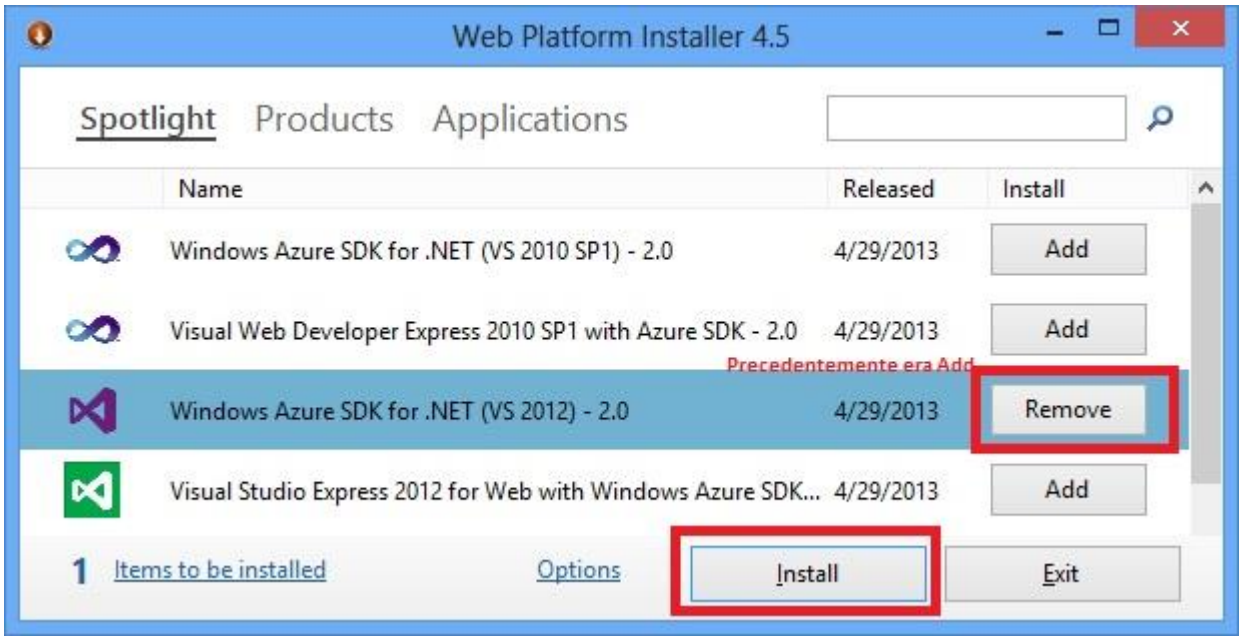


Figura 26 – Platform Installer e l'installazione dell'SDK di Windows Azure

Fai clic su Installa (*Install* - Figura 26) , se stai utilizzando Visual Studio 2010 o Visual Web Developer 2010 Express, installa MVC4 (Figura 28) (consiglio di installare la versione di VS 2012 - Figura 27)

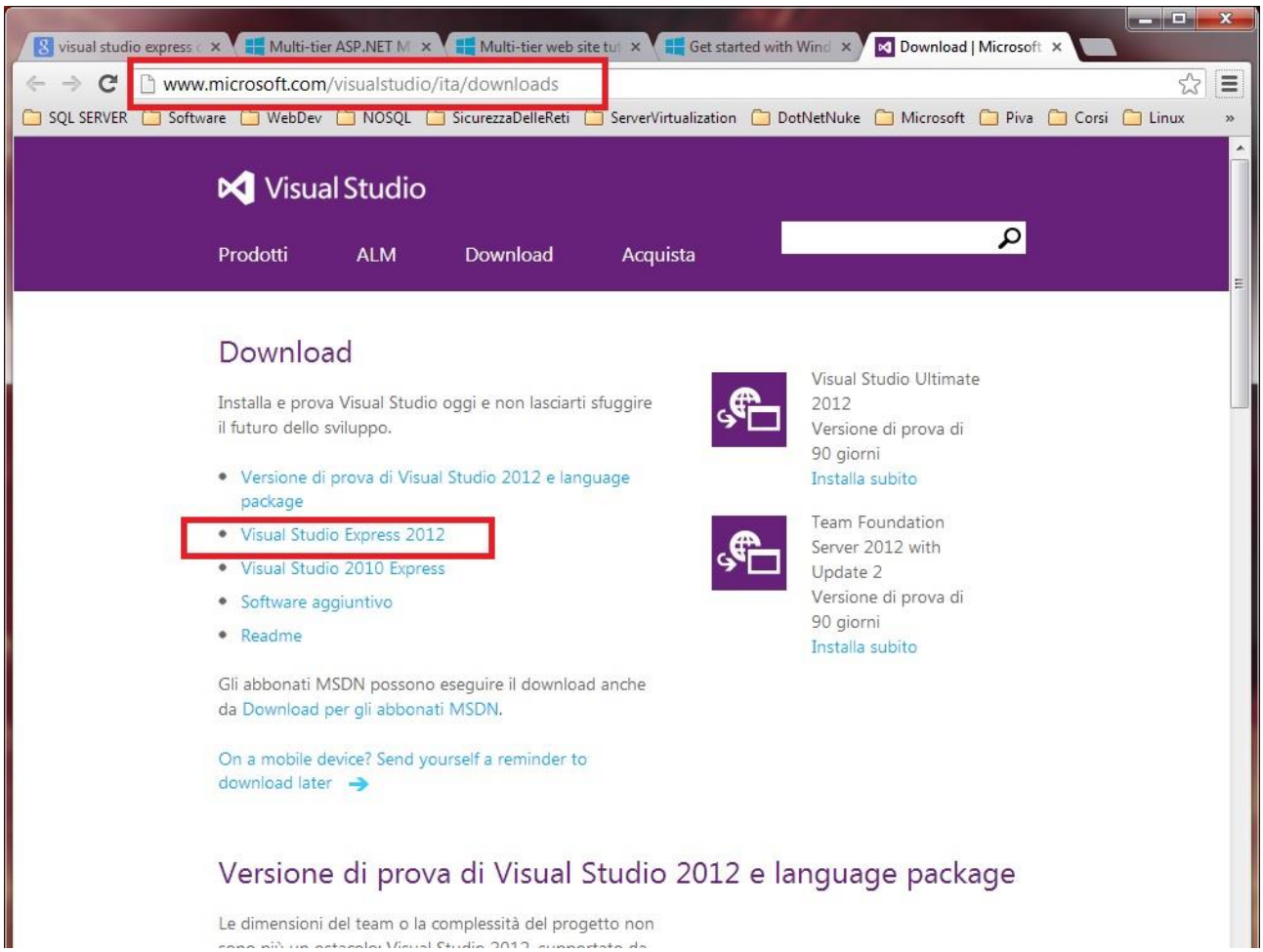


Figura 27 – Sito di riferimento per il download e l'installazione di VS 2012 versione Express (gratuita)

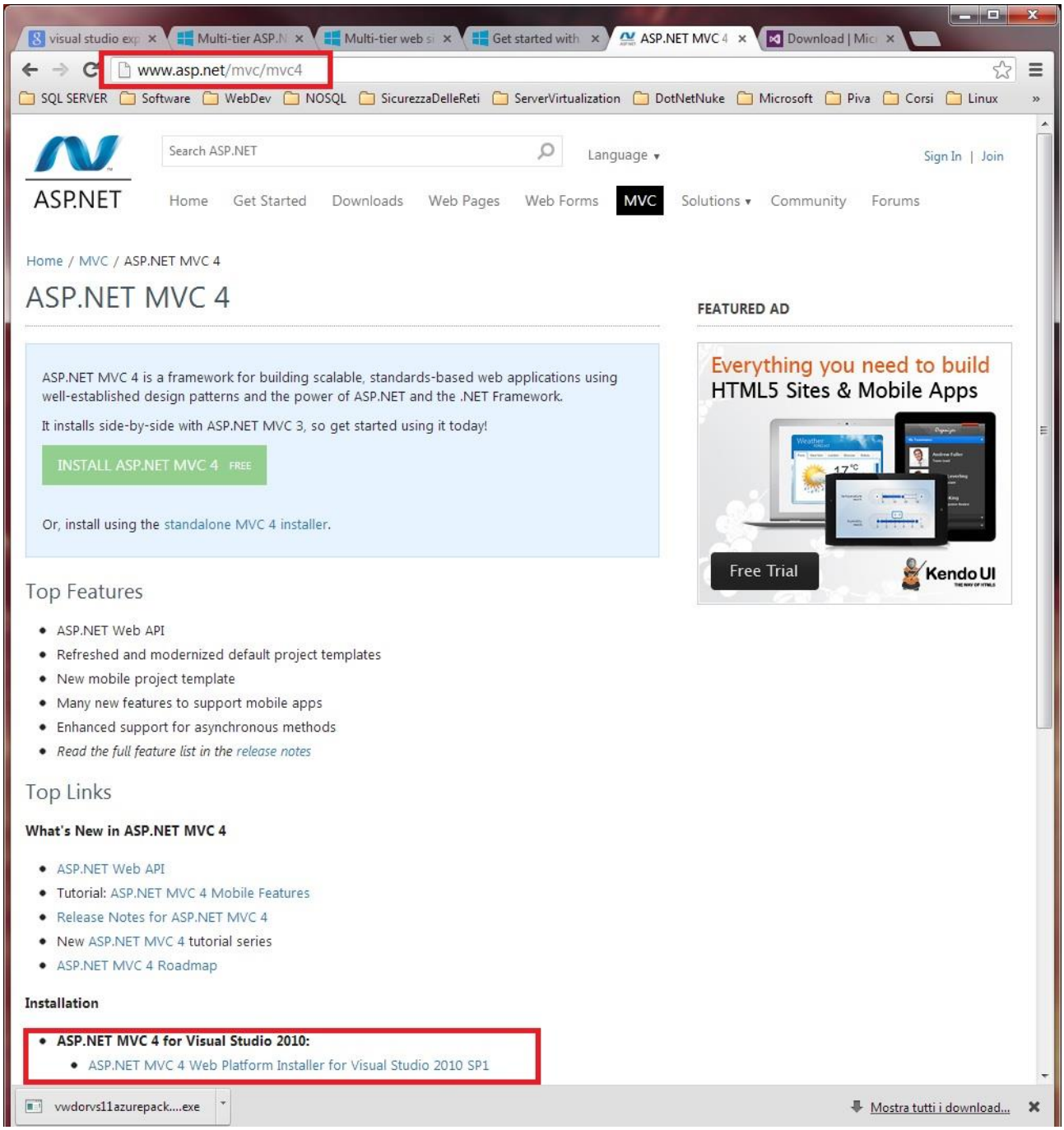


Figura 28 – Sito di riferimento per installare ASP.NET MVC4

Windows Azure Tools per Microsoft Visual Studio

Prima di cominciare l'installazione, accertati di soddisfare I requisiti di sistema e, in particolare che siamo presenti sulla tua macchina l'IIS 7.x e I relative component.

Per fare questo, su Windows 7, Windows 8, Windows Server 2008R2 apri il pannello di controllo, seleziona programmi e funzionalità, di seguito Attivazione e disattivazione delle funzionalità di Windows, espandi la sezione relativa all'IIS, e accertati che i seguenti file siano selezionati: ASP.NET, CGI, WCF HTTP Attivazione e contenuto statico (Figura 29)

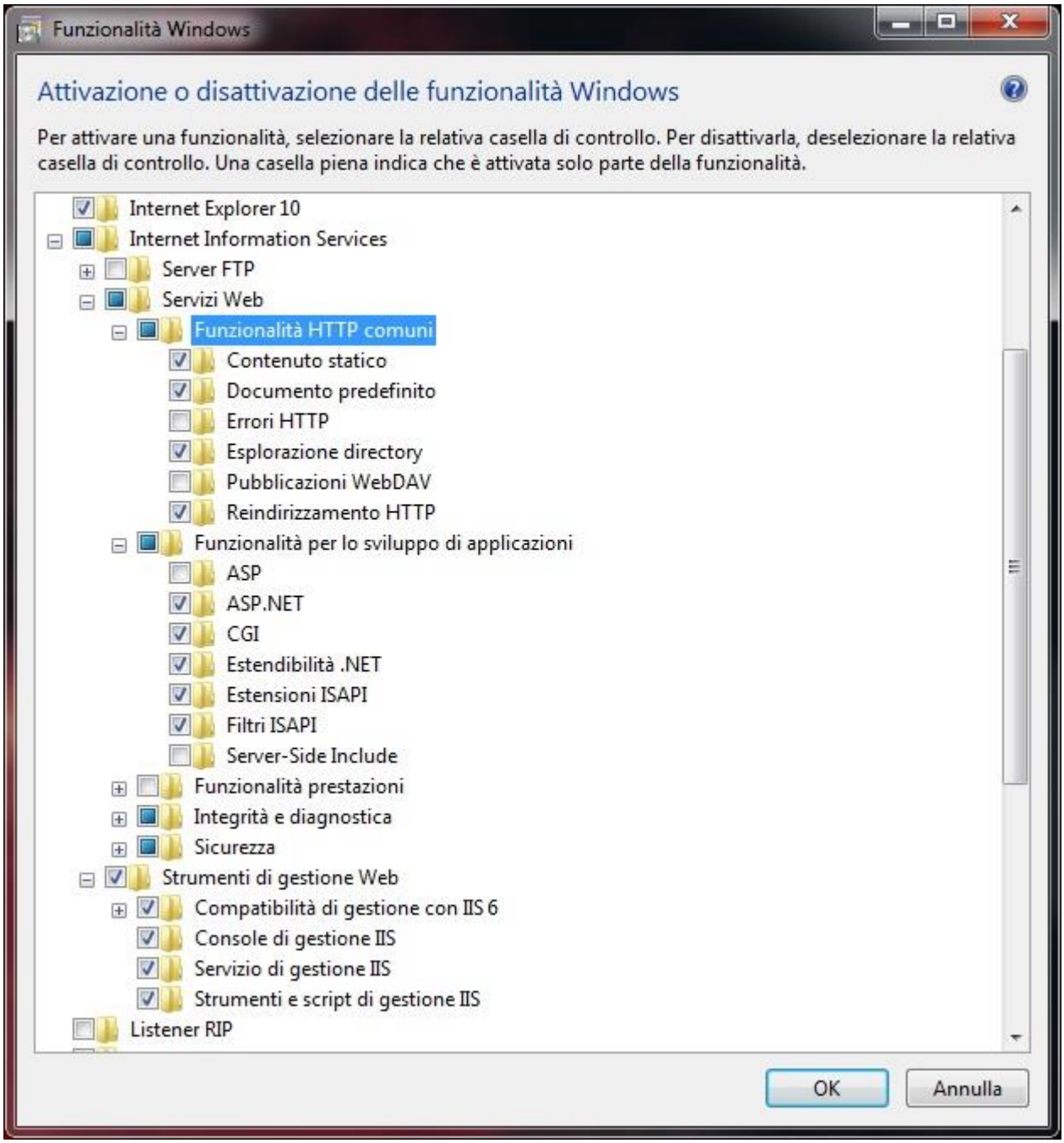


Figura 29 – Attivazione delle funzionalità del sistema operativo per progettare applicazioni cloud

Il Template per il progetto Web Role

Per effettuare il debug e il test delle tue applicazioni nel Compute Emulator (la versione locale che simula il comportamento del vero Windows Azure Fabric), hai bisogno di eseguire Visual Studio con i privilegi di Amministratore.

Il processo d'installazione dell'SDK e dei relativi strumenti porta con sé alcuni nuovi template e wizard per facilitare la creazione di un progetto Azure. Come vedrai nella prossima figura, all'interno della sezione dedicata ai progetti in Visual C# o Visual Basic troverai una nuova sezione denominata Cloud, che rappresenta il punto d'ingresso per qualunque progetto Azure.

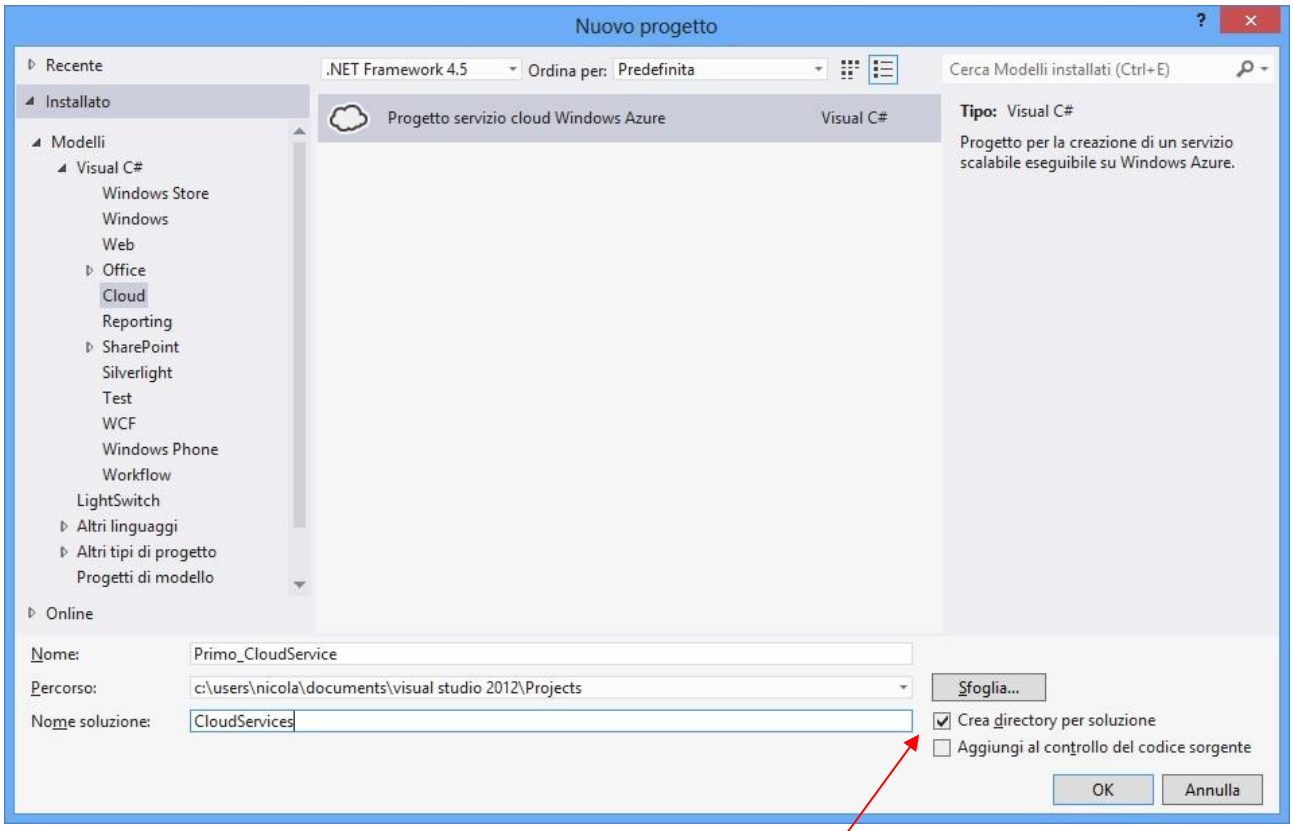


Figura 30 – Creazione progetto cloud

In Visual Studio 2012

File -> Nuovo Progetto -> Cloud -> seleziona la versione del Framework -> imporre un nome al tuo nuovo progetto, quindi scegli una directory del File System ed il nome della soluzione. Al termine clicca su OK.

A questo punto, Visual Studio 2012 (per impostazione) crea una directory per ospitare la soluzione nonché il progetto basato sul *template* selezionato. Nel caso di un progetto Azure, invece, è necessario un passaggio ulteriore, dato che Windows Azure supporta diversi tipi di progetto. Una volta selezionato il *template* per il cloud, infatti, dovrai scegliere che tipo di progetto intendi creare (Figura 31).

Tipo di progetto	Descrizione
ASP.NET Web Role	Classico progetto ASP.NET con una pagina Default.aspx che può essere caricata su Windows Azure.
ASP.NET MVC 3 o ASP.NET MVC 4 Web Role	Progetto ASP.NET basato su pattern MVC 3 o MVC 4 introdotti in .NET 4.0 e .NET 4.5
WCF service Web Role	Progetto ASP.NET basato su un template per un progetto di tipo WCF Service
Worker Role e Worker Role Cache	Progetti che eseguono processi di supporto al Web Role[41]
Silverlight Enterprise Application	Applicazione Line-of-business di WCF RIA Service

Tabella 2

[41] http://www.cloudtalk.it/it/c_000005vm/introduzione-a-microsoft-windows-azure

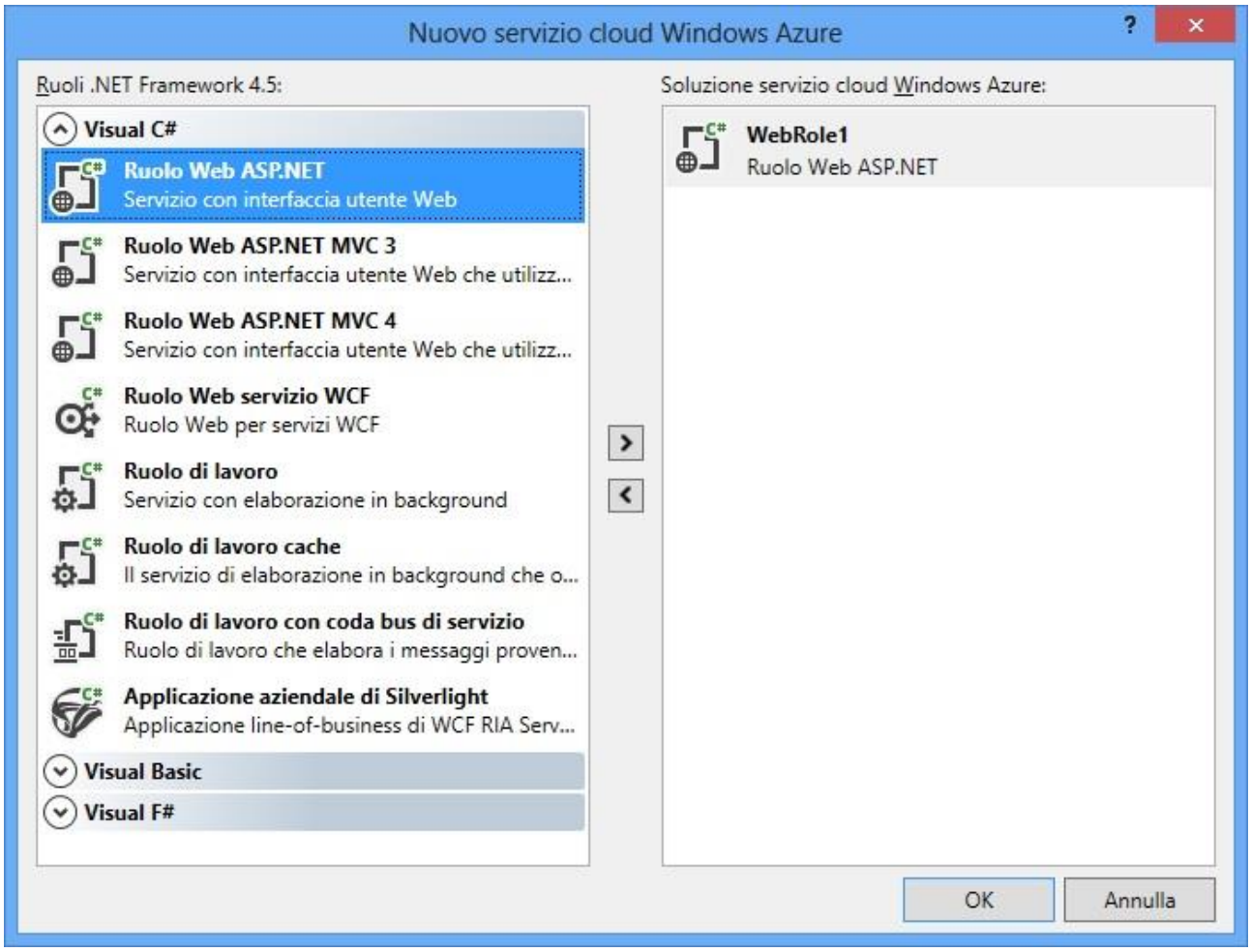


Figura 31 – Aggiunta del WebRole alla soluzione

Dato che questo è il nostro primo esempio ci limiteremo a creare una semplice ASP.NET Web Application. Selezioniamo Ruolo Web ASP.NET (ASP.NET Web Role) e, cliccando sul tasto che ha la freccia orientata verso destra (oppure doppio clic sul ruolo/role), in Ruolo Web ASP.NET comparirà nel pannello di destra denominato Soluzione servizio cloud Windows Azure (traduzione pessima di Windows Azure Solution). È possibile rinominare il progetto Ruolo Web ASP.NET, selezionandolo nel pannello di destra e quindi cliccando sull'icona a forma di matita: il nome inserito diventerà il nome del progetto Ruolo Web ASP.NET.

Aperto l'IDE[42], nella finestra di Esplora Soluzioni, abbiamo due progetti: uno è il servizio Cloud e uno è il Web Role. Eccezion fatta per il file WebRole.cs tutto il resto è identico ad una classica applicazione web ASP.Net (Figura 32).

Il Listato 1 seguente mostra la pagina di default modificata, con il testo di default rimosso e con una nuova label denominata *TimeLabel*.

[42] In informatica un **Integrated Development Environment** (**IDE**), in italiano **ambiente di sviluppo integrato** è un software che, in fase di programmazione, aiuta i programmatori nello sviluppo di codice sorgente di un programma.

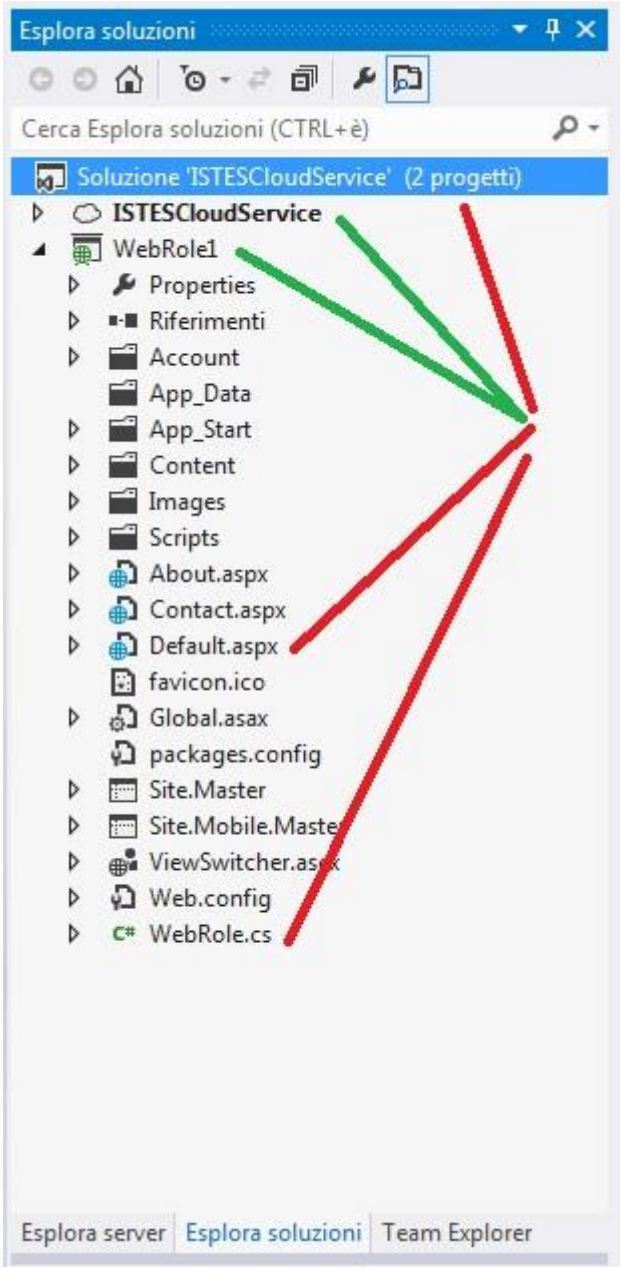


Figura 32 – La soluzione con due progetti

Il Listato 1 non contiene alcunché di particolare. La pagina usa una *master page*, come suggerito dallo stesso *template* e contiene un normale controllo ASP.NET per renderizzare il contenuto. Non troviamo niente di speciale nel *code behind* della pagina, dal momento che questa non utilizza nessuna delle caratteristiche di Azure.

Il code behind (Listato 2) opera un override del metodo *OnPreRender* per assegnare l'ora corrente alla Label (*TimeLabel*) contenuta nel Listato 1.

Avremmo potuto ottenere lo stesso risultato usando l'evento *Load* o il metodo *OnLoad* o una qualsiasi altra tecnica ASP.NET valida.

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master"
     AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebRole1._Default" %>

<asp:Content ID="HeaderContent" runat="server"
     ContentPlaceHolderID="HeadContent"></asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <p><asp:Label ID="TimeLabel" runat="server" /></p>
</asp:Content>
```

Listato 1 – Codice della pagina Default.aspx del progetto WebRole1 (soluzione ISTESCloudService)

```
namespace WebRole1
{
    using System;
    using System.Web.UI;

    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            TimeLabel.Text = DateTime.Now.ToString();
        }
    }
}
```

Listato 2 – Codice (code behind) della pagina Default.aspx.cs del progetto WebRole1

Esegui il progetto cliccando con il tasto destro sul file Default.aspx e seleziona Visualizza nel Browser (Figura 33) (non eseguire il progetto con F5 o Ctrl+F5).

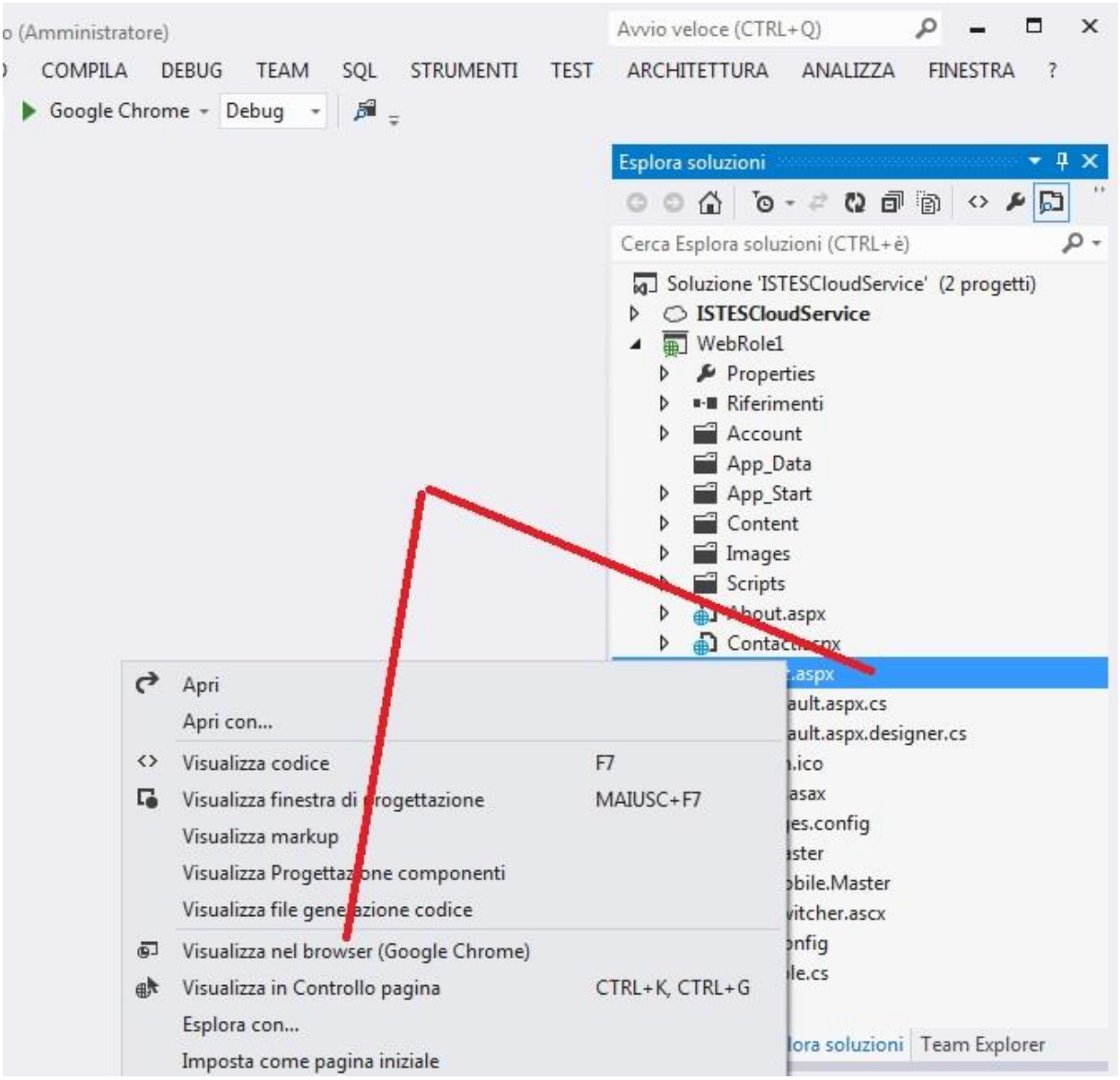


Figura 33 – Visualizzare nel browser la pagina di Default.aspx

A questo punto, abbiamo un'applicazione ASP.NET che gira sull'ASP.NET Development Server, una classica pagina Default.aspx (Figura 34), un altrettanto classico file Web.config e un tipico file Global.asax. Ancora una volta non è richiesto alcunché di speciale per creare un progetto Windows Azure in ASP.NET. È possibile utilizzare qualsiasi controllo web, HTML o mobile, collegarli ad un data source, come faresti di solito in ASP.NET e usare il Web.config per configurare l'autenticazione, le autorizzazioni, il tracing, il caching, l'HTTP Handler, i moduli e così via.

La magia di Windows Azure non risiede in quello che puoi o non puoi fare in un progetto, ma nella capacità di sfruttare le caratteristiche di deployment e di scalabilità della piattaforma.

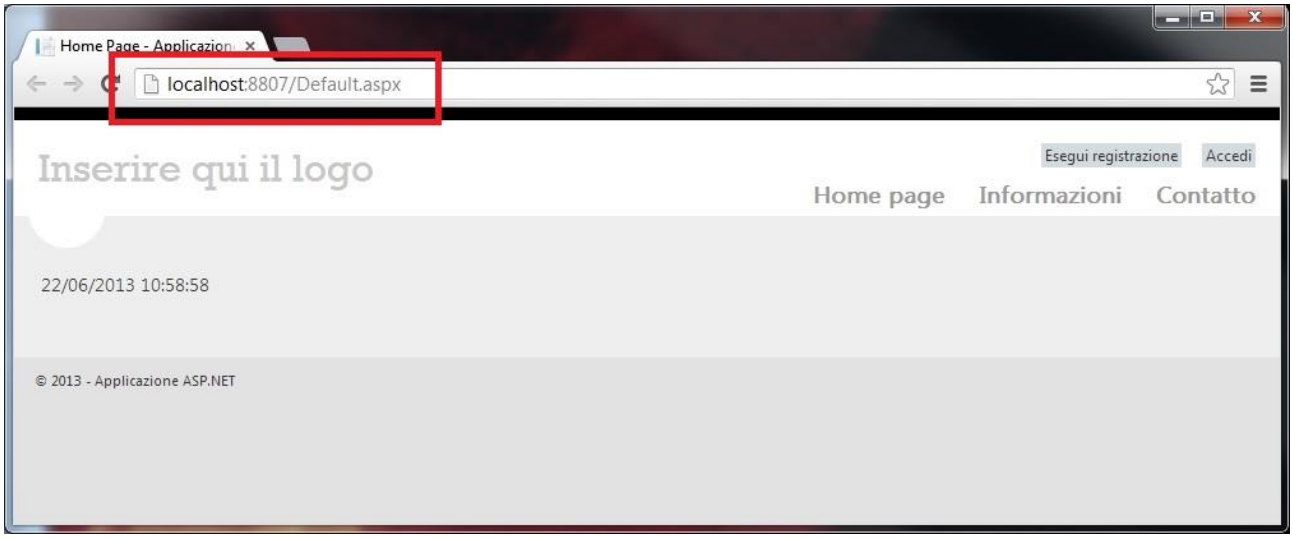


Figura 34 – Particolare dell'indirizzo e della porta, della pagina di Default.aspx

Il progetto *Cloud*

In questa sezione esamineremo il progetto cloud contenuto all'interno della soluzione ed esporremo alcune delle API per il cloud che puoi usare per le tue pagine.

La prima cosa da notare è la presenza di un progetto cloud all'interno della tua soluzione. Grazie alla finestra delle proprietà (Figura 35) puoi verificare che il progetto presenta un file con estensione .ccproj ed è collocata in una directory separata all'interno della soluzione.

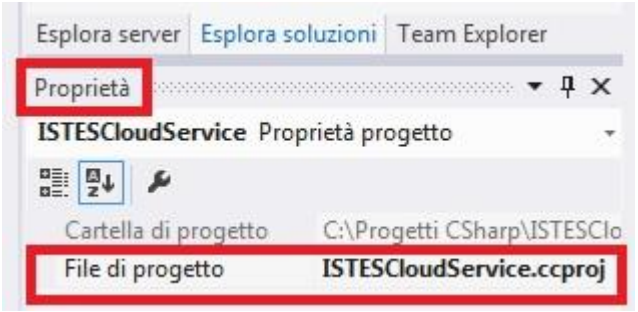


Figura 35 – Proprietà del progetto cloud

Visual Studio rappresenta il contenuto del progetto cloud come un elemento *root* con lo stesso nome del progetto, una cartella Ruoli (*Roles*) e tre elementi denominati ServiceConfiguration.Cloud.cscfg, ServiceConfiguration.Local.cscfg e ServiceDefinition.csdef (Figura 36).

Per testare le API di Windows Azure o il comportamento reale della tua applicazione nel cloud, devi eseguire quest'ultima nel Compute Emulator, e, per fare ciò, hai bisogno dei privilegi di amministratore.

Per il momento, tuttavia, l'applicazione viene eseguita in una infrastruttura cloud simulata, nella quale Visual Studio ha effettuato il deployment della soluzione che processa l'URL richiesta (Figura 37). Infatti stai utilizzando la porta 81 e non la 8807 come nel caso precedente (Figura 34 – questa corrisponde alla porta utilizzata da ASP.NET Development Server sulla mia macchina, sulla tua potrebbe essere diversa).

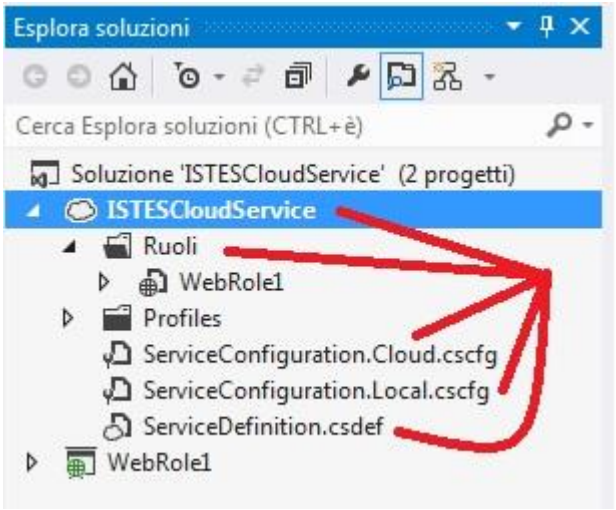


Figura 36 – Elementi del progetto cloud

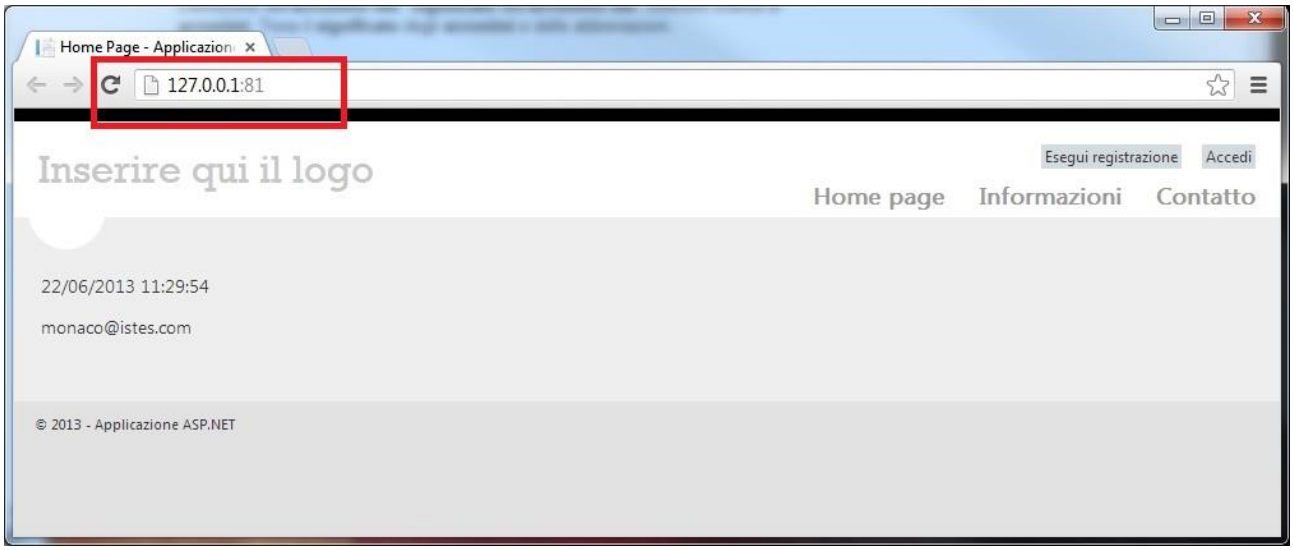


Figura 37 – URL processata (in locale) dal progetto cloud

La differenza non sta semplicemente nella porta utilizzata, ma nel complessivo ambiente in cui l'applicazione è in esecuzione. Vai nell'Area Notifiche della *taskbar di Windows* (io, mentre sto scrivendo il presente testo, mi sto dividendo tra un Windows 7 Professional 64 bit e un Windows 8 Professional a 64 bit) e clicca sull'icona di Windows Azure, compare un messaggio che conferma

Figura 38 – Particolare di Windows Azure Emulator

l'avvio del Compute e dello Storage Emulator. Se clicchi tasto destro e selezioni Show Compute Emulator UI vedrai una cosa simile alla Figura 39.

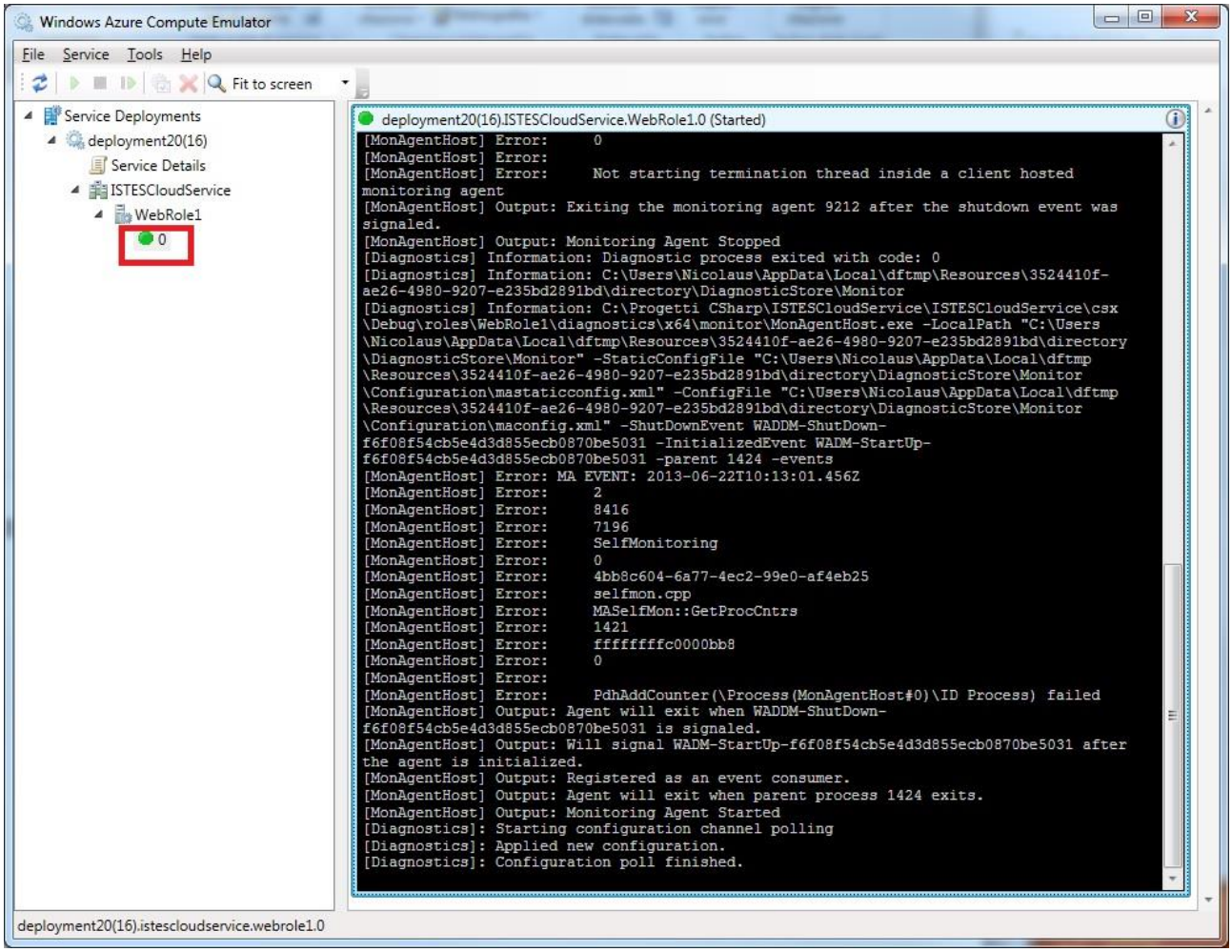


Figura 39 – Visualizzazione dell'istanza nel Windows Azure Compute Emulator

L'albero nel pannello di sinistra del Compute Emulator (Figura 39) fornisce informazioni in merito al deployment locale del progetto demo. Il numero accanto alla voce deployment (deployment20) identifica in modo univoco il singolo deployment (nel mio caso ne ho fatti 20 da quando ho installato l'SDK di Windows Azure). Nell'interfaccia utente puoi vedere che il progetto denominato *ISTESCloudService* contiene un ruolo chiamato *Web Role1*, che ha soltanto un'istanza, denominata semplicemente "0". Se selezioni l'istanza nel pannello di destra noterai numerosi messaggi di trace. Questo pannello riporta tutti gli eventi sollevati dal *runtime*, insieme con alcune utili informazioni diagnostiche.

La voce *Service Details* nel pannello di sinistra è utile per scoprire a *runtime* come l'emulatore ha configurato il servizio, la porta usata dai Web Role e altre informazioni sul servizio medesimo.

Adesso abbiamo chiarito la differenza tra esecuzione all'interno di ASP.NET Development Server e quella nell'ambiente simulato del Compute Emulator.

Un'ultima cosa.

Chiudiamo il browser e ritorniamo a Visual Studio. Aumentiamo il numero di istanze del ruolo Web Role1, assegnando il valore 5 nel campo Conteggio istanze (Instances Count). Ripremiamo F5 per eseguire il progetto ancora una volta (Figura 40).

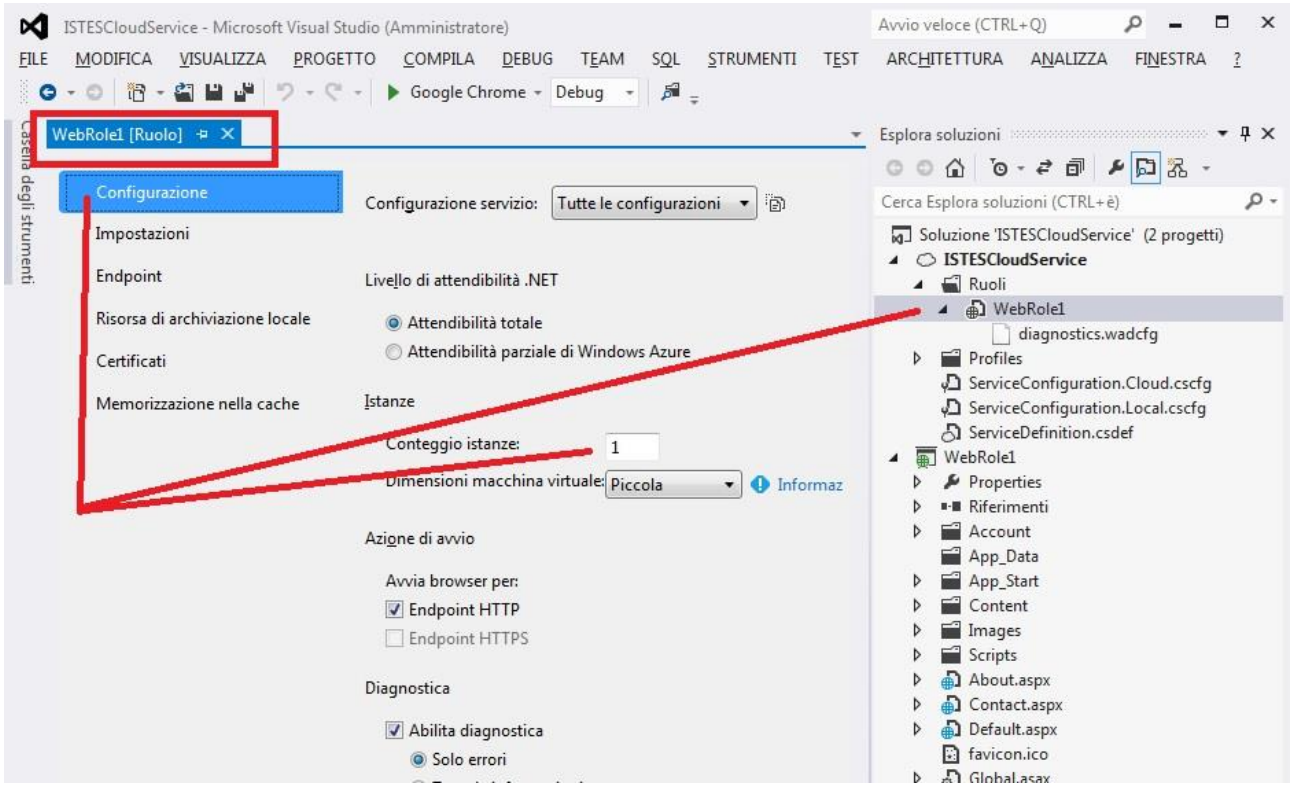


Figura 40 – Aggiornamento delle iswtanze del WebRole

Nel browser continuerai a vedere la stessa identica pagina, ma se adesso apri l'interfaccia utente del Compute Emulator vedrai un cosa simile alla seguente figura (Figura 41)

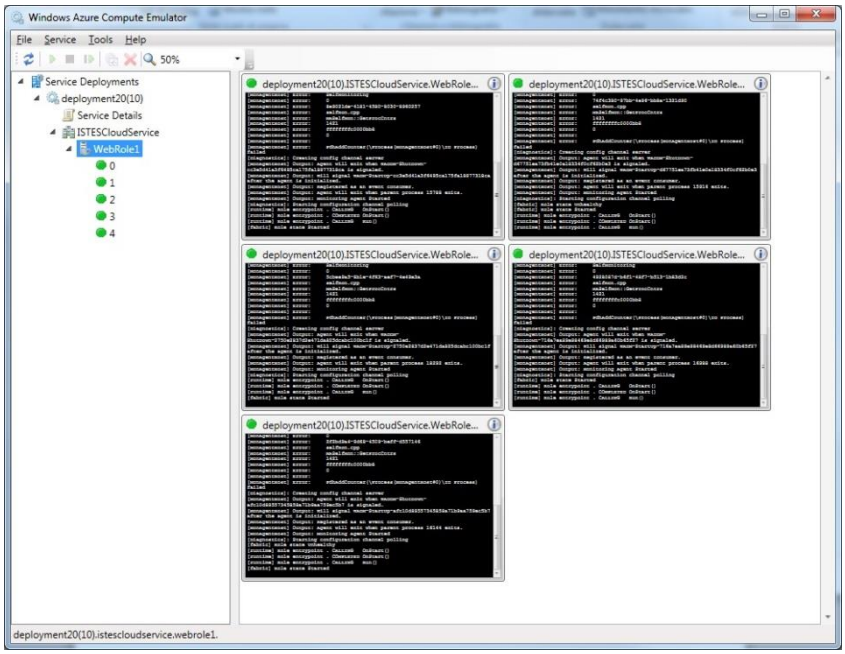


Figura 41 – Visualizzazione delle istanze de lWebRole nel Windows Azure Compute Emulator

In locale, il Compute Emulator simula le istanze utilizzando i processi. Infatti se apri *Task Manager* di Windows, vedrai 5 processi denominati "WaIISHost.exe", dove "Wa" sta per Windows Azure e "IISHost" significa che il processo è in esecuzione all'interno di IIS. Nel tuo Task Manager dovresti vedere qualcosa del genere (Figura 42):

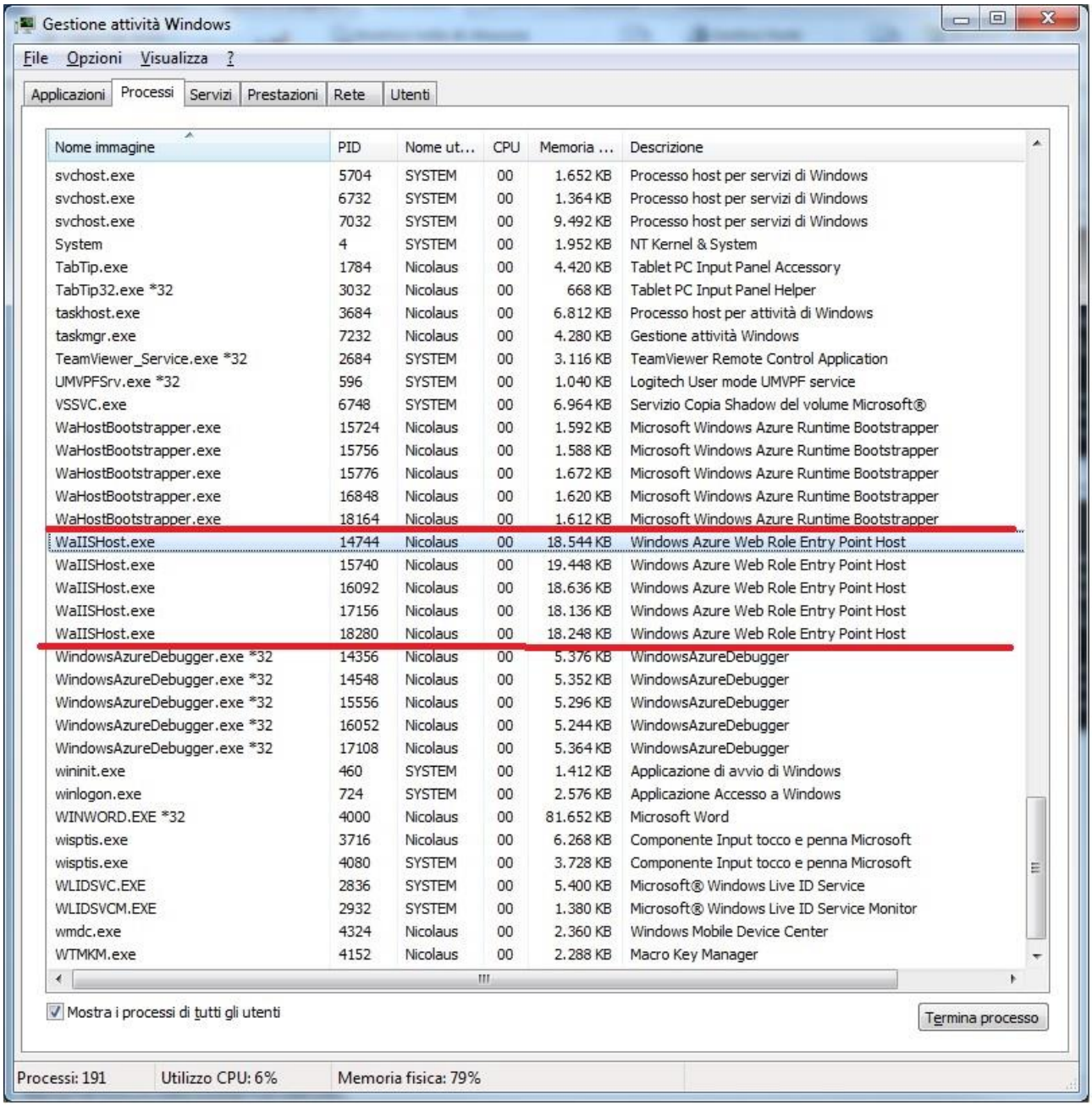


Figura 42 – Task Manager e le istanze gestite come processi dal WACE

I processi si possono terminare ma il Compute Emulator si limiterà ad riavviarli nuovamente ogni volta che ciò si renda necessario, così come farebbe Windows Azure con i tuoi Web Role in esecuzione sul cloud.

Effettuare il *deployment* su Windows Azure

Clicca col tasto destro del mouse sul progetto e seleziona il menù Pubblica (*publish*) per avviare il processo di pubblicazione. Il tuo Windows Explorer aprirà la sottodirectiory app.publish contenuta nella directory bin/Release del progetto cloud; il percorso dovrà essere qualcosa del tipo:

C:\directorydilavoro\nomesoluzione\nomeprogettocloud\bin\release\app.publish

Visual Studio crea due file in questa cartella: il file con l'estensione *.cspkg* rappresenta il package del servizio cloud e contiene tutti i *role* ospitati nella soluzione, mentre il file con l'estensione *.cscfg* è il file di configurazione.

La procedura di pubblicazione apre la seguente Figura 43.

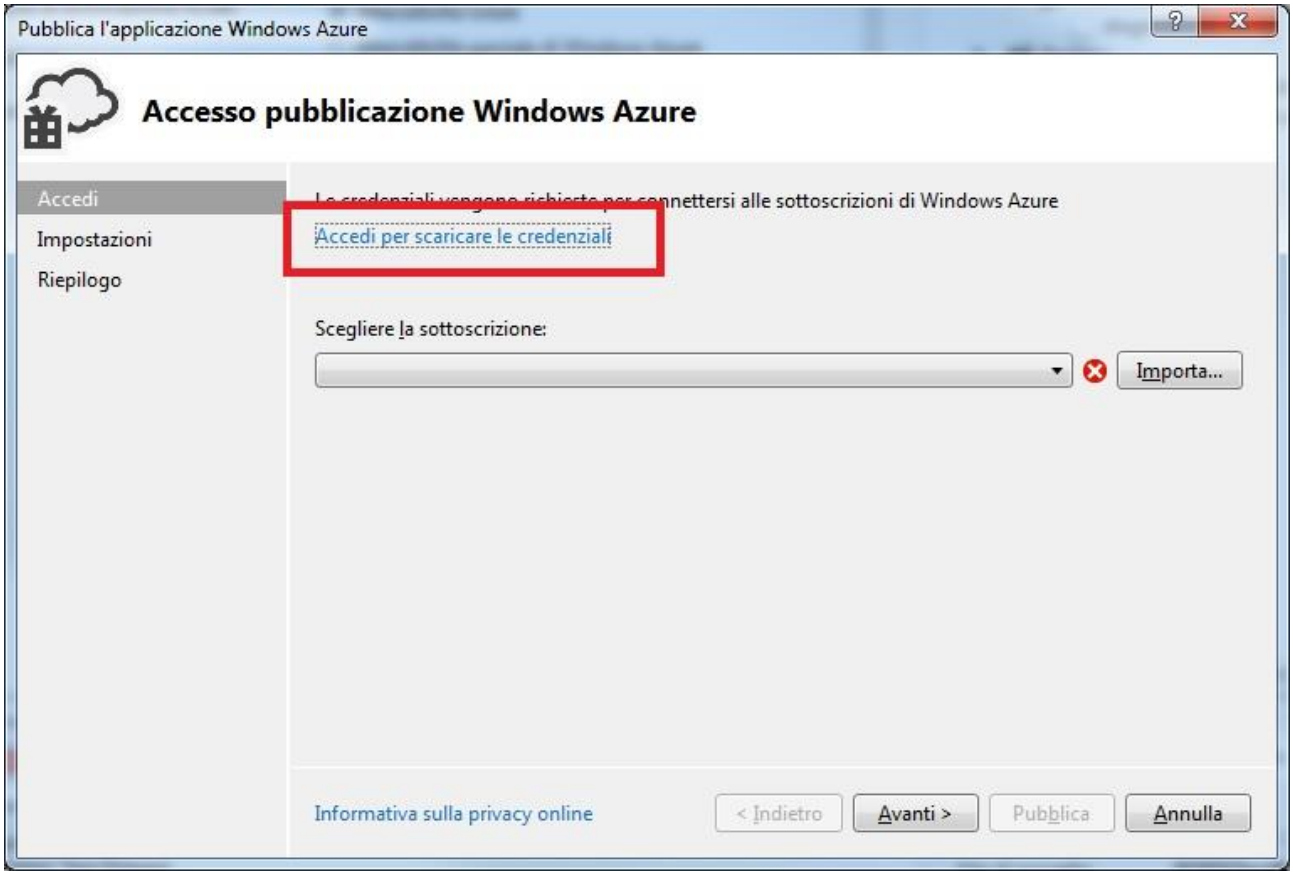


Figura 43 – Deployment di una soluzione passo 1

Bisogna fare il login con l'account Live ID di Microsoft e si viene proiettati in una pagina che ha nell'intestazione "È in corso la generazione del file di sottoscrizione" …

Nella schermata di accesso alla procedura guidata fare clic sul pulsante Importa.

Caricare le credenziali e le informazioni della sottoscrizione badando bene che questo file contiene un certificato che fungerà da credenziale per l'amministrazione di tutti gli aspetti delle sottoscrizioni e dei servizi correlati. Archiviare il file in una posizione sicura o eliminarlo dopo averlo usato.

Scegliere la sottoscrizione appropriata.

Di seguito, nelle impostazioni, selezionare il servizio cloud, l'ambiente di prova (*staging*) o di produzione, la configurazione della compilazione se in *Debug* o in *Release* e lasciare la scelta del servizio Cloud nell'ultima opzione (Figura 44).

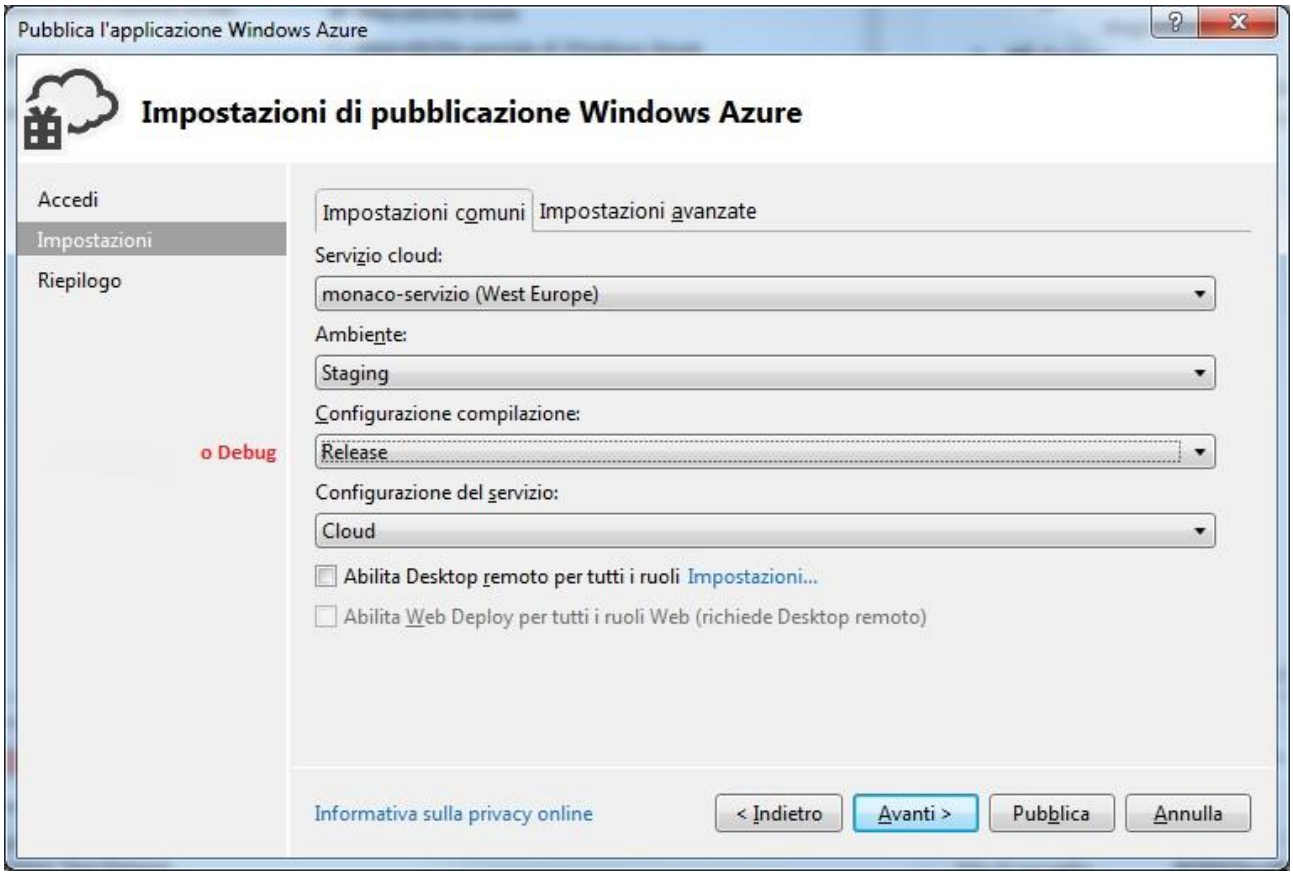


Figura 44 - – Deployment di una soluzione passo 2

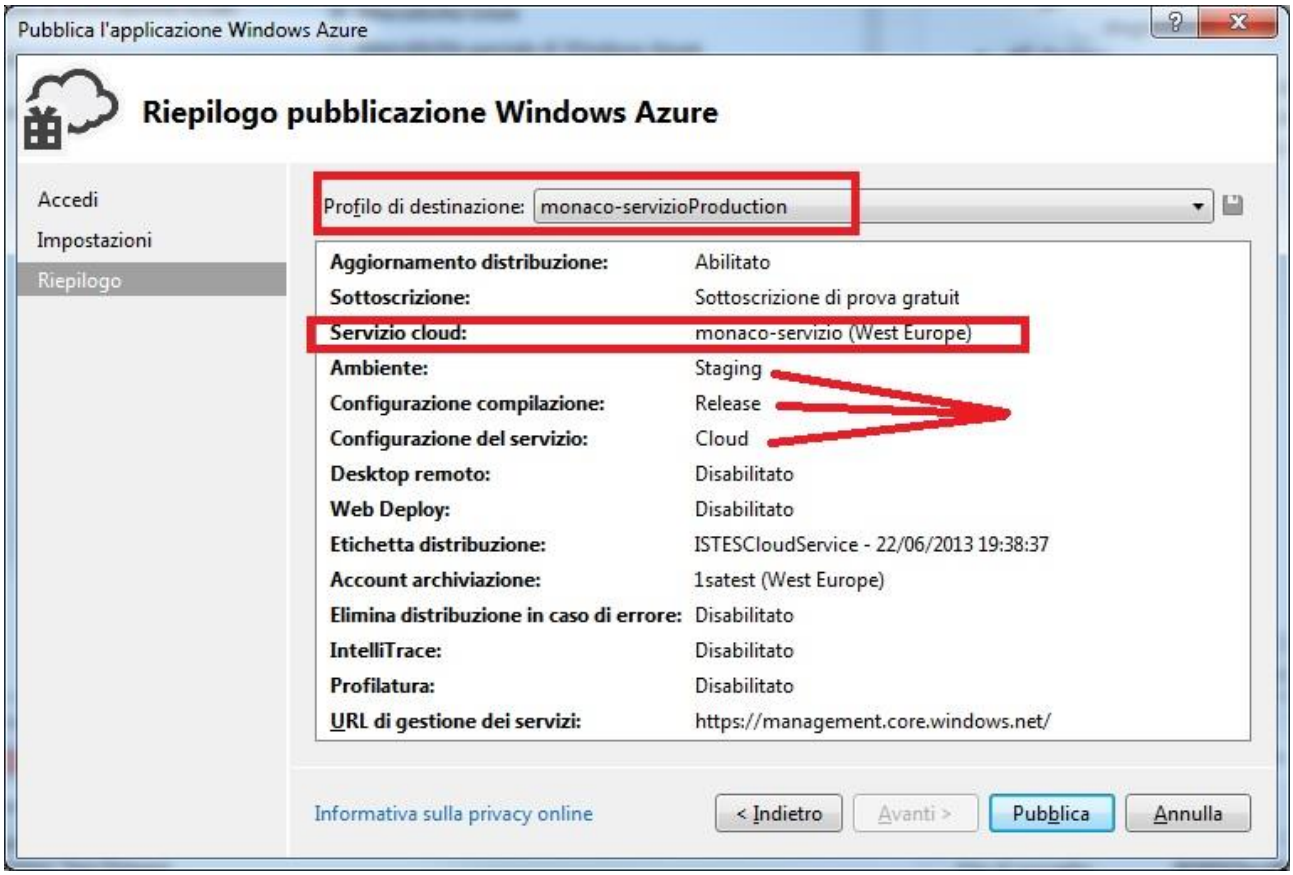


Figura 45 - – Deployment di una soluzione passo 3

Bisogna ricordarsi che, a questo punto, si è cominciato a pagare: ogni volta quindi che si effettua un *deployment* di un *hosted service*, il tassametro comincerà a girare.

Dalla schermata principale si possono compiere diversi tipi di operazioni sui servizi caricati. Prima di tutto si dovrebbe testare l'URL scelto nel *wizard* di creazione di un nuovo servizio (*Create a New Service*).

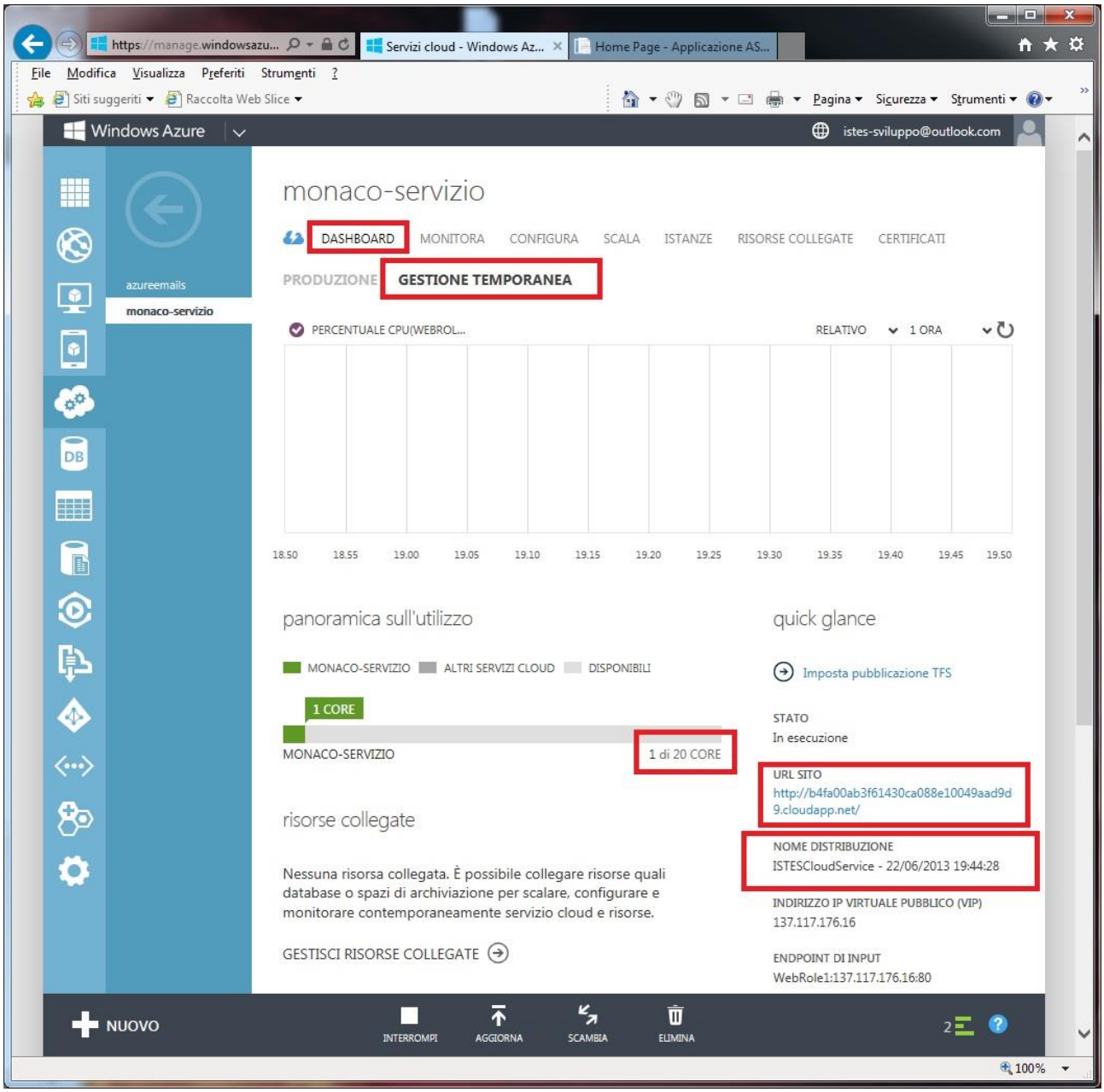


Figura 46 – Servizio hostato

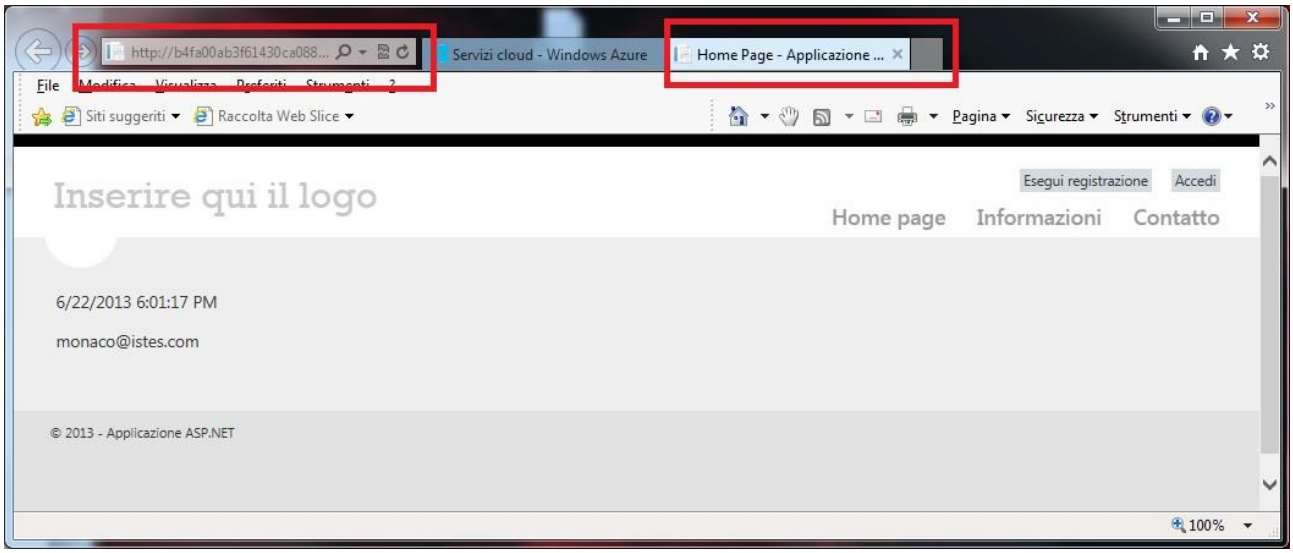


Figura 47 – Particolare dell'URL del sito

Si può anche aggiornare l'ambiente di produzione o di *staging* in qualunque momento semplicemente caricando una nuova versione del package. Adesso che abbiamo capito come funziona la procedura di deployment manuale, nella prossima sezione conosceremo delle tecniche più efficienti per raggiungere lo stesso risultato.

Configurazione e upgrade

La magia che sta dietro un ambiente di tipo PaaS e in particolare dietro Windows Azure è la completa astrazione rispetto alle proprietà fisiche del file system, alle directory virtuali e al load balancer.

Come abbiamo visto è possibile caricare un package usando Visual Studio (o Eclipse) o sostituire quello presente sul cloud con una versione più recente in qualunque istante (Figura 48 - Figura 49). Questa operazione, per quanto facile, solleva tuttavia un problema: che cosa accade se la versione più recente non funziona nel cloud?

Figura 48 – Aggiornamento del servizio passo 1

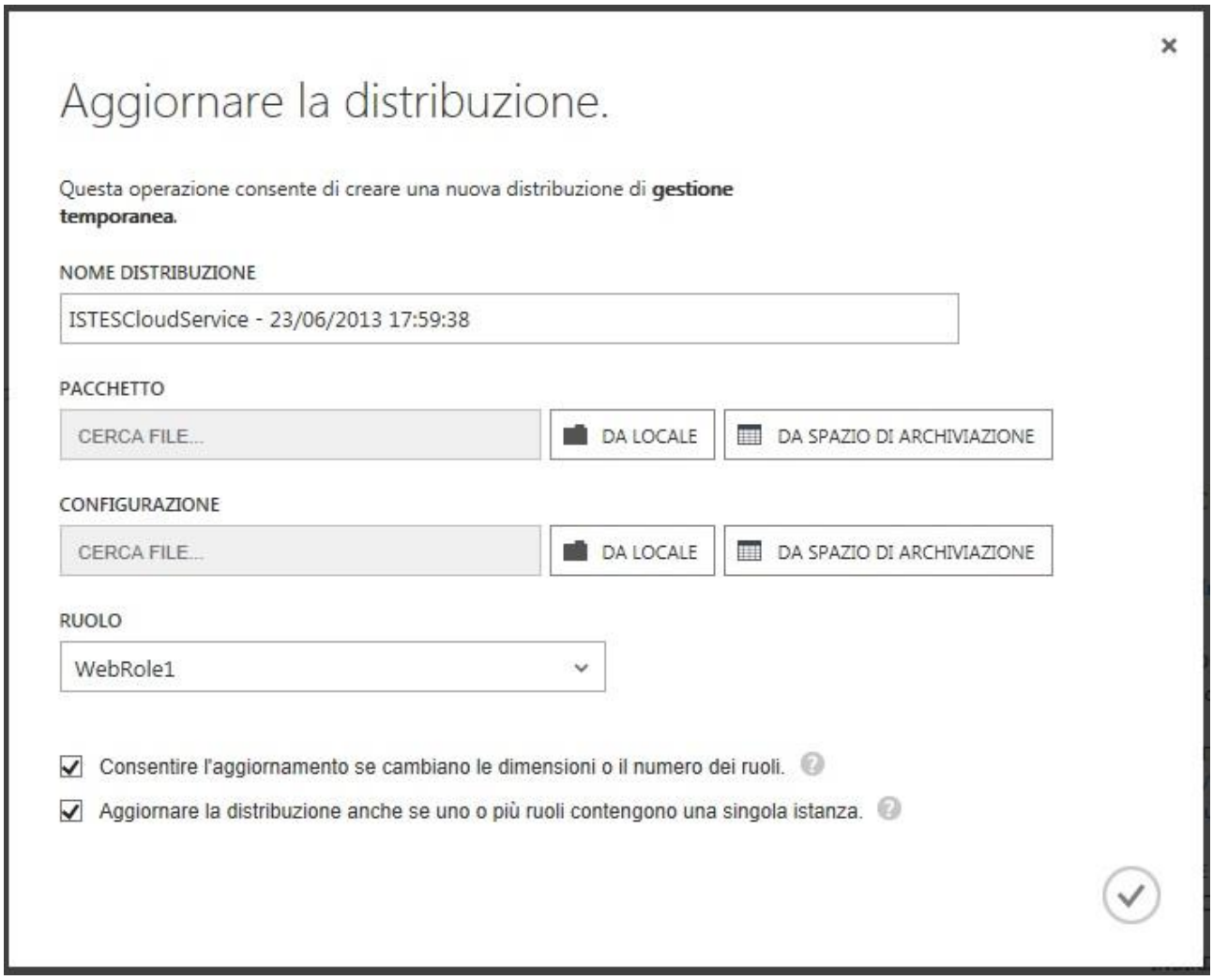


Figura 49 – Aggiornamento del servizio passo 2

Idealmente, la cosa migliore sarebbe caricare e testare la nostra soluzione in un ambiente separato e solo dopo testata spostarla in un *endpoint* reale. In Windows Azure si può simulare questo scenario effettuando il *deployment* nell'ambiente di gestione temporanea (*staging*), usare l'URL temporaneo assegnato da Windows Azure (tramite GUID) per testare l'applicazione e quindi, una volta che siamo soddisfatti, passare dall'ambiente di gestione temporanea (*staging*) a quello di produzione. L'operazione di passaggio tra i due ambienti è praticamente immediata (Figura 50), dato che Windows Azure Fabric deve solo invertire i relativi DNS.

Figura 50 – Scambio di area di gestione per il servizio (da staging a produzione)

In Visual Studio puoi organizzare qualsiasi aggiornamento (pagine, servizi, modifiche al *global.asax* o al *web.config*), caricarlo nell'ambiente di *staging*, testarlo e quindi muoverlo nell'ambiente di produzione.

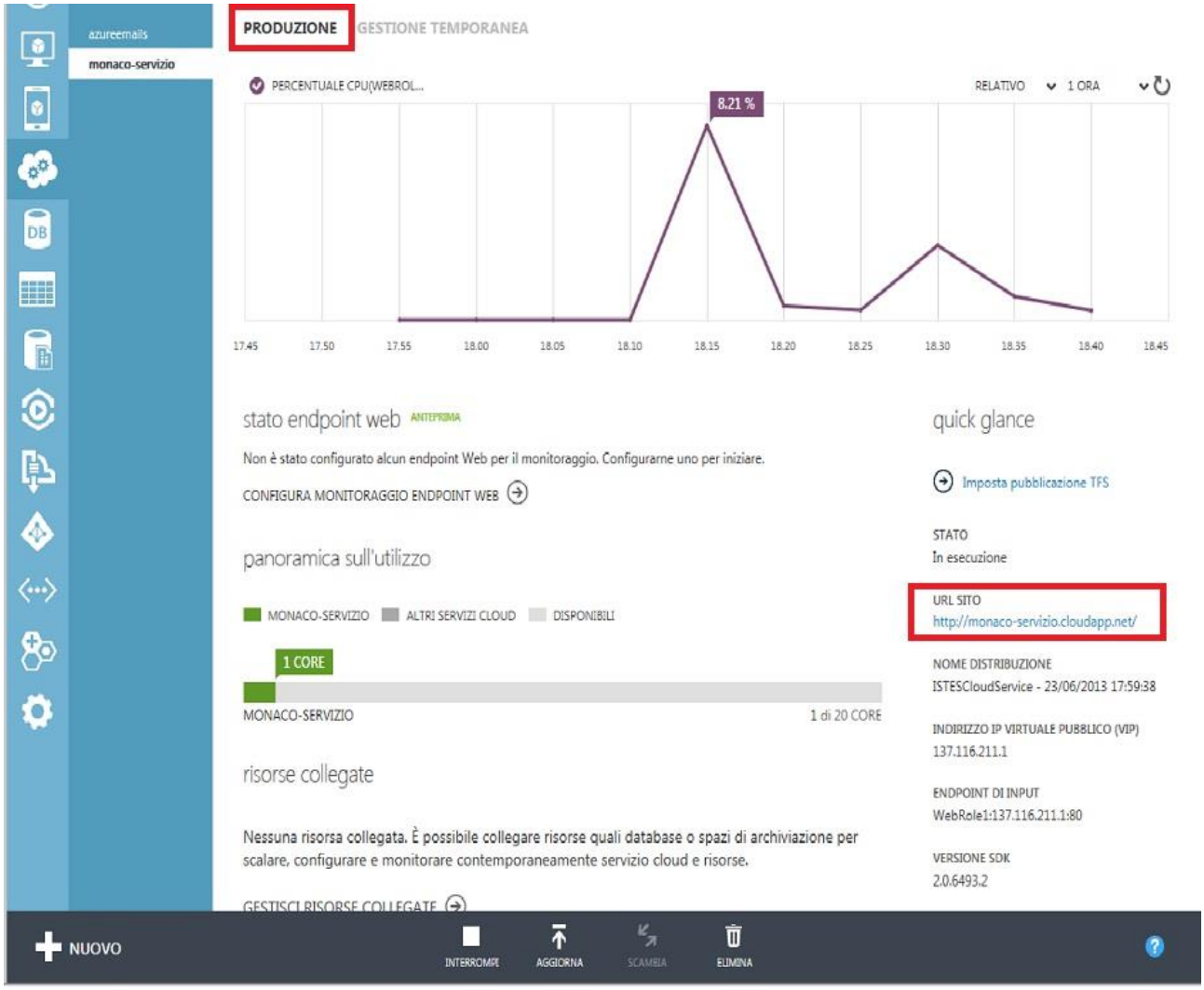


Figura 51 – Particolare del nuovo URL del sito per il servizio hostato

Ritorniamo al progetto di Visual Studio

È possibile usare la finestra di dialogo con le proprietà del progetto per configurare le impostazioni dell'applicazione: in questa finestra è possibile verificare ed eventualmente modificare, i valori contenuti nel file *ServiceConfiguration.cscfg*.

1. Fai doppio clic sull'elemento Web Role denominato WebRole1, contenuto nella cartella Ruoli del progetto cloud. La scheda (*Tab*) Impostazioni (*Settings*), contiene un'unica impostazione denominata *Microsoft.WindowsAzure.Plugins.Diagnostic.ConnectionString*.
2. Aggiungi una nuova impostazione cliccando sul pulsante Aggiungi Impostazione (*Add Settings*), nella mini-toolbar. Assegna alla nuova impostazione il nome *EmailAdmin* e il valore di una tua email (Figura 52). Windows Azure SDK fornisce una API denominata *RoleEnvironment* che espone un semplice metodo per leggere questo nuovo valore dal file di configurazione. Non c'è bisogno di alcuna nuova reference perché il template ASP.NET modificato, che abbiamo usato per creare il progetto, già include quanto necessario.

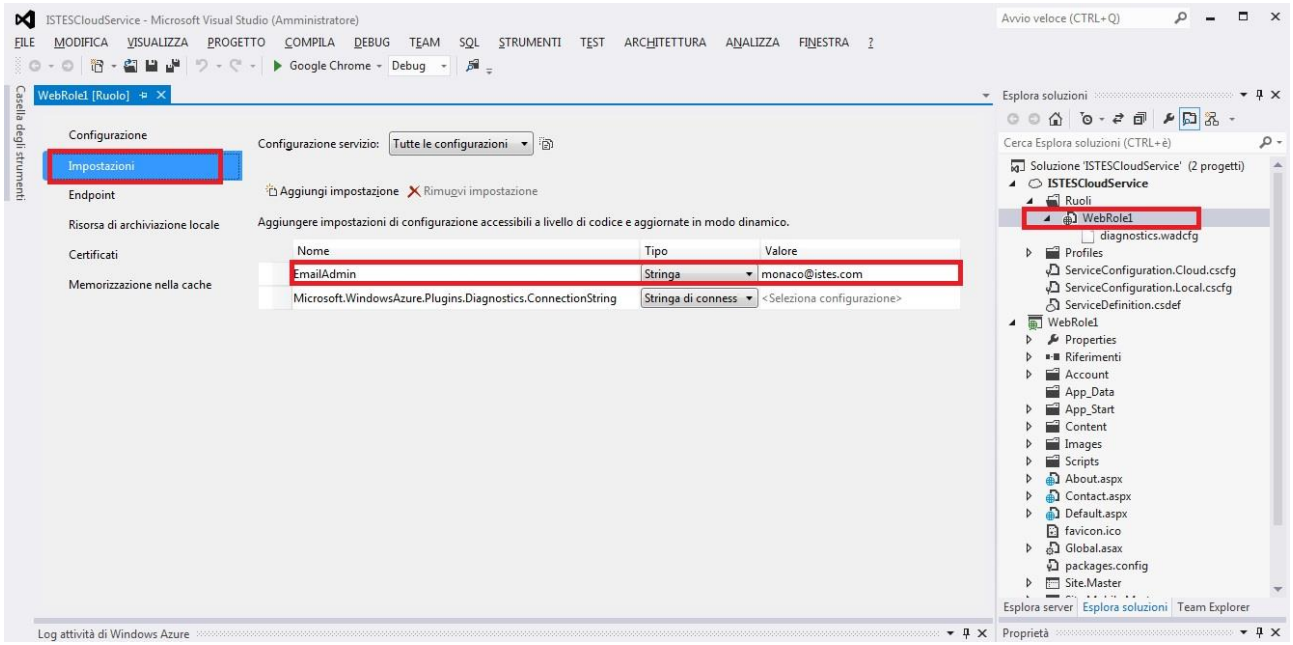


Figura 52 – Particolare dell'assegnazione di una nuova impostazione

3. Testa la nuova impostazione di configurazione aggiungendo una nuova Label alla pagina di default (Default.aspx, parte evidenziata).

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebRole1._Default" %>

<asp:Content ID="HeaderContent" runat="server"
ContentPlaceHolderID="HeadContent"></asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <p><asp:Label ID="TimeLabel" runat="server" /></p>
    <p><asp:Label ID="EmailAdminLabel" runat="server" /></p>
</asp:Content>
```

Listato 3 - Codice della pagina Default.aspx del progetto WebRole1

Per leggere il valore della configurazione il codice utilizza il metodo statico *GetConfigurationSettingValue(string)* esposto dalla classe *RoleEnvironment*, mostrata di seguito (Listato 4)

```
namespace WebRole1
{
    using System;
    using System.Web.UI;
    using Microsoft.WindowsAzure.ServiceRuntime;
    using System.Configuration;

    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            TimeLabel.Text = DateTime.Now.ToString();

            if (RoleEnvironment.IsAvailable)
            {
                EmailAdminLabel.Text =
                    RoleEnvironment.GetConfigurationSettingValue("EmailAdmin");
            }
            else
            {
                EmailAdminLabel.Text = ConfigurationManager.AppSettings["EmailAdmin"];
            }
        }
    }
}
```

Listato 4 - Codice (code behind) della pagina Default.aspx.cs del progetto WebRole1

Resta ancora da introdurre uno dei più importanti file contenuti nel progetto WebRole1. Il progetto consiste in un classico progetto ASP.NET Web Application con tre importanti variazioni: la presenza di tre *reference* ad altrettante API di Windows Azure, una nuova *trace listener configuration* e una classe denominata WebRole definita nel file WebRole.cs. Il Listato 5 mostra la definizione della classe:

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace WebRole1
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // Per informazioni sulla gestione delle modifiche alla configurazione,
            // vedere l'argomento MSDN all'indirizzo
            http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

Listato 5 – Classe WebRole.cs con il particolare del metodo OnStart

Come si può vedere dal listato, la classe WebRole deriva dalla classe *RoleEntryPoint*, definita nel *namespace Microsoft.WindowsAzure.ServiceRuntime*. WebRole compie l'override del metodo *OnStart* della classe base *RoleEntryPoint* per fornire un entry point per quando Windows Azure Fabric avvierà il servizio. Questo metodo viene eseguito ogni volta che si riavvia un servizio prima sospeso o viene incrementato il numero di istanze. È importante ricordare che il package viene caricato su un numero variabile di macchine virtuali, sulla base di quanto specificato nelle impostazioni di configurazione contenute nel file *ServiceConfiguration.cscfg*.

Un classico WebRole ha il seguente aspetto (Listato 6):

```
namespace WebRole1
{
    using System.Linq;
    using Microsoft.WindowsAzure.ServiceRuntime;

    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            RoleEnvironment.Changing += RoleEnvironment_Changing;

            return base.OnStart();
        }

        void RoleEnvironment_Changing(object sender, RoleEnvironmentChangingEventArgs e)
        {
            // se la configurazione è cambiata
            if (e.Changes.Any(change => change is
                RoleEnvironmentConfigurationSettingChange))
            {
                // imposta e.Cancel a true per riavviare l'istanza del ruolo
                e.Cancel = true;
            }
        }
    }
}
```

Listato 6 – Classe WebRole.cs integrata con il metodo RoleEnvironment_Changing

La prima linea del metodo *OnStart* informa il *RoleEnvironment* che esiste un *event handler*[43] per la gestione dell'evento denominato *Changing*. Questo evento viene sollevato prima che una qualsiasi modifica alla configurazione del servizio venga applicata alle istanze del *role* attualmente in esecuzione. L'event handler riceve un'istanza della classe *RoleEnvironmentConfigurationSettingChange*, la quale espone due proprietà: *Cancel* e *Changes*. La proprietà *Cancel* viene utilizzata per segnalare a Fabric che si vuole riavviare l'istanza, mentre la proprietà in sola lettura *Changes* contiene una collezione di *RoleEnvironmentChange* che richiedono di essere applicate all'istanza del *role*.

Il codice utilizza la sintassi LINQ per ispezionare la collezione e trovare tutti quei cambiamenti che consistono in modifiche alle impostazioni di configurazione. Se la query ritorna il valore *true*, il codice setta *e.Cancel* a *true* per forzare il riavvio dell'istanza del *role*.

[43] Un delegato **event handler** rappresenta il metodo di gestione dell'evento che non dispone di dati dell'evento stesso. (http://msdn.microsoft.com/it-it/library/system.eventhandler.aspx).
In programmazione un **event handler** è una chiamata asincrona di una subroutine che manipola gli input ricevuti da un programma (chiamato *listener* in Java) (http://en.wikipedia.org/wiki/Event_(computing)).

Dal codice sopra illustrato (Listato 6) si può notare che, di default, in un progetto Web Role creato con Visual Studio, viene forzato il riavvio delle istanze ogni volta che viene modificata una qualche impostazione nel file *ServiceConfiguration.cscfg*.

Il File di definizione del Servizio

Se apriamo il file *ServiceDefinition.csdef* nel progetto cloud, vedremo qualcosa di simile al Listato 7.

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="Primo_CloudService"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition"
schemaVersion="2013-03.2.0">
  <WebRole name="WebRole1" vmsize="Small">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="Endpoint1" endpointName="Endpoint1" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="http" port="80" />
    </Endpoints>
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
  </WebRole>
</ServiceDefinition>
```

Listato 7 – File ServiceDefinition.csdef

Dopo la classica dichiarazione XML, la definizione continua con un elemento XML denominato *ServiceDefinition*, il quale specifica il nome che verrà utilizzato dal *Compute Emulator* e da Windows Azure Fabric per definire il contratto per questo servizio.

Da notare che la porta 80 è configurata nel file di definizione, ma il Compute Emulator utilizza la porta 81: questo comportamento dipende dalla presenza di IIS sulla mia macchina, il quale si riserva la porta 80.

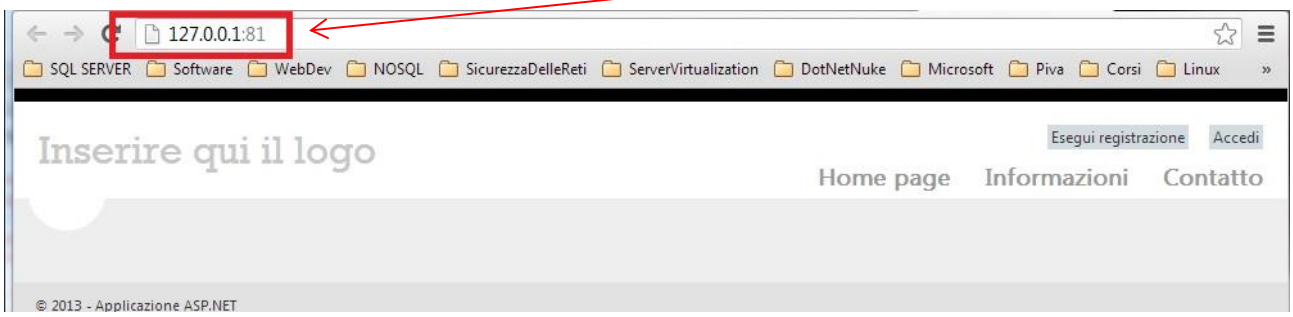


Figura 53 – Particolare dell'indirizzo locale dell'applicazione

L'elemento *ConfigurationSettings* è in qualche modo curioso, dal momento che a prima vista sembra una duplicazione dell'omonima impostazione contenuta nel file *ServiceConfiguration.cscfg*. In realtà il file *ServiceDefinition* definisce l'esistenza di una certa impostazione, non il suo valore che invece viene assegnato dal *ServiceConfiguration*. Non è possibile assegnare un valore ad una impostazione nel *ServiceConfiguration.cscfg* che non sia stato prima definito nel file di definizione del servizio. Quest'ultimo

viene compilato assieme al codice dei *role* e caricato su ciascuna delle istanze del servizio presenti sul cloud, informandole che esiste un'impostazione con quello stesso nome (nel nostro caso *EmailAdmin*).

Windows Azure *Storage*

Fino adesso abbiamo visto che ogni servizio cloud per Windows Azure presenta una file di definizione del servizio stesso con estensione .csdef, che imposta la richiesta di componenti, servizi, endpoint, ecc. a Windows Azure. Abbiamo anche visto che questo file assieme a quello di configurazione del servizio, con estensione .cscfg, può contenere impostazioni dell'applicazione che possono essere lette attraverso la classe *RoleEnvironment*.

Lo storage locale (local storage) rappresenta una caratteristica esposta dal sistema operativo per fornire uno spazio in cui le applicazioni possono salvare e leggere i dati. È una sorta di hard disk tradizionale sebbene sia esposto come risorsa logica e non fisica (alla pari di qualunque *feature* di Windows Azure): cioè non c'è alcun disco contrassegnato da una lettera o da un percorso di rete.
Una volta ottenuto lo spazio richiesto, per usarlo è sufficiente ricorrere alla classe RoleEnvironment per ottenere il punto iniziale dove cominciare a salvare i dati. È possibile creare sottocartelle e file, come faresti normalmente con le classi incluse nel Microsoft .NET Framework *System.IO*, leggere, aggiornare e cancellare i file presenti nello storage.
L'unica differenza risiede nel modo in cui possiamo raggiungere la *root* dello spazio disco.

Lo strumento Windows Azure Compute Emulator simula le caratteristiche dello storage locale sulla tua machina di sviluppo, in modo da permetterti di testare il comportamento della tua applicazione localmente, senza doverla pubblicare sul cloud.

Creiamo un nuovo progetto Windows Azure (Figura 54) e aggiungiamo un ASP.NET Web Role (lasciando il nome di default WebRole1).

Quando Visual Studio ha finito di creare l'infrastruttura del progetto, nel progetto cloud, facciamo doppio clic sul nodo WebRole1 per aprire la schermata di configurazione del role (*Role Configuration*).

Facciamo clic sulla scheda (*tab*) Risorse di archiviazione locale (*Local Storage*) e, nella mini toolbar (sul lato destro), clicchiamo sul pulsante Aggiungi risorsa di archiviazione locale (*Add Local Storage*).

Denominiamo il nuovo storage locale MyStorage inserendo il valore 50 nella textbox Dimensioni (*Size*).

La figura che ci dobbiamo trovare alla fine della configurazione è quella di Figura 55.

Dopo avere salvato la configurazione L'IDE (nota 39 pag.63) di Visual Studio provvede a modificare il file *ServiceDefinition.csdef* per richiedere a Windows Azure, durante il *deployment* dell'*hosted service*, lo spazio disco desiderato, come mostrato nel Listato 8. In questo caso Windows Azure crea una macchina virtuale con 50Mb di spazio disco libero e lo espone con il nome MyStorage.

È possibile leggere la gran parte delle impostazioni usando le classi contenute nel *namespace*[44] *Microsoft.WindowsAzure.ServiceRuntime* e nel relativo *assembly*[45] e, dato che questa libreria è referenziata di default all'interno del template per un progetto cloud, è anche possibile usare le relative API.

[44] Un **namespace** è una collezione di nomi di entità, definite dal programmatore, omogeneamente usate in uno o più file sorgente. Lo scopo del namespace è quello di evitare confusione ed equivoci nel caso siano necessarie molte entità con nomi simili.

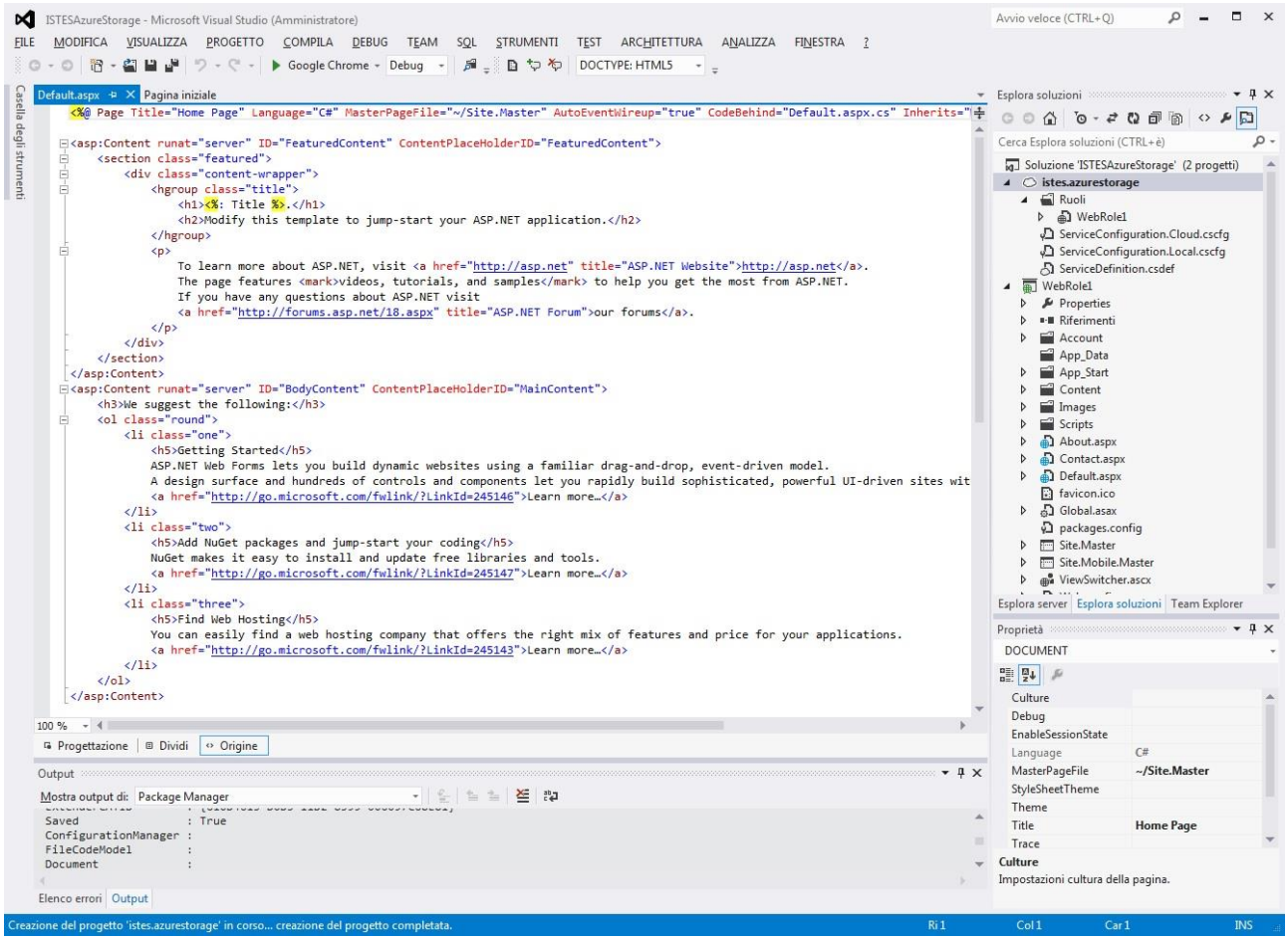


Figura 54 – Creazione di un nuovo progetto Windows Azure

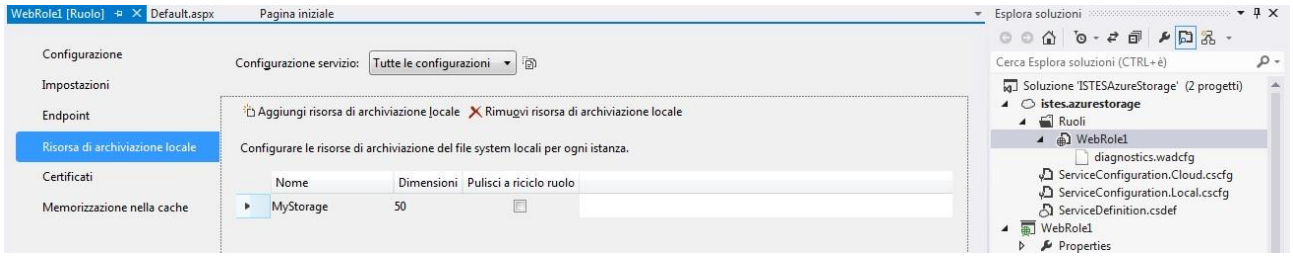


Figura 55 – Assegnazione di spazio disco alla risorsa di archiviazione locale

[45] L'**assembly** è un software che trasforma le istruzioni in memoria in linguaggio macchina. Si tratta dunque di un compilatore.

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="istes.azurestorage"
       xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition"
       schemaVersion="2013-03.2.0">
  <WebRole name="WebRole1" vmsize="Small">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="Endpoint1" endpointName="Endpoint1" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="http" port="80" />
    </Endpoints>
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
    <LocalResources>
      <LocalStorage name="MyStorage" cleanOnRoleRecycle="false" sizeInMB="50" />
    </LocalResources>
  </WebRole>
</ServiceDefinition>
```

Listato 8 – File ServiceDefinition.csdef del progetto cloud istes.azurestorage

Attraverso la classe *RoleEnvironment*, puoi accedere alle risorse locali configurateusando il metodo *GetLocalResource*, il quale accetta come parametro una stringa corrispondente al nome della risorsa richiesta. Il metodo ritorna un oggetto del tipo *LocalResource*, il quale espone, accanto alla proprietà *Name* (che indica il nome della risorsa), altre due proprietà: la proprietà *MaximumSizeInMegabytes*, che indica la dimensione massima dello spazio disco allocato per quello storage locale (e definita nel file di configurazione del servizio), e la proprietà *RootPath*, che contiene le informazioni necessarie per accedere allo storage.

Adesso modifichiamo il file *Default.aspx* (Listato 9) aggiungendo un controllo Label denominato localStorage (togliendo tutto il resto del codice prodotto in automatico all'associazione del Web Role al progetto cloud)

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master"
    AutoEventWireup="true"
    CodeBehind="Default.aspx.cs" Inherits="WebRole1._Default" %>

<asp:Content runat="server" ID="HeaderContent" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="MainContent">
    <h2>Benvenuti in ISTES Azure Storage:</h2>
    <p>
        <asp:Label ID="localStorage" runat="server" />
    </p>
</asp:Content>
```

Listato 9 – File Default.aspx del progetto WebRole1

Modifichiamo adesso il file contente il *code behind* della pagina (*Default.aspx.cs*) in modo che richiami il metodo *GetLocalResource*, il quale, a sua volta, richiederà a Windows Azure una risorsa specifica. Usiamo l'istanza di tipo *LocalResource* restituita da *GetLocalResource* in modo che il testo della *Label* mostri la proprietà *RootPath*:

```
namespace WebRole1
{
    using Microsoft.WindowsAzure.ServiceRuntime;
    using System;
    using System.Web.UI;

    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            LocalResource resource = RoleEnvironment.GetLocalResource("MyStorage");

            localStorage.Text = resource.RootPath;
        }
    }
}
```

Listato 10 – File Default.aspx.cs (code behind) del progetto WebRole1

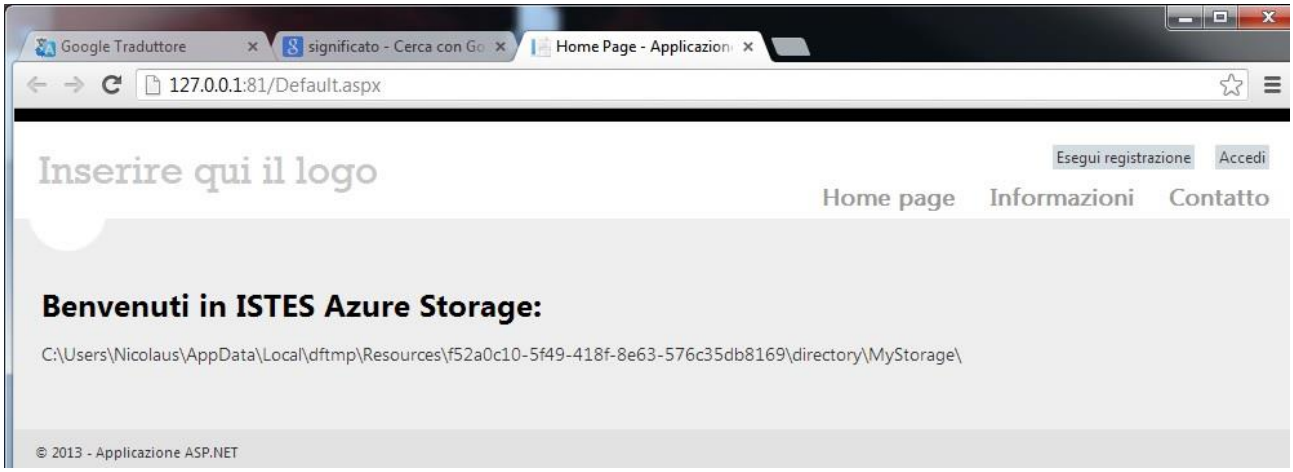


Figura 56 – Finestra browser in locale con il particolare della proprietà RootPath

Azure *Storage* Account

Possiamo assimilare lo storage locale ad un file system locale in un server on-premises. Database SQL (ex SQL Azure) può fungere da store relazionale per un'applicazione nel cloud richiede di essere pagato separatamente rispetto alla sottoscrizione per Windows Azure.

Una valida alternativa per salvare i dati nel cloud è offerta dall'Azure Storage Account. Questo è un posto perfetto per salvare blob[46] (anche di grandi dimensioni) e le entità dell'applicazione; inoltre fornisce un meccanismo di gestione delle code per disaccoppiare il *front end* dell'applicazione dal *back end*.

Le risorse immagazzinate nello Storage Account sono sempre disponibili perché Windows Azure provvede a replicare su differenti nodi in modo da garantire sia la fault-tolerance che la scalabilità.

[46] Un **blob** (Binary LargeObject) è destinato alla memorizzazione di dati di grandi dimensioni in formato binario. È una categoria all'interno di un container. Ci sono due titpi di blob: Page Blob e Block Blob.
http://stackoverflow.com/tags/azure-storage-blobs/info

L'accesso alle risorse sullo Storage Account può avvenire via REST e HTTP cosicché qualsiasi applicazione, in esecuzione su qualsiasi piattaforma, è in grado di salvare, recuperare e manipolare dati nel cloud.

Creiamo uno Storage Account nel portale di Windows Azure[47].

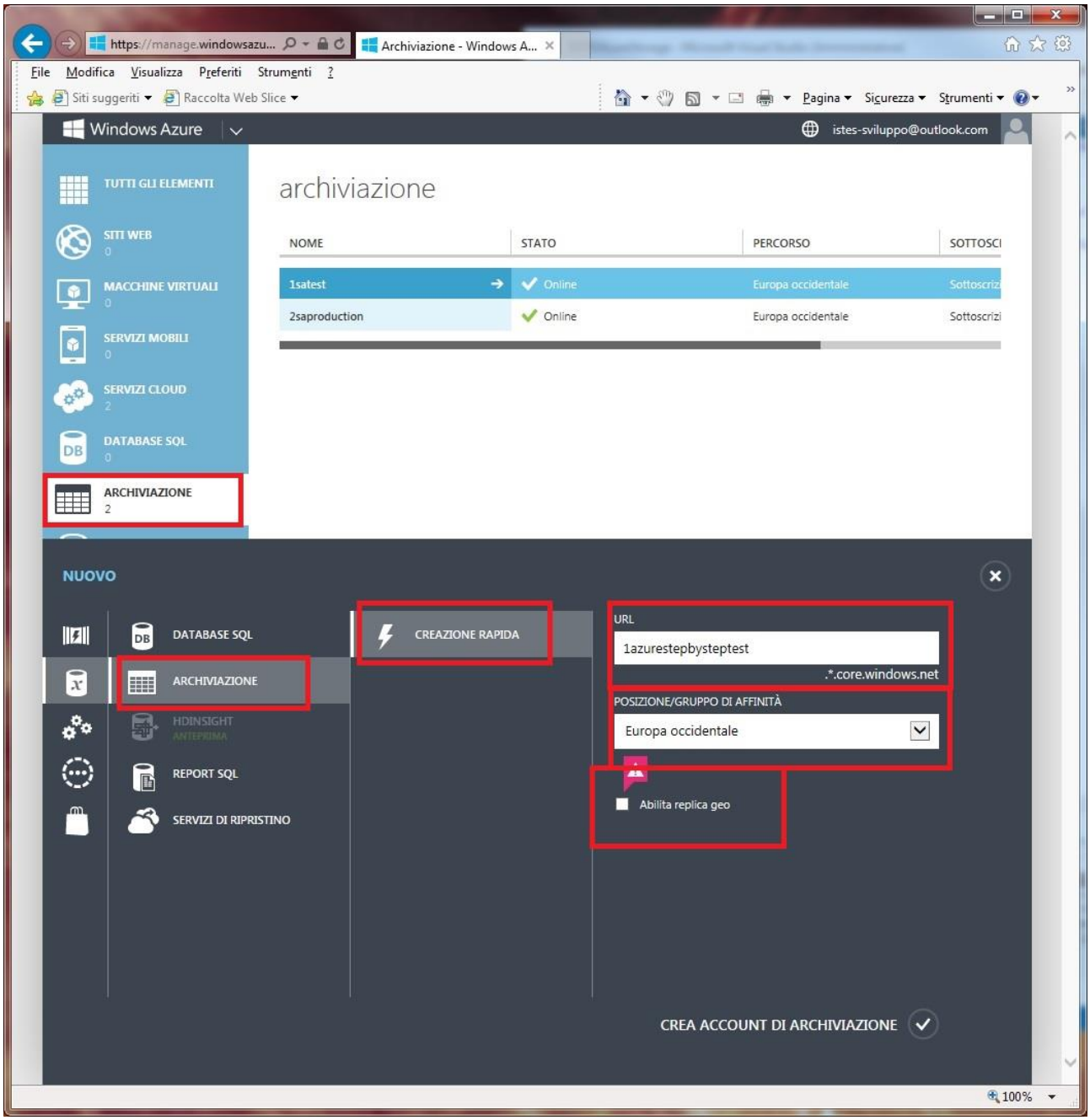


Figura 57 – Creazione di uno Storage Account

Abbiamo creato uno Storage Account, scegliendo una "regione" in cui immagazzinare i dati, oltre che una URL pubblica che qualunque applicazione potrà usare per accedere.

[47] Nel campo URL della figura 57, è possibile inserire una URL univoca dai 3 ai 24 caratteri alfabetici e numerici ma non altro.

Personalmente differenzio l'account di archiviazione per la fase di test da quello per la fase di produzione, utilizzando una notazione in prefisso "1" e suffisso "test" per lo *storage account* di test, mentre utilizzo il prefisso "2" e suffisso "production" per lo *storage account* di produzione.

Il nome pubblico assegnato diviene la *root* delle URL per qualunque tipo di risorsa. Per esempio per richiedere un blob nello Storage Account, è necessaria una richiesta HTTP GET alla seguente URI: http://1azurestepbysteptest.blob.core.windows.net .

Per connettersi alle risorse presenti nello storage account, è necessario includere una delle chiavi di accesso al quale si può accedere tramite la procedura di gestione del Sistema Operativo Windows Azure

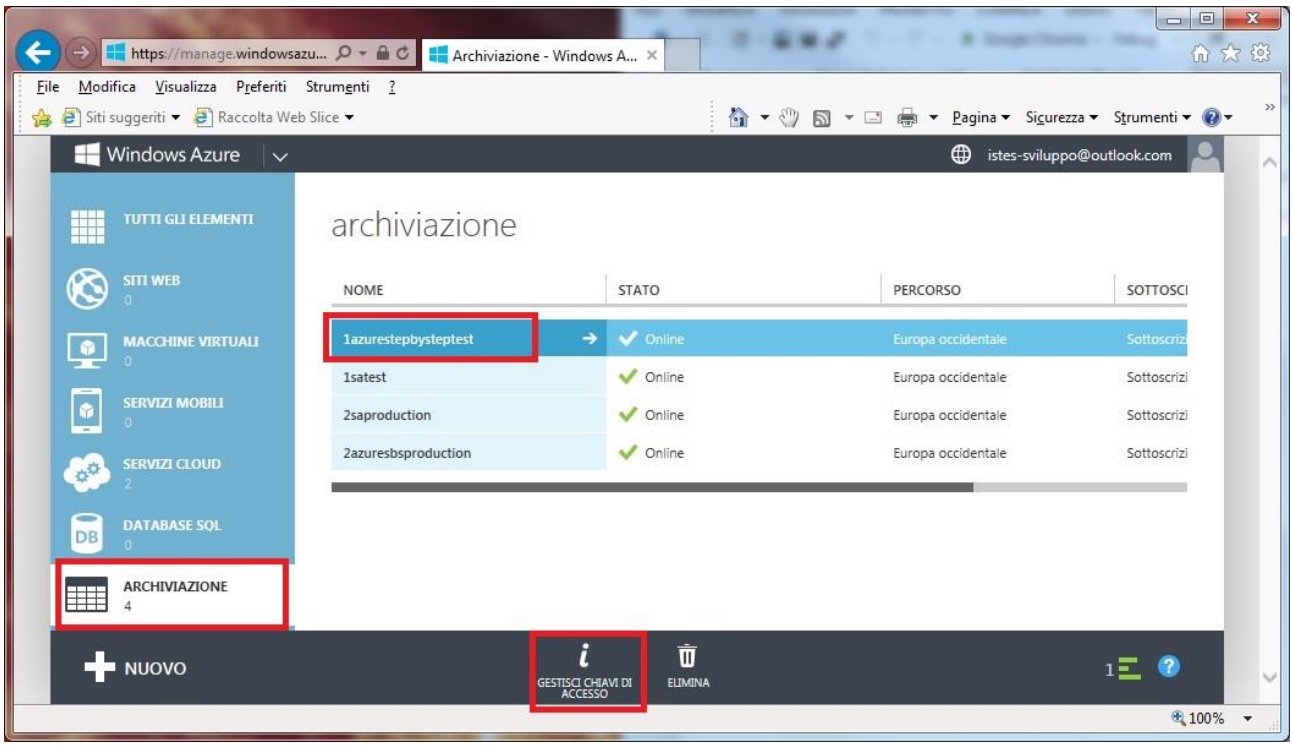


Figura 58 – Gestione delle chiavi di accesso dello Storage Account

Figura 59 – Copia del nome e della chiave di accesso primaria dello Storage Account

È possibile copiare il nome dell'account di archiviazione e la chiave, negli appunti, per riutilizzarli nel codice o in altri strumenti di gestione.

Azure Storage Explorer

Puoi scaricare Azure Storage Explorer dal sito di codeplex (http://azurestorageexplorer.codeplex.com/). In questo momento siamo alla versione 5 Preview 1 (giugno 2012)[48].

Questo strumento permette di gestire blob, tabelle e code direttamente sul tuo Storage Account, per cui si rivela un ottimo compagno di viaggio per ogni progetto che utilizzi lo Storage Account.

Si rimanda all'help molto particolareggiato dell'applicazione, lo studio delle peculiarità di Azure Storage Account (Figura 60).

Nella Versione di Visual studio 2012 c'è la possibilità di vedere i dati che la tua applicazione ha inserito nello storage (in blob, tabelle o code), anche nella sezione Esplora Server (*Server Explorer*) da Visualizza -> Esplora Server (Figura 62)

[48] Al momento ci sono altri strumenti (*tools*) che si interfacciano con Windows Azure Storage, ad esempio http://clumsyleaf.com/products/tablexplorer oppure http://www.cerebrata.com/Products/CloudStorageStudio/Details.aspx?t1=0&t2=6

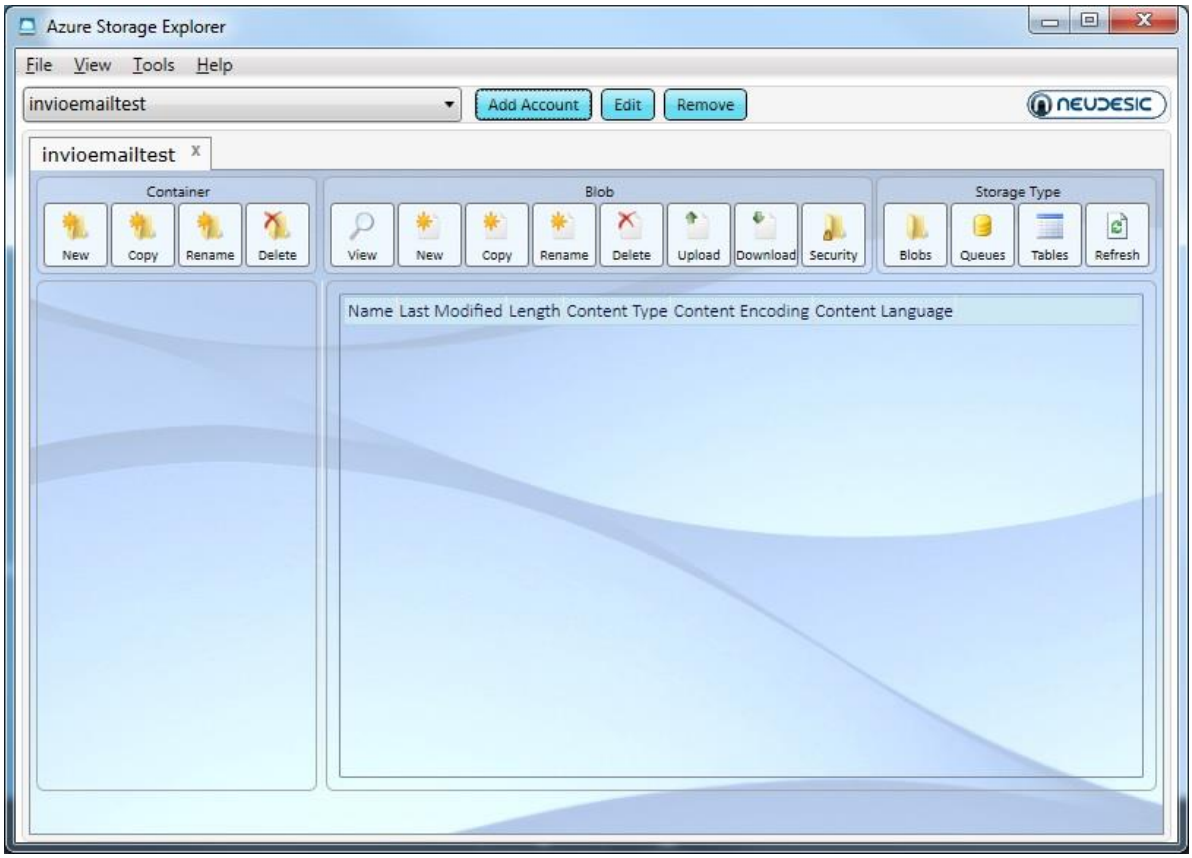


Figura 60 – Azure Storage Explorer

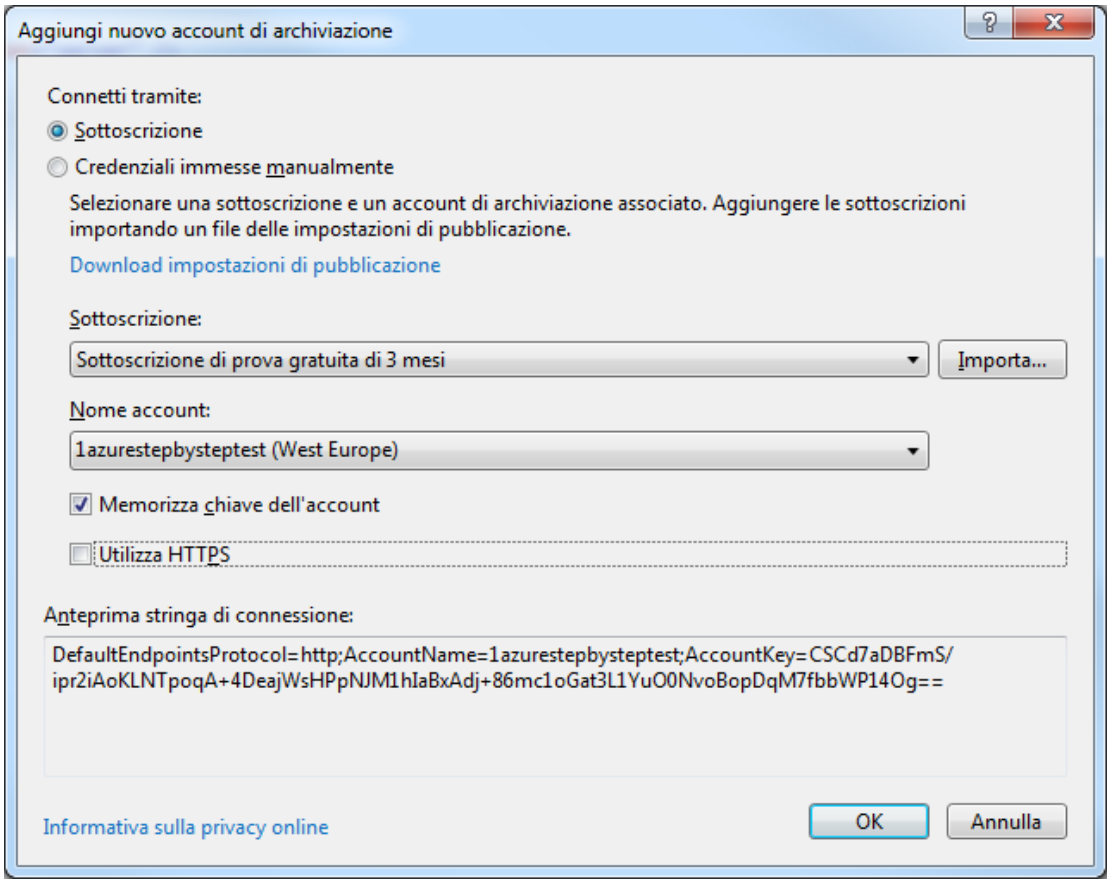


Figura 61 – Aggiunta di un nuovo account di archiviazione con Visual Studio

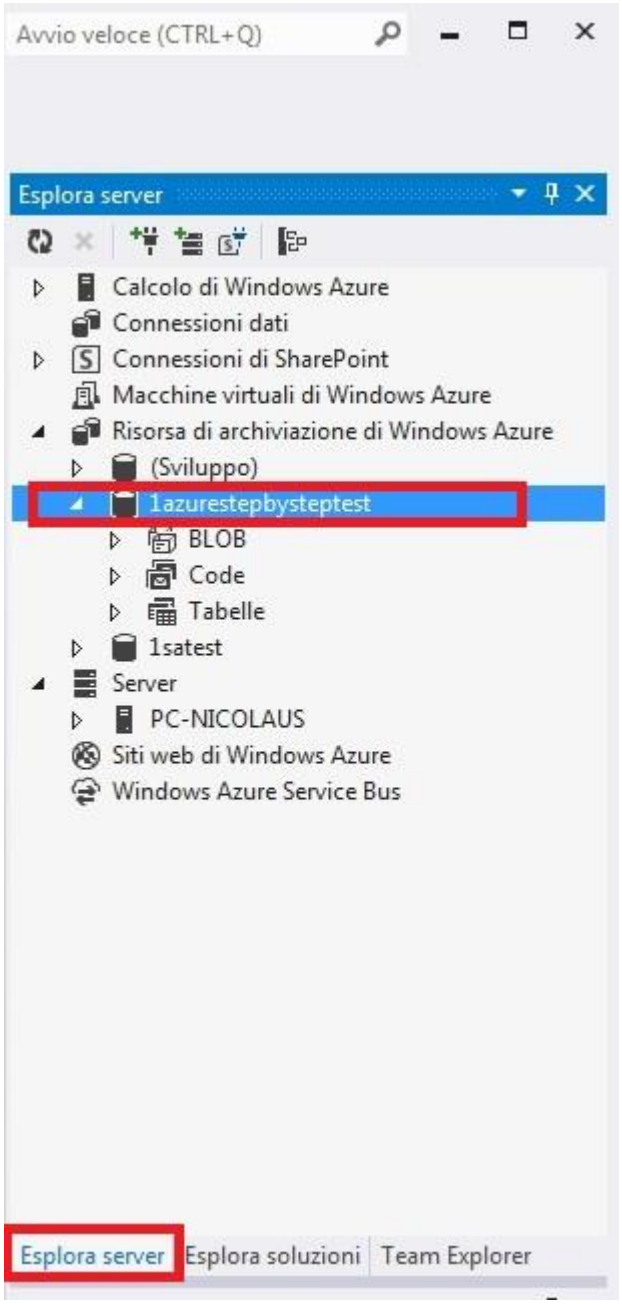


Figura 62 – Esplora Server di Visual Studio

Per completare il tour introduttivo su blob e container, e prima di passare a conoscere le API che si dovranno usare nel codice, bisogna provare ad aggiungere delle immagini nel blob container (da Figura 63 a Figura 66).

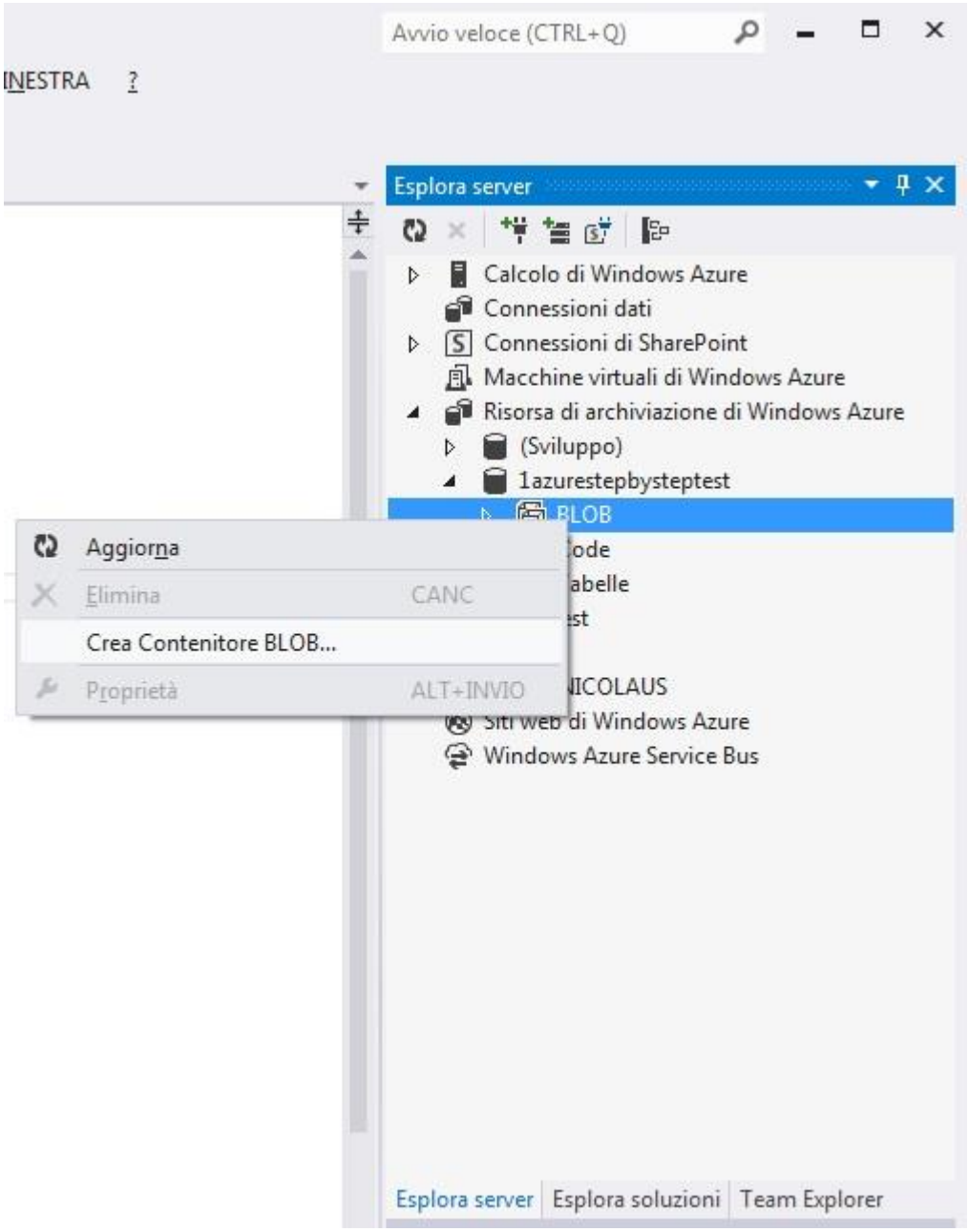


Figura 63 – Come inserire delle immagini nel blob container passo 1

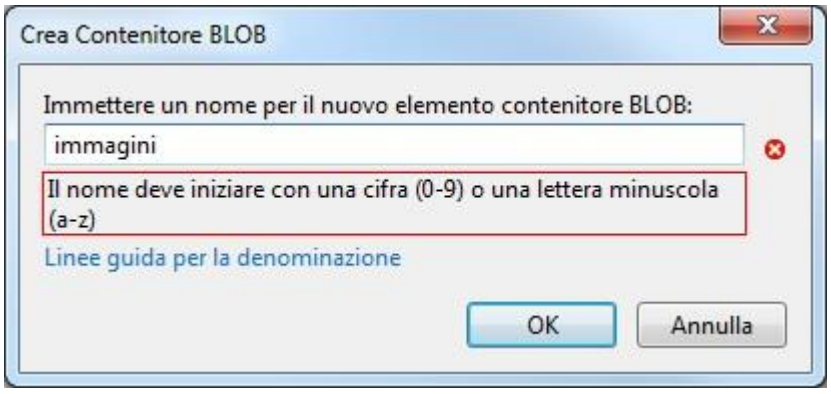


Figura 64 – Come inserire delle immagini nel blob container passo 2

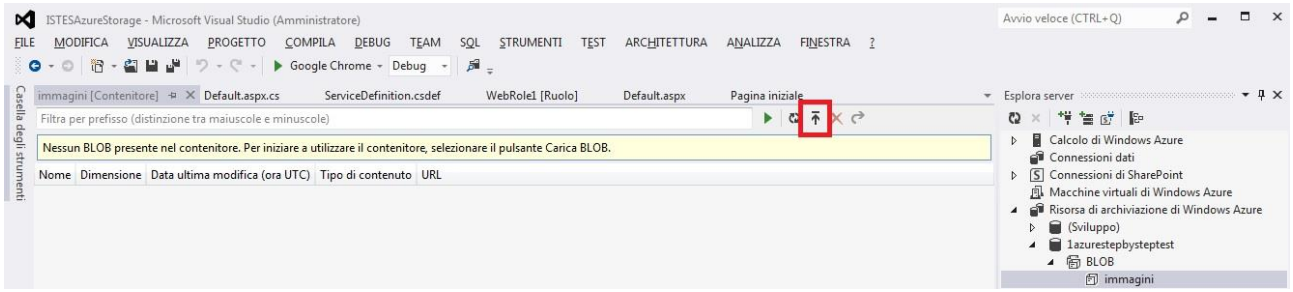


Figura 65 – Come inserire delle immagini nel blob container passo 3

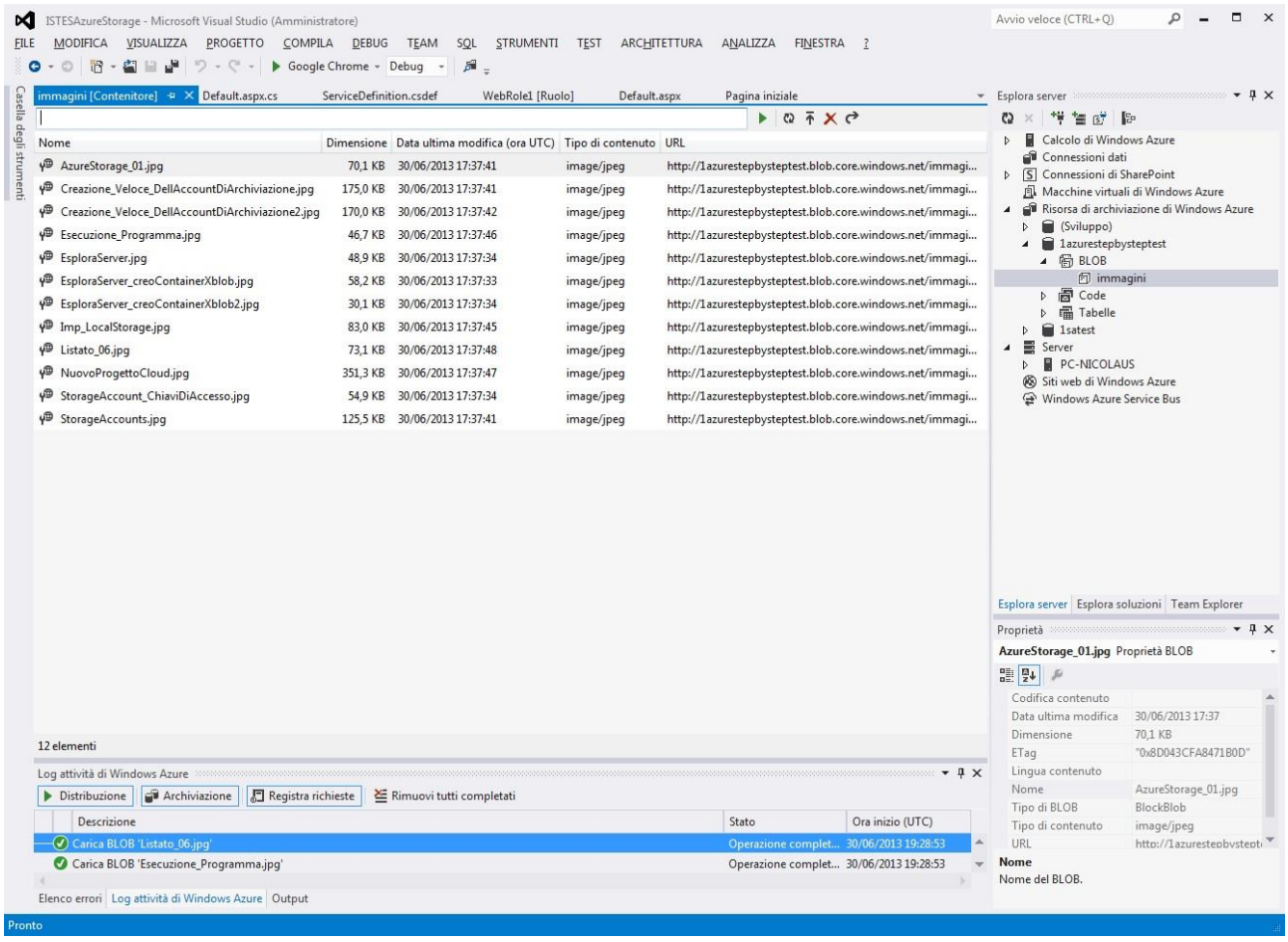


Figura 66 – Come inserire delle immagini nel blob container passo 4

Le immagini caricate nel contenitore (*container*) immagini, presentano nomi che rappresentano le risorse all'interno dello storage. Ciascun nome deve essere univoco all'interno del container, perché il nome contribuisce a formare l'URI utilizzata per le operazioni REST su ciascun elemento dello storage. Come puoi vedere il percorso (ossia l'URI) per le richieste, è composto dal nome dello Storage Account seguito dalla URI standard, dal nome del container e da quello del blob.

Poiché il container è pubblico, qualunque client può compiere un'operazione di tipo GET sui blob ivi contenuti, senza che sia necessaria alcuna chiave di accesso.

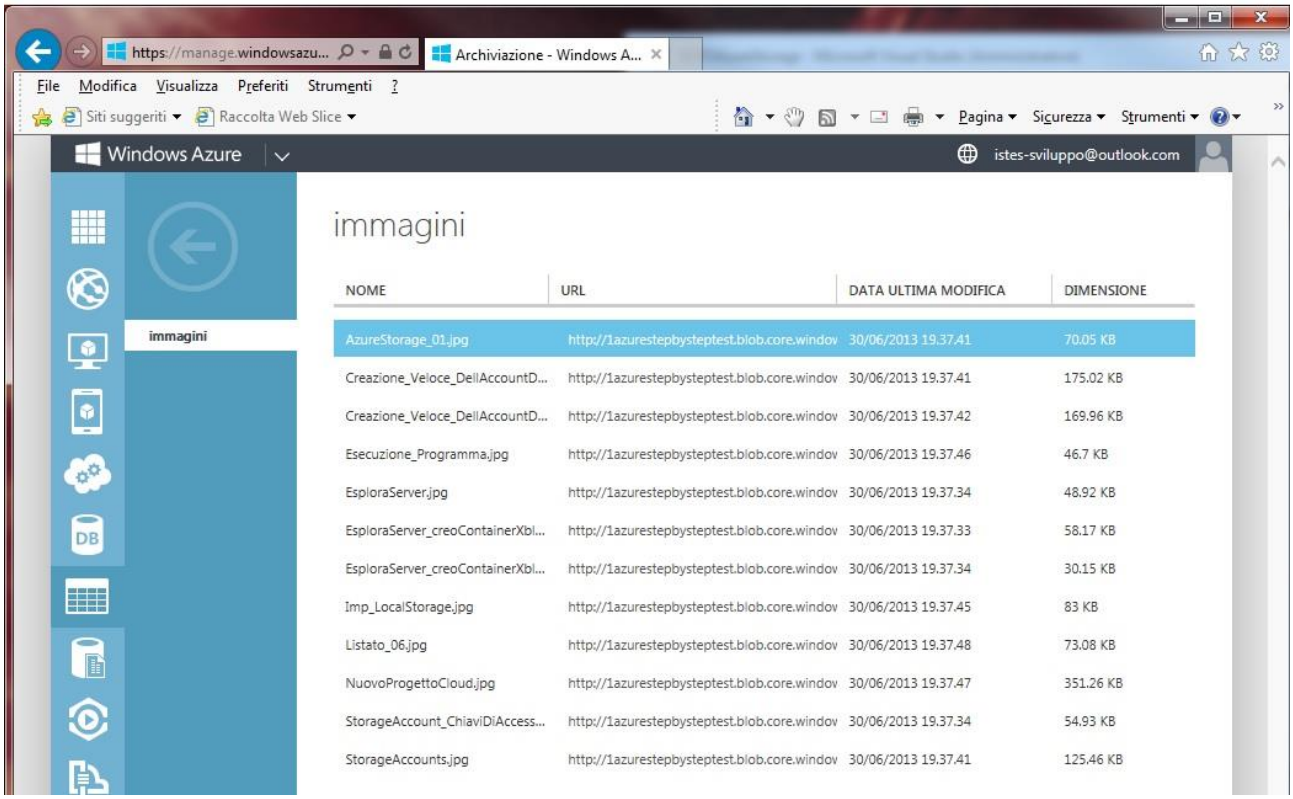


Figura 67 – Le immagini inserite viste in Azure

All'interno del container creato bisogna andare in modifica contenitore e renderlo pubblico all'accesso (Figura 67).

Figura 68 – Modifica dei metadati del container

Così facendo ho la possibilità di richiamare il singolo blob inserendo l'URI nella barra degli indirizzi di un browser.

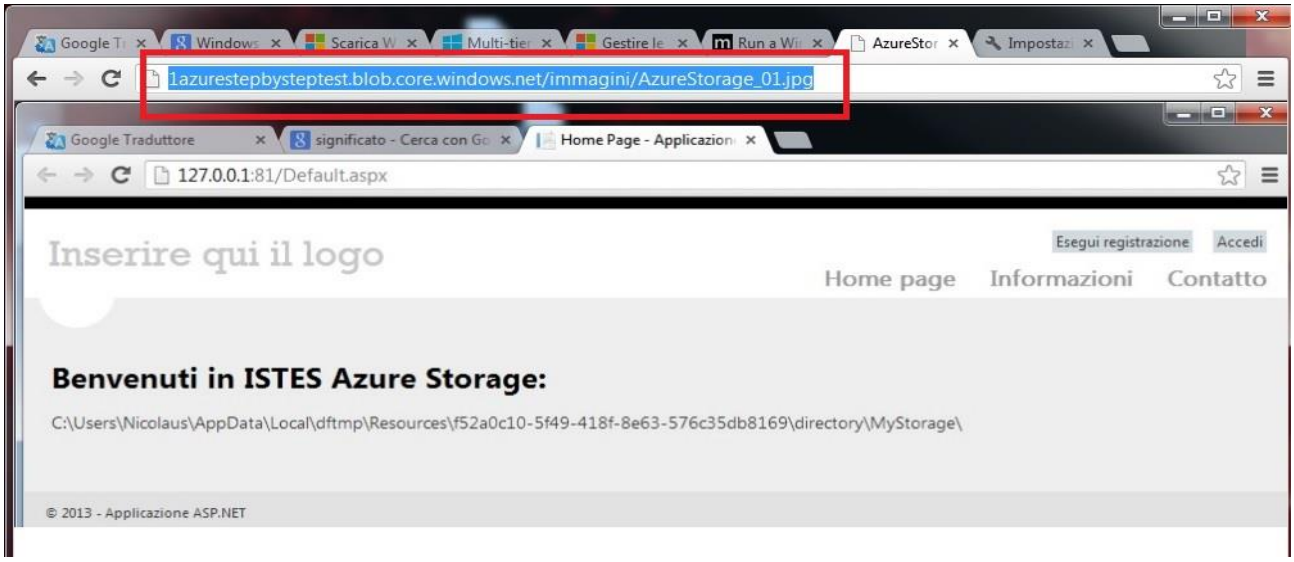


Figura 69 – Dettaglio dell'URL di un'immagine del container

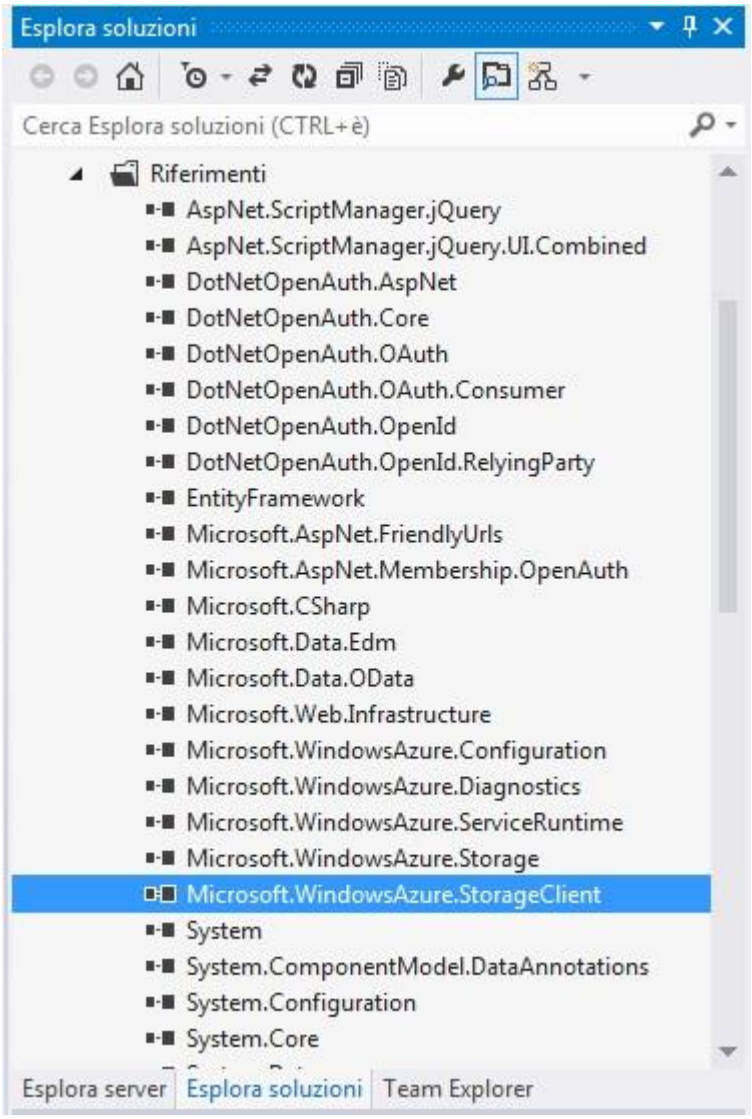


Figura 70 – aggiunta di una riferimento dello Storage Client di Windows Azure

Fino adesso ci siamo divertiti sperimentando con Esplora Server (*Server Explorer*), il carico (ma si può anche scaricare e cancellare blob e container; di Azure Storage Explorer abbiamo appena accennato poiché è un applicativo di terze parti e non Microsoft) di blob nello storage di Azure. Nel prossimo paragrafo introduco le API da utilizzare per compiere le medesime operazioni di *Create*, *Read*, *Update*, *Delete* (CRUD) via codice.

Blob API

Per prima cosa è necessario ottenere le informazioni di connessione per costruire l'URL da usare per la richiesta REST, e quindi devi fornire l'*authorization header*, aggiungendolo ad ogni richiesta. Può essere utile includere queste informazioni nel file di configurazione del servizio , così come si farebbe con qualunque altra informazione di configurazione. È anche possibile ottenere il riferimento ad un container semplicemente aggiungendo il suo nome all'URL (es. http://1azurestepbysteptest.blob.core.win

dows.net/immagini/**AzureStorage_01.jpg**). Per listare il contenuto di un container, basta aggiungere una *query string* del tipo comp=list (http://1azurestepbysteptest.blob.core.windows.net/immagini/?comp=list) (Figura 71). Infine, può essere specificato un tempo di timeout per ciascuna operazione.

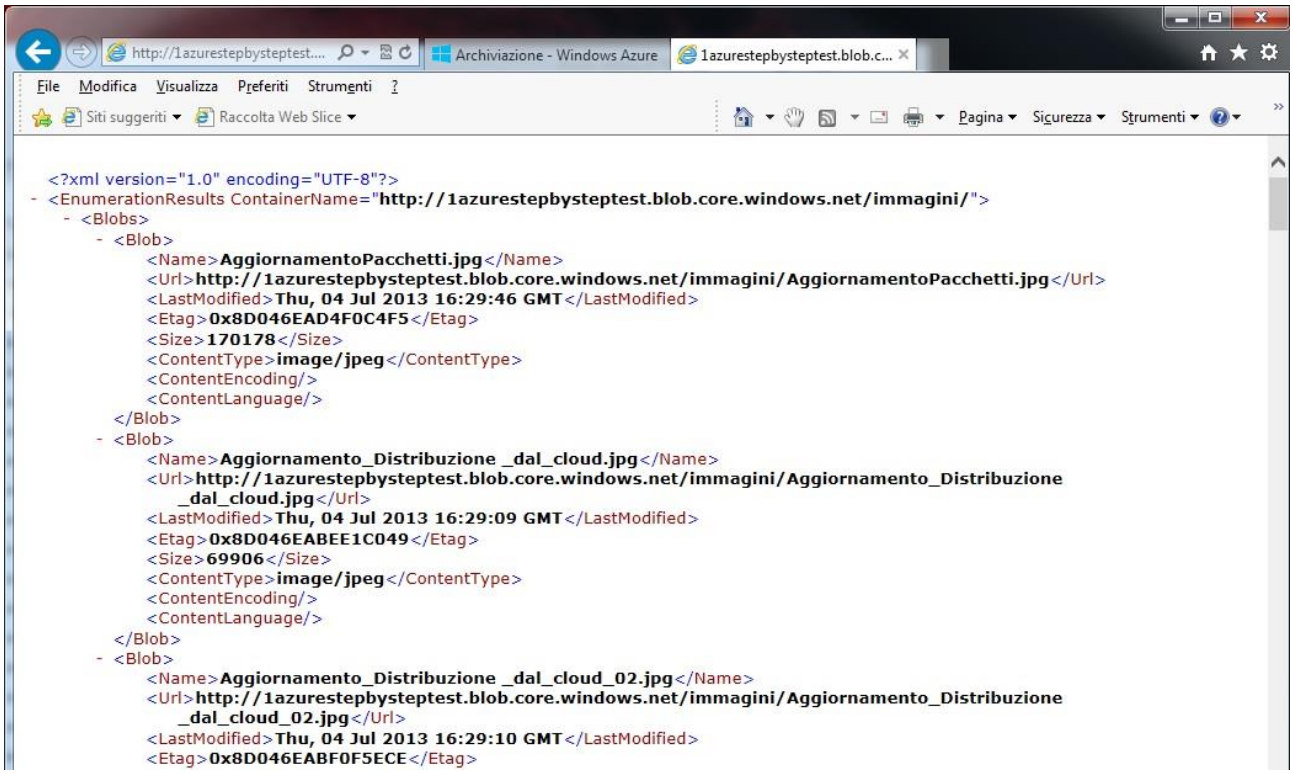


Figura 71 – Aggiunta di una querystring all'URL del container per visualizzare tutti i blob

Riapriamo il progetto precedente Windows Azure.

Aggiungiamo, se non è già presente, un riferimento alla libreria Microsoft.WindowsAzure.StorageClient (Figura 70).

Creiamo una nuova impostazione nella configurazione del Web Role in modo da immagazzinare le informazioni dell'account e sulla chiave di accesso. Fai doppio clic su WebRole1 nel nodo Ruoli (*Roles*) del progetto cloud, e quindi clicca su Aggiungi impostazione (*Add setting*) nella sezione Impostazioni (*Settings*). Nomina la nuova impostazione **DataConnectionString** e seleziona Stringa di connessione (*Connection String*) quale tipo (Figura 72).

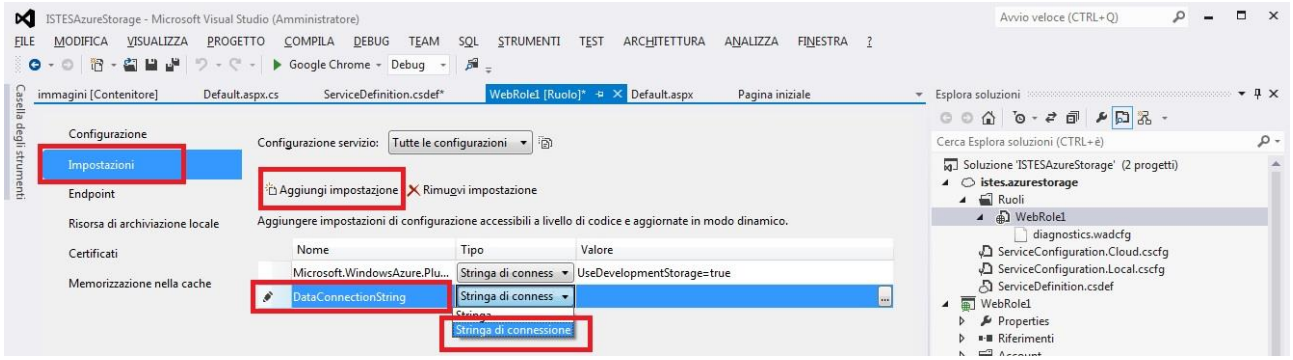


Figura 72 – Creazione di una nuova impostazione al WebRole

Clicca sul pulsante con l'ellissi (…) per aprire la finestra di dialogo Storage Connection String (Figura 73).

Aggiungiamo un nuovo elemento Web Form al progetto e lo nominiamo **StorageAccountBlobs**.

Inseriamo un controllo *Repeater* nella pagina e chiamiamolo **gridBlob**. Al suo interno collochiamo un *ItemTemplate* per mostrare la proprietà URI dei blob che ci apprestiamo a ricercare dallo Storage Account (Figura 74 e Listato 11).

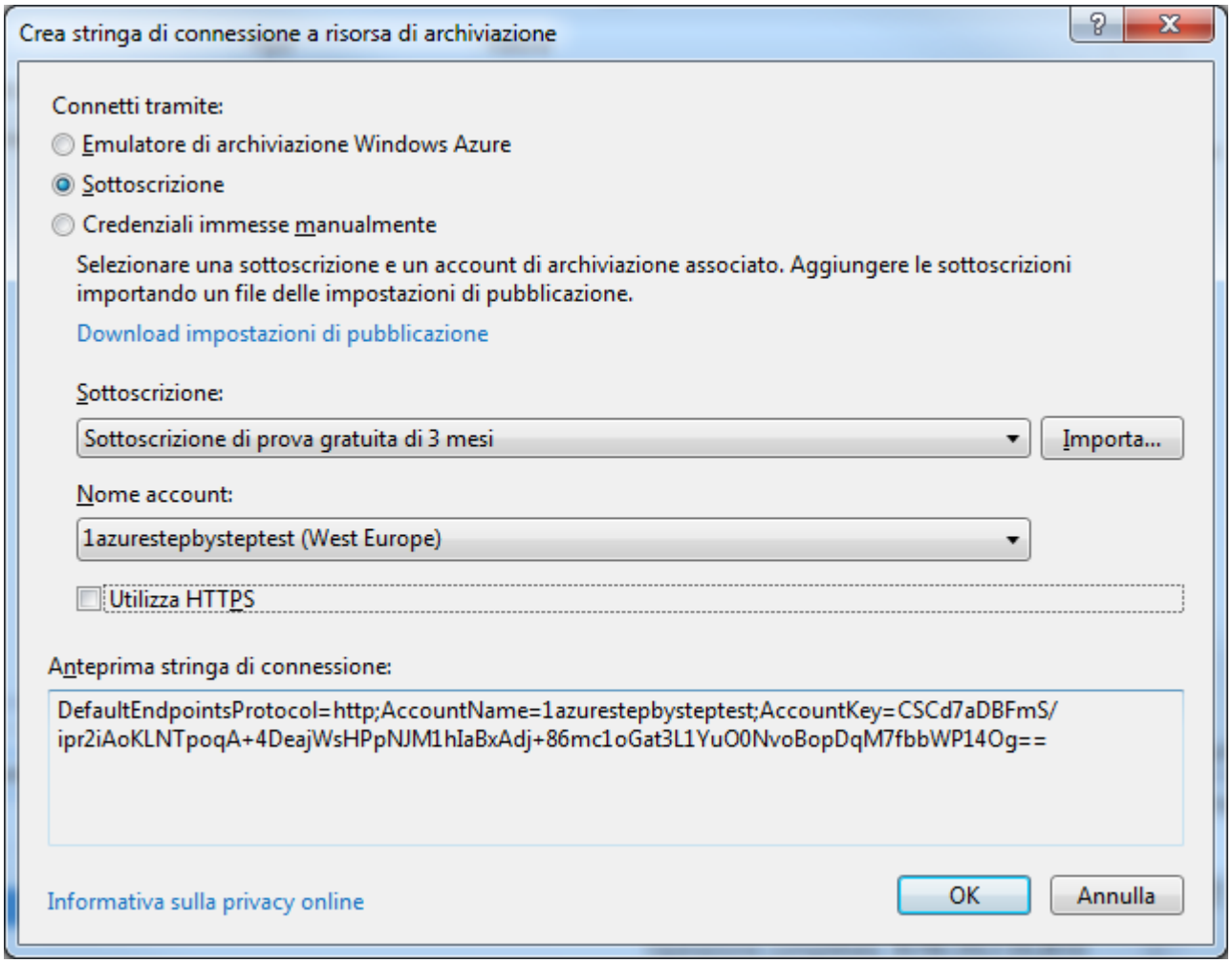


Figura 73 – Finestra di dialogo locale per la Storage Connection String

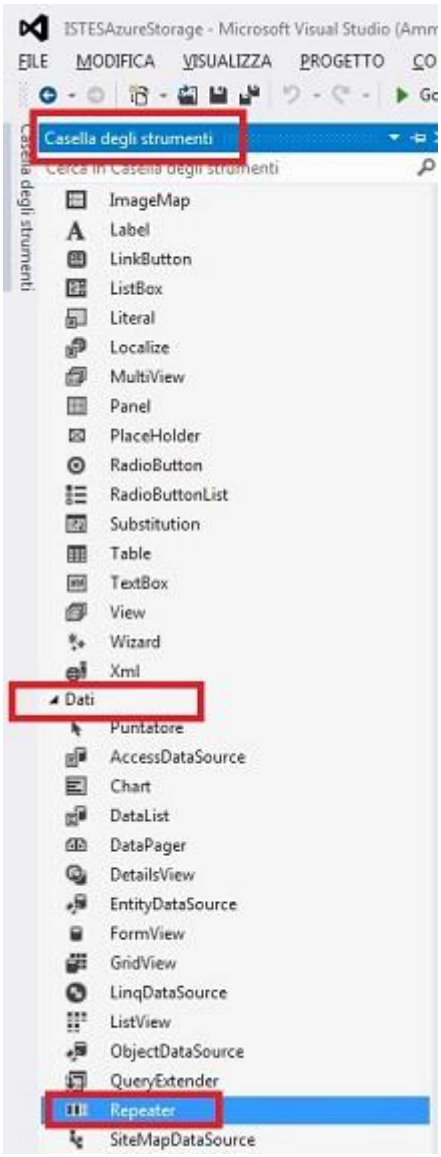


Figura 74 – Particolare del controllo repeater nella Caselal degli Strumenti

```
<%@ Page Language="C#" AutoEventWireup="true"

    CodeBehind="StorageAccountBlobs.aspx.cs"
    Inherits="WebRole1.StorageAccountBlobs" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:repeater ID="gridBlob" runat="server">
            <ItemTemplate>
                <p>
                    <%# Eval("Uri") %>
                </p>
            </ItemTemplate>
        </asp:repeater>
    </div>
    </form>
</body>
</html>
```

Listato 11 – File StorageAccountBlobs.aspx

Nel file contenente il *code behind*, **StorageAccountBlobs.aspx.cs**, bisogna mettere in *binding* i blob contenuti nel container **immagini** del nostro Storage Account al controllo *GridView*.

Le prime tre linee del metodo *Page_Load* recuperano le informazioni che abbiamo inserito nelle impostazioni di configurazione (Figura 73). Il metodo statico *FromConfigurationSettings* della classe *CloudStorageAccount* legge le informazioni nel file *ServiceConfiguration.cscfg* e restituisce un'istanza del tipo *CloudStorageAccount*. Questa istanza contiene tutte le informazioni per costruire un proxy per il servizio di storage account esposto da Windows Azure nel cloud ovvero localmente nel Windows Azure Storage Emulator. Il metodo *CreateCloudBlobClient* crea effettivamente il proxy necessario per compiere una qualsiasi operazione su blob e container.

Come si può vedere, il proxy ha un metodo per recuperare un riferimento (*reference*) al container, denominato *GetContainerReference*, il quale accetta il nome del container quale unico parametro. Una volta ottenuto un riferimento al container, possiamo vedere quali sono i blob ivi contenuti, semplicemente invocando il metodo *ListBlobs*. Questo metodo restituisce un tipo che implementa

IEnumerable>IListBlobItem>, che possiamo "bindare" (legare) a qualunque controllo che supportii il *data binding*.

```
namespace WebRole1
{
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.StorageClient;
    using System;

    public partial class StorageAccountBlobs : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            CloudStorageAccount account =
                CloudStorageAccount.FromConfigurationSetting("DataConnectionString");

            CloudBlobClient blobClient = account.CreateCloudBlobClient();

            CloudBlobContainer container = blobClient.GetContainerReference("immagini");

            gridBlob.DataSource = container.ListBlobs();
            gridBlob.DataBind();
        }
    }
}
```

Listato 12 – File StorageAccountBlobs.aspx.cs

Il codice è quasi completo ma la classe *CloudStorageAccount* necessita di alcuni ulteriori ritocchi nella configurazione. Digitiamo il codice sottostante nel metodo *Application_Start* del file *Global.asax.cs*, in modo da essere sicuri che qualunque cambiamento nei valori di configurazione dello storage account, determini un riciclo del ruolo (*role*) in modo da aggiornare i valori in cache. Il codice seguente proviene direttamente da MSDN:

```
        void Application_Start(object sender, EventArgs e)
        {
            // Codice eseguito all'avvio dell'applicazione
            AuthConfig.RegisterOpenAuth();
            RouteConfig.RegisterRoutes(RouteTable.Routes);

            #region Setup CloudStorageAccount Configuration Setting Publisher
            // questo codice imposta un gestore per aggiornare l'istanza
            // della classe CloudStorageAccount quando le loro impostazioni
            // di configurazione canbiano, nel file di configurazione del
            // servizio
            CloudStorageAccount.SetConfigurationSettingPublisher(
                (configName, configSetter) =>
                {
                    // fornisce la configurazione con un valore iniziale
                    configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));
                    RoleEnvironment.Changed += (s, arg) =>
                        {
                            if
(arg.Changes.OfType<RoleEnvironmentConfigurationSettingChange>().Any((change) =>
                                (change.ConfigurationSettingName == configName)))
                            {
                  // le impostazioni di configurazione sono cambiate
                  // propago il valore
                                if
(!configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))
```

```
                // In questo caso il cambiamento elle credenziali dello storage account
                // nel servizio di configurazione è così significativo che il ruolo
                // richiede di essere riciclato rispetto all'ultima configurazione
                // per esempio l'endpoint è cambiato
                                RoleEnvironment.RequestRecycle();

                            }
                        };
                });
            #endregion
```

Listato 13 – File Global.asax.cs con metodo Application_Start modificato

Impostiamo StorageAccountBlob.aspx come pagina di default.

Premiamo F5 per avviare il debug del nostro progetto.

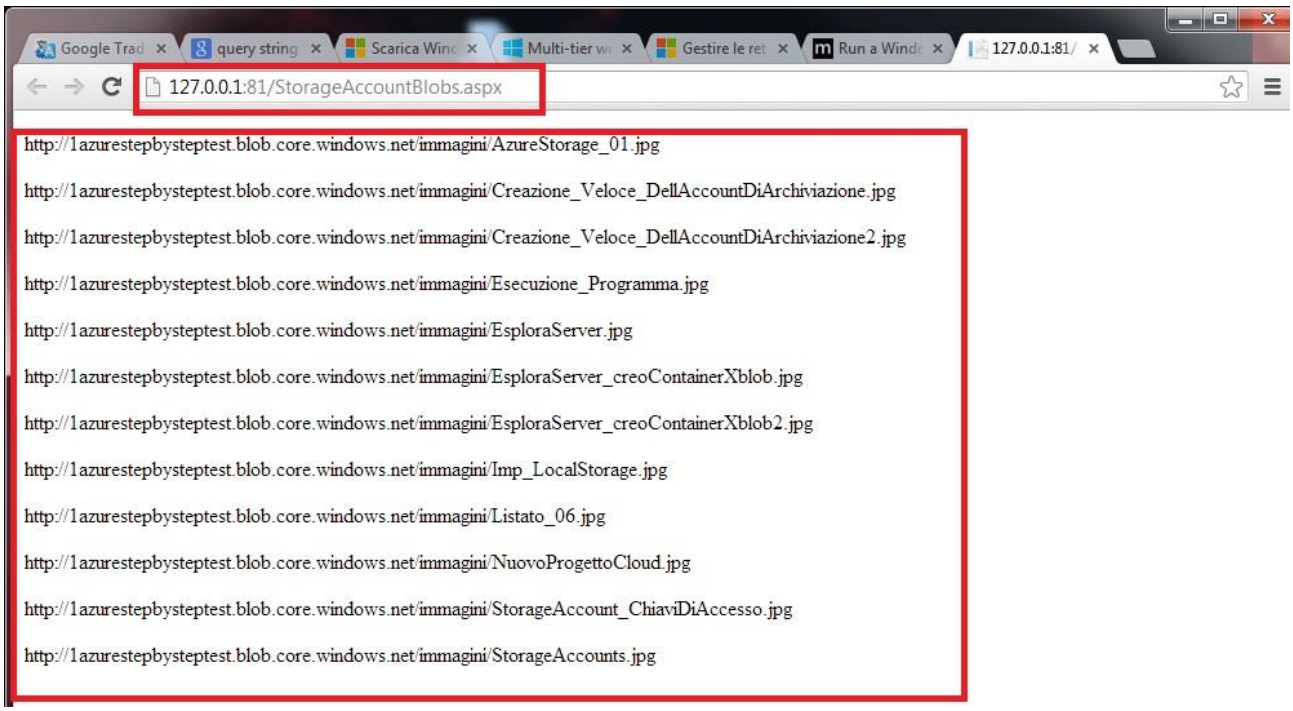


Figura 75 – Tutti i blob del container in locale ma con i riferimenti URL di Windows Azure

Vediamo la lista degli URI di ciascun blob nello Storage Account selezionato. Visto che l'URI punta ad un blob che rappresenta un'immagine, è possibile sostituire il codice del paragrafo HTML dell'elemento *ItemTemplate* del controllo *Repeater* con un tag *<img>* come mostrato nel codice seguente:

```
<asp:repeater ID="gridBlob" runat="server">
    <ItemTemplate>
        <img src='<%# Eval("Uri") %>' />
    </ItemTemplate>
</asp:repeater>
```

Listato 14 – File StorageAccountBlobs.aspx con controllo repeater modificato

Vedrai che la pagina del browser ti mostrerà tutte le immagini presenti nel container.

Nota:
il metodo *Eval* è semplice da usare, ma non è performante, in quanto usa *Reflection* per determinare il tipo di oggetto in *binding*. Un approccio migliore è quello di effettuare manualmente un cast verso la classe

messa in *binding*: non solo un tale codice si esegue più rapidamente, ma il compilatore può anche verificare la *type safety* del codice, cosa non possibile se si usa il metodo *Eval*. Il prossimo listato mostra il codice migliore per compiere questa operazione.

È importante notare l'uso della direttiva *Import Namespace* nella parte superiore della pagina.

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeBehind="StorageAccountBlobs.aspx.cs"
    Inherits="WebRole1.StorageAccountBlobs" %>
<%@ import Namespace="Microsoft.WindowsAzure.StorageClient" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <title>Mostra immagini</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
                <p>
                    <asp:Repeater ID="Repeater1" ID="GradBlob" runat="server">
                        <ItemTemplate>
                            <img src='<%#((IListBlobItem)Container.DataItem).Uri %>' />
                        </ItemTemplate>
                    </asp:Repeater>
                </p>
            </div>
        </form>
    </body>
</html>
```

Listato 15 – Web Form (StorageAccountBlobs.aspx) per testare blob con cast diretto

Upload di nuovi blob

La seguente procedura mostra come utilizzare le classi *StorageClient* per aggiungere una nuova immagine nello *Storage Account*. Sfrutteremo il controllo ASP.NET *File Upload* per caricare un'immagine nell'*hosted service*, che, a sua volta, la aggiungerà come blob nello *Storage Account*.

```
protected void upload_Click(object sender, EventArgs e)
{
     CloudStorageAccount account =
         CloudStorageAccount.FromConfigurationSetting("DataConnectionString");

     CloudBlobClient blobClient = account.CreateCloudBlobClient();

     CloudBlobContainer container = blobClient.GetContainerReference("immagini");

     CloudBlob blob = container.GetBlobReference(imageUpLoad.FileName);

     blob.UploadFromStream(imageUpLoad.FileContent);
}
```

Listato 16 – Metodo inserito nel code behind del file StorageAccountBlob.aspx.cs

La prima parte di codice non contiene niente di nuovo rispetto a quanto già visto nel metodo *Page_Load* usato negli esempi precedenti. Le righe successive del codice chiedono al container una *reference* al blob il cui nome corrisponde a quello passato come parametro al metodo *GetBlobReference*. Questo metodo non crea nulla, né si connette con lo Storage Account; si limita a creare una nuova istanza della classe *CloudBlob*

dotata di una URI completa, a partire da quella del container. L'ultima linea di codice crea un nuovo blob passando al servizio il contenuto del file ricevuto.

Se si esegue il codice, e si ha il contenitore (*container*) carico di immagini (… già da prima), compare una figura tipo la Figura 76.

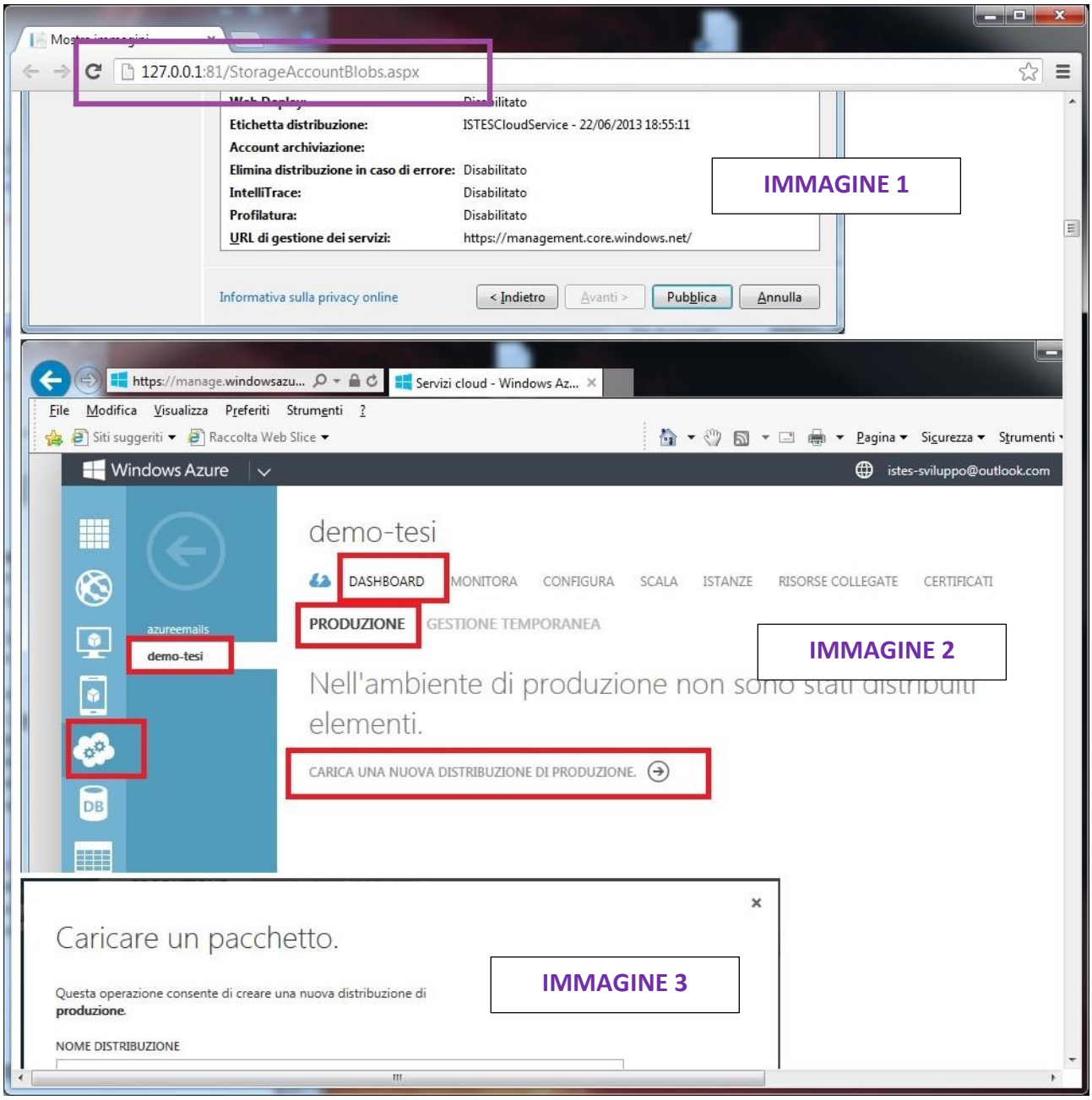


Figura 76 – I blob in sequenza in locale

Creare un container da codice

Per completare il nostro esempio, occorre aggiungere il codice per creare un container . Il posto perfetto è durante l'inizializzazione dell'applicazione. Dato che si tratta di una normale applicazione ASP.NET, possiamo collocare questo codice nell'*Application_Start*.

Bisogna aprire il file *Global.asax* e cercare il metodo *Application_Start*.

Subito dopo la regione *Setup CloudStorageAccount Configuration Setting Publisher* (aggiunta nell'esempio precedente), invochiamo il metodo *CreateIfNotExist* sull'istanza del *CloudContainer*.

```
    CloudStorageAccount account =
        CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
    CloudBlobClient blobClient = account.CreateCloudBlobClient();
    CloudBlobContainer container = blobClient.GetContainerReference("immagini");
    container.CreateIfNotExist();

    BlobContainerPermissions permissions = container.GetPermissions();
    permissions.PublicAccess = BlobContainerPublicAccessType.Container;
    container.SetPermissions(permissions);
}
```

Listato 17 – File Global.asax.cs con metodo Application_Start modificato

Il test mostra che si può creare un container via codice, mostrando l'utilità delle *Storage Client API*. Mediante una semplice modifica nel file di configurazione, si può adattare la soluzione in modo da testarla localmente o sul tuo reale Storage Account.

Per apportare questa modifica ed usare l'emulatore locale, apriamo le impostazioni di configurazione del Web Role e scegliamo l'opzione *Use Development Storage* per l'impostazione *DataConnectionString*.

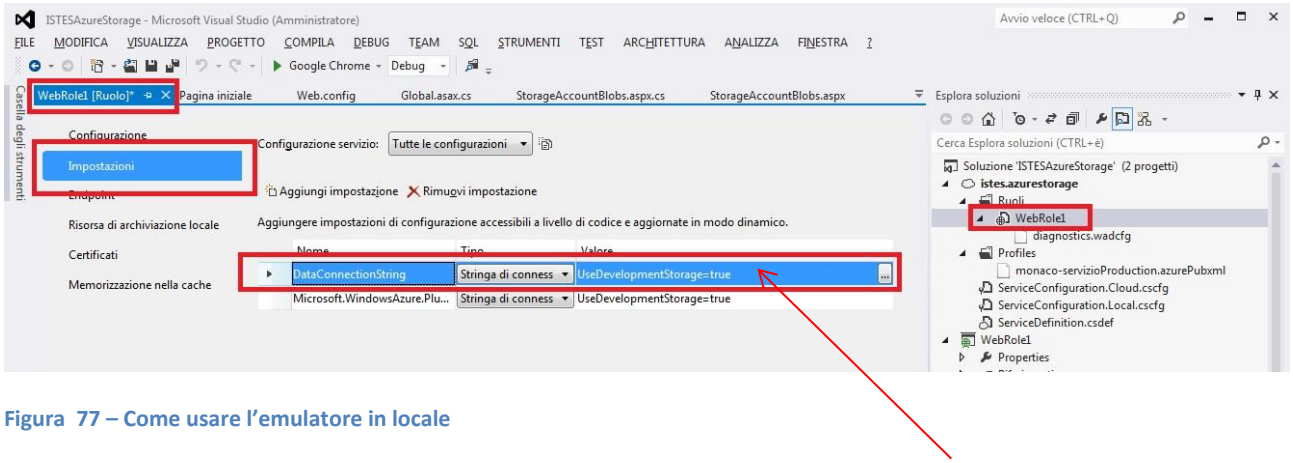


Figura 77 – Come usare l'emulatore in locale

Nella text box del *DataConnectionString* è sufficiente digitare ***UseDevelopmentStorage=true*** (come illustrato in Figura 77)

Adesso possiamo testare la soluzione aggiungendo nuove immagini al container creato da *Windows Azure Compute Emulator* mediante il codice che abbiamo inserito nel metodo *Application_Start*. Si può tornare sullo Storage Account online in qualsiasi momento, facendo attenzione durante le operazioni di *deployment*: se stiamo testando la soluzione usando lo storage di *Windows Azure Storage Emulator* e installiamo il file *ServiceConfiguration.cscfg* su Windows Azure, il *role* non funzionaerà, perché non esiste alcun *Windows Azure Storage Emulator* nel cloud.

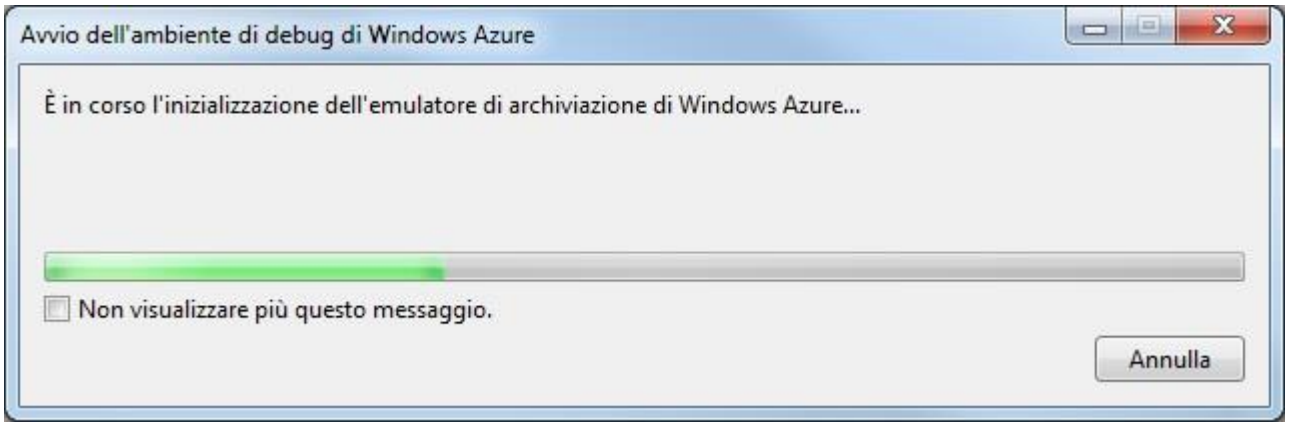


Figura 78 – Avvio dell'ambiente di debug

Si può usare l'interfaccia di Windows Azure Storage Emulator per avviare e fermare il servizio, o resettare le istanze, come è mostrato nella Figura 79.

Figura 79 – Interfaccia di Windows Azure Storage Emulator

L'emulatore è un processo Windows (*DDService.exe*) che simula i vari *endpoint* esposti per i blob, le tabelle (*table*) e le code (*queue*), usando differenti porte sulla macchina locale. Nella Figura 79, per esempio, la porta 10000 è usata per esporre il servizio di blob. Una richiesta al servizio locale deve utilizzare un diverso URI rispetto a quello del servizio cloud reale, ma la libreria dello *storage client* gestisce questi problemi al posto nostro. Se vogliamo lavorare a livello più basso e comporre la richiesta HTTP/REST manualmente, dobbiamo assicurarci di formare l'URI nel modo corretto.

Table, Queue e Worker Role

Lo Storage Account di Windows Azure è un posto perfetto per salvare non soltanto blob (inclusi quelli di grandi dimensioni), ma anche le entità dell'applicazione (*application entities*), come quelle di business, in tabelle (*tables*). Lo Storage Account fornisce inoltre un sistema di code (*queues*) che aiuta a disaccoppiare il front-end dal back-end.

La definizione ufficiale del Table Service (versione 2.0) data da Microsoft è la seguente:
"Il servizio di archiviazione *Table Service* memorizza grandi quantità di dati strutturati. Il servizio è un archivio dati *NoSQL* che accetta chiamate autenticate da dentro e fuori il cloud di Windows Azure. Le tabelle (*tables*) di Windows Azure sono l'ideale per l'archiviazione di dati strutturati ma non relazionali".

Il Table Service è esposto attraverso API REST per poter lavorare con le tabelle e i dati in esse contenuti.

Lo storage locale non è il posto più adatto per memorizzare i dati permanenti o temporanei, perché si trova in locale rispetto alla specifica macchina virtuale sulla quale l'applicazione è in esecuzione. Una strada per salvare i dati in modo che possano sopravvivere a eventuali crash dell'applicazione o del nodo, ed essere condivisi tra diversi nodi, è costituita dal Blob Service, che, come si è visto, può essere esposto all'esterno.

Gli stessi concetti visti per il Blob Service si applicano al Table Service. Lo store è esposto via HTTP e può essere acceduto da qualunque applicazione in esecuzione su qualsivoglia piattaforma, dal momento che utilizza il protocollo standard REST. L'accesso ai dati avviene attraverso una chiave condivisa. La libreria *StorageClient* include classi e metodi che semplificano il codice necessario a gestire i dati tramite il Table Service.

Nonostante queste analogie tra i Blob Service e i Table Service, i due servizi presentano anche notevoli differenze. Il Table Service:

- Organizza i dati in tabelle piuttosto che in container.
- Ciascuna tabella è rappresentata da un insieme flessibile di proprietà, mentre un container si limita a contenere i file.
- Ciascuna tabella è composta da righe, ciascuna delle quali rappresenta un'entità, non un file.
- Gli sviluppatori hanno il controllo dell'organizzazione logica delle tabelle. Per esempio è possibile partizionare una tabella usando una chiave specifica per suddividerla su specifici nodi. Al contrario, la gestione dei blob è automatizzata.
- Ognuna delle righe di una tabella dispone di una chiave primariache identifica l'entità che contiene; un blob e identificato da un file e dal un suo nome.

I servizi di storage sono indipendenti dall'applicazione ospitata sul cloud: in altre parole, qualunque client può accedere al servizio di storage usando il protocollo HTTP.

Per usare il Table Service è necessario innanzitutto creare uno Storage Account sul portale di Windows Azure. Al momento della creazione di uno Storage Account, Windows Azure alloca una certa quantità di spazio in alcuni dei nodi che si trovano nel data center selezionato, in modo da poter salvare i dati, per gestire i quali il sistema fornisce un URL che si può usare nelle query REST. In questa sezione, useremo l'URL http://account.table.core.windows.net, dove con il termine account si intende il nome univoco scelto per lo Storage Account pubblico.

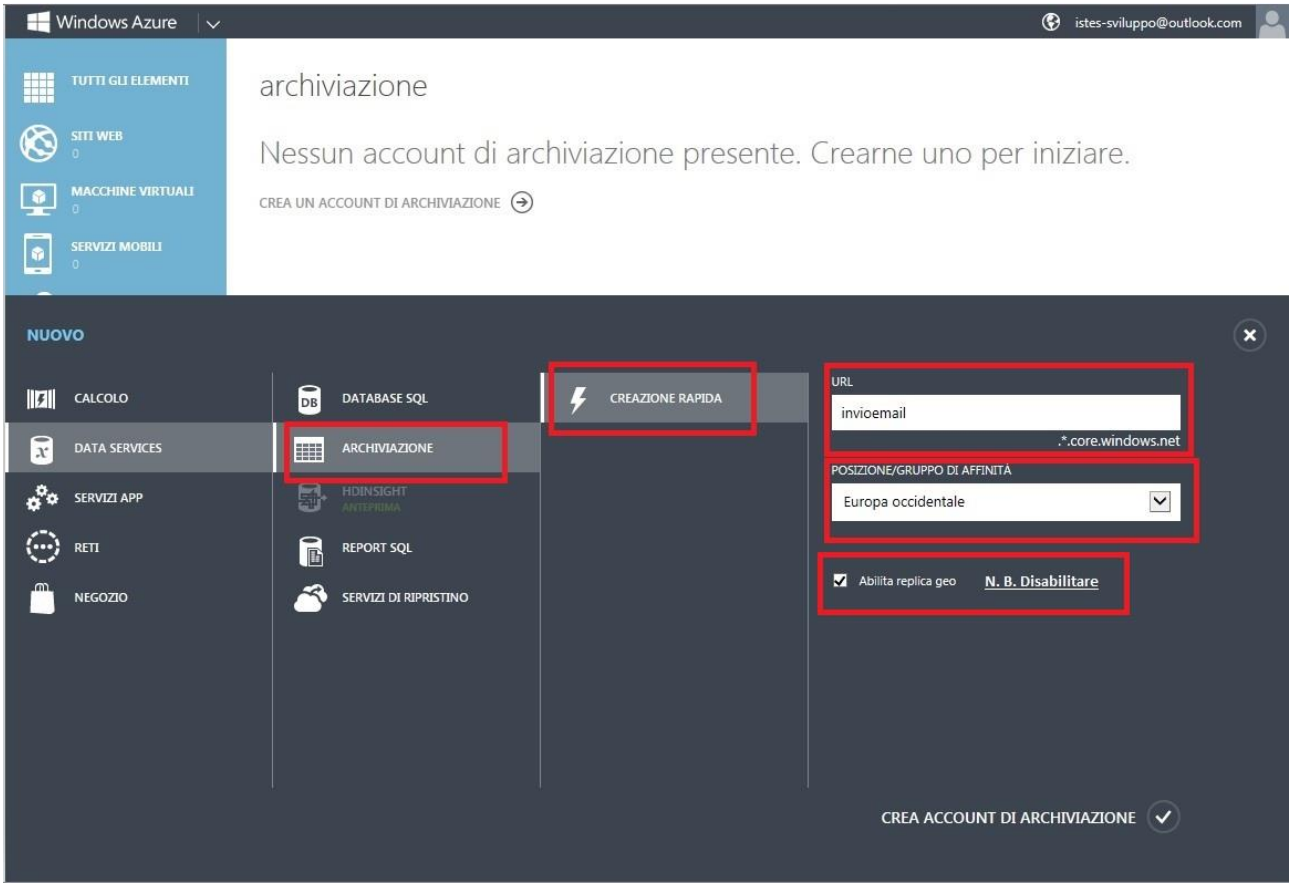


Figura 80 – Creazione di una account di archiviazione sul cloud Windows Azure

Usare le API del Table Service

Riprendiamo l'ultima soluzione creata con Visual Studio 2012. Nel progetto cloud fare doppio clic sul WebRole1, nella sezione Ruoli (*Roles*); di seguito bisogna cliccare per modificare la stringa di connessione Figura 81

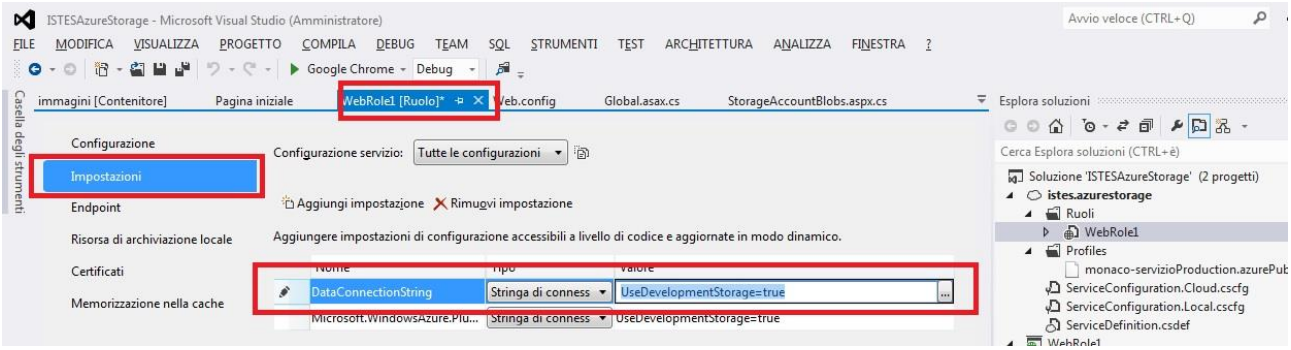


Figura 81 – Modifica della stringa di connessione nella impostazioni del progetto cloud

Bisogna cliccare sul pulsante con le ellissi (…) per aprire la finestra di dialogo Storage Account Connection String mostrata nella Figura 82. Il Development Storage locale simula il Table Service nello stesso modo in cui simula il Blob Service. Il primo dei due risponde alle richieste indirizzate dal localhost utilizzando la porta 10002, mentre il secondo si riserva la porta 10001 (Figura 79).

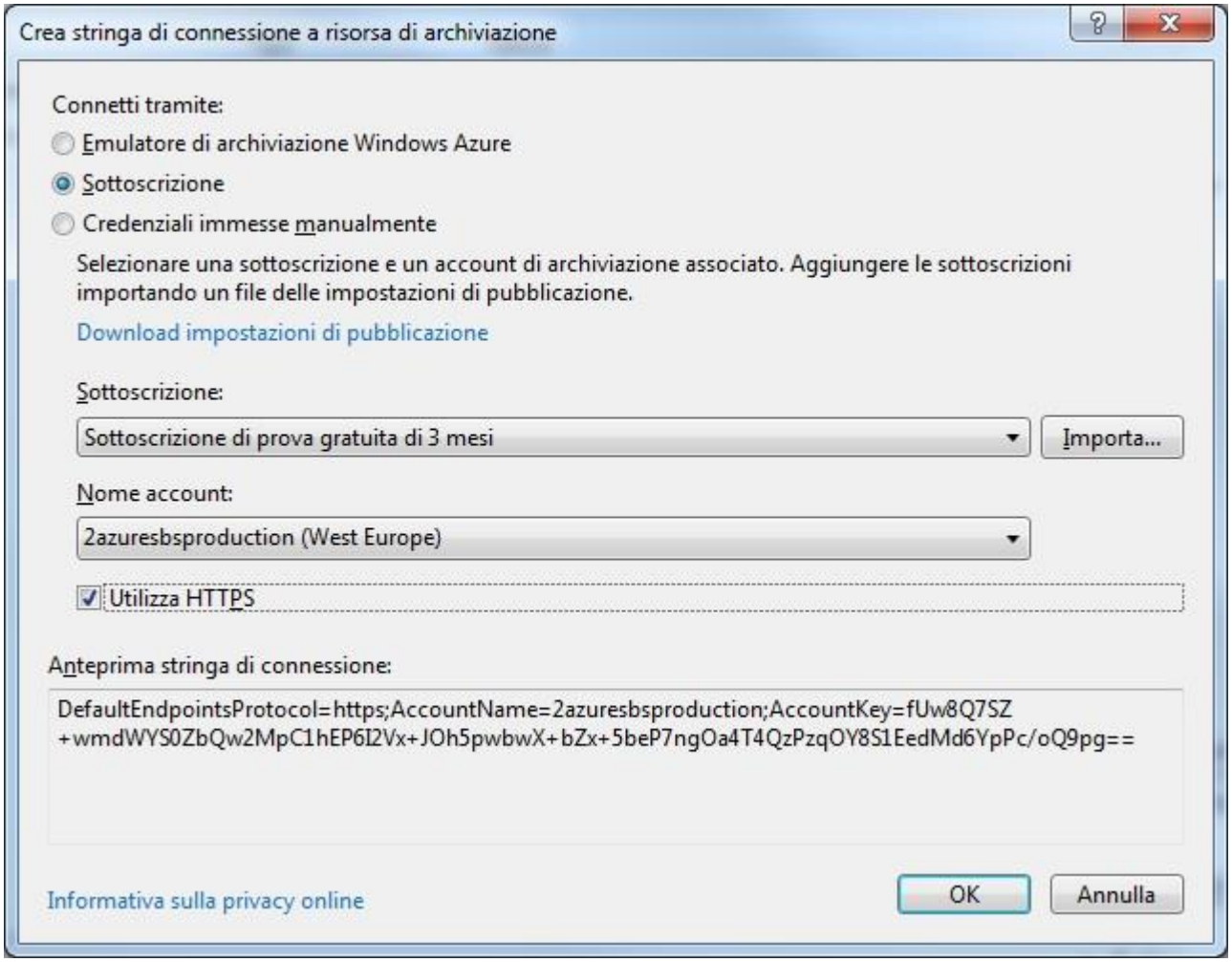


Figura 82 – Finestra di conferma dell'account di archiviazione Windows Azure

Prima di gestire le entità usando il Table Service, bisogna definire la struttura della riga che vogliamo usare per le operazioni di salvataggio o lettura. Una tabella può contenere entità eterogenee, ciascuna con una diversa struttura rispetto alle altre. In questo modo si può usare la stessa tabella per salvare, ad esempio, clienti e ordini.

Negli scenari più comuni si preferisce usare più tabelle per salvare per salvare differenti tipi di entità, come si fa con un classico database; ma dobbiamo tenere presente che il Table Service salva i dati in uno store **non relazionale**. Il servizio non può introdurre un vincolo (*constraint*) su un'entità: spetta farlo al programmatore via codice. Da questo punto di vista, non c'è differenza tra immagazzinare entità diverse in tabelle separate, rispetto a immagazzinare le medesime entità, tutte insieme, nella stessa tabella.

La chiave di partizione è uno dei tre campi richiesti per ciascuna tabella. Ciascuna riga necessita di un campo *RowKey* che, assieme al campo *PartitionKey*, forma quello che possiamo considerare la chiave primaria di quella entità all'interno dello store. Il terzo campo è un timestampautogenerato definito *Timestamp*.

La libreria *StorageClient* espone una classe base denominata *TableServiceEntity*, la qualecontiene questi tre campi. I campi PartitionKey e RowKey sono rappresentati da due stringhe, mentre il Timestamp è posto dal tipo .NET *DateTime*.

Definire un'Entità

Aggiungiamo una nuova classe al progetto nominandola *Message*. Diamo un identificatore di accesso pubblico e deriviamola da *TableServiceEntity*.

```
namespace WebRole1
{
    using Microsoft.WindowsAzure.StorageClient;

    public class Message : TableServiceEntity
    {
        public string Body { get; set; }

    }
}
```

Listato 18 – File Message.cs del progetto WebRole1

Useremo questa classe Message per immagazzinare il testo di un messaggio che l'utente dovrà inserire in una pagina (che creeremo a breve). Prima di questo, però, dobbiamo creare il contesto (*context*) che incapsulerà ogni richiesta al Table Service.

Creare il Contesto Client-Side

Aggiungiamo una nuova classe al progetto e nominiamola *MessageServiceContext*.

```
namespace WebRole1
{
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.StorageClient;
    using System;

    public class MessageServiceContext : TableServiceContext
    {
        public MessageServiceContext(string baseAddress,
            StorageCredentials credentials) : base(baseAddress, credentials)
        {

        }

        public void AddMessage(string body)
        {
            Message m = new Message();
            m.PartitionKey = "A";
            m.RowKey = Guid.NewGuid().ToString();

            m.Body = body;

            this.AddObject("Messages", m);
            this.SaveChanges();
        }
    }
}
```

Listato 19 – File MessageServiceContext.cs del progetto WebRole1

Abbiamo inserito un costruttore che invoca, a sua volta, il costruttore della classe base, passandogli le credenziali dello storage e l'indirizzo base. Non c'è niente di speciale da fare in questo costruttore, perché sarà quello della classe base a occuparsi di recuperare le informazioni necessarie leggendole dal file di configurazione. Il costruttore accetta due parametri e li passa direttamente al costruttore della classe base. Il metodo *AddMessage* accetta una stringa che rappresenta il testo del messaggio e crea un'istanza della

classe *Message*. Questo metodo è utile per evitare duplicazioni nell'assegnazione della *PartitionKey*, della *RowKey* e del nome della corrispondente tabella (*Messages*)[49].

Adesso dobbiamo creare una pagina ASP.NET che prenda un messaggio inserito dall'utente, crei una nuova entità *Message* e l'aggiunga al *context*, chiedendo a quest'ultimo di salvare il messaggio nel Table Service.

Usare il Table Service

Il *Context* lato client rappresentato dalla classe *AzureStorageServiceContext*, creata nella precedente procedura, ci permette di compiere qualunque operazione di Create, Read, Update e Delete (CRUD) definita nello standard OData.

Nel prossimo esempio creeremo una semplice applicazione di chat. Gli utenti possono inserire un messaggio nella pagina ASP.NET e il contesto salverà il messaggio in una riga della tabella Messages. La pagina visualizzerà la lista dei messaggi salvati, recuperandoli dal Table Service.

Creiamo una pagina ASP.NET Web Form denominata *StorageAccountTable*. Inseriamo un controllo *TextBox* in cui l'utente digiterà un messaggio e denominiamolo *messageTextBox*. Inseriamo un controllo *Button* sotto la *TextBox* e assegniamo il nome *addButton*. Impostiamo l'attributo *OnClick* del controllo *Button* a *addButton_Click* per mettere in *binding* il click con il corrispondente *event handler* (**Errore. L'origine riferimento non è stata trovata.**).

Errore. L'origine riferimento non è stata trovata.
Nel code-behind della pagina, *StorageAccountTable.aspx.cs*, recuperiamo le informazioni relative all'account dalla classe CloudStorageAccount e le passiamo al costruttore della classe MessageServiceContext creata nellesempio precedente (Listato 19).

```
namespace WebRole1
{
    using Microsoft.WindowsAzure;
    using System;

    public partial class StorageAccountTable : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void addButton_Click(object sender, EventArgs e)
        {
            var account = CloudStorageAccount.
                FromConfigurationSetting("DataConnectionString");
            var context = new MessageServiceContext(
                account.TableEndpoint.ToString(), account.Credentials);

            context.AddMessage(messageTextBox.Text);
        }
    }
}
```

Listato 20 – File StorageAccountTable.aspx.cs

[49] Per evitare di complicare troppo il codice della pagina chiamante, possiamo esporre alcuni metodi che incapsulano le linee di codice necessarie per istanziare la classe *Message*, assegnarle una *PartitionKey* e una *RowKey* in modo automatico e aggiungere l'istanza alla **tabella Messages**.

La prima linea dell'*event handler* denominato *addButton_Click* legge il valore dell'impostazione *DataConnectionString* usata per creare la classe *CloudStorageAccount* (come abbiamo già visto per il Blob Service). La terza linea di codice invoca il metodo *AddMessage*, che compie l'operazione di aggiungere il messaggio allo storage.

Infine, ma non meno importante, bisogna creare la tabella che conterrà i messaggi. Analogamente a quanto già visto nella precedente sezione riguardante i blob, possiamo creare la tabella per queste righe nel metodo *Application_Start* del file *Global.asax.cs* (come mostrato nel seguente codice).

```
    …
    CloudStorageAccount account =
        CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
    CloudBlobClient blobClient = account.CreateCloudBlobClient();
    CloudBlobContainer container = blobClient.GetContainerReference("immagini");
    container.CreateIfNotExist();

    BlobContainerPermissions permissions = container.GetPermissions();
    permissions.PublicAccess = BlobContainerPublicAccessType.Container;
    container.SetPermissions(permissions);

    CloudTableClient tableClient = account.CreateCloudTableClient();
    tableClient.CreateTableIfNotExist("Messages");
}
```

Listato 21 – File Global.asax.cs con metodo Application_Start modificato

Prima di testare la soluzione, bisogna ricordarsi di impostare la pagina StorageAccountTable.aspx come Pagina iniziale (*start page*) del progetto (al momento non abbiamo alcuna gestione degli errori). Abbiamo una figura come questa.

Figura 83 – Finestra browser in locale per archiviare dati nella tabella

Effettuare una Query sul Table Service

Adesso possiamo aggiungere una *query* alla pagina per richiedere ed eventualmente filtrare, i messaggi salvati nello Storage Account. La procedura per compiere queste operazioni è molto simile a quella vista in precedenza; bisogna aggiungere una proprietà pubblica per esporre una *query* che incapsuli la logica necessaria a richiedere i messaggi al servizio e quindi invocare tale metodo dall'evento *Load* della pagina, in modo da visualizzare l'elenco dei messaggi salvati.

Il metodo *CreateQuery<T>* è ereditato dalla classe *DataSeerviceContext*. Il metodo restituisce un'interfaccia *IQueryable* che rappresenta l'*expression tree* a partire dal quale il codice può costruire la query. Il metodo *CreateQuery* richiede il nome della tabella in cui salvare l'entità. Lo scopo di questo lavoro è incapsulare il nome della tabella e le operazioni di costruzione della query all'interno di una classe, per liberare il codice

della pagina dal dover compiere più volte lo stesso compito. In questo modo da una pagina, dobbiamo solo richiedere la proprietà *Messages* ogni volta che vogliamo costruire una query per il Table Service per ottenere le entità *Message*.

Nell'evento *Page_Load*, creiamo un'istanza del *CloudStorageAccount* usando il metodo statico *FromConfigurationSetting*. Usiamo l'istanza per passare il parametro al costruttore della classe *MessageServiceContext.cs*. Aggiungiamo il controllo *Repeater* alla pagina *StorageAccountTable.aspx*.

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeBehind="StorageAccountTable.aspx.cs"
    Inherits="WebRole1.StorageAccountTable" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <title>Inserisci messaggio</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
                <p>
                    <asp:TextBox ID="messageTextBox" runat="server" />
                </p>
                <p>
                    <asp:Button Text="Aggiungi" ID="addButton" runat="server"
                        OnClick="addButton_Click"></asp:Button>
                </p>
                <p>
                    <asp:Repeater ID="gridTable" runat="server">
                        <ItemTemplate>
                            <p>
                                <%#((WebRole1.Message)Container.DataItem).Body %>
                            </p>
                        </ItemTemplate>
                    </asp:Repeater>
                </p>
            </div>
        </form>
    </body>
</html>
```

Listato 22 – File StorageAccountTable.aspx

```
namespace WebRole1
{
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.Storage.Table;
    using Microsoft.WindowsAzure.StorageClient;
    using System;

    public partial class StorageAccountTable : System.Web.UI.Page
    {
        /// <summary>
        /// costruttore della classe
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void Page_Load(object sender, EventArgs e)
        {
            var account =
                CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
            var context = new MessageServiceContext(
                account.TableEndpoint.ToString(), account.Credentials);

            gridTable.DataSource = context.Messages;
            gridTable.DataBind();

        }

        /// <summary>
        /// event handler
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void addButton_Click(object sender, EventArgs e)
        {
            var account = CloudStorageAccount.
                FromConfigurationSetting("DataConnectionString");
            var context = new MessageServiceContext(
                account.TableEndpoint.ToString(), account.Credentials);

            context.AddMessage(messageTextBox.Text);
        }
    }
}
```

Listato 23 – Il code-behind della pagina StorageAccountTable.aspx.cs

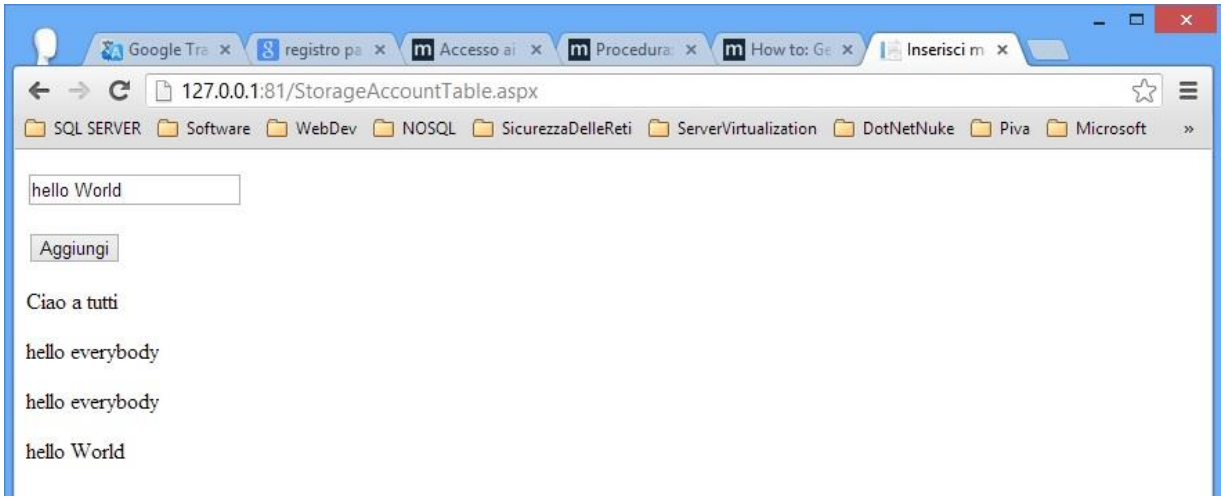


Figura 84 – Parole inserite nella table dello Storage Account

Dando un'occhiata più da vicino, la classe *TableServiceContext*, costruisce una *query REST* e la invia al Table Service configurato nel file *ServiceConfiguration.cscfg*, impiegando l'*header* di autorizzazione con il valore della chiave condivisa.

Il Queue Service

Nelle precedenti sezioni abbiamo visto come sfruttare in servizio di blob (Blob Service) per immagazzinare e recuperare file dallo Storage Account. Di seguito abbiamo introdotto il Table Service, con il quale è possibile gestire le entità di un'applicazione. Ma esiste anche un altro tipo di dato nello Storage Account: un messaggio. Così come un blob risiede in un contenitore (*container*), un'entità risiede in una tabella (*table*), un messaggio risiede in una coda (*queue*).

Come indica il nome stesso, una coda non è un posto nel quale un software inserisce, modifica o legge dati, quanto piuttosto un luogo in cui un'applicazione può salvare messaggi che normalmente rappresentano un'operazione ancora da completare. Questi messaggi possono essere recuperati in un secondo momento dall'applicazione per processarli.

I messaggi possono anche essere tolti dalla coda, da una diversa applicazione responsabile della gestione di quei processi.

Lo Storage Account nel cloud espone un servizio intelligente di gestione delle code che permette di disaccoppiare un'applicazione da un'altra. Il primo esempio che balza in mente è il disaccoppiamento del front-end di Azure, il Web Role, rispetto al back end, o Worker Role.

Un Worker Role è semplicemente un Role che, di default, non è esposto all'esterno ed è dedicato al compimento di determinate operazioni nel back end. Un esempio di progetto Worker Role è rappresentato da un'applicazione per la gestione degli ordini, i quali arrivano sotto forma di messaggi creati dal front end. In questo tipo di approccio, una volta che il front end ha inserito il messaggio nella coda, è libero di passare ad altro. Se, invece, tutti gli ordini provenienti dagli utenti dovessero essere processati immediatamente dalla pagina (o dal Business Layer) del front end, il Web Role non risulterebbe scalabile; infatti fintantoché un thread è impegnato a processare un ordine, non può servire altre richieste, come invece sarebbe possibile se l'intero processo fosse spostato nel back end.

Non ci sono differenze in ciò che un Worker Role può fare rispetto ad un Web Role. La differenza principale tra i due è che un Worker Role è e rimane in esecuzione nell'ambiente di prova (*staging*) o di produzione, dove continua a ripetere in loop le sue funzioni, mentre un Web Role rimane dietro al protocollo HTTP/HTTPS e riceve le richieste che arrivano da internet o da altri role nel cloud.

Vedremo ora come inserire un messaggio nella coda usando il Web Role e come recuperare il messaggio mediante il Worker Role. L'esempio illustrato in questa sezione usa un messaggio che contiene soltanto una stringa che rappresenta l'ordine da passare dal front end al back end. Useremo questo approccio per inviare un ordine serializzato.

Nel progetto creiamo una nuova pagina ASP.NEDT denominata *StorageAccountQueue*.

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeBehind="StorageAccountQueue.aspx.cs"
    Inherits="WebRole1.StorageAccountQueue" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <title>Coda</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
                <p>
                    <asp:TextBox ID="orderTextBox" runat="server"></asp:TextBox>
                </p>
                <p>
                    <asp:Button ID="addButton" runat="server"
                        Text="Add" OnClick="addButton_Click" />
                </p>
            </div>
        </form>
    </body>
</html>
```

Listato 24 – File StorageAccountQueue.aspx

```
namespace WebRole1
{
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.StorageClient;
    using System;

    public partial class StorageAccountQueue : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void addButton_Click(object sender, EventArgs e)
        {
            var account =
                CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
            CloudQueueClient queueClient = account.CreateCloudQueueClient();

            CloudQueue queue = queueClient.GetQueueReference("orders");
            CloudQueueMessage m = new CloudQueueMessage(orderTextBox.Text);
            queue.AddMessage(m);
        }
    }
}
```

Listato 25 – Il code behind della pagina StorageAccountQueue.aspx.cs

Questo codice è molto simile a quello usato per il Blob Service; anche in questo caso è necessario ottenere un servizio proxy per cominciare a lavorare con lo Storage Account. La classe *CloudStorageAccount* fornisce un metodo denominato *CreateCloudQueueClient* che funge proprio a questo scopo. Il metodo restituisce un oggetto *CloudQueueClient* che rappresenta il proxy per il Queue Service. È sufficiente invocare il metodo

GetQueueReference("nomecoda") per ottenere una reference ad una coda, e quindi, usare la classe CloudQueue per inserire o leggere messaggi in o dalla coda.

La classe *CloudQueueMessage* rappresenta un messaggio che verrà serializzato automaticamente in modo da passare il contenuto nella sottostante richiesta REST.

Creiamo la coda nello stesso posto in cui abbiamo creato la tabella nella sezione precedente, ossia all'interno del metodo *Application_Start* della classe Global, nel file *Global.asax.cs*, come mostrato nel seguente estratto:

```
.....
    CloudQueueClient queueClient = account.CreateCloudQueueClient();
    CloudQueue queue = queueClient.GetQueueReference("orders");
    queue.CreateIfNotExist();
}
```

Listato 26 – File Global.asax.cs con modifica del metodo Application_Start

Impostiamo la nuova pagina *StorageAccountQueue.aspx* come pagina di avvio ed eseguiamo il progetto.

Creare il progetto Worker Role

Dopo avere configurato il role in modo che punti allo stesso Storage Account del progetto WebRole, scriveremo del codice per togliere i messaggi dalla coda degli ordini.

Dalla finestra Esplora Soluzioni (*Solution Explorer*) espandi il progetto cloud e clicca sulla cartella Ruoli (*Roles*) con il tasto destro. Nel menù contestuale scegli aggiungi e di seguito Nuovo progetto di Ruolo di lavoro

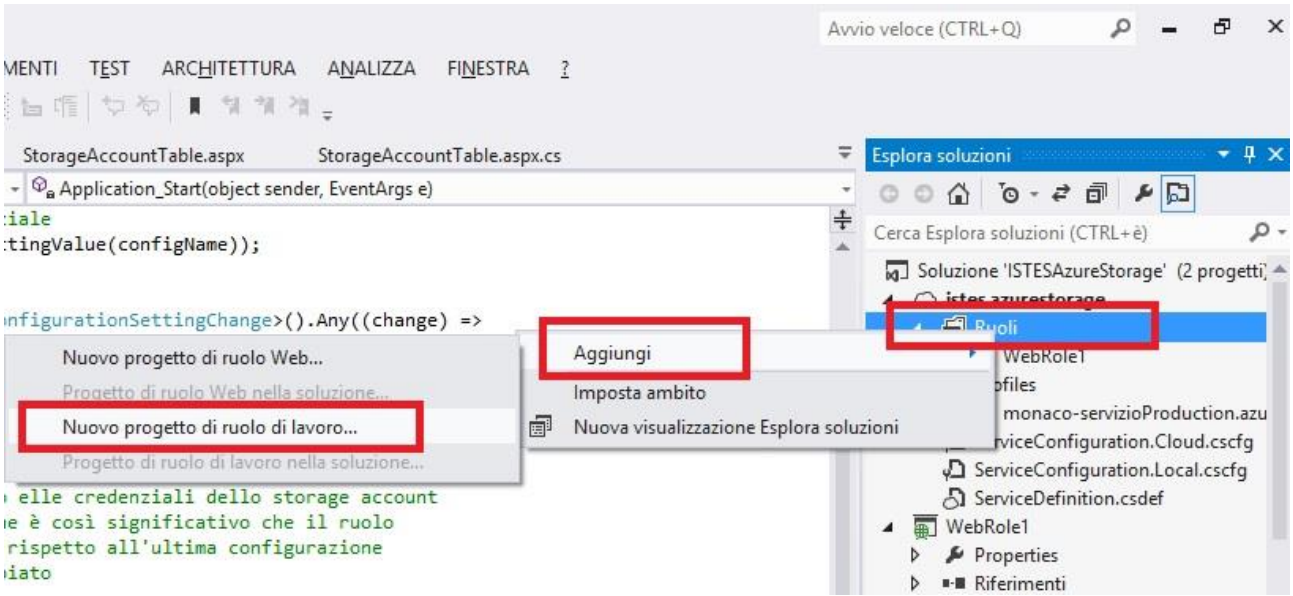


Figura 85 – Creazione di un progetto Worker Role passo 1

Nella finestra che si è aperta verifica che nel pannello di sinistra sia selezionato il linguaggio C# e nel pannello di destra controlla che il *template* selezionato sia Ruolo di Lavoro (*Worker Role*)

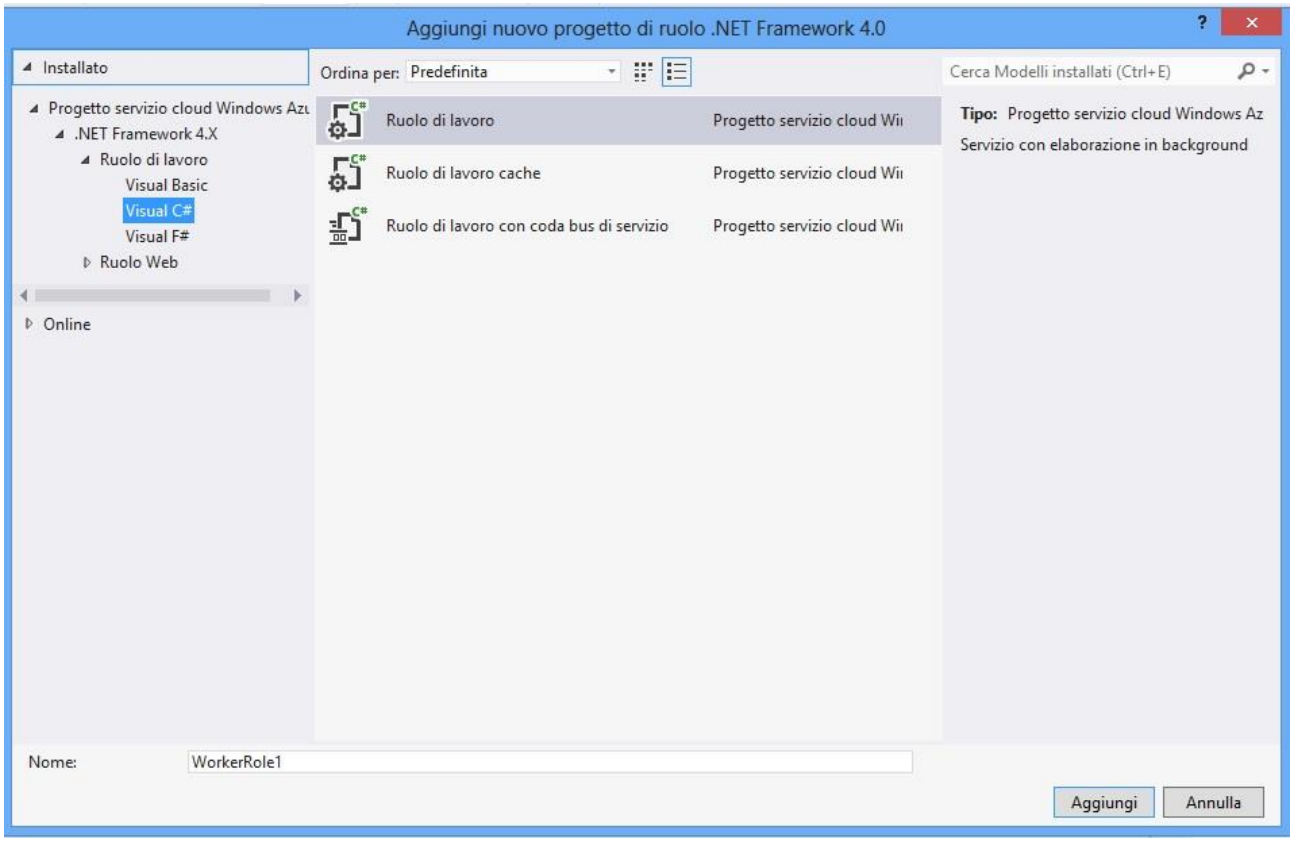


Figura 86 – Creazione di un progetto Worker Role passo 2

Lasciamo pure il nome WorkerRole1 e clicchiamo su aggiungi.

La soluzione adesso contiene un progetto WebRole, un progetto WorkerRole ed un progetto cloud.

Configurare il progetto Worker Role

Per prima cosa dobbiamo aprire l'editor visuale o il file *ServiceConfiguration.Cloud.cscfg* e il file *ServiceConfiguration.Local.cscfg* per configurare questo nuovo progetto con le informazioni relative allo Storage Account.

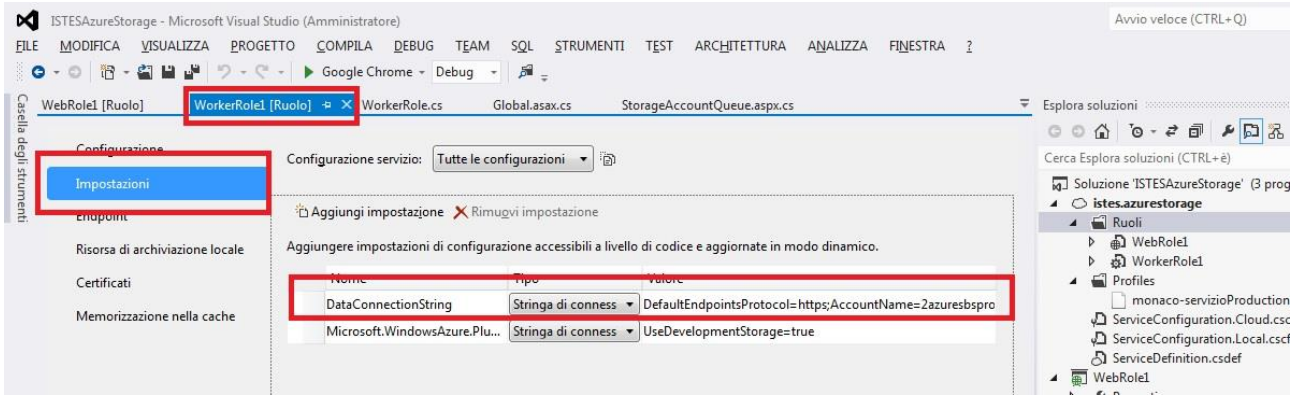


Figura 87 – Configurazione del progetto Worker Role

Adesso il nuovo progetto Worker Role possiede un puntatore allo Storage Account usato per immagazzinare il messaggio.

Ricordiamo che un progrtto richiede il codice per configurare il *Configuration Setting Publisher* prima di poter utilizzare la classe *CloudStorageAccount*.

Apriamo il file *Global.asax.cs* nel progetto *WebRole1*, copiamo la regione di codice definita *Setup CloudStorageAccount Configuration Setting Publisher*, all'interno del metodo *Application_Start* , apriamo il file *WorkerRole.cs* nel progetto *WorkerRole1*, troviamo il metodo *OnStart* e incolliamo il codice prima dell'istruzione *return*.

```
namespace WorkerRole1
{
    using System.Diagnostics;
    using System.Linq;
    using System.Net;
    using System.Threading;
    using Microsoft.WindowsAzure.ServiceRuntime;

    public class WorkerRole : RoleEntryPoint
    {
        public override void Run()
        {
            // Implementazione di lavoro di esempio. Sostituire con la logica
personalizzata.
            Trace.TraceInformation("WorkerRole1 entry point called", "Information");

            while (true)
            {
                Thread.Sleep(10000);
                Trace.WriteLine("Working", "Information");
            }
        }

        public override bool OnStart()
        {
            // Impostare il numero massimo di connessioni simultanee
            ServicePointManager.DefaultConnectionLimit = 12;

            #region Setup CloudStorageAccount Configuration Setting Publisher

            // questo codice imposta un gestore per aggiornare l'istanza
            // della classe CloudStorageAccount quando le loro impostazioni
            // di configurazione canbiano, nel file di configurazione del
            // servizio
            Microsoft.WindowsAzure.CloudStorageAccount.SetConfigurationSettingPublisher(
                (configName, configSetter) =>
                {
                    // fornisce la configurazione con un valore iniziale
                    configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));
                    RoleEnvironment.Changed += (s, arg) =>
                    {
                        if
(arg.Changes.OfType<RoleEnvironmentConfigurationSettingChange>().Any((change) =>
                            (change.ConfigurationSettingName == configName)))
                        {
                            // le impostazioni di configurazione sono cambiate
                            // propago il valore
                            if
(!configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))

                                // In questo caso il cambiamento elle credenziali dello
storage account
                                // nel servizio di configurazione è così significativo che
il ruolo
```

```
                                // richiede di essere riciclato rispetto all'ultima
configurazione
                                // per esempio l'endpoint è cambiato
                                RoleEnvironment.RequestRecycle();
                            }
                        };
                });
            #endregion

            return base.OnStart();
        }
    }
}
```

Listato 27 – File Globalasax.cs (copiamo la regione di codice definita CloudStorageAccount Configuration Setting Publisher)

Come si vede, il template propone un loop infinito con ritardo di 10 secondi tra due successive operazioni. È possibile fare qualsiasi cosa in questo metodo: chiamare un servizio web esterno, accedere al Table Service o al Blob Service, usare Database SQL o, come vedremo nel prossimo esempio, "scodare " e processare un messaggio.

Creiamo un'istanza della classe *CloudStorageAccount* usando il metodo statico *FromConfigurationSetting*. Di seguito creiamo il proxy per il Queue Service chiamando il metodo *CreateCloudQueueClient* esposto dalla classe *CloudStorageAccount*. Otteniamo un oggetto *CloudQueue* usando il metodo *GetQueueReference* del proxy appena creato. All'interno del loop proposto dal template, usiamo il metodo *CloudQueue.GetMessage* per ottenere un *CloudQueueMessage*. Verifichiamo di avere ottenuto un messaggio confrontando il risultato con *null* prima di invocare il metodo *AsString* per analizzare il contenuto.

```
    using Microsoft.WindowsAzure.ServiceRuntime;
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.StorageClient;

    public class WorkerRole : RoleEntryPoint
    {
        public override void Run()
        {
            // Implementazione di lavoro di esempio. Sostituire con la logica
personalizzata.
            Trace.WriteLine("WorkerRole1 entry point called", "Information");

            CloudStorageAccount account =
                CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
            CloudQueueClient queueClient = account.CreateCloudQueueClient();
            CloudQueue queue = queueClient.GetQueueReference("orders");

            while (true)
            {
                CloudQueueMessage m = queue.GetMessage();
                if (m != null)
                {
                    queue.DeleteMessage(m);
                    Trace.WriteLine(m.AsString, "Information");
                }
                Thread.Sleep(2000);
            }
        }

        public override bool OnStart()
        {
```

```
            // Impostare il numero massimo di connessioni simultanee
            ServicePointManager.DefaultConnectionLimit = 12;

            #region Setup CloudStorageAccount Configuration Setting Publisher

            // questo codice imposta un gestore per aggiornare l'istanza
            // della classe CloudStorageAccount quando le loro impostazioni
            // di configurazione canbiano, nel file di configurazione del
            // servizio
            Microsoft.WindowsAzure.CloudStorageAccount.SetConfigurationSettingPublisher(
                (configName, configSetter) =>
                {
                    // fornisce la configurazione con un valore iniziale
                    configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));
                    RoleEnvironment.Changed += (s, arg) =>
                    {
                        if
(arg.Changes.OfType<RoleEnvironmentConfigurationSettingChange>().Any((change) =>
                            (change.ConfigurationSettingName == configName)))
                        {
                            // le impostazioni di configurazione sono cambiate
                            // propago il valore
                            if
(!configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))

                                // In questo caso il cambiamento elle credenziali dello
storage account
                                // nel servizio di configurazione è così significativo che
il ruolo
                                // richiede di essere riciclato rispetto all'ultima
configurazione
                                // per esempio l'endpoint è cambiato
                                RoleEnvironment.RequestRecycle();
                        }
                    };
                });
            #endregion

            return base.OnStart();
        }
    }
}
```

Listato 28 – File WorkerRole.cs

Nella finestra del Windows Azure Compute Emulator si possono vedere le informazioni relative al messaggio order1 al termine del trace. Una volta che il processo è in esecuzione, possiamo inserire nuovi ordini dalla pagina web e verificare il processo nel trace del WACE. Si può anche cambiare il valore dell'impostazione *DataConnectionString* per far puntare lo Storage Account locale nel cloud.

Conclusioni

Abbiamo visto una introduzione al cloud computing, a cominciare dai principi basilari fino ad un a sintetica introduzione alla strategia seguita da Microsoft per il cloud. Abbiamo visto alcuni dei principi comuni e delle teorie dietro questa nuova ondata nell'industria informatica.

Windows Azure rappresenta il sistema operativo della piattaforma di cloud computing di Microsoft. Al momento questo è basato su Windows Server 2012: non richiede alcun tipo di configurazioneo di installazione, né è necessario applicare patch o installare altri componenti Windows. Le applicazioni possono essere caricate e gestite tramite un semplice portale web. Inoltre Windows Azure mette a disposizione servizi di storage consistenti in uno Storage Account per immagazzinare blob, tabelle e code.

Database SQL (ex SQL Azure) è la versione cloud di SQL Server: può essere usato nello stesso modo in cui si farebbe con un qualsiasi database tradizionale.

Quando ci si avvicina alla programmazione verso il cloud computing di Microsoft, si comincia con l'installazione dell'SDK: abbiamo spiegato le diverse componenti della piattaforma (e abbiamo creato il nostro primo progetto). Abbiamo effettuato il test in locale con il Compute Emulator, abbiamo operato il deployment dell'applicazione su Windows Azure utilizzando il metodo pubblica (*publish*) del progetto cloud. Abbiamo analizzato le possibili definizioni e configurazioni di un progetto cloud in Visual Studio.

Abbiamo familiarizzato con lo storage locale per mettere in cache le risorse remote e approfondito l'Azure Storage Account utilizzando un account reale. Abbiamo iniziato a conoscere le API utilizzate in un progetto cloud ed infine testando la soluzione nell'emulatore locale.

Per finire abbiamo fornito un0introduzione completa al Table Service e su come usarlo nelle nostre soluzioni per immagazzinare e recuperare entità. Abbiamo visto come usare il Queue Service esposto dallo Storage Account per disaccoppiare il front end di un'applicazione dal suo back end. Il servizio di coda nel cloud è simile al Microsoft Message Queue che si potrebbe usare in una soluzione on-premises. Nell'ultima parte di questa sezione, abbiamo descritto le funzioni svolte da un progetto Worker Role in una soluzione che realizza un semplice sistema di order-processing nel back end.

Considerazioni dell'autore

Non ho inserito in questo contesto (o accennato minimamente) al Live ID di Microsoft, la sottoscrizione al portale,a i certificati (che controllano il molto importante aspetto della sicurezza), a Database SQL ma soprattutto alla **tariffazione**: non esisterà progetto cloud che un programmatore creerà che non sarà soggetto ad un tariffario.

La tariffazione è un aspetto principale per qualsiasi soluzione che si interfacci con una richiesta di un cliente, poiché l'azienda software/il programmatore autonomo (Freelance), deve essere in grado di stimare correttamente e congruamente il preventivo che dovrà fare accettare al cliente …

… ma questa è un'altra storia.

Indice delle figure

Indice dei listati