Microsoft

The Windows Azure Application Model

December 2011

Contents

The Windows Azure Application Model

Windows Azure enables you to deploy and monitor your application code running inside a Microsoft data center. When you create an application and run it on Windows Azure, the code and configuration together is called a Windows Azure hosted service. Hosted services are easy to manage, scale up and down, reconfigure, and update with new versions of your application's code. This article focuses on the Windows Azure hosted service application model.

Windows Azure Application Model Benefits

When you deploy your application as a hosted service, Windows Azure creates one or more virtual machines (VMs) that contain your application's code, and boots the VMs on physical machines residing in one of the Windows Azure data centers. As client requests to your hosted application enter the data center, a load balancer distributes these requests equally to the VMs. While your application is hosted in Windows Azure, it gets three key benefits:

- **High availability.** High availability means Windows Azure ensures that your application is running as much as possible and is able to respond to client requests. If your application terminates (due to an unhandled exception, for example), then Windows Azure will detect this, and it will automatically re-start your application. If the machine your application is running on experiences some kind of hardware failure, then Windows Azure will also detect this and automatically create a new VM on another working physical machine and run your code from there. NOTE: In order for your application to get Microsoft's Service Level Agreement of 99.95% available, you must have at least two VMs running your application code. This allows one VM to process client requests while Windows Azure moves your code from a failed VM to a new, good VM.

- **Scalability.** Windows Azure lets you easily and dynamically change the number of VMs running your application code to handle the actual load being placed on your application. This allows you to adjust your application to the workload that your customers are placing on it while paying only for the VMs you need when you need them. When you want to change the number of VMs, Windows Azure responds within minutes making it possible to dynamically change the number of VMs running as often as desired.

- **Manageability.** Because Windows Azure is a Platform as a Service (PaaS) offering, it manages the infrastructure (the hardware itself, electricity, and networking) required to keep these machines running. Windows Azure also manages the platform, ensuring an up-to-date operating system with all the correct patches and security updates, as well as component updates such as the .NET Framework and Internet Information Server. Because all the VMs are running Windows Server 2008, Windows Azure provides additional features such as diagnostic monitoring, remote desktop support, firewalls, and certificate store configuration. All these features are provided at no extra cost. In fact, when you run your application in Windows Azure, the Windows Server 2008 operating system (OS) license is included. Since all of the VMs are running Windows Server 2008, any code that runs on Windows Server 2008 works just fine when running in Windows Azure.

Hosted Service Core Concepts

When your application is deployed as a hosted service in Windows Azure, it runs as one or more *roles*. A *role* simply refers to application files and configuration. You can define one or more roles for your application, each with its own set of application files and configuration. For each role in your application, you can specify the number of VMs, or *role instances*, to run. The figure below show two simple examples of an application modeled as a hosted service using roles and role instances.

Figure 1: A single role with three instances (VMs) running in a Windows Azure data center

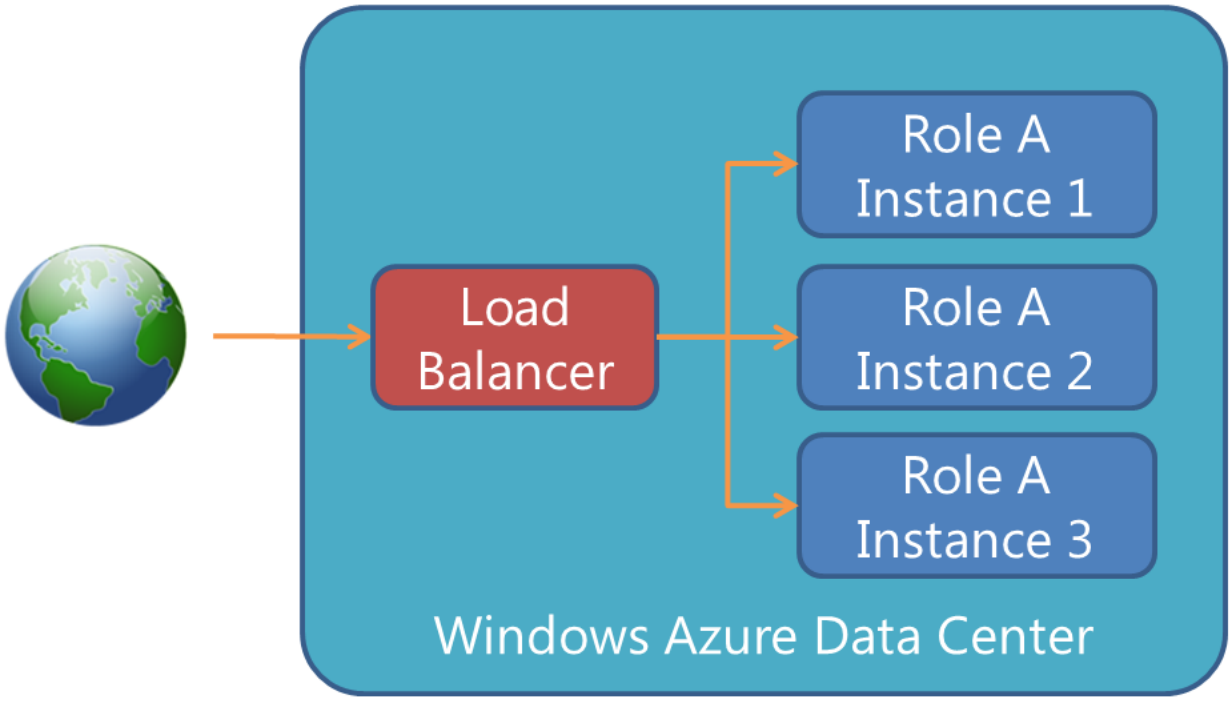
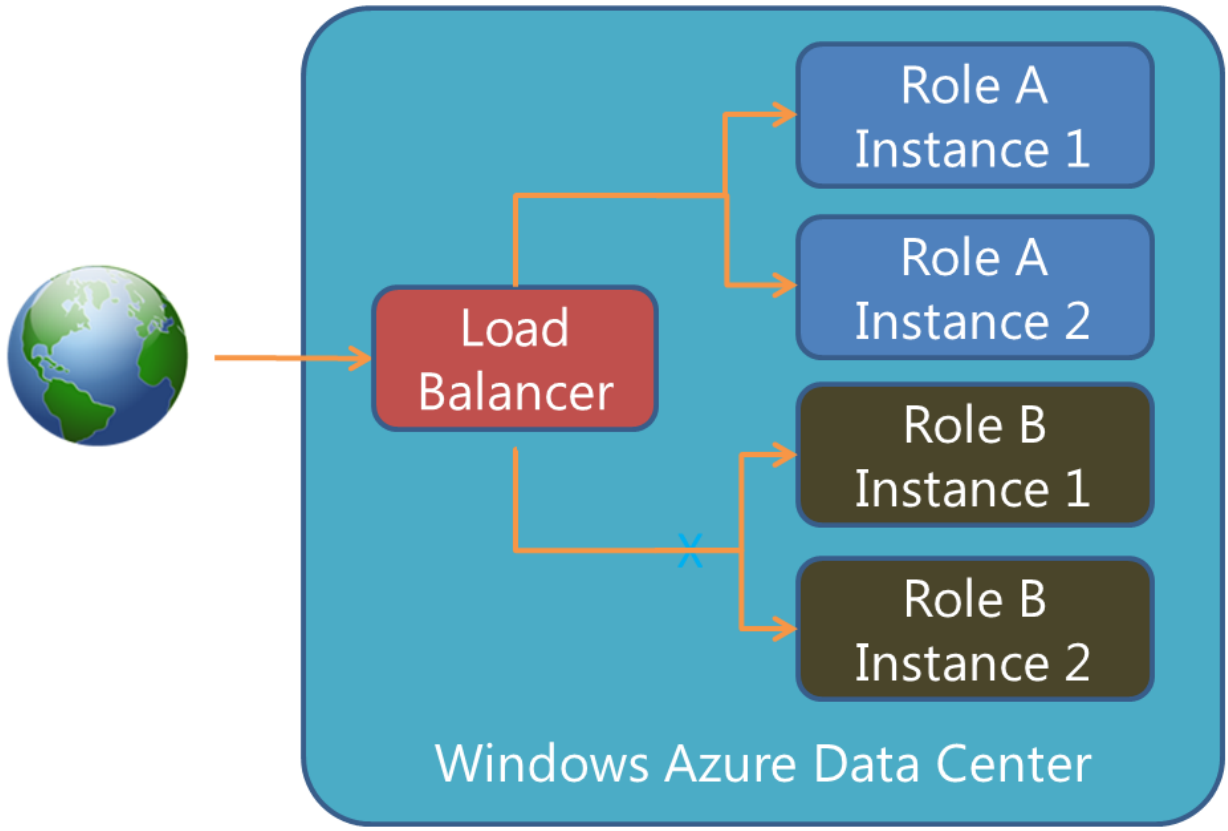
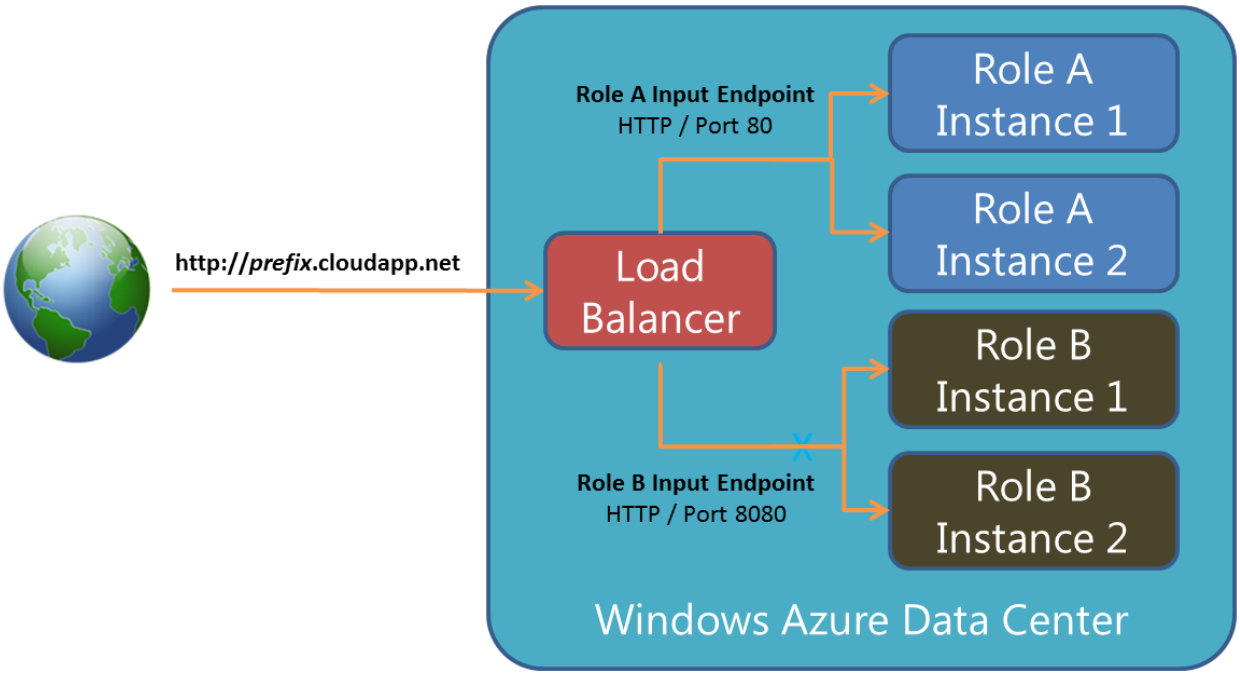


Figure 2: Two roles, each with two instances (VMs), running in a Windows Azure data center



Role instances typically process Internet client requests entering the data center through what is called an *input endpoint*. A single role can have 0 or more input endpoints. Each endpoint indicates a protocol (HTTP, HTTPS, or TCP) and a port. It is common to configure a role to have two input endpoints: HTTP listening on port 80 and HTTPS listening on port 443. The figure below shows an example of two different roles with different input endpoints directing client requests to them.



When you create a hosted service in Windows Azure, it is assigned a publicly addressable IP address that clients can use to access it. Upon creating the hosted service you must also select a URL prefix that is mapped to that IP address. This prefix must be unique as you are essentially reserving the *prefix*.cloudapp.net URL so that no one else can have it. Clients communicate with your role instances by using the URL. Usually, you will not distribute or publish the Windows Azure *prefix*.cloudapp.net URL. Instead, you will purchase a DNS name from your DNS registrar of choice and configure your DNS name to redirect client requests to the Windows Azure URL. For more details, see Configuring a Custom Domain Name in Windows Azure.

Hosted Service Design Considerations

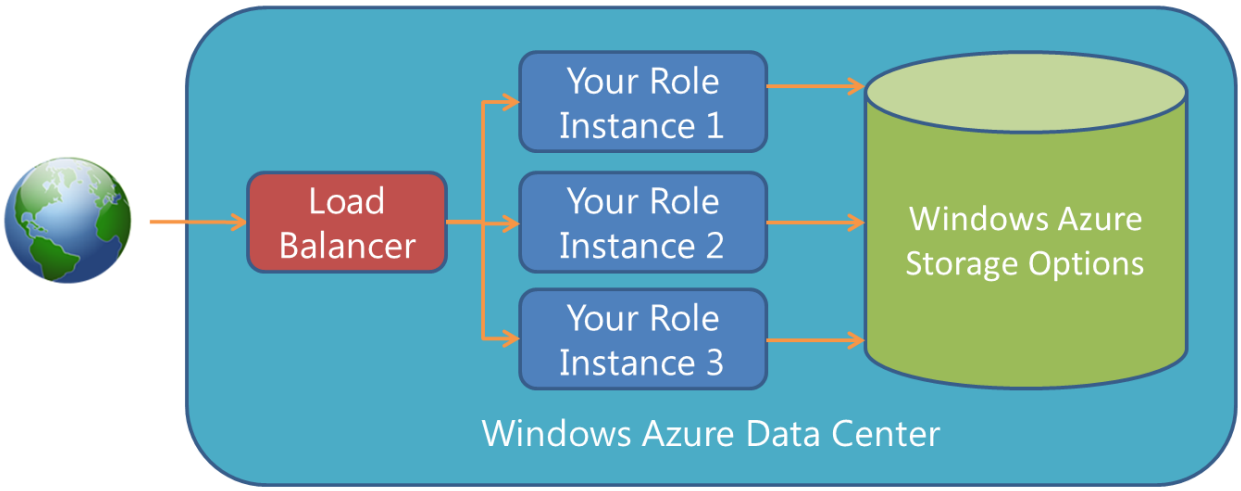
When designing an application to run in a cloud environment, there are several considerations to think about such as latency, high-availability, and scalability.

Deciding where to locate your application code is an important consideration when running a hosted service in Windows Azure. It is common to deploy your application to data centers that are closest to your clients to reduce latency and get the best performance possible. However, you might choose a data center closer to your company or closer to your data if you have some jurisdictional or legal concerns about your data and where it resides. There are six data centers around the globe capable of hosting your application code. The table below shows the available locations:

Country/Region	Sub-regions	
United States	South Central	North Central
Europe	North	West
Asia	Southeast	East

When creating a hosted service, you select a sub-region indicating the location in which you want your code to execute.

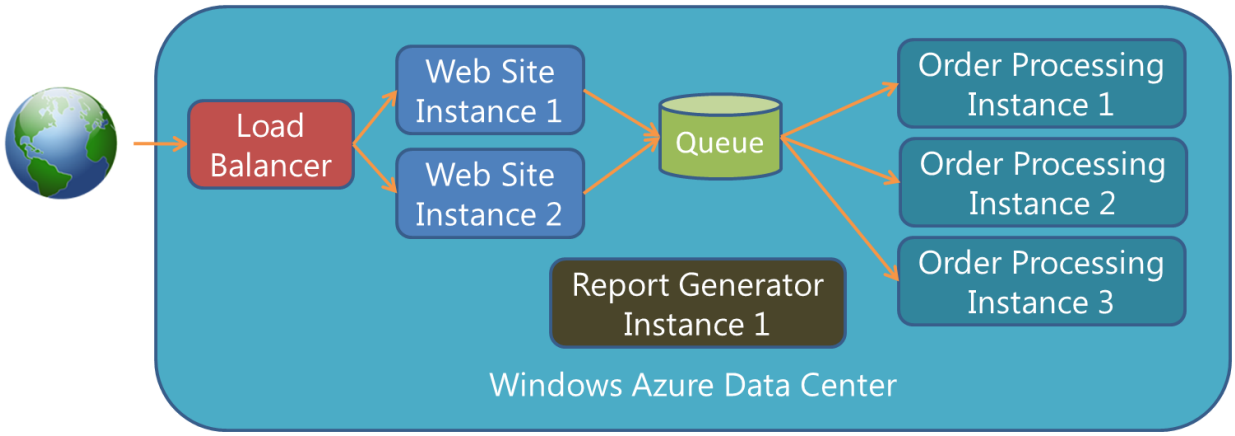
To achieve high availability and scalability, it is critically important that your application's data be kept in a central repository accessible to multiple role instances. To help with this, Windows Azure offers several storage options such as blobs, tables, and SQL Azure. Please see the Data Storage Offerings in Windows Azure article for more information about these storage technologies. The figure below shows how the load balancer inside the Windows Azure data center distributes client requests to different role instances all of which have access to the same data storage.



Usually, you want to locate your application code and your data in the same data center as this allows for low latency (better performance) when your application code accesses the data. In addition, you are not charged for bandwidth when data is moved around within the same data center.

Designing your Application for Scale

Sometimes, you may want to take a single application (like a simple web site) and have it hosted in Windows Azure. But frequently, your application may consist of several roles that all work together. For example, in the figure below, there are two instances of the Web Site role, three instances of the Order Processing role, and one instance of the Report Generator role. These roles are all working together and the code for all of them can be packaged together and deployed as a single unit up to Windows Azure.



The main reason to split an application into different roles each running on its own set of role instances (that is, VMs) is to scale the roles independently. For example, during the holiday season, many customers may be purchasing products from your company, so you might want to increase the number of role instances running your Web Site role as well as the number of role instances running your Order Processing role. After the holiday season, you may get a lot of products returned, so you may still need a lot of Web Site instances but fewer Order Processing instances. During the rest of the year, you may only need a few Web Site and Order Processing instances. Throughout all of this, you may need only one Report Generator instance. The flexibility of role-based deployments in Windows Azure enables you to easily adapt your application to your business needs.

It's common to have the role instances within your hosted service communicate with each other. For example, the web site role accepts a customer's order but then it offloads the order processing to the Order Processing role instances. The best way to pass work form one set of role instances to another set of instances is using the queuing technology provided by Windows Azure, either the Queue Service or Service Bus Queues. The use of a queue is a critical part of the story here. The queue allows the hosted service to scale its roles independently allowing you to balance the workload against cost. If the number of messages in the queue increases over time, then you can scale up the number of Order Processing role instances. If the number of messages in the queue decreases over time, then you can scale down the number of Order Processing role instances. This way, you are only paying for the instances required to handle the actual workload.

The queue also provides reliability. When scaling down the number of Order Processing role instances, Windows Azure decides which instances to terminate. It may decide to terminate an instance that is in the middle of processing a queue message. However, because the message processing does not complete successfully, the message becomes visible again to another Order Processing role instance that picks it up and processes it. Because of queue message visibility, messages are guaranteed to eventually get processed. The queue also acts as a load balancer by effectively distributing its messages to any and all role instances that request messages from it.

For the Web Site role instances, you can monitor the traffic coming into them and decide to scale the number of them up or down as well. The queue allows you to scale the number of Web Site role instances independently of the Order Processing role instances. This is very powerful and gives you a lot of flexibility. Of course, if your application consists of additional roles, you could add additional queues as the conduit to communicate between them in order to leverage the same scaling and cost benefits.

Hosted Service Definition and Configuration

Deploying a hosted service to Windows Azure requires you to also have a service definition file and a service configuration file. Both of these files are XML files, and they allow you to declaratively specify deployment options for your hosted service. The service definition file describes all of the roles that make up your hosted service and how they communicate. The service configuration file describes the number of instances for each role and settings used to configure each role instance.

The Service Definition File

As I mentioned earlier, the service definition (CSDEF) file is an XML file that describes the various roles that make up your complete application. The complete schema for the XML file can be found here: http://msdn.microsoft.com/en-

us/library/windowsazure/ee758711.aspx. The CSDEF file contains a WebRole or WorkerRole element for each role that you want in your application. Deploying a role as a web role (using the WebRole element) means that the code will run on a role instance containing Windows Server 2008 and Internet Information Server (IIS). Deploying a role as a worker role (using the WorkerRole element) means that the role instance will have Windows Server 2008 on it (IIS will not be installed).

You can certainly create and deploy a worker role that uses some other mechanism to listen for incoming web requests (for example, your code could create and use a .NET HttpListener). Since the role instances are all running Windows Server 2008, your code can perform any operations that are normally available to an application running on Windows Server 2008.

For each role, you indicate the desired VM size that instances of that role should use. The table below shows the various VM sizes available today and the attributes of each:

VM Size	CPU	RAM	Disk	Peak Network I/O
Extra Small	1 x 1.0 GHz	768 MB	20GB	~5 Mbps
Small	1 x 1.6 GHz	1.75 GB	225GB	~100 Mbps
Medium	2 x 1.6 GHz	3.5 GB	490GB	~200 Mbps
Large	4 x 1.6 GHz	7 GB	1TB	~400 Mbps
Extra Large	8 x 1.6 GHz	14 GB	2TB	~800 Mbps

You are charged hourly for each VM you use as a role instance and you are also charged for any data that your role instances send outside the data center. You are not charged for data entering the data center. For more information, see Windows Azure Pricing. In general, it is advisable to use many small role instances as opposed to a few large instances so that your application is more resilient to failure. After all, the fewer role instances you have, the more disastrous a failure in one of them is to your overall application. Also, as mentioned before, you must deploy at least two instances for each role in order to get the 99.95% service level agreement Microsoft provides.

The service definition (CSDEF) file is also where you would specify many attributes about each role in your application. Here are some of the more useful items available to you:

- **Certificates**. You use certificates for encrypting data or if your web service supports SSL. Any certificates need to be uploaded to Windows Azure. For more information, see Managing Certificates in Windows Azure. This XML setting installs previously-uploaded certificates into the role instance's certificate store so that they are usable by your application code.
- **Configuration Setting Names**. For values that you want your application(s) to read while running on a role instance. The actual value of the configuration settings is set in the service configuration (CSCFG) file which can be updated at any time without requiring you to redeploy your code. In fact, you can code your applications in such a way to detect the changed configuration values without incurring any downtime.
- **Input Endpoints**. Here you specify any HTTP, HTTPS, or TCP endpoints (with ports) that you want to expose to the outside world via your *prefix*.cloadapp.net URL. When Windows Azure deploys your role, it will configure the firewall on the role instance automatically.
- **Internal Endpoints**. Here you specify any HTTP or TCP endpoints that you want exposed to other role instances that are deployed as part of your application. Internal endpoints allow all the role instances within your application to talk to each other but are not accessible to any role instances that are outside your application.

- **Import Modules**. These optionally install useful components on your role instances. Components exist for diagnostic monitoring, remote desktop, and Windows Azure Connect (which allows your role instance to access on-premises resources through a secure channel).
- **Local Storage**. This allocates a subdirectory on the role instance for your application to use. It is described in more detail in the Data Storage Offerings in Windows Azure article.
- **Startup Tasks**. Startup tasks give you a way to install prerequisite components on a role instance as it boots up. The tasks can run elevated as an administrator if required.

The Service Configuration File

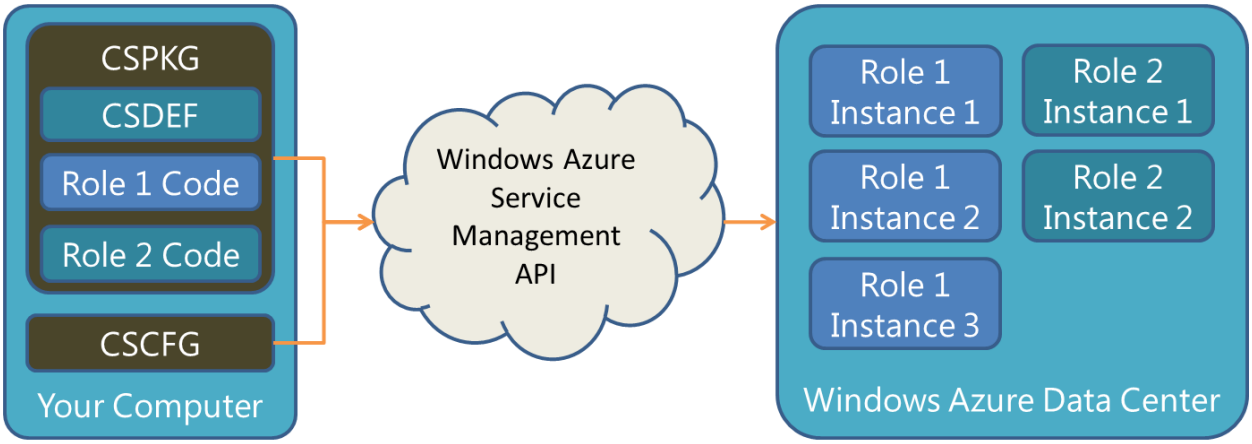
The service configuration (CSCFG) file is an XML file that describes settings that can be changed without redeploying your application. The complete schema for the XML file can be found here: http://msdn.microsoft.com/en-us/library/windowsazure/ee758710.aspx. The CSCFG file contains a Role element for each role in your application. Here are some of the items you can specify in the CSCFG file:

- **OS Version**. This attribute allows you to select the operating system (OS) version you want used for all the role instances running your application code. This OS is known as the *guest OS*, and each new version includes the latest security patches and updates available at the time the guest OS is released. If you set the osVersion attribute value to "*", then Windows Azure automatically updates the guest OS on each of your role instances as new guest OS versions become available. However, you can opt out of automatic updates by selecting a specific guest OS version. For example, setting the osVersion attribute to a value of "WA-GUEST-OS-2.8_201109-01" causes all your role instances to get what is described on this web page: http://msdn.microsoft.com/en-us/library/hh560567.aspx. For more information about guest OS versions, see Managing Upgrades to the Windows Azure Guests OS.
- **Instances**. This element's value indicates the number of role instances you want provisioned running the code for a particular role. Since you can upload a new CSCFG file to Windows Azure (without redeploying your application), it is trivially simple to change the value for this element and upload a new CSCFG file to dynamically increase or decrease the number of role instances running your application code. This allows you to easily scale your application up or down to meet actual workload demands while also controlling how much you are charged for running the role instances.
- **Configuration Setting Values**. This element indicates values for settings (as defined in the CSDEF file). Your role can read these values while it is running. These configuration settings values are typically used for connection strings to SQL Azure or to Windows Azure Storage, but they can be used for any purpose you desire.

Creating and Deploying a Hosted Service

Creating a hosted service requires that you first go to the Windows Azure Management Portal and provision a hosted service by specifying a DNS prefix and the data center you ultimately want your code running in. Then in your development environment, you create your service definition (CSDEF) file, build your application code and package (zip) all these files into a service package (CSPKG) file. You must also prepare your service configuration (CSCFG) file. To deploy your role, you upload the CSPKG and CSCFG files with the Windows Azure Service Management API. Once deployed, Windows Azure, will provision role instances in the data center (based upon the configuration data), extract your application code from the package, copy it to the role instances, and boot the instances. Now, your code is up and running.

The figure below shows the CSPKG and CSCFG files you create on your development computer. The CSPKG file contains the CSDEF file and the code for two roles. After uploading the CSPKG and CSCFG files with the Windows Azure Service Management API, Windows Azure creates the role instances in the data center. In this example, the CSCFG file indicated that Windows Azure should create three instances of role #1 and two instances of Role #2.



For more information about deploying, upgrading, and reconfiguring your roles, see the Deploying and Updating Windows Azure Applications article.

References

- Creating a Hosted Service for Windows Azure
- Managing Hosted Services in Windows Azure
- Migrating Applications to Windows Azure
- Configuring a Windows Azure Application