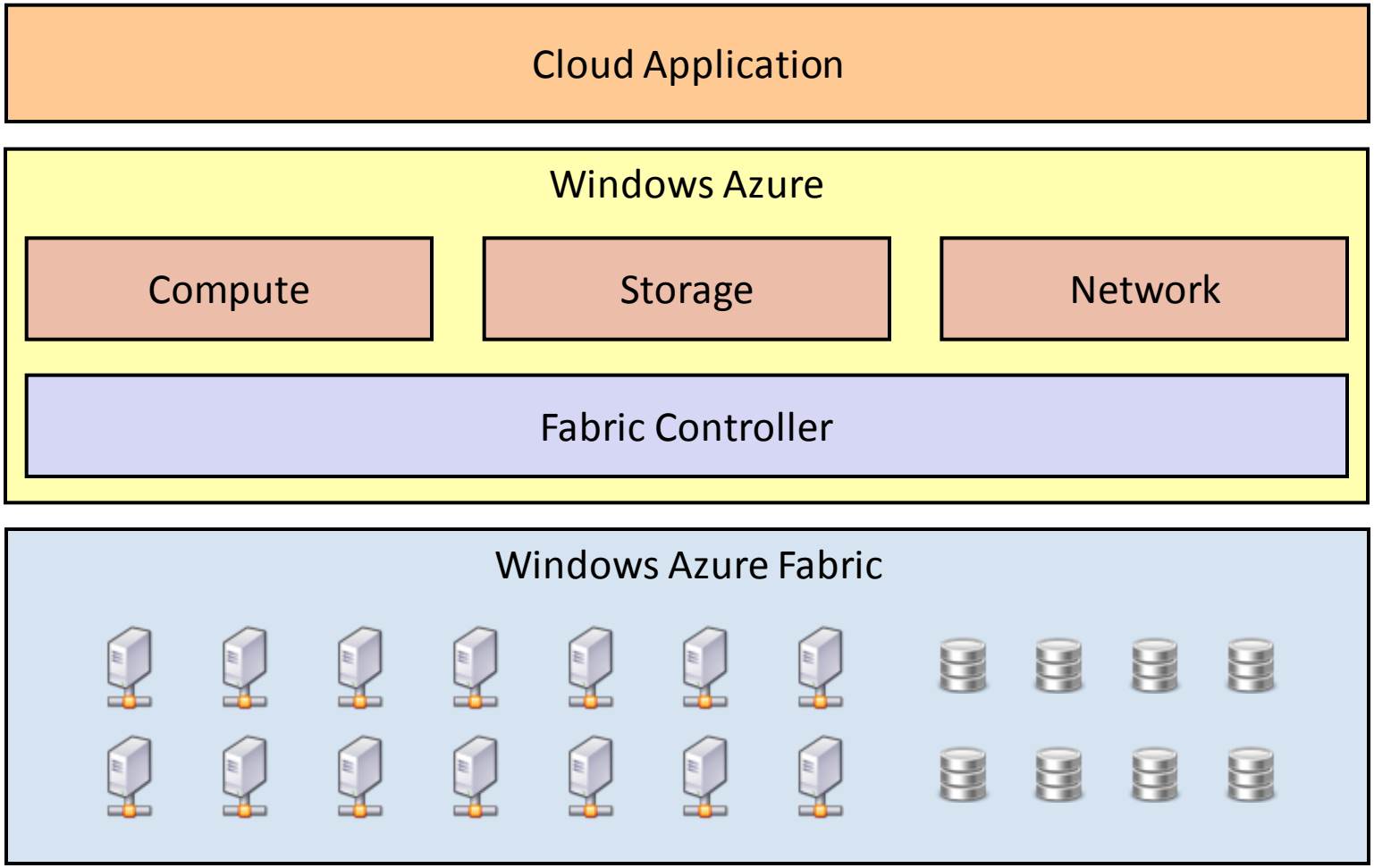
Razvoj PHP aplikacija na Microsoft Windows Azure platformi

Uvod

- **Azure je Microsoftova platforma za računarske oblake**
- **Azure se klasifikuje kao PaaS (Platform as a Service)**
- **Moguće je programiranje na jezicima .NET, PHP, Ruby, Java, node.js i Python**
- **Globalni opseg sistema, 6 centara podataka:**
- **Severna amerika**
  - North-central US - Chicago, IL
  - South-central US - San Antonio, TX
- **Azija**
  - East Asia - Hong Kong, China
  - South East Asia - Singapore
- **Evropa**
  - West Europe - Amsterdam, Netherlands
  - North Europe - Dublin, Ireland
- **24 CDN čvora (CDN=content delivery network) sistem servera i mrežnih resursa za ubrzanje i povećanje dostupnosti Azure servisa (npr. u EMEA u Amsterdam, NL**
- **Doha, QT, Dublin, IE, London, GB, Moscow, RU, Paris, FR , Stockholm, SE , Vienna, AT, Zurich, CH**

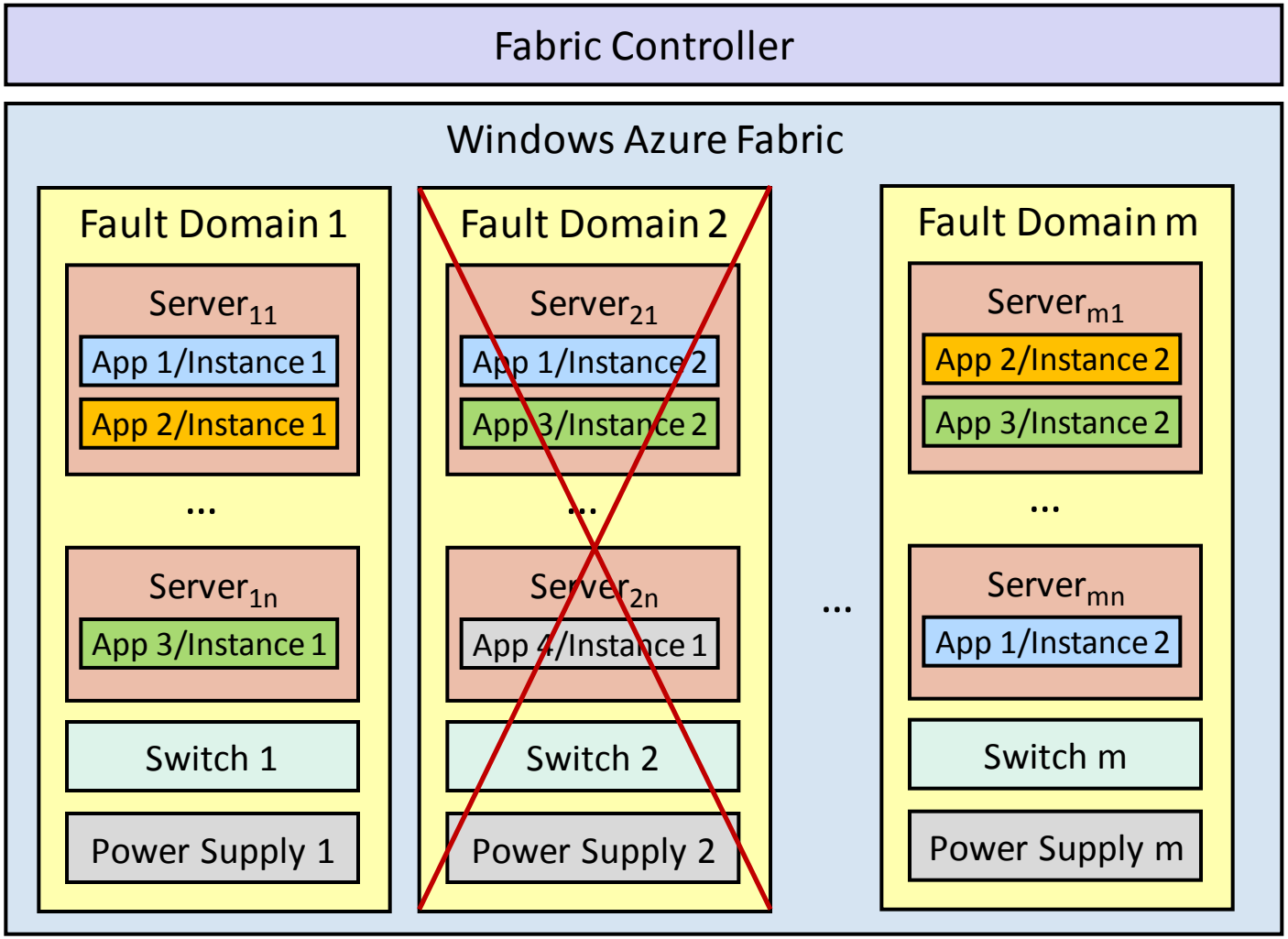
Arhitektura Windows Azura



Arhitektura Windows Azura

- *Fabric* (“tkanje”) je mreža povezanih čvorova:
  - Serverskih računara
  - Brzih svičeva, rutera, balansera opterećenja
  - fiber-optičkih veza
- *Azure Fabric Controller* je servis koji nadgleda, održava i dodeljuje mašine.
- Osnovni servisi Azura su:
  - *Compute:* (računarski) Hosting (“ugošćavanje”) skalabilnih (“koji se mogu propocionalno smanjiti ili uvećati”) aplikativnih servisa na Windows Serveru 2008.
  - *Storage:* (skladištenje podataka) Upravljanje nerelacionim podacima visoke raspoloživosti (stalno je dostupan za rad)
  - *Network:* (mreža) Resursi za komunikaciju sa eksternim aplikacijama (van oblaka) – servisna magistrala.

Otpornost na otkaze i raspoloživost



Otkazni domeni

- Domeni otkazivanja (fault domains): Jedinica otkazivanja u centru podataka (kod Azura, to je jedan rek sa mašinama).
- Pod uslovom da se želi da instanca korisničkog servisa u oblaku bude otporna na otkaze (tj. da važi Mikrosoftova garancija o uptime-u iz SLA ugovora) svaka servisna “uloga” mora imati više od jedne instance (specificira korisnik u .csdef fajlu). Azure tada raspoređuje instance na bar dva otkazna domena, tako da ako jedna instanca padne, druga će još biti raspoloživa.

Domeni dogradnje (upgrade)

- Domeni ažuriranja (Upgrade Domains): Logička jedinica puštanja aplikacije u rad (unit of deployment). Mehanizam koji omogućava da aplikacija bude visoko raspoloživa (dakle u radnom stanju) i tokom procesa ažuriranja same aplikacije
- Azure distribuira instance servisnih “uloga” ravnomerno u više domena dogradnje. Kada se ažurira aplikacija, u jednom trenutku ažuriraće se samo jedan domen ažuriranja, da bi se minimizirao uticaj upgrade-a na izvršavanje aplikacije.
- Proces dogradnje: zaustavljaju se instance uloga koje su se izvršavale, izvršava se ažuriranje koda tih instanci, potom se one ponovo aktiviraju. Zatim se prelazi na sledeći domen. Dogradnja aplikacije se završava kada su procesirani svi njeni domeni.
- Programer ima uticaj samo na to koliko će biti domena dogradnje (preko csdef fajla) ali ne i koje instance će biti u kom domenu

Uloga Fabric Controller-a

- Aplikativni programeri specifikaciju potrebne resurse u modelu servisa. Ovaj model obuhvata uloge koje aplikacija obavlja i kako one međusobno komuniciraju, koji su sistemski zahtevi (da li treba IIS na primer), koliko CPUa je potrebno, potreban propusni opseg, itd. Može se čak odrediti koji gostujući operativni sistem da se koristiti, da li se zahteva posvećeni server je dovoljan virtuelni, broj otkaznih domena i domena nadogradnje.
- Fabric Controller automatski obezbeđuje zahtevane resurse (mapira servisni model na fizičke resurse, delove fabrica).

Uloga Fabric Controller-a

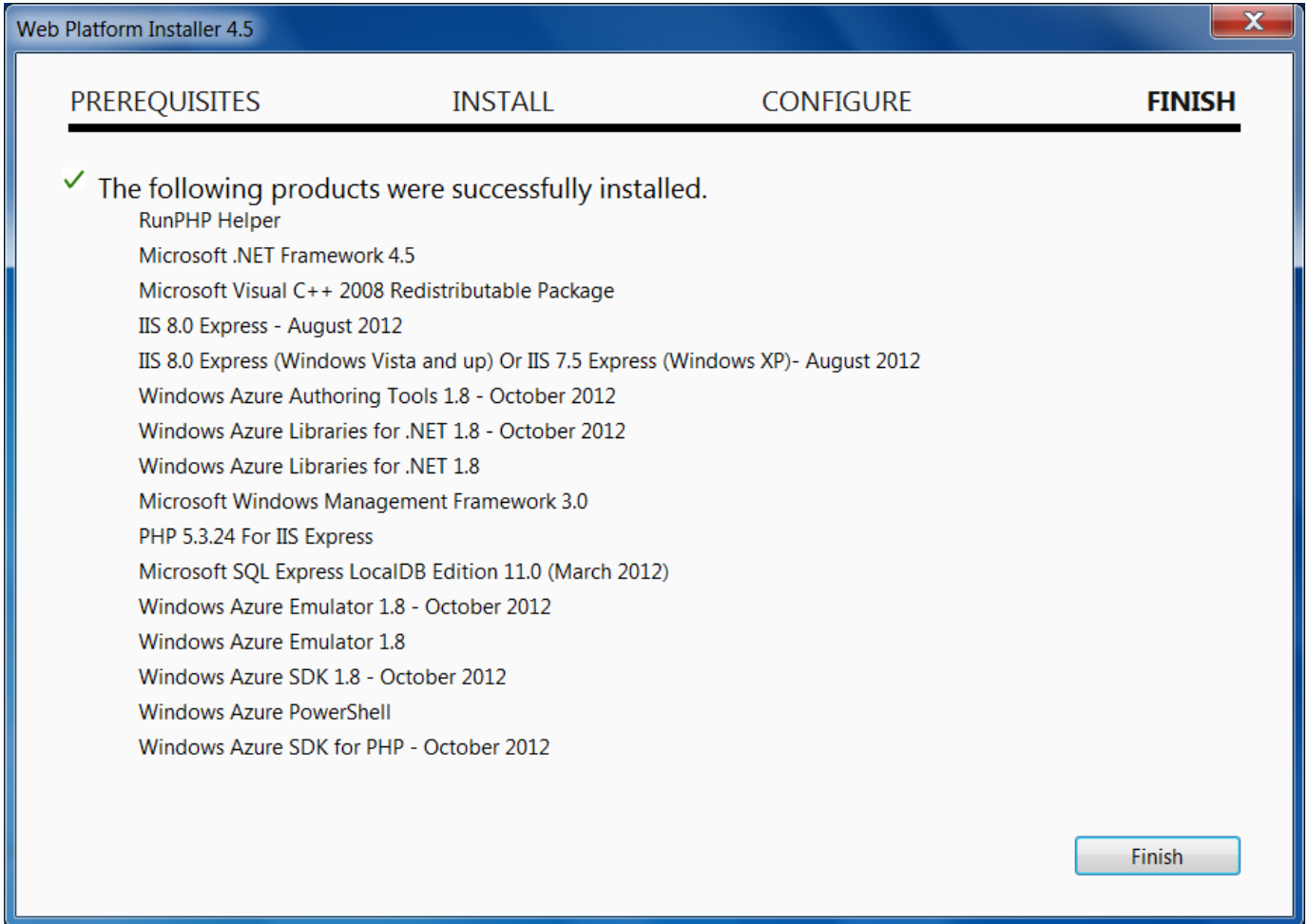
- Fabric Controller čini “obične” resurse otpornima na otkaze i visoko raspoloživima.
  - Rana detekcija aplikativnih otkaza.
  - Pokreće dodatne instance servisnih uloga po potrebi.

- Kontroler fabric-a i sam je implementiran kao visoko raspoloživi servis, ima veći broj instanci i podaci o stanju celog fabrica replicirani su u svim instancama. I kontroler ima fault i upgrade domene kao i svaki drugi servis u oblaku.
- Na fizičkim serverima stalno je aktivan proces koji javlja kontroleru stanje servera (tipa initializing, idle, izvršava određene instane, faulted, inoperable). Kontroler je zadužen da server iz tekućeg prevede u neko željeno tj. planirano stanje

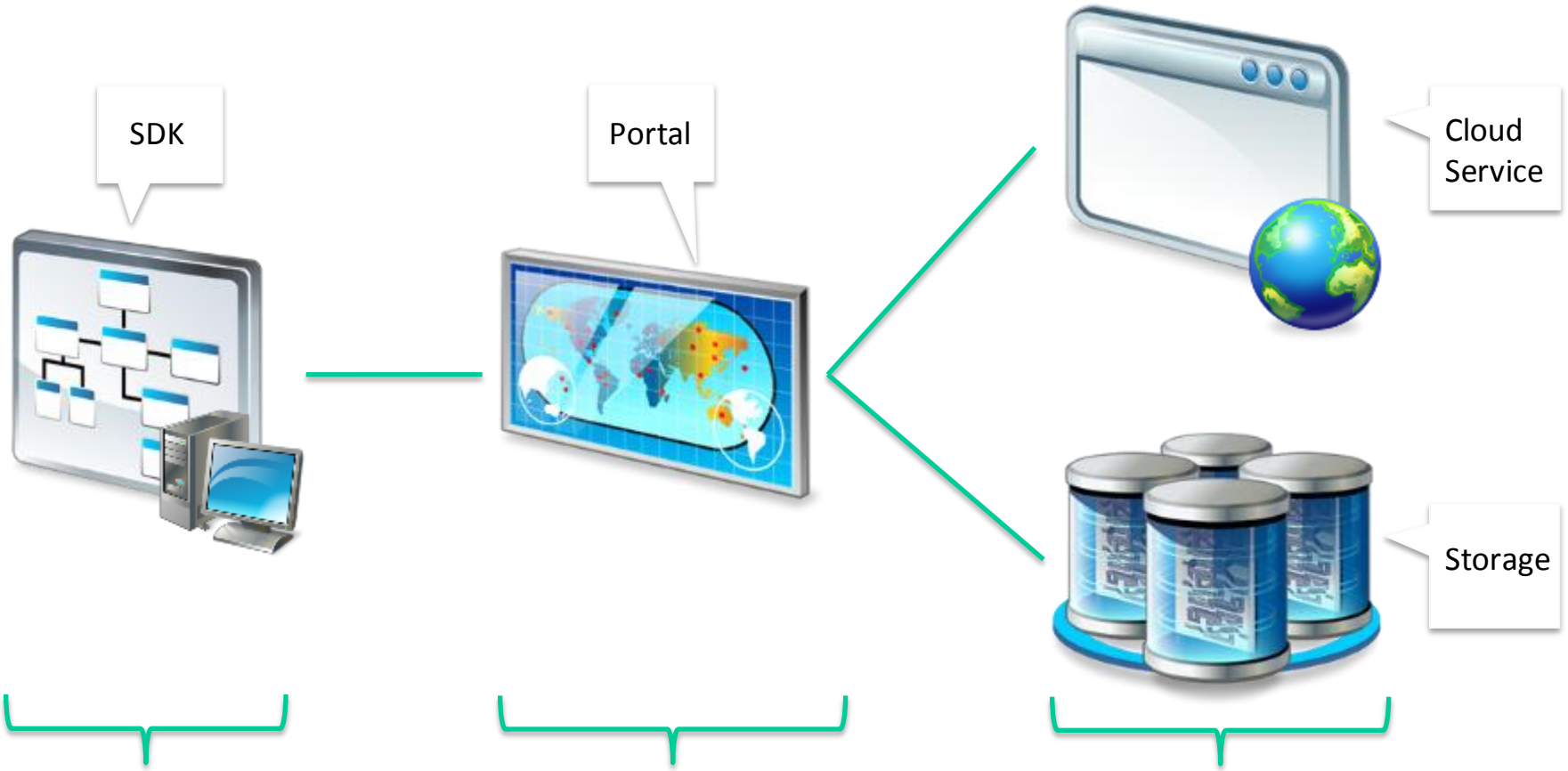
Programiranje PHP Cloud Servisa na Azure

Instalacija programerskih alata

- **Potrebni alati: Azure PHP SDK**
- **http://www.windowsazure.com/en-us/develop/php/common-tasks/download-php-sdk/**
- **Kliknuti na link Microsoft Web Platform Installer (ispod naslova Windows Azure PowerShell and Windows Azure Emulators)**



Programerski pogled na Windows Azure



Računarske “uloge” (Compute roles)

Web Role

IIS Host

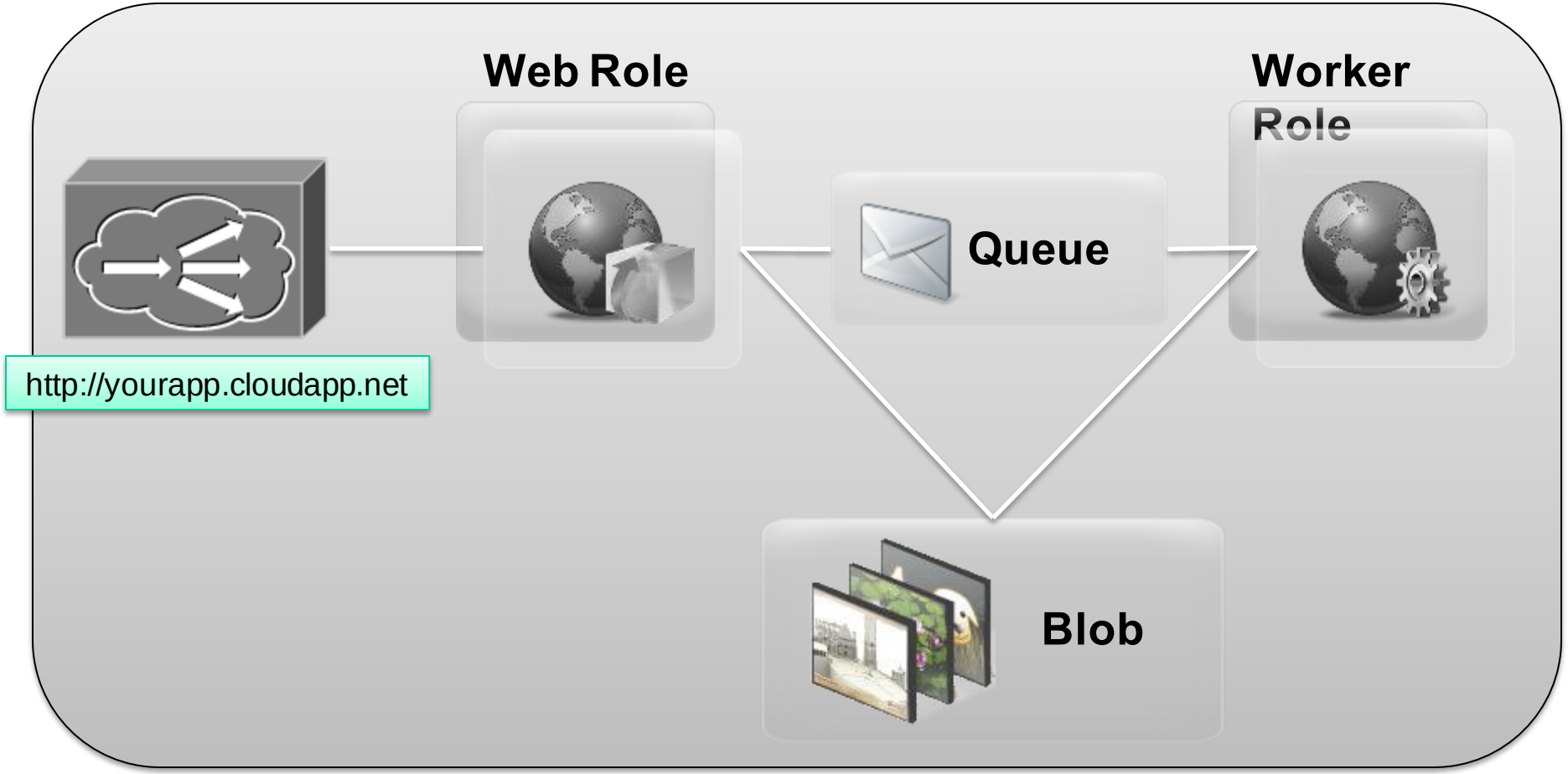
Your Code

Worker Role

System Host

Your Code

Primer arhitekture cloud servisa



Azure Emulator za pomoć u razvoju aplikacije

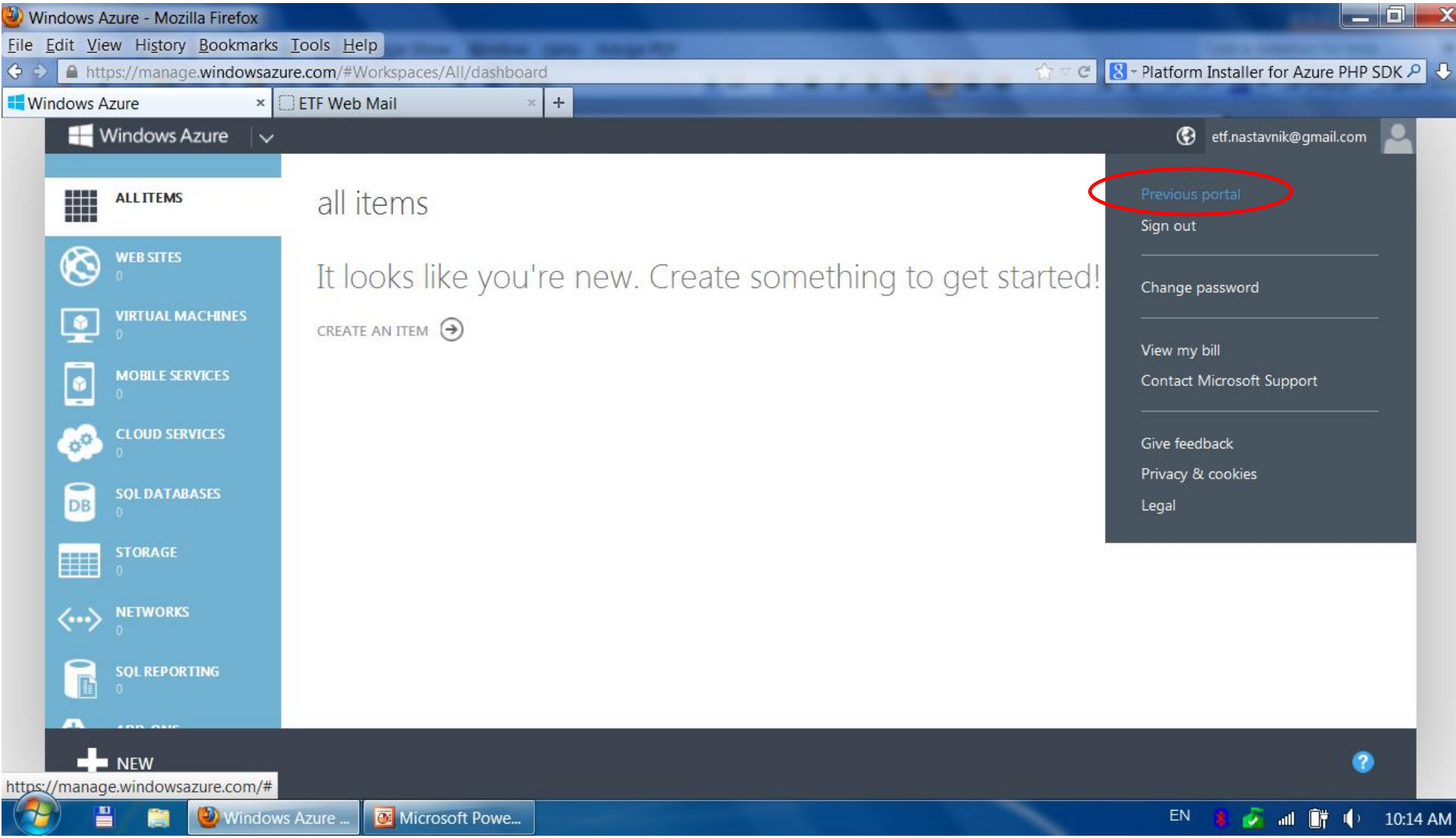
Local Machine

Windows Azure Simulation Environment

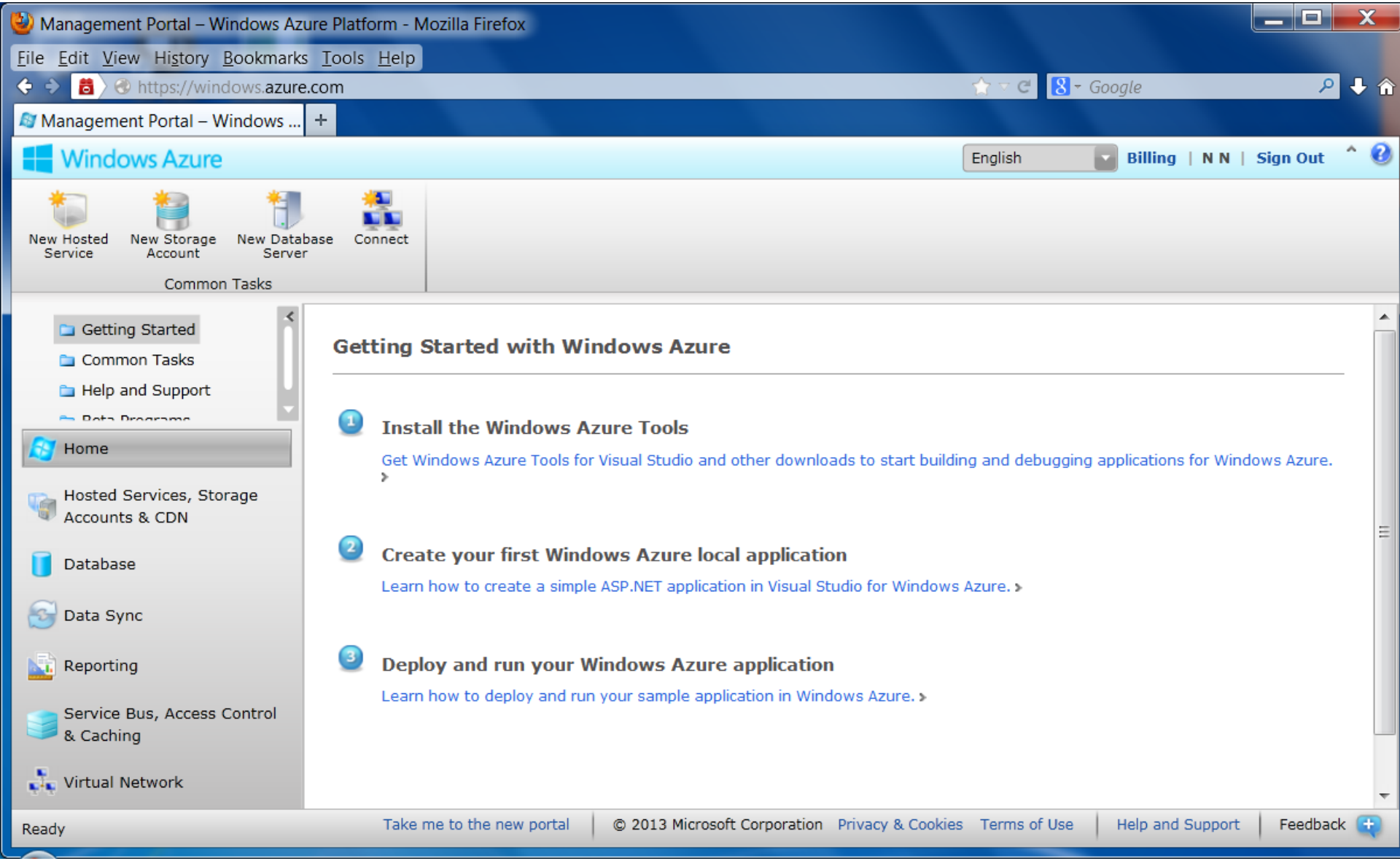
Development Fabric

Development Storage

Windows Azure Management Portal
windows.azure.com



Windows Azure Management Portal (starija verzija)



Pravljenje novog projekta tipa Cloud Service

- Azure SDK for PHP je biblioteka PHP funkcija za pristup Azure oblaku i skup alata za rad iz komandne linije

PRAVLJENJE NOVOG PROJEKTA

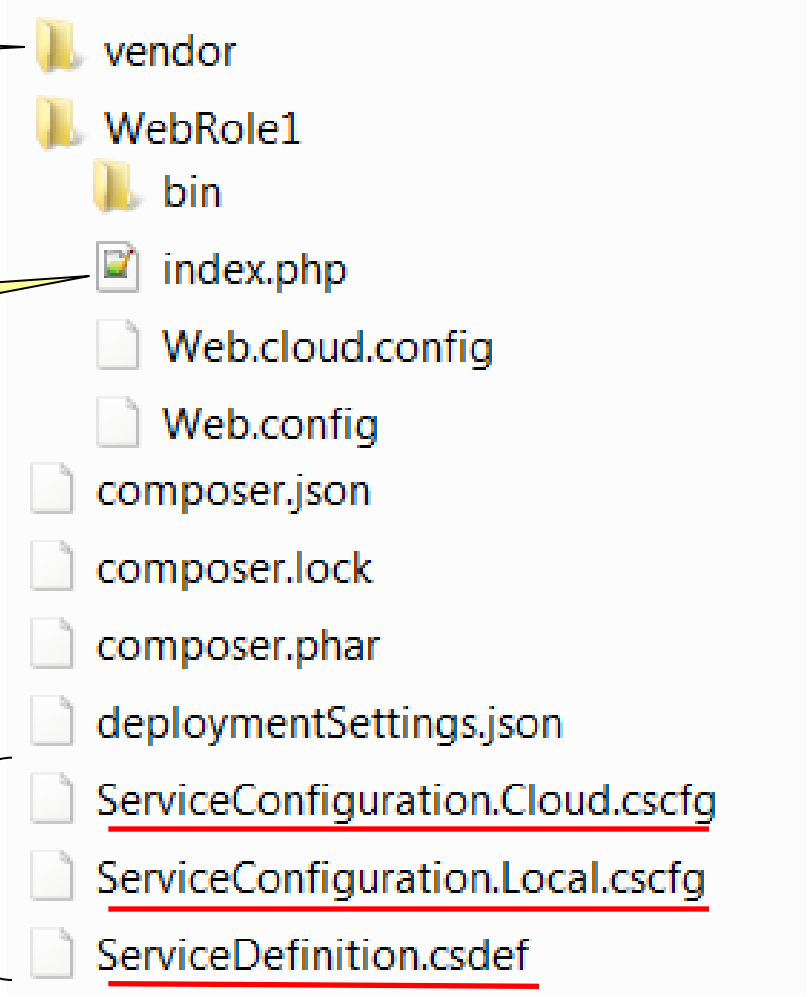
- Startovati Windows Azure PowerShell (run as administrator)
- Otkucati **[console]::CursorSize=10**
- Otkucati: **cd *<lokacija gde ćemo čuvati projekat>***
- Za osnovni kod (kostur projekta) kucati: **New-AzureServiceProject *<naziv proj>***
- U osnovni kod dodati jednu Web ulogu: **Add-AzurePHPWebRole <naziv>**
- Azure biblioteka za korišćenje funkcija oblaka mora se uključiti u projekat na način opisan na http://www.windowsazure.com/en-us/develop/php/common-tasks/download-php-sdk/ (varijanta Install via Composer)

Generisani kostur aplikacije

Ovde je smešten bibliotečki kod

Početni web fajl (i ostale php fajlove smestiti ovde)

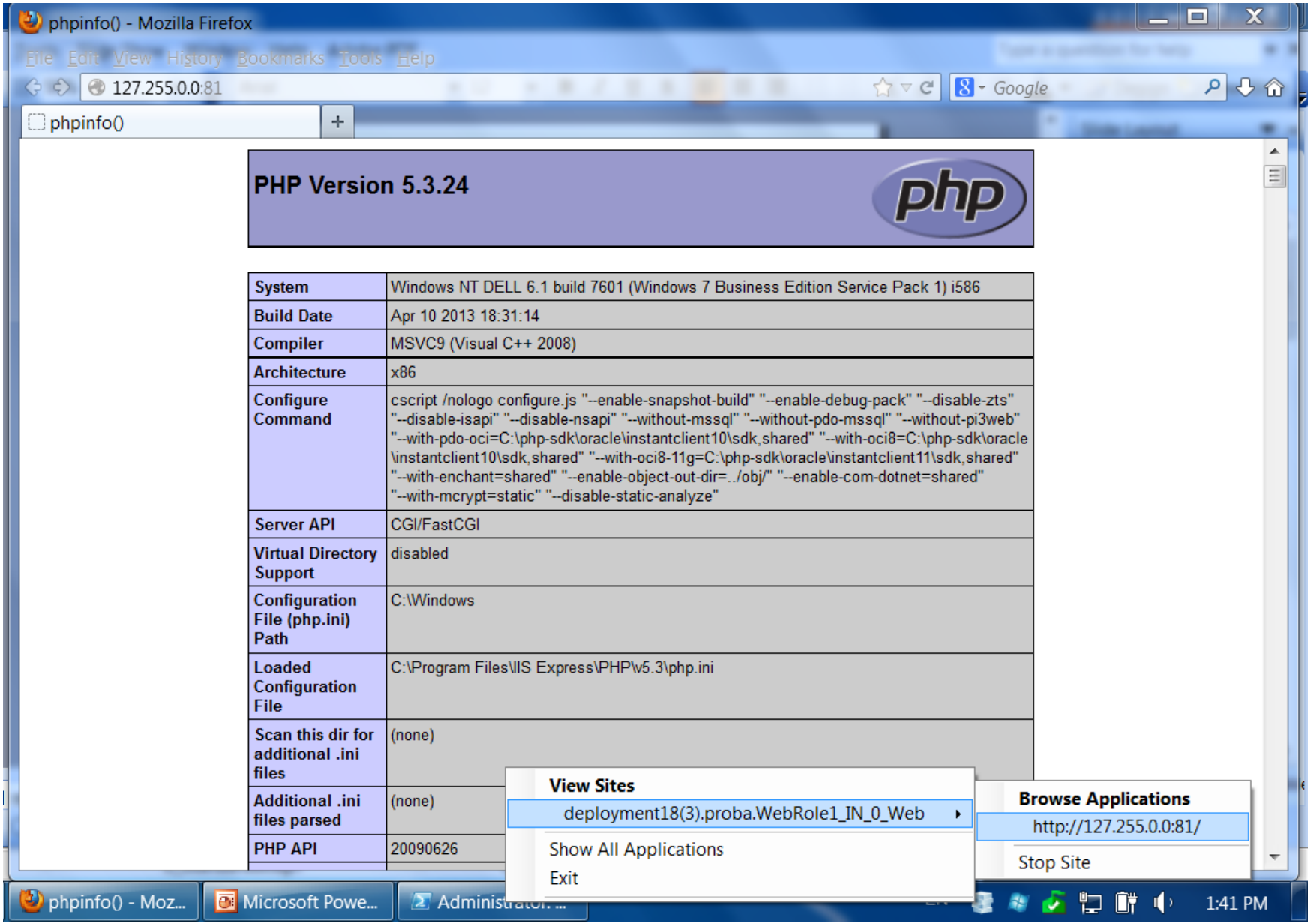
Definicija i konfiguracija cloud servisa



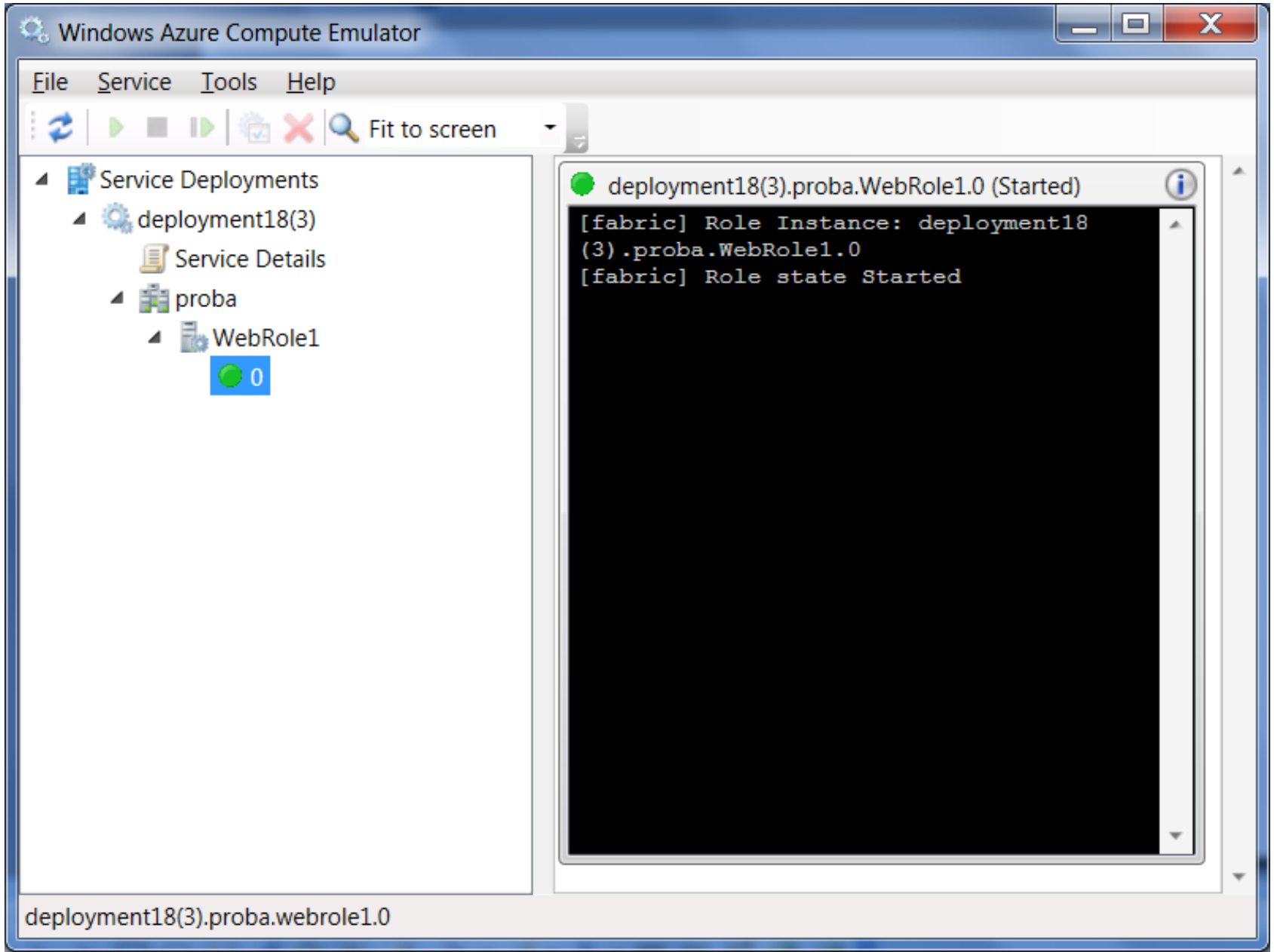
Pokretanje servisa u Azure emulatoru

- Startovati Windows Azure PowerShell (run as administrator)
- Otkucati **[console]::CursorSize=10**
- Otkucati: **cd *<koren projekta>***
- Pokrenuti emulator: **Start-AzureEmulator**
  (Pravi deployment paket, pokreće emulator ako nije pokrenut, instalira aplikaciju na emulatoru i startuje aplikaciju)
- **Hint: da bi se videle eventualne greške u PHP kodu u browseru, editovati:**
  C:\Program Files\IIS Express\PHP\v5.3\php.ini
  i staviti:
  display_errors = On
  display_startup_errors = On

U slučaju da je sve u redu...



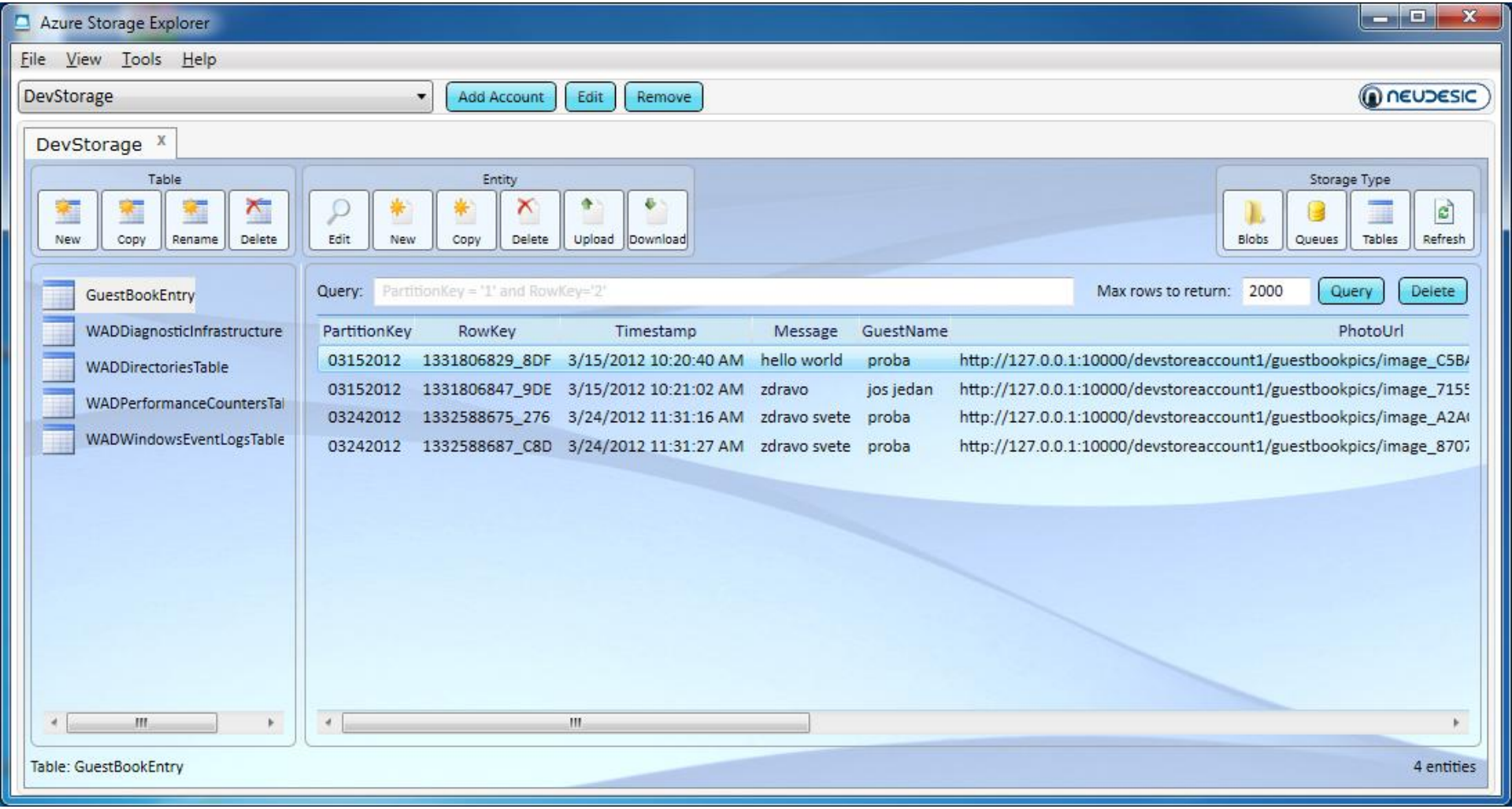
Emulator ima UI u kome se vide statusne poruke



Pokretanje servisa na Azure oblaku

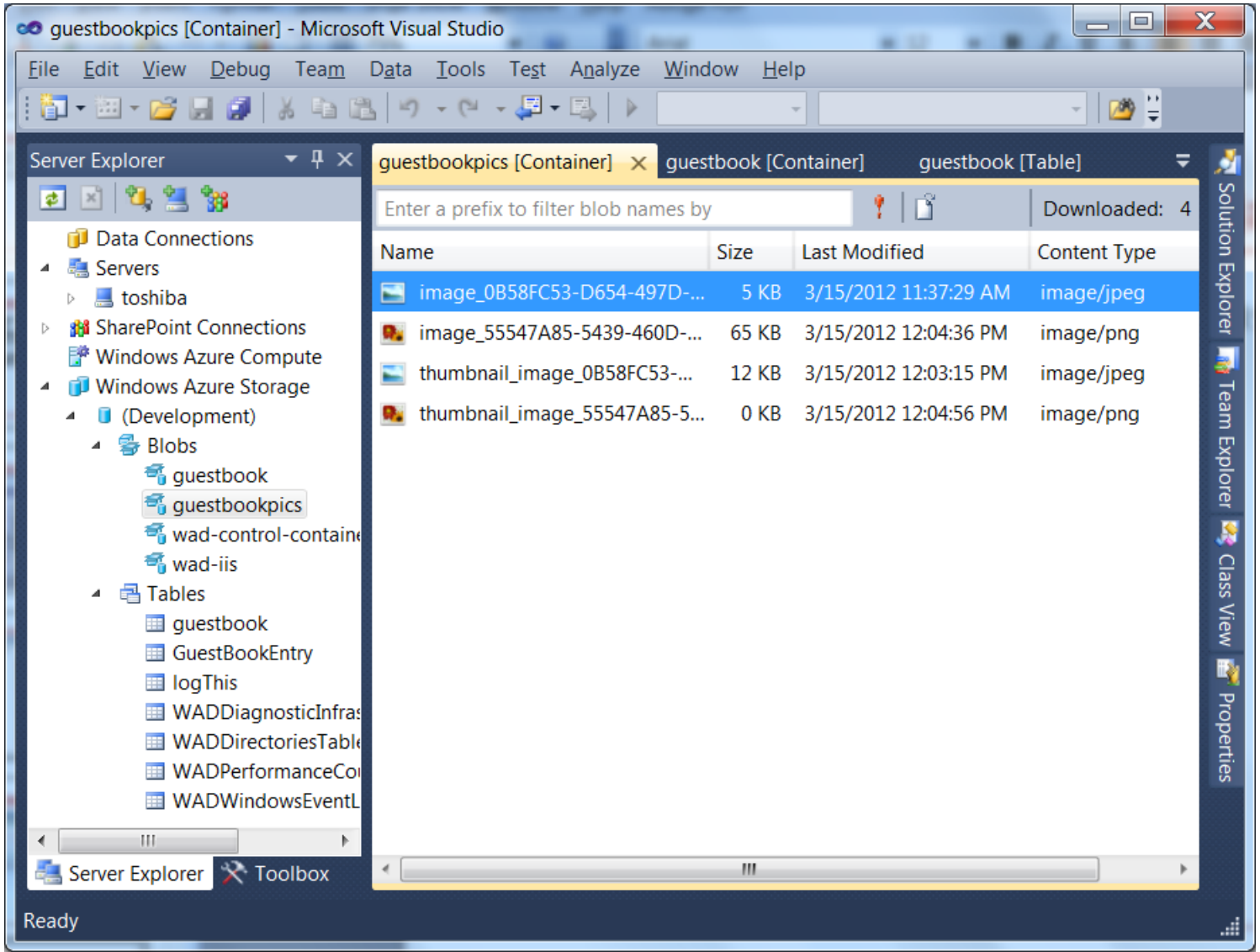
- **Isto se radi iz WindowsAzure PowerShella, komandom: Publish-AzureServiceProject**
- **Međutim, prethodno treba sa nekoliko komandi postaviti informacije o parametrima Azure naloga, storage naloga itd.**
- **Detaljan opis dat je na:** http://www.windowsazure.com/en-us/develop/php/how-to-guides/powershell-cmdlets/

Pregled storage-a bez Visual Studija (http://azurestorageexplorer.codeplex.com/)

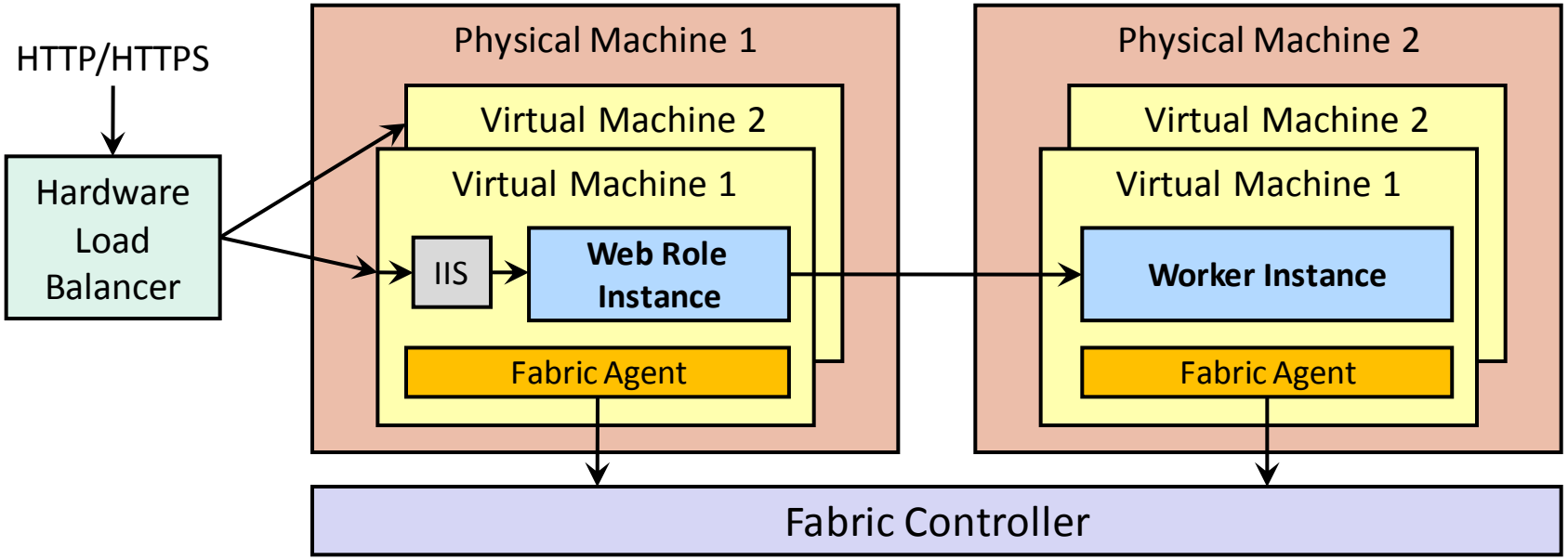


Pregled storage-a iz Visual studija

- **Mora se instalirati Windows Azure Tools for Visual Studio**



Računarski servisi aplikacije : uloge Web i Worker

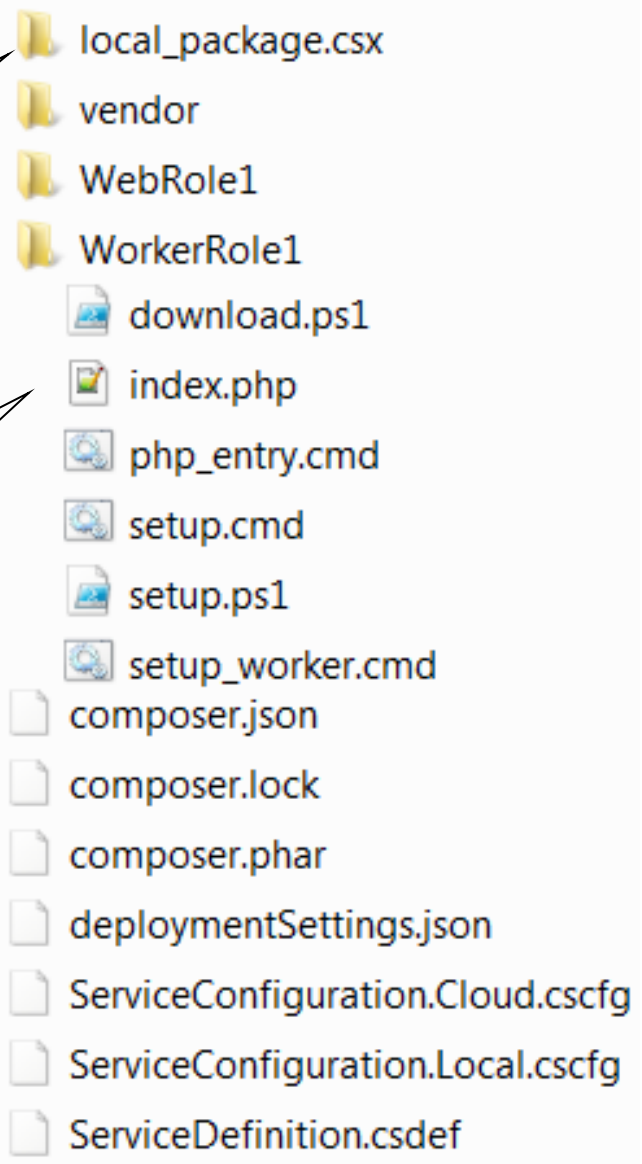


- **Web uloga:** Interaktivna aplikacija hostovana u IISu:
  - Web aplikacija ili web servis
- **Worker uloga:** Dugotrajni pozadinski proces
  - Najčešće izolovan od spoljnog (van oblaka) okruženja
  - Mogu se napraviti dostupni od strane spoljne aplikacije
- **Fabric Agent** sakuplja metrike o resursima (korišćenje, otkazi, …)

Dodavanje worker uloge u servisni projekat

- U WindowsAzure PowerShellu postaviti se na koren projekta i otkucati:
  Add-AzurePHPWorkerRole <naziv>

Ovo je napravljeno prilikom startovanja u emulatoru



Ovde je “glavni program” workera, treba da ima sledeću formu:

```
<?php
    while (true) {
        // procesiranje
        sleep(10); // 10 sec
    }
?>
```

Servisni model: definicija servisa

- Fajl **ServiceDefiniton.csdef** definiše globalnu strukturu servisa:
  - *vmsize:* CPU jezgra (1 – 8) i memorija za VM (1.7 – 15 GB)
  - *full/partial trust:* podrška za izvršavanje mašinskog ("native") koda
  - *Endpoint:* interne i eksterne komunikacione tačke (http, https, tcp)
  - *LocalStorage:* privremeni skladišni prostor na serveru koji izvršava instancu
  - *ConfigurationSettings:* imena konfiguracionih parametara
- Definicija servisa obrađuje se u vreme puštanja u rad (deployment time).

```
<ServiceDefinition name="MyService" …>
  <WebRole name="MyWebRole" enableNativeCodeExecution="false" vmsize="Medium">
    <InputEndpoints>
      <InputEndpoint name="HttpIn" protocol="http" port="80" />
    </InputEndpoints>
    <ConfigurationSettings>
      <Setting name="name1" />
      …
   </ConfigurationSettings>
  </WebRole>
</ServiceDefinition>
```

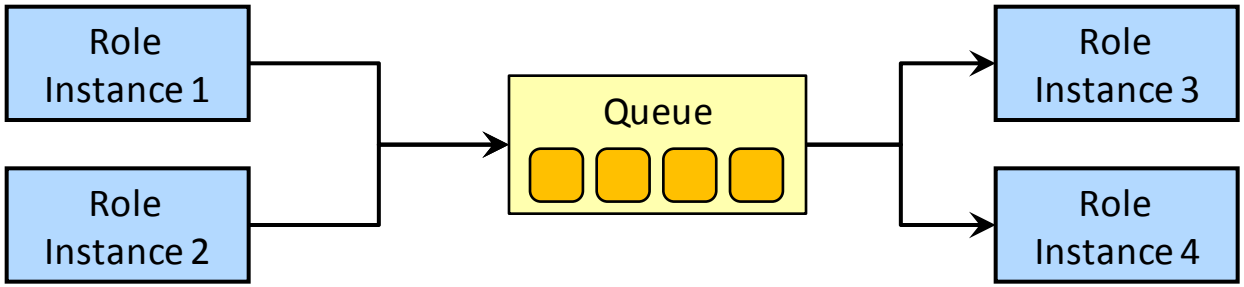
Servisni model: podešavanje servisa

- Fajl **ServiceConfiguration.[Local|Cloud].cscfg**
  - određuje broj *instanci* svake uloge, kao i
  - Vrednosti za k*onfiguraciona podešavanja*.
- Konfiguracija servisa može se menjati u vreme izvršavanja.

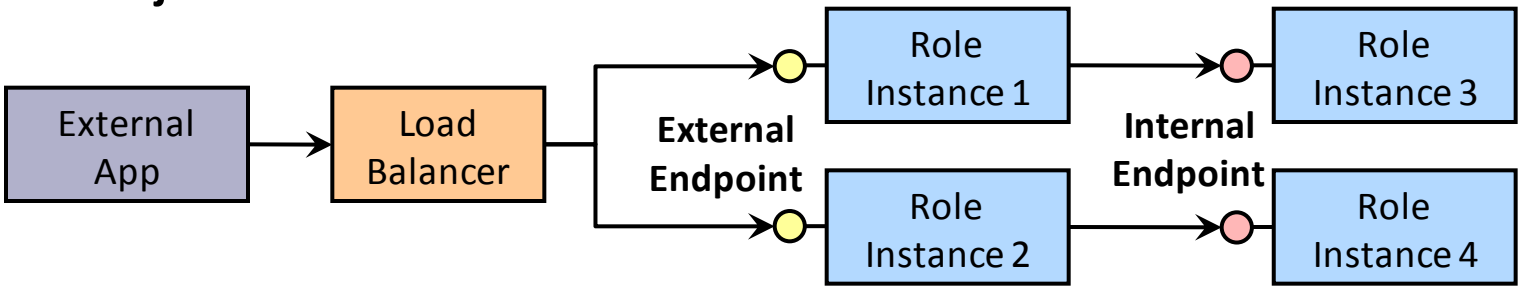
```
<ServiceConfiguration serviceName="MyService" xmlns= "…">
  <Role name="MyWebRole">
    <Instances count="3" />
    <ConfigurationSettings>
      <Setting name="name1 " value="value1" />
    </ConfigurationSettings>
  </Role>
</ServiceConfiguration>
```

Komunikacija među servisnim ulogama

- Instance uloga mogu komunicirati **asinhrono putem redova (queues)**.
  - Preporučeni method za pouzdano slanje poruka
  - Više objašnjenja o *Azure Queues* u nastavku



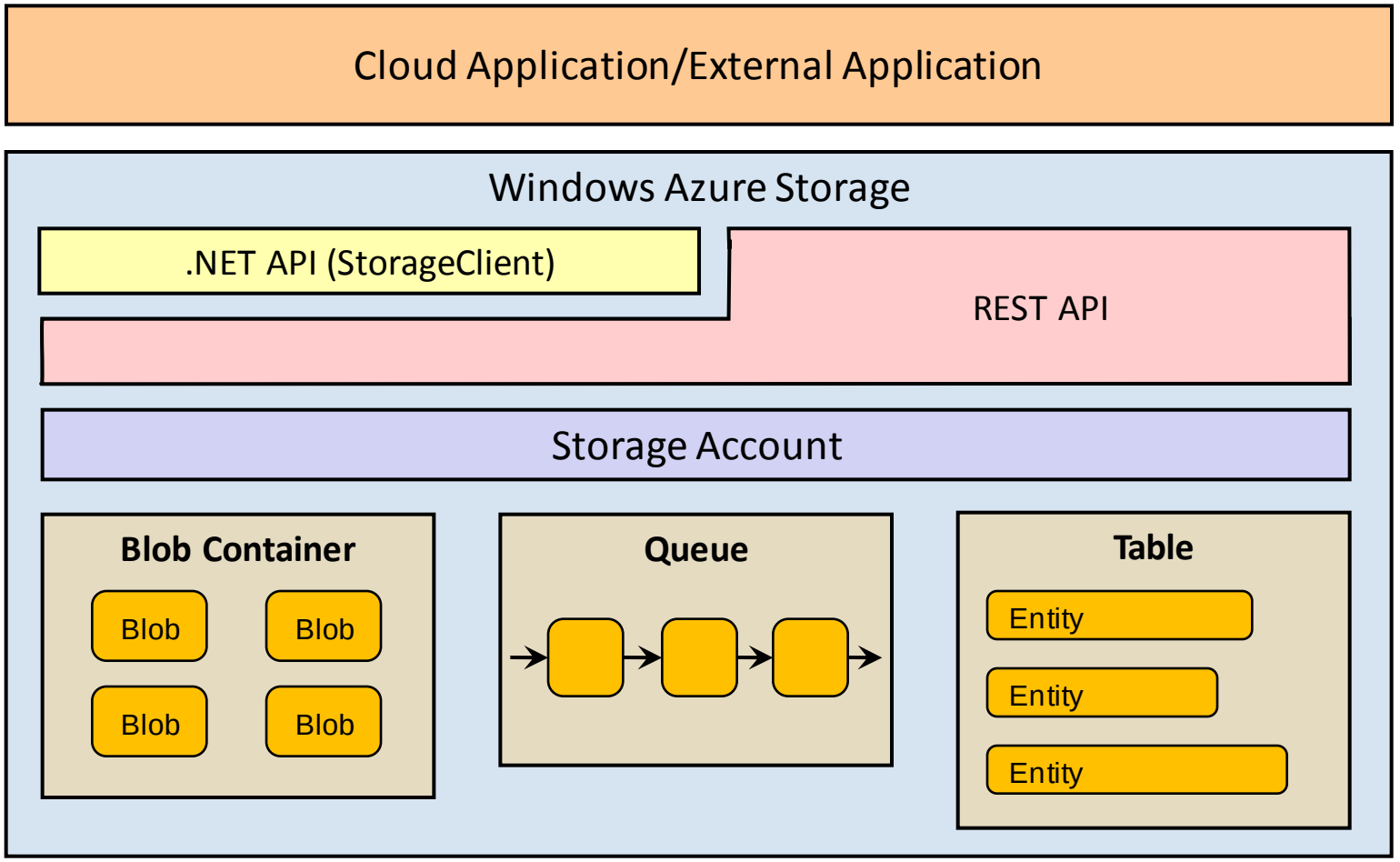
- Instance uloga mogu komunicirati i direktno koristeći **TCP ili HTTP(S) konekcije**.



Skladištenje podataka

- Aplikacija u oblaku nema pristup fajl sistemu virtuelne mašine na kojoj se izvršava.
- Aplikacija u oblaku podatke može čuvati u:
  - Binarnim objektima (Blob),
  - Tabelama (nerelaciona baza podataka)
  - Redovima (Queues)
  - Relacionoj bazi (SQL Azure)
- **Blob-ovi** obezbeđuju skladište za nestrukturirane podatke
  - Blob-ovi se organizuju u kontejnere.
- **Tabele** su strukturisane kolekcije entiteta.
  - Entiteti su kolekcije parova ime/vrednost.
  - Tabele nemaju shemu.
- **Redovi** obezbeđuju skladištenje i pristup podacima male veličine po FIFO principu.
  - Redovi obezbeđuju komunikaciju između instanci servisa.

Arhitektura Azure skladišta (storage)



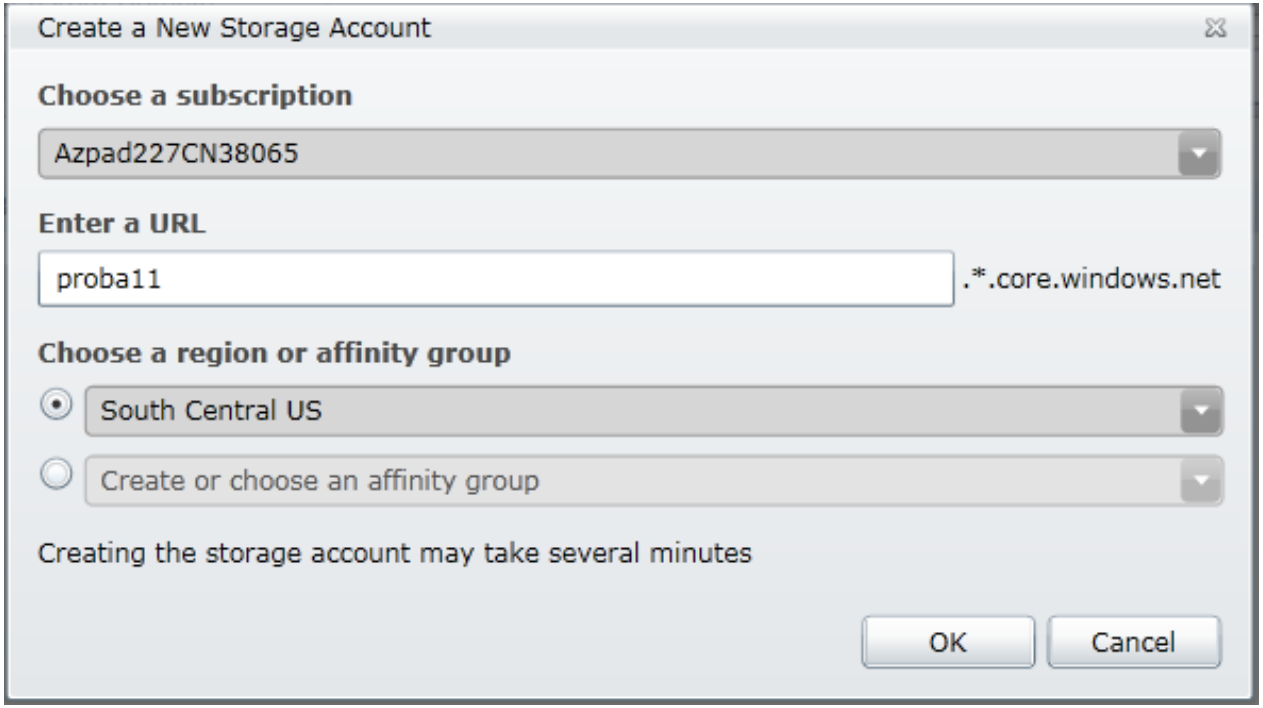
Osobine Storage-a

- Windows Azure Storage
  - ima **veliki smeštajni kapacitet**,
  - Je **skalabilano** i
  - **visoko pouzdano** skladištenje podataka.

- Skalabilnost
  - Podaci se distribuiraju između velikog broja skladišnih čvorova (umreženih servera).
  - Pristup skladištu je sa balansiranjem opterećenja.
- Pouzdanost
  - Prave se replike podatka na različitim sladišnim čvorovima pri upisu (3 replike u različitim domenima otkaza).
  - Ako mašina otkaže, podatak se replikuje na novi skladišni čvor.

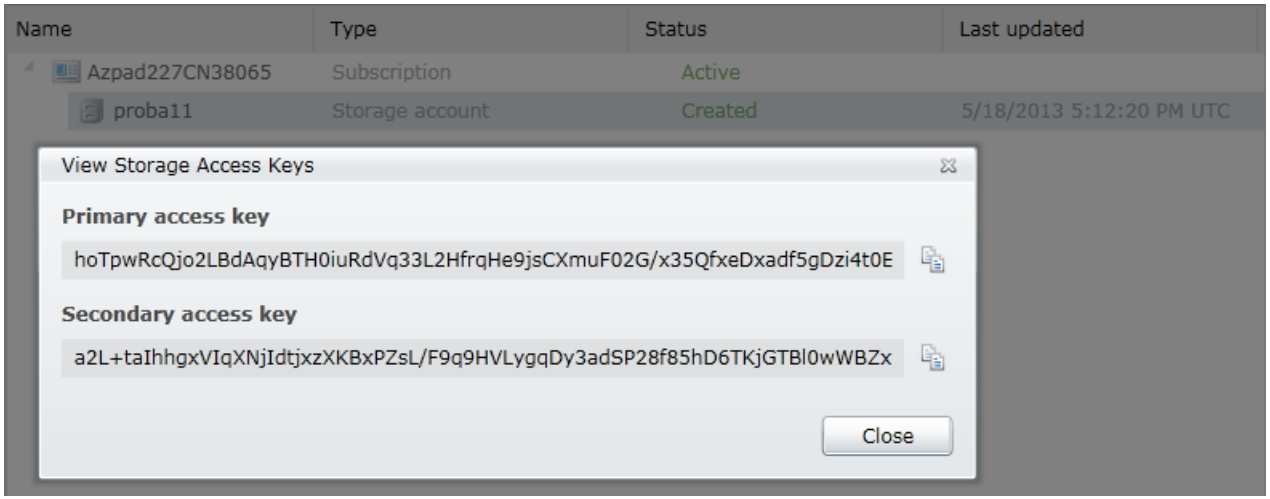
Skladišni nalog

- Skladišni nalog je ulazna tačka za sve skladišne servise.
- Nalozi se kreiraju preko Azure Management Portala.



Skladišni nalog

- Skladišnim nalozima može se pristupiti jednim od dva **tajna ključa**:
  - Na klijentskoj strani računa se hash kode (koristeći SHA256).
  - Hash kode i tajni ključ koriste se za računanje HMAC-SHA256 potpisa (Hash Message Authentication Code).
  - HMAC se priključuje svakom REST zahtevu.
  - Server skladišta koristi potpis da proveri autentičnost zahteva.



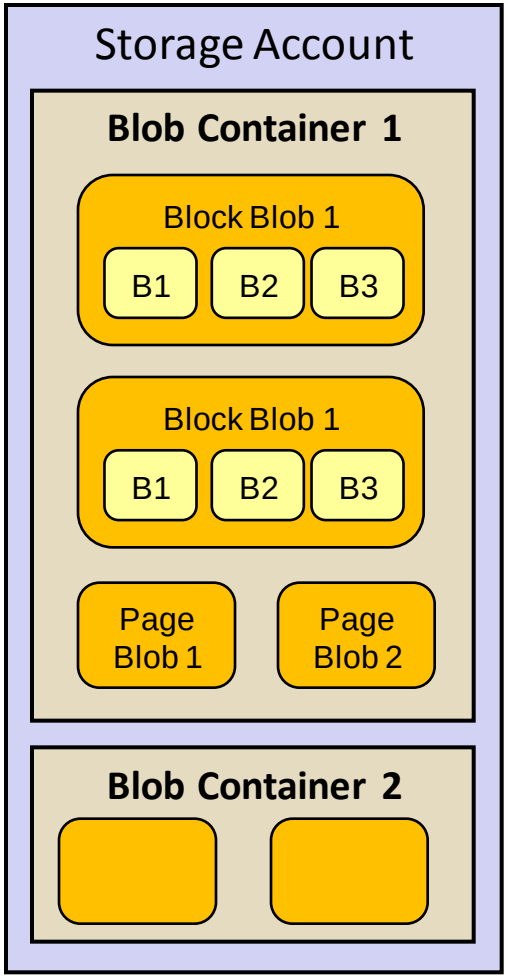
Pristup skladištu putem RESTa

- Svaka stavka u skladištu identifikuje se uniformnim resursnim identifikatorom URI:
  - Blob: *<http|https>://<account name>*.blob.core.windows.net/*<container>/<blob name>*
  - Table: *<http|https>://<account name>*.table.core.windows.net/*<table name>*()
  - Queue: *<http|https>://<account name>*.queue.core.windows.net/*<queue name>*
- HTTP "glagol" predstavlja akciju koja treba da se izvrši nad resursom:
  - PUT: create container/queue/table, save blob/entity/queue item, set metadata, …
  - GET: list blobs in container, get blob/entity/queue item, …
  - DELETE: delete container/queue/table/blob/entity/queue item, …
- REST API obezbeđuje platformski-nezavistan interfejs ka Azure skladištu.
- Primer:

```
PUT http://myaccount.blob.core.windows.net/mycontainer/myblockblob HTTP/1.1
x-ms-version: 2009-09-19
x-ms-date: Sun, 27 Sep 2009 22:33:35 GMT
Content-Type: application/octet-stream
x-ms-blob-type: BlockBlob
x-ms-meta-m1: some metadata
Authorization: SharedKey myaccount:YjN4fAR8/AmBrqBz7MG2uFinQh4dscbj598g=
Content-Length: nn
Request Body
```

Blob-ovi

- **Kontejneri blob-ova** obezbeđuju grupisanje skupa blob objekata.
  - Politika deljenja se postavlja na nivou kontejnera: public read ili private.
  - Kontejneri mogu imati meta-podatke.
- **Blob** pamti veliku količinu podataka.
  - Blob-ovi mogu imati meta-podatke.
- **Tipovi Blob-ova**
  - Blokovski blob-ovi su optimizovani za striming
    - Maksimalna veličina bloba: 200 GB
    - Blobovi se uploaduju u blokovima (max. 4 MB)
  - Stranični blobovi su optimizovani za direktan pristup
    - Maksimalna veličina bloba: 1TB (u emulatoru 2Gb)
    - Kontejner za Azure diskove (Azure Drive).



Primeri upotrebe blob API-ja

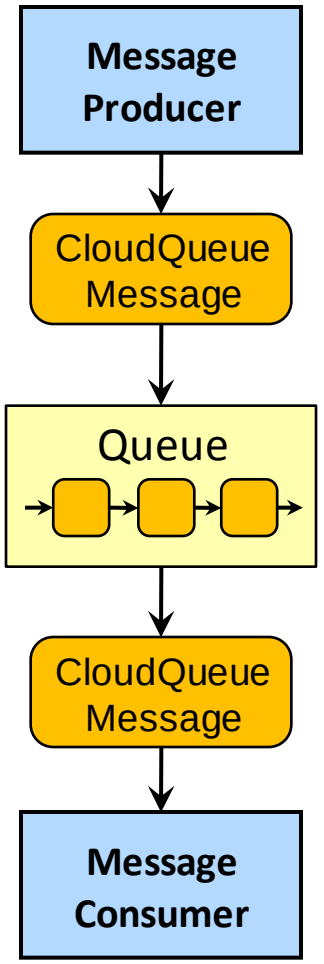
Programski primeri za najčešće scenarije upotrebe Blob-ova:

- How to: Create a container
- How to: Upload a Blob into a Container
- How to: List the Blobs in a container
- How to: Download a Blob
- How to: Delete a Blob
- How to: Delete a Blob container

mogu se naći na:
http://www.windowsazure.com/en-us/develop/php/how-to-guides/blob-service/

Redovi (Queues)

- **Redovi čekanja obezbeđuju pouzdan asinhroni mehanizam komunikacije.**
- **Pouzdanost**
  - Poruke se repetitivno isporučuju sve dok potrošač ne potvrdi da ih je uspešno procesirao →
  - Poruke se mogu procesirati i više puta po potrebi
- **Rasprezanje proizvođača i potrošača**
  - Različiti delovi aplikacije mogu biti implementirani različitim tehnologijama i programskim jezicima.
  - Proizvođač i potrošač ne moraju biti aktivni istovremeno.
- **Skalabilnost**
  - Redovi kompenzuju komunikacione špiceve i otklanjaju posledice otkaza servisnih komponenata.
  - Ako dužina reda raste → povećati broj potrošačkih instanci.



Redovi – Osobine i ograničenja

- **Red**
  - Broj poruka nije limitiran.
  - Mogu biti pridruženi meta-podaci.
  - Nije garantovan redosled isporuke poruka.
  - Poruke se isporučuju najmanje jednom → potrošač mora biti u stanju da se snađe sa porukama koje su isporučene više od jednog puta.
- **Poruka**
  - Poruka može biti veličine do 8KB.
  - Struktura poruke:
    - *MessageID:* a GUID
    - *VisibilityTimeout:* Ako nije potvrđeno da je poruka uzeta za procesiranje obrađena u ovom intervalu, poruka će se ponovo pojaviti u redu (default = 30 seconds)
    - *MessageTTL:* Time-to-live interval (maximum = default = 7 days)
    - *Payload:* String ili niz bajtova.

Primeri Queue API-ja

Programski primeri za najčešće scenarije upotrebe Queue-a:

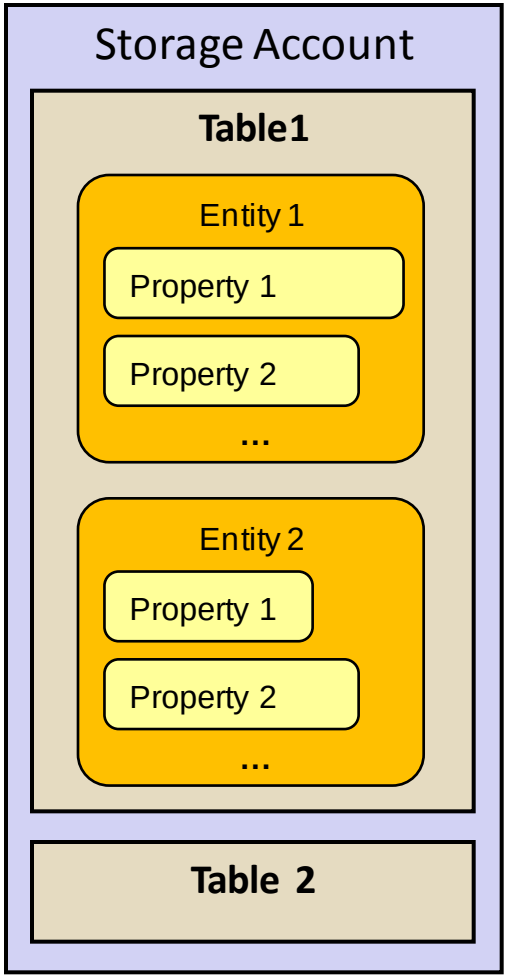
- How to: Create a queue
- How to: Add a message to a queue
- How to: Peek at the next message
- How to: De-queue the next message
- How to: Change the contents of a queued message
- Additional options for de-queuing messages
- How to: Get queue length
- How to: Delete a queue

mogu se naći na:

http://www.windowsazure.com/en-us/develop/php/how-to-guides/queue-service/

Tabele

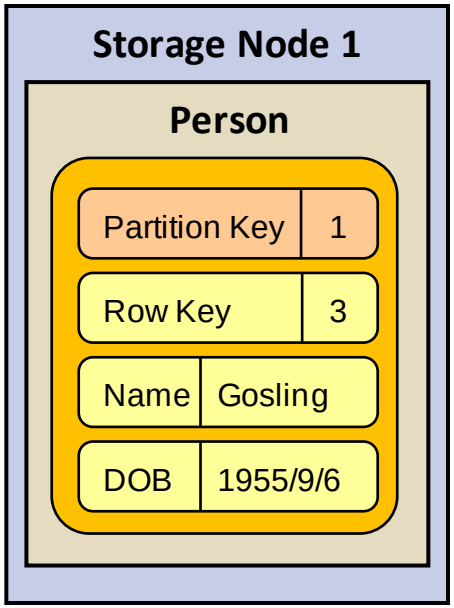
- Tabele obezbeđuju strukturirani smeštaj.
- Tabela sadrži skup **entiteta**.
- Svaki entitet sadrži skup **svojstava**.
- Svojstvo ima
  - Ime
  - Vrednost nekog standardnog tipa (Int32, Int64, double, string, bool, DateTime, GUID, byte array).
- Tabele nemaju fiksiranu shemu, tj. struktura svakog entiteta može biti različita.
- Svaki entiet ima dva svojstva ključa:
  - The partition key
  - the row key.
- Veličina entiteta ograničena je na 1MB.

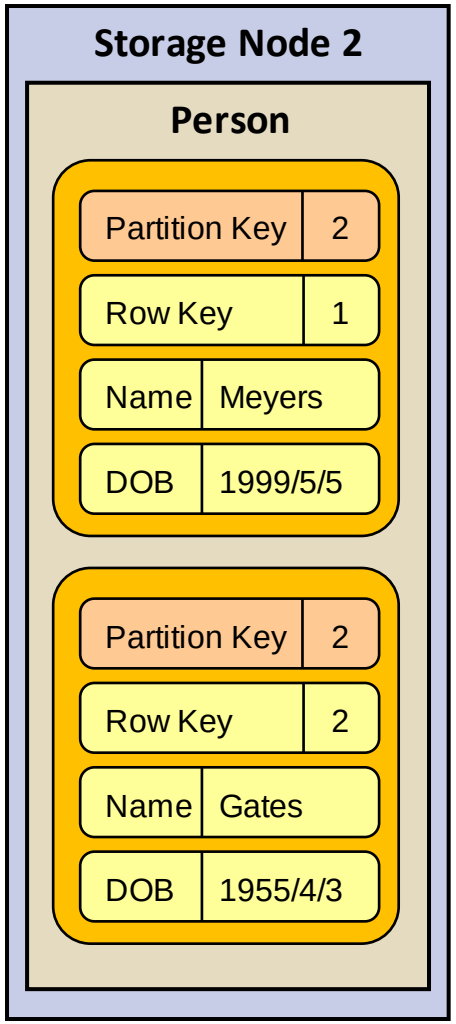


Particionisanje tabela

- Aplikacija kontroliše raspodelu entiteta dodelom particionih ključeva entitetima.
- Ako dva entiteta imaju različite particione ključeve mogu biti smeštena na različitim skladišnim čvorovima.
- Samo entiteti unutar iste tabele i iste particije mogu biti deo **transakcije nad grupom entiteta**.

- Problem: prava granularnost particija?
  - Poboljšana skalabilnost kada puno klijenata pristupa različitim skladišnim čvorovima.
  - Slabe performanse kada jedan klijent pristupa entitetima u različitim skladišnim čvorovima.





Primeri API operacija nad tabelom

Programski primeri za najčešće operacije nad Azure tabelama:

- **How to: Create a table**
- **How to: Add an entity to a table**
- **How to: Retrieve a single entity**
- **How to: Retrieve all entities in a partition**
- **How to: Retrieve a subset of entities in a partition**
- **How to: Retrieve a subset of entity properties**
- **How to: Update an entity**
- **How to: Batch table operations**
- **How to: Delete a table**

opisani su na:

http://www.windowsazure.com/en-us/develop/php/how-to-guides/table-service/

Rad sa entitetima

- **Definisanje entiteta**
  - Tabela skladišti entitete (koji nalikuju redovima rel. tabela). Entitet ima primarni ključ i skup osobina (properties). Osobina je par ime-vrednost (određenog tipa) kao kolona u rel. tabeli.
  - Dva entiteta u istoj tabeli mogu imati različite skupove osobina.
  - Klasa Entity() obezbeđuje neke osnovne (ugrađene) propertije:
    - PartitionKey (exposed trough getPartitionKey() and setPartitionKey())
    - RowKey (exposed trough getRowKey() and setRowKey())
    - Timestamp (exposed trough getTimestamp() and setTimestamp())
    - Etag value (exposed trough getEtag() and setEtag()) – služi za proveru da li je entitet menjan posle poslednjeg čitanja iz skladišta

Definisanje entiteta

- **Standardni tipovi:**
- Edm.Binary - An array of bytes up to 64 KB in size. This will be mapped to a string in PHP.
- Edm.Boolean - A boolean value. This will be mapped to a boolean in PHP.
- Edm.DateTime - A 64-bit value expressed as Coordinated Universal Time (UTC). The supported DateTime range begins from 12:00 midnight, January 1, 1601 A.D. (C.E.), Coordinated Universal Time (UTC). The range ends at December 31st, 9999. This will be mapped to a DateTime objectin PHP.
- Edm.Double - A 64-bit floating point value. This will be mapped to a double in PHP.
- Edm.Guid - A 128-bit globally unique identifier. This will be mapped to a string in PHP.
- Edm.Int32 - A 32-bit integer. This will be mapped to an integer in PHP.
- Edm.Int64 - A 64-bit integer. This will be mapped to an integer in PHP.
- Edm.String - A UTF-16-encoded value. String values may be up to 64 KB in size. This will be mapped to a string in PHP.

Postavljanje upita

- **Upit se može postaviti u vidu stringa (uslov filtriranja), na primer (konstruisanje $tableRestProxy nije prikazano):**

```
$filter = "Location eq 'Office' and DueDate lt '2012-11-5'";
try {
    $result = $tableRestProxy->queryEntities("mytable", $filter);
} catch(ServiceException $e){ ...}
```

- **Operatori u upitima:**

Operator	Query expression
Equal	eq
GreaterThan	gt
GreaterThanOrEqual	ge
LessThan	lt
LessThanOrEqual	le
NotEqual	ne

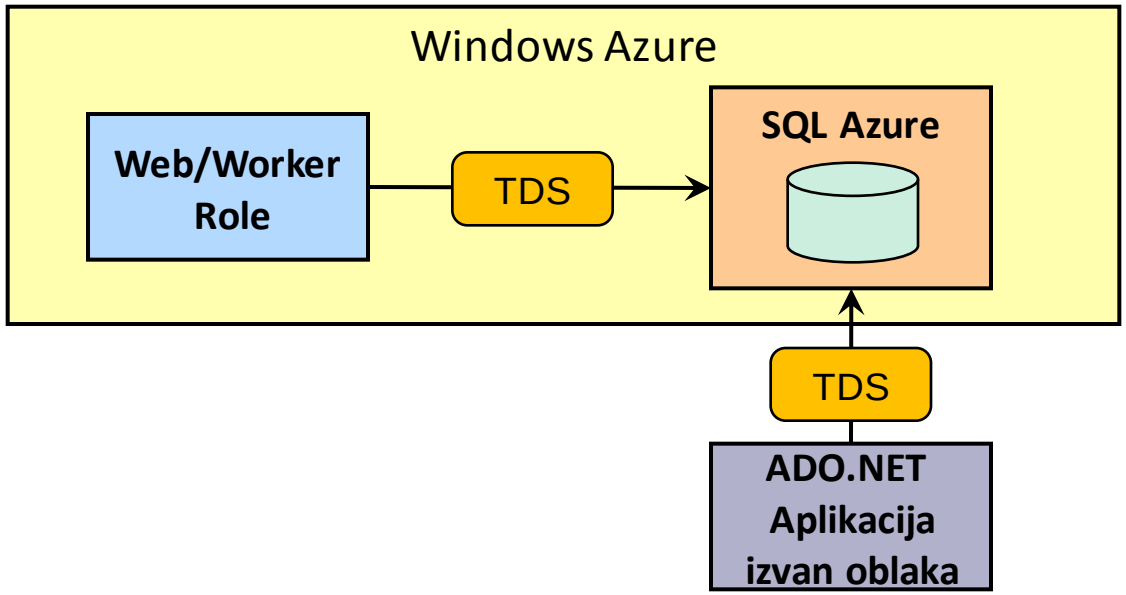
Additionally, the following operators are supported for Boolean properties:

Operator	Query expression
And	and
Not	not
Or	or

Detalljniji opis na: http://msdn.microsoft.com/en-us/library/windowsazure/dd894031.aspx

Azure SQL Database

- SQL Azure je relaciona baza podataka koja se nudi u obliku servisa iz oblaka.
- SQL Azure u osnovi ima iste funkcije kao Microsoft SQL Server koji se izvršava na klasičnom serveru.
- SQL Azure servis ima pristupnu tačku po *Tabular Data Stream* (TDS) protokolu. TDS je mrežni network protokol koji koristi SQL Server.



Azure SQL glavne odlike

- **Upravljivost**
  - Manji administrativni overhed
  - Dostupnost u kratkom vremenu
- **Visoka raspoloživost**
  - Servis replikuje podatke na veći broj fizičkih servera
  - Nema potrebe za backupom
- **Skalabilnost**
  - Jednostavno se povećava ili smanjuje broj baza.
  - Jedna baza je locirana na jednom skladišnom čvoru → ograničena skalabilnosdt unutar baze → alternativa je table storage.
- **Programerima poznato razvojno okruženje**
  - Relacioni model podataka
  - SQL Server tehnologija
- **Veličina baze je ograničena na 50GB**

Primer pristupa SQL Azure servisu iz PHPa

```
<?php
// pristupa se na isti način kao lokalnom sql serveru (slično kao i za mysql)
  $serverName = "tcp:nl8ex46z3u.database.windows.net,1433";
  $userName = 'si3psi@nl8ex46z3u';
  $userPassword = ‘xxxxx';
  $dbName = "TestDB";
  $table = "tablePHP";

  $connectionInfo = array("Database"=>$dbName, "UID"=>$userName,
    "PWD"=>$userPassword, "MultipleActiveResultSets"=>true);

  sqlsrv_configure('WarningsReturnAsErrors', 0);
  $conn = sqlsrv_connect( $serverName, $connectionInfo);
  if($conn === false)
  {
   FatalError("Failed to connect...");
  }

  CreateTable($conn, $table, "Col1 int primary key, Col2 varchar(20)");
```

Primer pristupa SQL Azure servisu iz PHPa

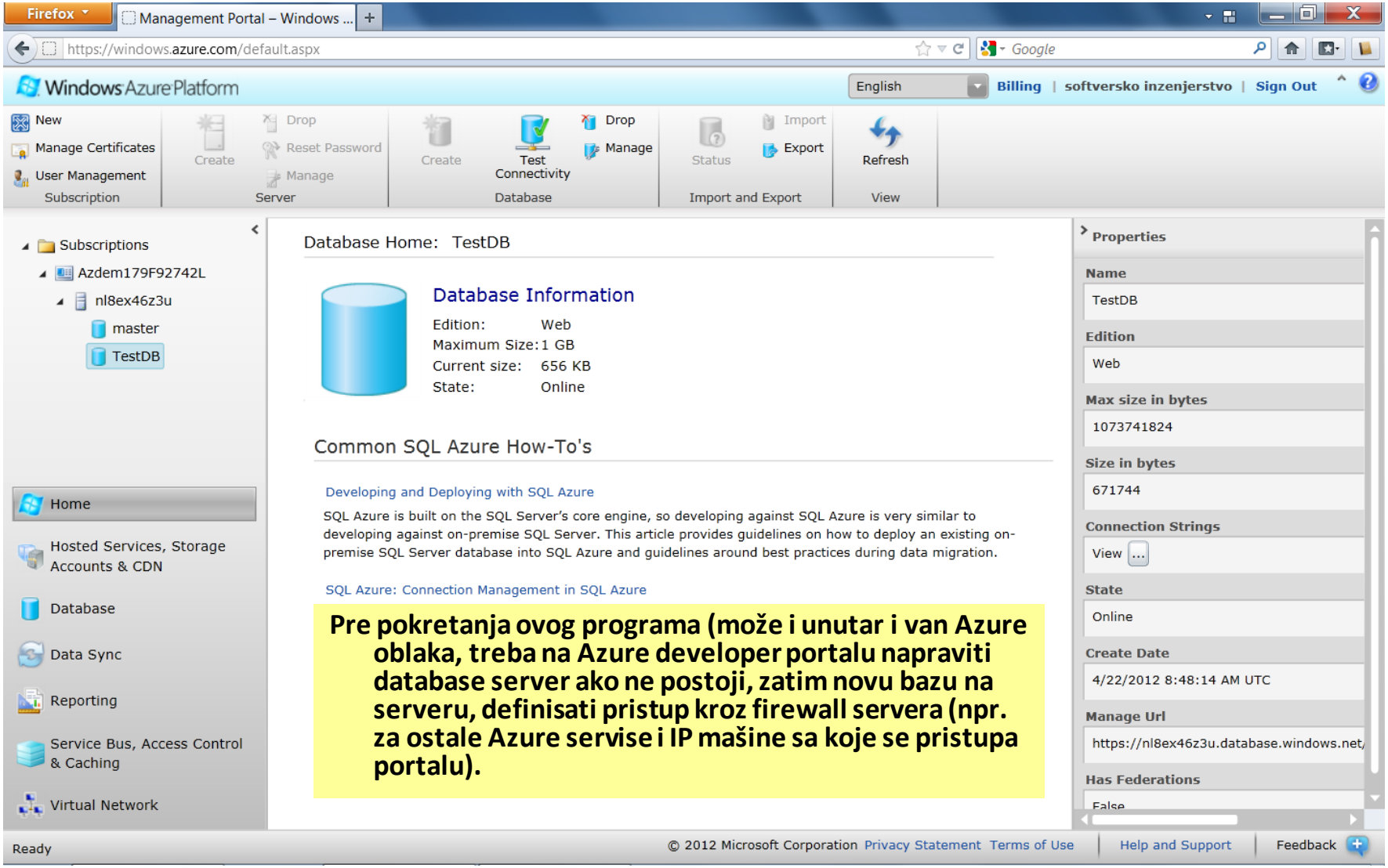
```
$tsql = "INSERT INTO [$table] (Col1, Col2) VALUES (1, 'string1'), (2, 'string2')";
 $stmt = sqlsrv_query($conn, $tsql);
 if ($stmt === false)
 {
    FatalError("Failed to insert data into test table: ".$tsql);
 }
 sqlsrv_free_stmt($stmt);

 $tsql = "SELECT Col1, Col2 FROM [$table]";
 $stmt = sqlsrv_query($conn, $tsql);
 if ($stmt === false)
 {
    FatalError("Failed to query test table: ".$tsql);
 }
 else
 {
      while($row = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_NUMERIC))
    {
        echo "Col1: ".$row[0]."\n";
        echo "Col2: ".$row[1]."\n";
    }
    sqlsrv_free_stmt($stmt);
 }
 sqlsrv_close($conn);
```

Primer pristupa SQL Azure servisu iz PHPa

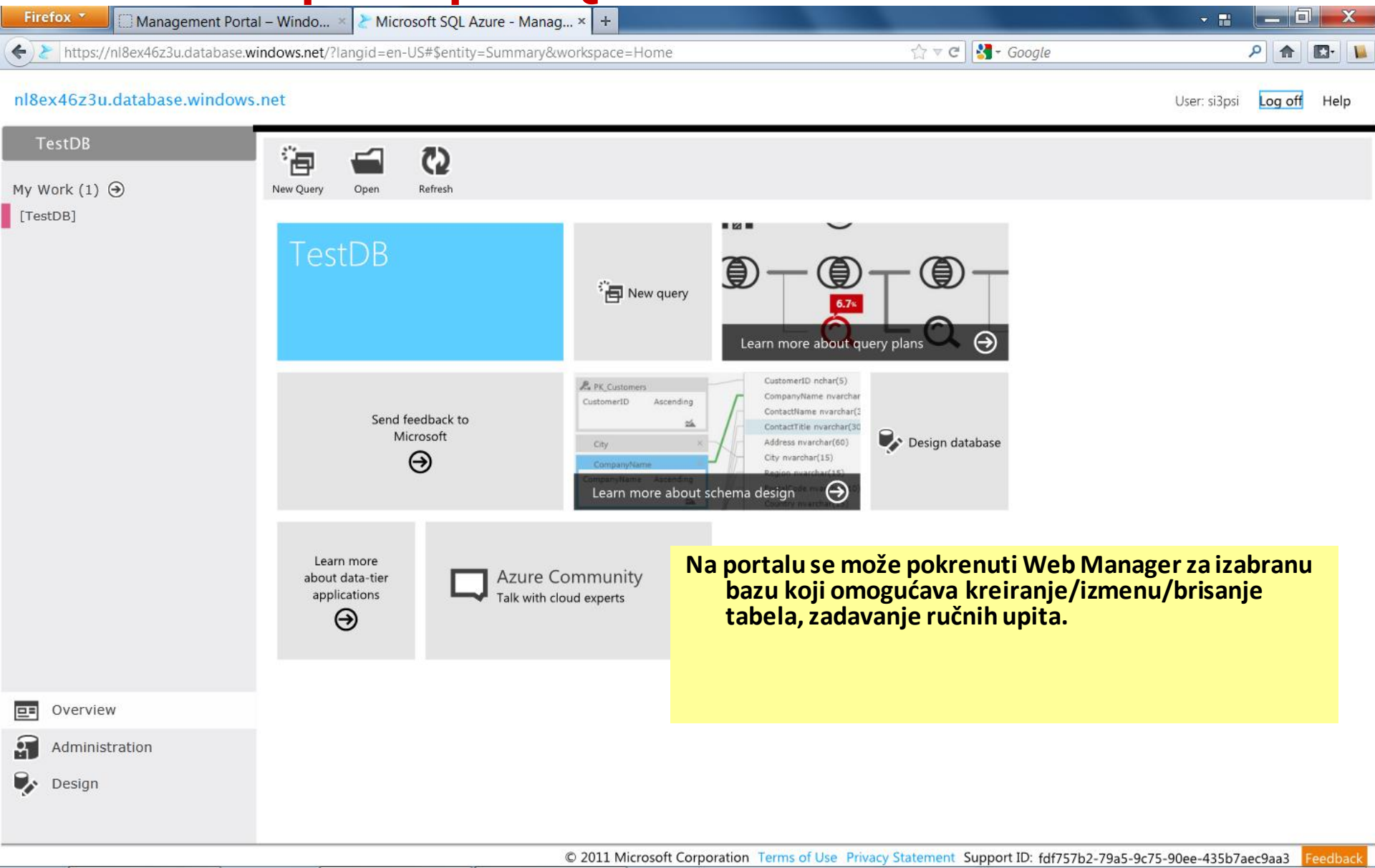
```
function CreateTable($conn, $tableName, $dataType)
{
  $sql = "CREATE TABLE [$tableName] ($dataType)";
  DropTable($conn,$tableName);
  $stmt = sqlsrv_query($conn, $sql);
  if ($stmt === false)
  {
    FatalError("Failed to create test table: ".$sql);
  }
  sqlsrv_free_stmt($stmt);
}

function DropTable($conn, $tableName)
{
   $stmt = sqlsrv_query($conn,
          "DROP TABLE [$tableName]");
   if ($stmt === false)
   {
   }
   else
   {
    sqlsrv_free_stmt($stmt);
   }
}
```

```
function FatalError($errorMsg)
{
   Handle_Errors();
   die($errorMsg."\n");
}


function Handle_Errors()
{
   $errors = sqlsrv_errors(SQLSRV_ERR_ERRORS);
   $count = count($errors);
   if($count == 0)
   {
     $errors = sqlsrv_errors(SQLSRV_ERR_ALL);
     $count = count($errors);
   }
   if($count > 0)
   {
    for($i = 0; $i < $count; $i++)
    {
      echo $errors[$i]['message']."\n";
    }
   }
}

?>
```

Primer pristupa SQL Azure servisu iz PHPa



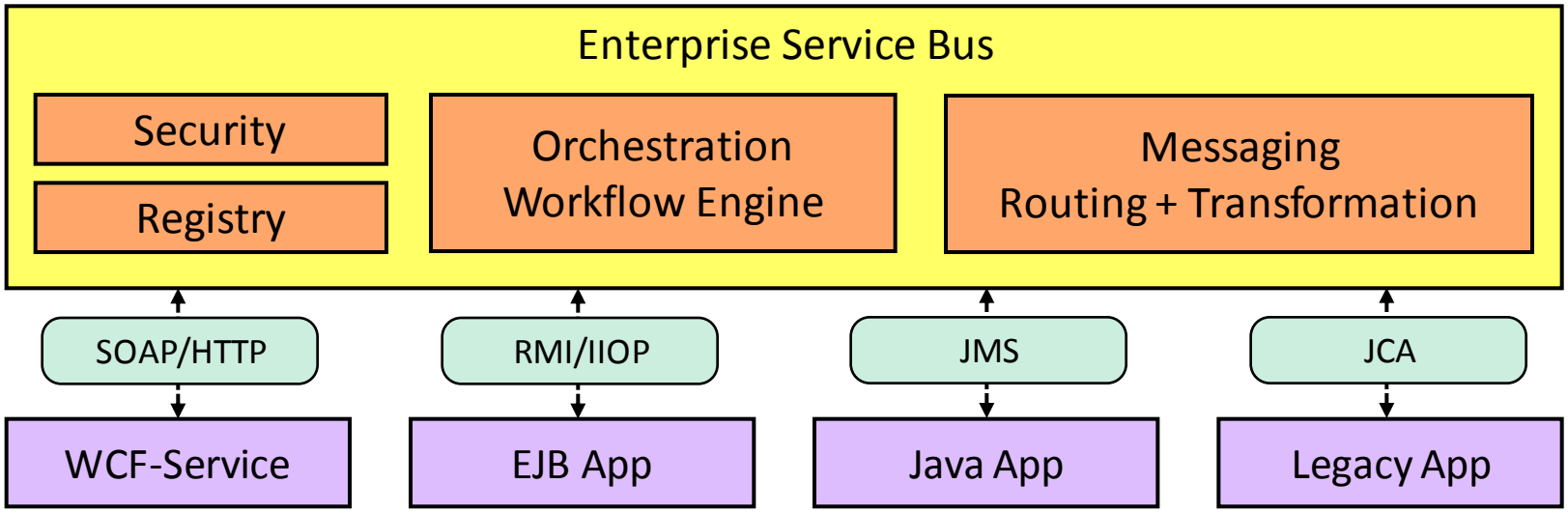
Primer pristupa SQL Azure servisu iz PHPa



Na portalu se može pokrenuti Web Manager za izabranu bazu koji omogućava kreiranje/izmenu/brisanje tabela, zadavanje ručnih upita.

Windows Azure Service Bus (serv. magistrala)

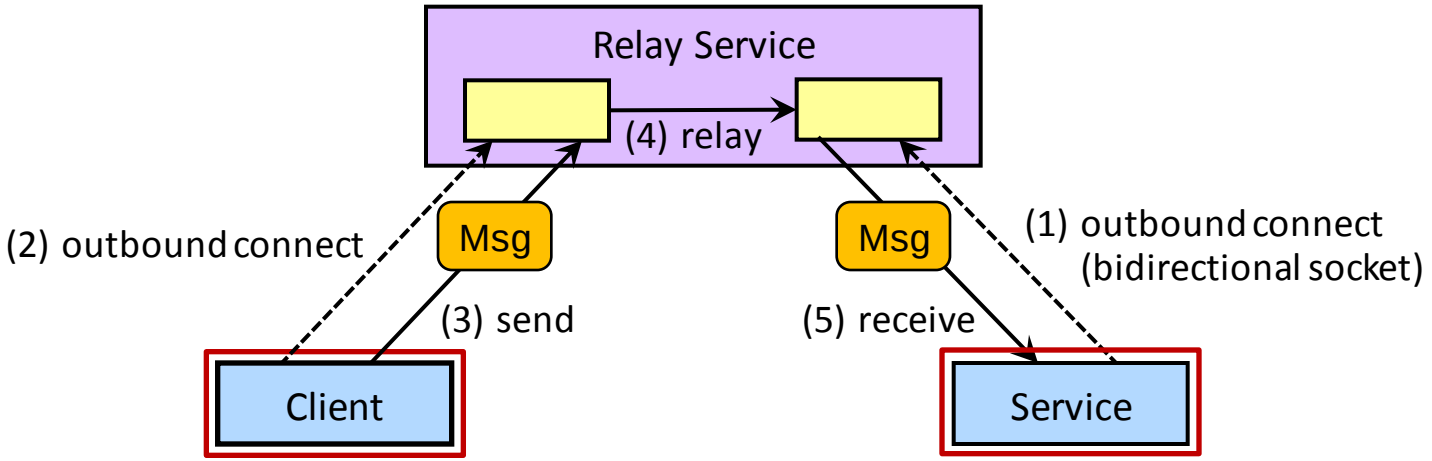
- Zasnovan na ideji **Enterprise Service Bus** (ESB)



- **Windows Azure Service Bus** obezbeđuje na nivou interneta:
  - Kontrolu identiteta i prava pristupa
  - Uniformno imenovanje, registar servisa
  - Infrastrukturu za poruke (messaging fabric)

Bidirekciona komunikacija putem Interneta

- Bidirekciona komunikacija je u principu ograničena na internetu:
  - Network Address Translation (NAT): IP-jevi nisu javno adresibilni.
  - Firewalls: Samo neki izlazni portovi i još manji broj ulaznih se dozvoljava (tipično 80 (HTTP) i 443 (HTTPS)).
- Osnovna ideja za zaobilaženje ovih restrikcija:
  - R**elejni servise** posreduje između klijenta i servisa.
  - Klijentu i Servisu potrebne su jedino izlazne konekcije.



Imenovanje i registar servisa

- Imenovanje
  - DNS je zasnovan na javnim IP adresama, hostovi iza NAT rutera ne mogu se identifikovati.
  - Imenovanje na servisnoj magistrali služi da identifikuje pristupne tačke servisa na način nezavistan od hosta.
  - Sistem imenovanja: *[scheme]://[service-namespace]*.
    servicebus.windows.net/*name1*/*name2*
    - *[scheme]:* "http"/"https" za HTTP pristupne tačke, "sb" za TCP prist. tačke
    - *[service-namespace]:* kreira se kada se otvori račun za Service Bus
    - *name1, name2:* specifikuje hijerahiju imenovanja u korisničkom imenskom prostoru
  - Example: sb://user1.servicebus.windows.net/azure/calculator
- Registar servisa
  - Pristupne tačke servisa mogu se napraviti javno dostupnim.
  - Informacija o prist. tačkama se vraća u Atom 1.0 formatu.

Content Delivery Network

- Windovs Azure Content Delivery Network (CDN) nudi programerima globalno rešenje za isporuku sadržaja velikog propusnog opsega tako što keširan blob-ove i statički sadržaj compute čvorova na fizičke čvorove u Sjedinjenim Američkim Državama, Evropi, Aziji, Australiji i Južnoj Americi.
- Time se poboljšava odziv aplikacija u uslovima trenutnog visokog opterećenja

Content Delivery Network

CDN se “uključuje” za storage preko Azure management portala.

- CDNu se pristupa pomoću URLa oblika:
  http://<*guid*>.vo.msecnd.net/<*myPublicContainer*>/<*BlobName*>