
The A-Y of running BizTalk Server in Microsoft Azure

Technical Whitepaper

Author: Johann Cooper

Brought to you by:

DATACOM

BizTalk360

Table of Contents

1 About the Author3

2 Document Control.....4

 2.1 Audience and Scope4

 2.2 Revision History4

 2.3 Contributors.....4

3 Abstract.....5

4 Introduction5

5 Should you make the move?7

6 BizTalk Server Topologies in Azure9

 6.1 Self-Contained Development Topology9

 6.2 Multi-Server Topology.....10

 6.3 On-Premise Connectivity based Multi-Server Topology.....12

 6.4 Multi-Region Multi-Server Topology.....13

7 Economics of running BizTalk Server in Azure14

8 BizTalk Server Performance in Azure19

9 Step by Step Guide to Setting up BizTalk Server in Azure21

 9.1 VNets – the backbone of your BizTalk IAAS topology21

 9.2 So what does a Cloud Service have to do with VMs?22

 9.3 Creating the VMs.....23

 9.4 Provisioning additional SQL data disks25

 9.5 Availability Sets – High Availability in Azure.....26

 9.6 Autoscaling – just how much BizTalk do you really need?27

 9.7 Endpoints and Load Balancing29

 9.8 Cloud Service Certificates and HTTPS Endpoints32

 9.9 Getting rid of WS-A Address Filter Mismatch errors.....33

10 Appendix33

 10.1 Monitoring your BizTalk Server Environments in Azure.....33

 10.2 Exploring WSFC in Azure34

 10.3 Demonstrating Autoscaling in Action.....34

 10.4 Glossary of Acronyms.....36

11 Acknowledgements.....38

1 About the Author

Johann Cooper is an Integration Consultant working at Datacom, which is an IT-based service provider in New Zealand, focusing primarily on BizTalk Server and Azure.

Johann is also an active blogger, frequently writing about BizTalk Server, WCF, and Azure on his personal blog site titled Adventures inside the Message Box. He is also the creator of the BizTalk BRE Pipeline Framework CodePlex project and is a huge fan of the BizTalk Business Rules Engine which tends to feature heavily in solutions he builds. It is not unheard of for colleagues to jokingly refer to him as Mr. BRE.

2 Document Control

2.1 Audience and Scope

This white paper is targeted primarily at BizTalk developers/consultants and IT infrastructure specialists who need to understand and appreciate Azure concepts and relate them back to their own knowledge of creating BizTalk Server environments.

The scope of this white paper is to illustrate how BizTalk Server can be implemented in Microsoft Azure and the associated economic impacts of doing so, and also explains some of the core Azure concepts that implementers will need to be familiar with. These concepts will be further illustrated with some potential topologies. To be clear, this paper aims to explore how you can leverage IAAS capabilities with Microsoft's BizTalk Server product and is not about Microsoft's PAAS offering Microsoft Azure BizTalk Services (MABS).

The goal of this white paper is not to be a one stop shop for all things Azure IAAS (the topic is too large to cover in anything smaller than a book, and is also too fast changing) but rather to be a good starting point for readers to understand concepts before they branch out to do more research.

It is assumed that readers are well versed in installing, configuring and to some extent using BizTalk Server and will not provide many steps detailing these areas, however will focus on the Azure concepts and implementation steps which would be rather new to those who haven't yet explored the detailed Azure features.

Given the fast moving pace of developments and new features in the Azure space, best efforts will be made to keep this white paper up to date. Please ensure that you are reading the most up to date version of the document from the BizTalk360 [website](#), and take note of the date when the document was published if you believe that certain sections seem to be out of date.

2.2 Revision History

Date	Revision	Overview of Changes	Author(s)
22/10/2014	1.0	First published version	Johann Cooper

2.3 Contributors

Name	Role
Johan Cooper	Author
Steef-Jan Wiggers	Reviewer
Michael Stephenson	Reviewer
Saravana Kumar	Reviewer
Craig Beetlestone	Reviewer
Mark Brimble	Reviewer

Colin Dijkgraaf	Reviewer
James Corbould	Reviewer
Ricardo Torre	Reviewer
Peter Nield	Reviewer
Brett Atkin	Reviewer

3 Abstract

Running BizTalk Server in Microsoft Azure offers many benefits that would not be available in an on premises deployment. Azure IAAS enables customers to bypass heavy investment into network and server infrastructure and to eliminate the need for heavy maintenance of that infrastructure. Leveraging Azure IAAS also eliminates a lot of the lead time required to provision equivalent components on premises, enabling projects to prevent costly holdups.

Microsoft boasts a per minute pay as you go model for VMs which means that you only pay for what you use, and has extended this to licensing for SQL Server and BizTalk Server as well. This pay as you go model is even more cost effective when combined with autoscaling, which allows you to spin up and down VMs as and when they are demanded, thus paying for the extra capacity only when it is actually used. Creative use of these models can help customers to save money and to maximize their return on investment in the BizTalk Server platform.

A major constraint when running BizTalk Server in Azure is the current lack of high availability for BizTalk Server environments. While Azure has catered for high availability on the BizTalk Server runtime layer, there is no high availability for SQL Server at this time (except when using SQL Always-On which is not applicable in BizTalk Server environments) due to a lack of support for Windows Server Failover Clustering (WSFC). Companies that require a 24x7 environment and can't afford downtime would be best to hold off on running BizTalk Server in Azure till WSFC is enabled. It appears that Microsoft are making some developments with SMB file shares in Azure, so hopefully the lack of support for WSFC will be addressed soon.

Azure is also a great place for BizTalk development, test and performance lab environments. MSDN subscribers benefit from greatly reduced VM prices (up to 40%), and will find that SQL Server and BizTalk Server VMs will not incur any licensing costs with the proviso that they are only used for development and test purposes. MSDN subscriptions are provided with a fair amount of monthly Azure credits (the amount depends on the type of subscriptions) and developers would do well to maximize their usage of these benefits.

4 Introduction

Microsoft's Azure IAAS offering has been available since April 2013 but not much has been written on how BizTalk Server can be used to leverage Azure VMs beyond setting up development environments, nor has the viability of BizTalk Server running in the cloud really been explored. This document aims to bridge that gap by discussing various BizTalk Server topologies that could be implemented in Azure, and also

explains some of the Azure concepts that BizTalk specialists will need to be aware of in order to evaluate whether an IAAS model is suitable for their requirements.

You'll notice that this white paper is facetiously titled the "A-Y" of running BizTalk Server in Azure IAAS rather than the "A-Z". This is because the nature of the cloud is such that change is constant and this document could never be seen as 100% authoritative as things might have changed since the day it was written. It is difficult to keep up to date with cloud technologies and best efforts will be made to keep this document up to date as and when the Azure landscape changes.

Once you have finished reading this white paper you should have some level of familiarity with the below topics from a BizTalk Server in Azure perspective.

- Reasons to consider and rule out Azure
- Topologies in Azure
- Cloud economics
- Performance considerations
- Azure Virtual Networks (VNETs)
- Cloud Services
- Availability Sets and High Availability
- Update domains (UDs) and Fault domains (FDs)
- Cloud Service endpoints (including load balanced endpoints)
- Autoscaling VMs
- Site-to-Site VPNs and ExpressRoutes
- Azure Traffic Manager
- Attaching extra data disks to your VMs
- Azure File Services

Even if you don't find any reason to use your learnings from this article from a BizTalk Server perspective, the article should still serve as an introduction to some of the important concepts and features available in Azure.

Note that all steps detailed and screenshots included in this document are based on the HTML5 second generation version of the Azure Portal. There is currently a WIP third generation version of the Azure Portal (see here for more details) with a Windows 8.x look and feel which at the time of writing this document only supports a subset of the features contained within the second generation portal. Chances are that the experience using this Portal would not be too different however this is not assessed in the current version of this document. Also note that while this White Paper lists instructions based on usage of the Azure Portal, there will be equivalent methods available via REST APIs and PowerShell commandlets.

5 Should you make the move?

There are plenty of reasons to consider running BizTalk Server in Azure IAAS, however there are also some reasons to exercise caution or avoid Azure altogether based on your requirements. Below are some potential use case scenarios and corresponding recommendations on whether moving to Azure makes sense or not.

A green traffic light indicates a recommendation to explore moving to the cloud, a yellow traffic light indicates that you should explore the possibility with caution, and a red traffic light indicates that there are major gaps or roadblocks that might currently prevent you from achieving your goals in Azure. Note that multiple use cases might apply to you and you should consider the implications of each one.



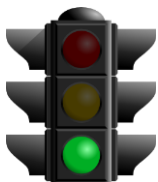
I need to build development, test and highly scaled performance environments for BizTalk Server without too much hassle.

There are many reasons to consider using Azure to host development, test, and performance lab BizTalk Server environments. First and foremost is that if you have an MSDN license you will have access to greatly discounted prices for VMs and will not have to pay for any software licenses that would normally be covered by MSDN benefits. You can also take advantage of per minute rates by switching VMs off when you're not using them so that you are only paying for what you use, and potentially even throwing them away when you're done. Perhaps most compelling is the fact that you will not have to jump through many hoops to have servers provisioned and scaled to the levels that you need. You might want to consider looking into an MSDN pay as you go subscription if your company is very committed to leveraging Azure for development and test purposes.



My company operates in a lean fashion and doesn't want to invest in expensive on premise infrastructure or expensive up front licenses.

Making use of the Azure IAAS platform allows you to save on investment in expensive on premise infrastructure and leverage relatively cheaper prices for your VMs through the economies of scale that Microsoft has built into their Azure data centres. Using Azure VM gallery images also means that you can take advantage of a pay as you go OPEX friendly pricing model for platforms such as SQL/BizTalk Server, rather than making expensive up-front CAPEX based licensing investments. Using the gallery VM images also eradicates any commitment to the BizTalk Server platform, which might be useful in scenarios where BizTalk Server is being used as a stopgap measure. It might also make sense to explore using PAAS integration offerings such as Microsoft Azure BizTalk Services (MABS), so please ensure that you explore all possibilities.



My company has already invested in SQL Server and BizTalk Server licenses and would like to leverage these in Azure to save on infrastructure costs.

As long as you have invested in Software Assurance for SQL Server and BizTalk Server you will be allowed License Mobility and will thus be allowed to use your existing licenses in the cloud. You will want to avoid the SQL Server/BizTalk Server gallery images and instead install SQL Server and BizTalk Server on vanilla Windows gallery images from their install media. This ensures that you will only be billed for the Windows license costs of the VM and not for the SQL Server/BizTalk Server licenses which you have already paid for.



My company is looking to invest in BizTalk Server Standard edition for B2B integration and doesn't necessarily require high availability.

Azure works well for BizTalk Server Standard environments, and offers multiple VM size configurations so that you can choose a scale that matches your throughput requirements. If you have idle periods throughout the day then you could potentially look at switching off your environment during those times, in which case you will not be billed during those periods except for storage.



My throughput requirements are high and my applications are very orchestration heavy, thus requiring lots of CPU, however throughput is also inconsistent with many quiet periods in the day.

BizTalk Server licensing since BizTalk 2013 is based on the number of cores available to each server on which the platform is running, with licenses available in 4 core packs. To satisfy high throughput requirements in an orchestration heavy solution on premise one would have to pay for enough licenses to satisfy peak CPU usage, even if demand is inconsistent. This applies regardless whether you were choosing to use a single highly scaled server or multiple servers.

You could save money in Azure by building a scaled out BizTalk Server Enterprise environment and leveraging technologies such as autoscaling and load balancing to only allocate extra VMs during periods of peak demand. Combining autoscaling with the pay as you go model associated with gallery images will allow you to ensure that you are only investing in what you use, thus avoiding large sunk costs.



My company doesn't have a cloud strategy as yet.

Choosing to use Azure technologies when your company doesn't yet have a view on whether and how it is going to leverage the cloud would be unwise except maybe for POCs. That said, now is the time for all companies to be assessing the cloud even if they aren't yet ready to make any definite steps. Perhaps it is worth you raising the conversation with your management and using your learnings from this White Paper to express the possible benefits to them if they aren't being explored already.



My BizTalk Server environment is mostly used to integrate with on premise applications.

It is altogether possible to connect Azure VNets to on premise networks using technologies such as Site-to-Site VPN and ExpressRoute enabling BizTalk Server environments in the cloud to connect to on premise applications. That said, if most of your integration requirements are on premise, and you don't have any plans to start migrating to the cloud then an investment in Azure might not be the best immediate fit for you.



I require a 24x7 highly available BizTalk Server environment with proven and possibly financially backed SLAs.

As of now Windows Server Failover Clustering (WSFC) is not supported in Azure except within SQL Always-On environments. This means that it not possible to cluster your SQL layer, the Enterprise Single Sign-On Master Secret, or host instances for adapters that don't support active-active deployments. This effectively

means that it is currently impossible to build a highly available BizTalk Server environment in Azure. You can cater for high-availability on the BizTalk runtime server layer (assuming you don't have clustering requirements for host instances), but the fact that some other layers are unprotected from outages compromises your entire environment.

You could potentially build multiple BizTalk environments, potentially within different regions, and use Azure Traffic Manager to direct traffic to currently active environments, however this would most likely prove to be cost-ineffective, is overly complicated if all you're after is high availability, and you still wouldn't qualify for an SLA from Microsoft even though you've built in redundancy. Hopefully this is an area that Microsoft will address soon.



I want to migrate my BizTalk Server 2010 environment (or older) into Azure, and am not yet prepared to upgrade to a more recent version.

According to [this article](#), only BizTalk Server 2013 and later versions are supported in Azure. You will not find gallery images for earlier version of BizTalk Server. While you could use a custom image of an earlier version of BizTalk Server in Azure it will not be supported, and thus it is recommended that you hold off until you are ready to upgrade to a recent version.

6 BizTalk Server Topologies in Azure

There are many different BizTalk Server topologies that can be implemented in Azure, some mirroring traditional on premise deployments, and some taking full advantage of Azure technologies to cater for scenarios that aren't quite possible on premises.

6.1 Self-Contained Development Topology

The most traditional scenario for running BizTalk Server in Azure is for development or testing purposes. In these scenarios it might be viable to run a single self-contained VM instance which will be used for BizTalk databases, the BizTalk runtime environment, as well as development tools such as Visual Studio. The effort involved in creating a development BizTalk Server VM is miniscule as the Azure gallery contains a BizTalk Server 2013 as well as a BizTalk Server 2013 R2 developer edition image that already has the relevant version of BizTalk Server and corresponding versions of SQL Server and Visual Studio installed. After a 10-20 minute wait while Azure provisions the VM it is ready to use. Note that you will still have to run through the BizTalk Server Configuration tool to configure your environment before the BizTalk databases and runtime will be available. This topology could be expressed by the below diagram.

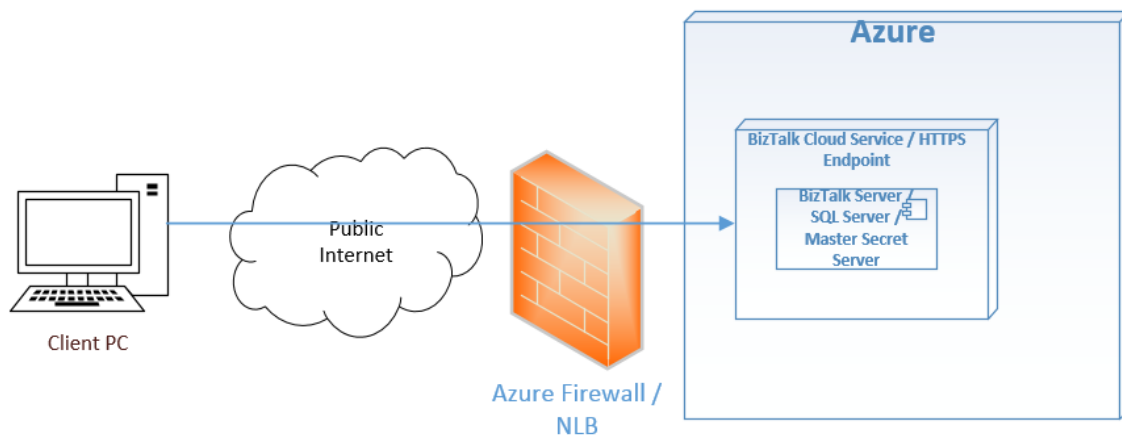


Figure 1 – Self-contained BizTalk Server Topology in Azure IAAS

6.2 Multi-Server Topology

The remainder of this document will mostly focus on more complicated topologies which involve more than just a single-instance VM which is very easy to provision, and step by step guides to setting up these more complex topologies are provided in [section 8](#) of this document.

For topologies which involve multiple VMs interacting with each other you will need to create a VNet (Virtual Network) to allow for private networking between your VMs. You'll want to consider creating two subnets with set IP address ranges in the VNet, one for the DC/DNS VM(s), and one for the rest of your VMs which guarantees that your DC/DNS VM(s) will always get the same IP address. Alternatively you could reserve a static IP address for your DC/DNS VM(s) to ensure that the IP address will never be released. You'd also want to register your DNS server's address against the VNet. The steps to create a VNet are detailed in [section 8.1](#) of this document.

Windows Server Failover Clustering (WSFC) does not appear to be officially supported in Azure (as per [this article](#)) except when used for SQL AlwaysOn. This means that we aren't supported in clustering SQL Server/MSDTC, Enterprise Single Sign-on (master secret), or BizTalk host instances which host receive locations that use adapters such as FTP, MSMQ or POP3. Hopefully this will change in the future but for now this is a limiting factor that basically means that you will not be able to build a high availability BizTalk environment in Azure. Theoretically it is still possible to create a cluster using WSFC even though it is not supported (see [appendix 9.2](#) for more info).

Assuming you haven't decided to walk down the untrodden path and tried to implement WSFC anyways, your SQL VM is going to be a weak link in your environment, and thus it makes sense to also make this VM the master Enterprise Single Sign-On server since if either were unavailable that would cause problems for your environment, so you might as well keep your single points of failure together. If you are implementing just a single BizTalk runtime server it might still make sense to keep the SSO master secret on the SQL Server VM since that makes adding more runtime servers to the group easier in the future. Note that installing the Enterprise Single Sign-On service on the SQL Server will not incur any additional licensing costs since this service is not considered to be a runtime component.

In contrast to the above, Azure offers many more possibilities for BizTalk runtime servers in terms of high availability and scaling out, though do note that the lack of high availability on the SQL layer effectively compromises high availability for the BizTalk environment. It makes good sense to take advantage of Cloud Services and group multiple BizTalk runtime VMs (all joined to the same BizTalk group) under a single Cloud Service. You would want to choose a relatively less powerful VM size since it makes more economic sense to scale out in Azure rather than scale up in most cases. The A2 (2 Core, 3.5 GB RAM) or A3 (4 Core, 7 GB RAM) sizes would be good starting points to evaluate and you might even want to consider the memory intensive A5 (2 Core, 14GB RAM) and A6 (4 Core, 28GB RAM) sizes if your BizTalk applications are memory intensive. The relationship between Cloud Services and VMs is explored in detail in [section 8.2](#) of this document.

All your BizTalk VMs within a Cloud Service should be placed in an Availability Set to ensure that you are protected from maintenance or unplanned outages caused by hardware failures. If WSFC is supported in the future then you would want to have you clustered SQL VMs in the same Cloud Service within an Availability Set as well. Having more than one running VM in your Availability Set provides you with a 99.95% uptime SLA in Azure, and you'd want to protect each layer of your solution within its own availability set.

Each of your BizTalk VMs should also have relevant ports that you want publicly available exposed through a Cloud Service endpoint (potentially load balanced with an appropriate probe), optionally with ACLs limiting access to the endpoint to relevant parties. High availability for your BizTalk runtime servers is detailed further in [section 8.5](#) of this document, while creating Cloud Service endpoints is detailed in [section 8.7](#) of this document.

You can now provision as many BizTalk Servers as you think your scale might demand, even if you don't think that you will consistently need that scale. By taking advantage of autoscaling you can ensure that you only pay for the scale that you currently need as VMs that aren't currently required would be deallocated and thus you would only be paying for the storage on these VMs. You can even setup schedules whereby all of your BizTalk VMs get shut down at night time or other periods of down time if they aren't required then. Autoscaling is explored further in [section 8.6](#) of this document.

The aforementioned topology can be expressed by the below deployment diagram.

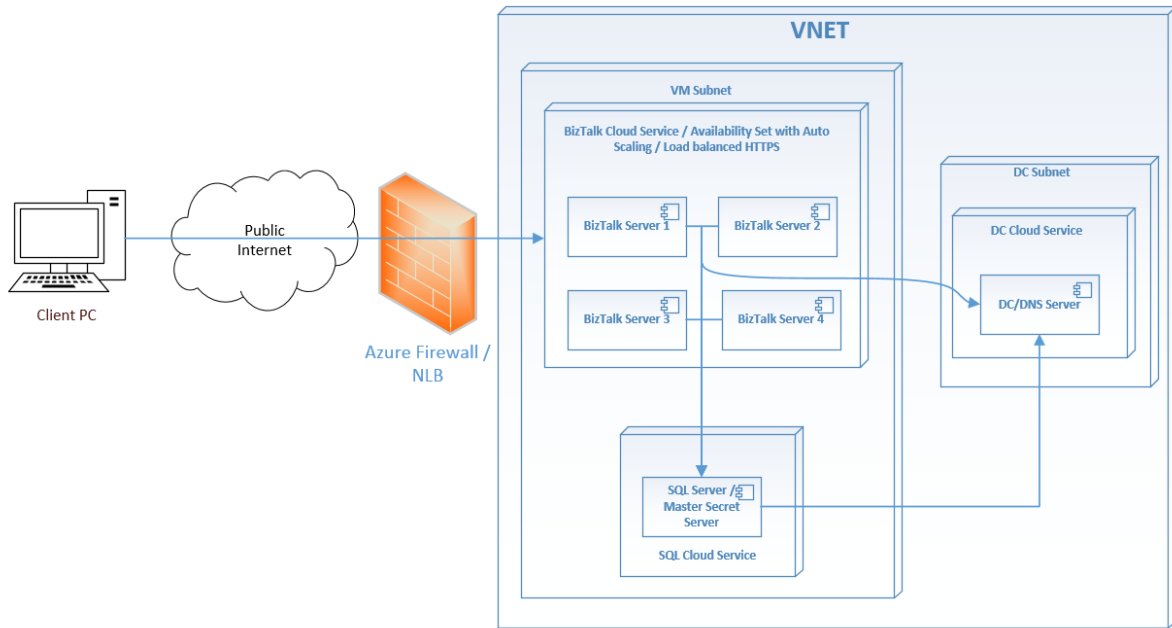


Figure 2 – Multi VM BizTalk Server Topology in Azure IAAS

6.3 On-Premise Connectivity based Multi-Server Topology

Another scenario you might face is where you need to have BizTalk Server in the cloud connecting to your on premises systems using non-Internet friendly adapters such as the FILE adapter or the DB2 adapter, or if you want to have your VNet in the cloud be an extension of your on premises network such that you are leveraging internal assets such as your internal domain controllers and DNS Servers (you could also consider having a DC on premises and in Azure with AD replication between them, see [this article](#) for more information). In order to meet such a requirement you would need to build a VPN tunnel between your VNet and your internal network, and even in this you have multiple choices available.

You can either choose to implement a Site-to-Site VPN connection (see [here](#) for more details) which most IT Pros should be familiar enough with, or if you require higher bandwidth and security then you should consider using ExpressRoute (see [here](#) for more details). The primary difference with ExpressRoute is that connections do not traverse the public internet, will be more secure and will have higher speeds and lower latency. The downside of this is that it takes more work to implement an ExpressRoute and will also involve some effort from your ISP, which will most likely rule out this option for most non-production deployments.

Such a topology can be expressed by the below deployment diagram (note that AD replication connections are not displayed for the purpose of brevity).

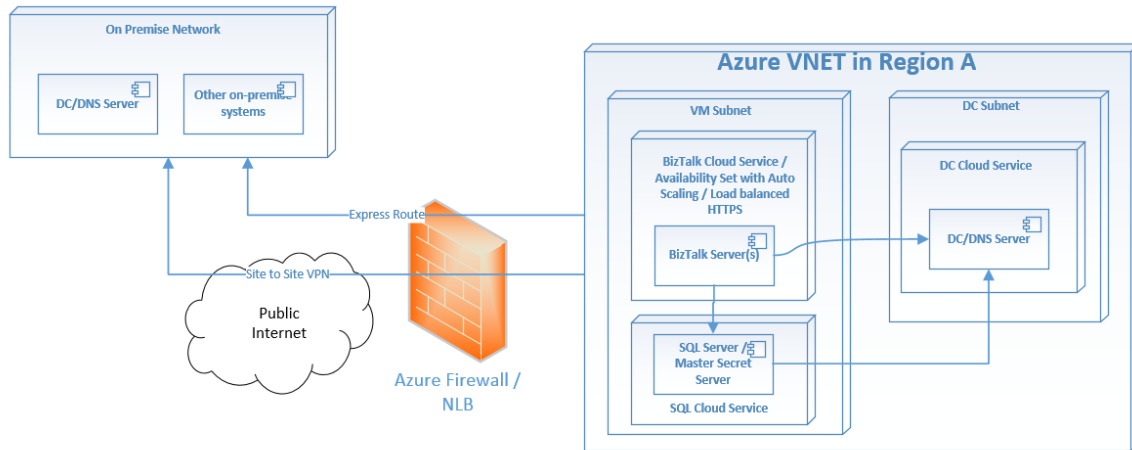


Figure 3 – Extending your on premises network to Azure

6.4 Multi-Region Multi-Server Topology

Another option that might interest companies that have stringent high-availability requirements, want to provide low latency services across multiple regions, or are just looking for a way to ease the move from their premises to Azure, is to take advantage of the Azure Traffic Manager. The Azure Traffic Manager enables you to expose a single endpoint for a given service that has the capability to distribute traffic to one of multiple target locations. You can read more about Azure Traffic Manager [here](#).

Traffic Manager allows traffic to be distributed to the target endpoints, potentially across multiple regions and potentially a mixture of Azure and non-Azure endpoints, based on multiple algorithms including round robin, failover, and performance based. Regardless which method you choose, Azure will probe the target endpoints to ensure they are healthy and will always remove unhealthy endpoints from the list of candidate endpoints. You can read more about the various load balancing methods in [this article](#), and they are also summarized below.

The round robin load balancing method allows you to distribute traffic across multiple endpoints and even allows you to assign higher priorities to certain endpoints. This load balancing method could be very useful if you want to direct a percentage of traffic to an alternate endpoint, perhaps because you are migrating from on premise to Azure and only want a certain percentage of traffic to be service in Azure.

The failover load balancing method implements an active-passive model whereby you must assign a primary target endpoint and one or many backup endpoints for high availability purposes. This load balancing method would be useful for global organizations that require high availability even across multiple geographic regions or that are looking to create their DR environment in Azure.

The performance based load balancing method directs traffic to the target that it believes will provide the lowest latency response to the client based on their geographic location. Note that this calculation for latency is not real time but is updated at regular intervals, and is not based on the amount of load that each target endpoint is currently servicing.

Such a topology can be expressed by the below deployment diagram (note that AD replication connections are not displayed for the purpose of brevity).

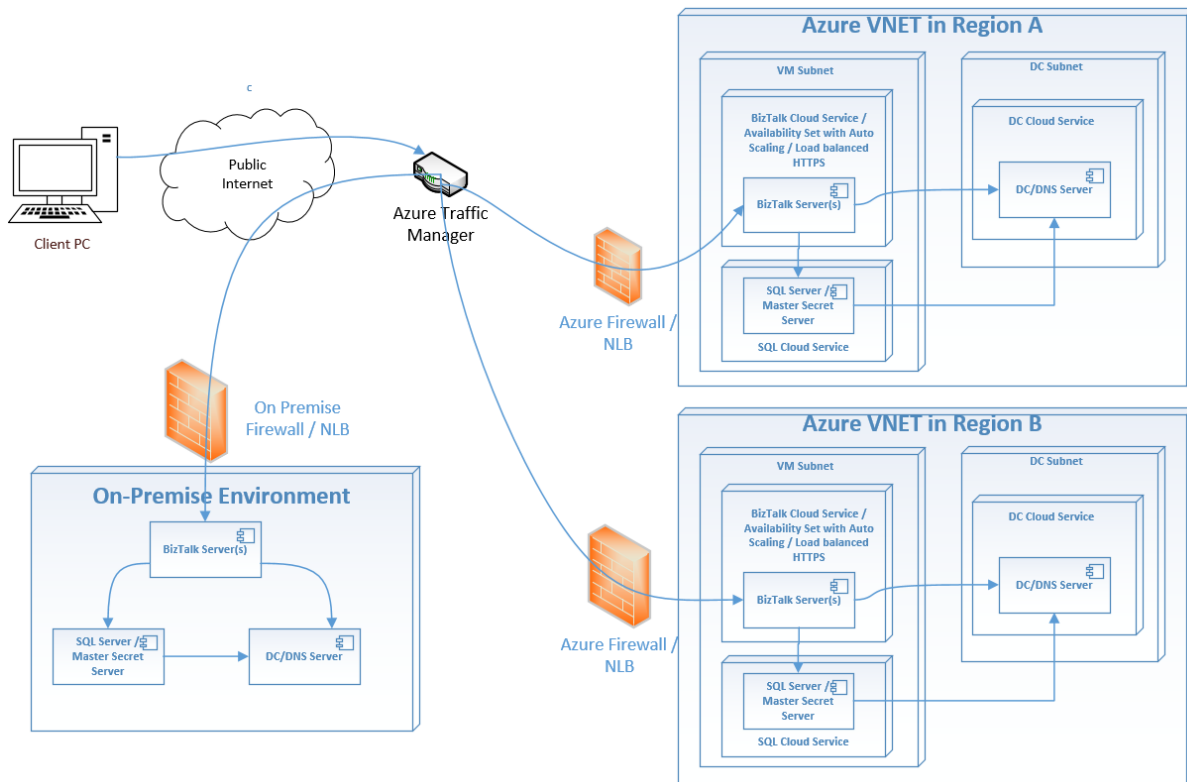


Figure 4 – Traffic Manager based Topology

7 Economics of running BizTalk Server in Azure

Running BizTalk Server in Azure is highly beneficial for MSDN subscribers who want to use Azure to spin up development or test environments. At the time this article was written MSDN subscribers get discounted rates on Azure VMs (up to 40% according to [this article](#)) and will not be charged for the usage of any MSDN software. This means that if an MSDN subscriber chooses to provision a VM based off a BizTalk Server image then they will only be charged the discounted VM rates and not for BizTalk Server. The same applies for SQL Server of course making running BizTalk Server development or test environments in Azure very cost effective for MSDN subscribers.

Creating a BizTalk environment against your individual MSDN subscription might be adequate for POC or training exercises, however most likely would not be an acceptable approach for full-blown test or other collaborative environments. For these scenarios it would pay to look into creating an [MSDN Dev and Test Pay-As-You-Go](#) subscription instead. The benefit of such a subscription is that it can be shared across multiple Azure accounts allowing for easy collaboration, whilst still enjoying the discounted MSDN rates and other associated benefits. Departments within the business or projects with a stake in the BizTalk environment can then pay for these subscriptions rather than having the account associated with an individual MSDN subscriber.

Do note that any Azure components associated with an MSDN based subscriptions must be for development or test purposes only. Financially backed SLAs will not apply for any MSDN based subscriptions.

Organizations that have a serious commitment to Azure technologies might find that an Azure Enterprise Agreement (EA) might be best suited to their needs. Azure EAs offer organizations that are willing to make upfront financial commitments to Azure many benefits in the way of discounts, annual payments, and a separate Enterprise Azure Portal that makes it easier to manage the multiple subscriptions which an enterprise would need. You can read more about Azure EAs [here](#).

When you spin up a VM in Azure based on a gallery image you pay per minute rates based on usage of the VM with no additional licensing implications, this being applicable to your SQL and BizTalk servers. However if you spin up a vanilla Windows Server VM from the Azure gallery on which you intend to install SQL/BizTalk Server, or upload your own VHD to create a custom VM then you will still be required to purchase licenses for any software that you implement on those VMs, have License Mobility through Software Assurance (read more on this [here](#)) which allows you to transfer your on premise licenses to Azure, or have an applicable MSDN subscription that covers the usage of said software.

Note that this pay as you go model requires no major up-front commitment and thus also gives you benefits akin to Software Assurance (SA), as upgrading to the newest version of a platform would not require any major investment beyond provisioning new VMs based on the gallery image for the latest platform.

Also note that when you are using gallery images you will not be licensed to perform in place upgrades of platforms that are part of the pay as you go license fee, since the license is for that specific version of the product. You would instead need to spin up new VMs, either with the updated gallery image or your own licensed software, and migrate your databases/applications to it. This might be an important consideration for you if your organization is an early adopter and always wants to be running the latest versions of the SQL/BizTalk Server platforms as new gallery images might not be released for a while after the new platforms are made generally available. As an example BizTalk Server 2013 R2 was announced as Generally Available (GA) at the end of June 2014 but as of early October 2014 there is only a BizTalk 2013 R2 Developer Edition gallery image available on Azure.

When you are running VMs in Azure you are getting the benefits of not having to provision your own hardware on site (this includes storage, compute, network etc...), you reduce your maintenance costs (this includes host server maintenance/patching, cooling, power etc...), and you can move towards more of an OPEX (operational expenditure) model rather than a CAPEX (capital expenditure) model. If you are running VMs based on gallery images (in which case the BizTalk and SQL platform costs are accounted for in your Azure subscription fee) rather than custom images then you gain the additional benefit of treating the license costs of your platforms as OPEX as well. There are a lot of reasons why a company might prefer it's IT spend to be treated as OPEX rather than CAPEX which makes Cloud models very attractive, the majority of these being around tax and bookkeeping implications; you can read a bit more about this [here](#).

You should consider using the gallery images of SQL Server and BizTalk Server if the below points apply to you.

- You don't have an existing SQL/BizTalk Server license.
- You have SQL/BizTalk Server licenses but you don't have or want to invest in License Mobility through Software Assurance.

- You want to avoid having to make any up-front large investments towards SQL/BizTalk Server licenses.
- You prefer for SQL/BizTalk Server licensing costs to be treated as OPEX rather than CAPEX.
- You don't necessarily want to make a long term commitment to the BizTalk Server platform and are looking for a stop-gap solution till you migrate to another platform such as Microsoft Azure BizTalk Services (MABS).
- On the flip side you should consider making use of your existing SQL/BizTalk Servers licenses if the below points apply to you.
- You have SQL/BizTalk Server licenses and are applicable for License Mobility through Software Assurance.
- You prefer to perform in-place upgrades on your SQL/BizTalk Server platforms when new platform versions are made available rather than spinning up new environments.
- You prefer to be an early adopter for new platform versions of SQL/BizTalk Server and don't want to wait for gallery images to be made available.

If your requirements are not 24/7, or your throughput is bursty then the story gets even better. Functionality such as autoscaling can be used to automatically add VMs to your load balanced BizTalk group during periods of peak demand, and tear down some/all the servers during idle periods. This would mean that you only pay for the times when your servers are active, and not for when they are inactive. Keep in mind however that a VM must be in a deallocated state, rather than just stopped, in order for the billing clock to stop ticking. Shutting down a VM from inside the guest operating system does not deallocate the VM; you must do this from the Azure Management Portal, Portal APIs, PowerShell, or through autoscaling.

Note that regardless whether your machine is allocated or deallocated you will still be paying for storage used by the VMs. Another additional cost to consider is that for network bandwidth. You'll find that Microsoft doesn't charge you for inbound traffic, but rather for outbound traffic only, which can be highly beneficial for BizTalk exchanges that have a higher ratio of message receipt to message delivery.

Another key piece of advice gleaned from the CAT team is that you should aim to create Azure deployments with the minimum possible specs per instance from a CPU perspective that make economic sense, and take advantage of additional instances with autoscaling to cater for peaks in demand. Note that from a BizTalk Server perspective the CAT team's advice would need a bit more context. First of all their advice is not at all applicable for a BizTalk Server Standard environment in which scaling out is not an option thus scaling up is the only way to cater for higher demand. Their advice makes more sense in a BizTalk Server Enterprise environment in which you could save money by scaling out with autoscaling rather than having higher specced instances. Also important is to ensure that the specs on the initial set of VMs is appropriate given that there will be a lead time for autoscaling to scale out your environment as required.

Based on the Azure VM pricelist as of October 2014, the price of a BizTalk Server VM from a BizTalk licensing perspective (assuming they aren't covered by an MSDN benefit in which case the cost would be nil) is the same for A0-A3 as well as A5-A6 instances, which is in line with on premises licensing costs where the minimum pricing unit is based on 4 cores. The same also applies to SQL Server VMs which follow a similar

core based licensing model to BizTalk Server. Thus the only price variable for these VM sizes is the Windows Server license. The Windows Server license scales up in price quite linearly based on number of cores for General Purpose VM sizes A0-A4, and the memory intensive A5-A7 VMs are a bit less than double the price of their General Purpose equivalents based on number of cores. For VMs with more than 4 cores, the BizTalk and SQL Server licenses scale up linearly as well.

The below table gives you an idea of approximate monthly costs of running a BizTalk Server environment based on the October 2014 pricelist in US Dollars. All of the below assume that gallery images are used for both SQL and BizTalk Server, and that the standard edition is used for SQL Server in all cases. This table only shows VM price approximations and excludes additional costs such as storage and networking. This table also assumes that you aren't taking advantage of MSDN or other discounts, and that you are using SQL/BizTalk Server gallery images except where explicitly stated. Please don't use this table as an absolute price list, its purpose is to make clear the Azure pricing model by showing price comparisons across various environment configurations.

Configuration	Monthly Cost
A2 BizTalk Standard VM / A3 SQL VM	\$ 1,058.00
A3 BizTalk Standard VM / A3 SQL VM	\$ 1,192.00
A6 BizTalk Standard VM / A6 SQL VM	\$ 1,638.00
A6 BizTalk Standard VM / A6 SQL VM (custom image leveraging existing BizTalk/SQL licenses)	\$ 982.00
A6 BizTalk Standard VM / A6 SQL VM (dev VM with MSDN subscription)	\$ 589.20
A9 BizTalk Standard VM / A6 SQL VM	\$ 5,864.00
A9 BizTalk Standard VM / A6 SQL VM (custom image leveraging existing BizTalk/SQL licenses)	\$ 4,137.00
A3 BizTalk Standard VM / A3 SQL VM / environment runs for 8 hours a day	\$ 397.33
A6 BizTalk Standard VM / A6 SQL VM / environment runs for 8 hours a day	\$ 546.00
A9 BizTalk Standard VM / A6 SQL VM / environment runs for 8 hours a day	\$ 1,954.67
A2 BizTalk Enterprise VM / A3 SQL VM	\$ 2,263.00
A3 BizTalk Enterprise VM / A3 SQL VM	\$ 2,397.00
A6 BizTalk Enterprise VM / A6 SQL VM	\$ 2,843.00

Configuration	Monthly Cost
A4 BizTalk Enterprise VM / A6 SQL VM	\$ 4,450.00
2x A3 BizTalk Enterprise VM / A6 SQL VM	\$ 4,451.00
2x A3 BizTalk Enterprise VM with 2nd VM used 25% of the time / A3 SQL VM	\$ 3,312.50
2x A6 BizTalk Enterprise VM with 2nd VM used 25% of the time / A6 SQL VM	\$ 3,870.00
4x A3 BizTalk Enterprise VM / A3 SQL VM	\$ 7,890.00
4x A6 BizTalk Enterprise VM / A6 SQL VM	\$ 9,005.00
4x A3 BizTalk Enterprise VM with 2nd VM used 75% of the time, 3rd VM 50% of the time, and 4th VM 25% of the time / A3 SQL VM	\$ 5,143.50
4x A6 BizTalk Enterprise VM with 2nd VM used 75% of the time, 3rd VM 50% of the time, and 4th VM 25% of the time / A6 SQL VM	\$ 5,924.00

Figure 5 – Azure BizTalk Environment Approximate VM Prices

You'll notice from the table above that if you have requirements which demand 16 cores during peak hours but the demand is not consistent, that it might make more economic sense to implement a BizTalk Server Enterprise environment with autoscaling than a high spec BizTalk Server Standard environment. On the flip side if your peak demand requires 4 cores at the maximum then it might make more sense from the price perspective to stick to a single BizTalk Server Standard runtime server from a price perspective.

You'll also notice that the cost of switching from a General Purpose to a Memory Intensive VM size is relatively marginal, thus will be well worth the extra money rather than scaling out from a RAM perspective. Take a look at the [Azure Pricing calculator](#) for more details, or if you want a breakdown of the costs of the Windows Server vs. the BizTalk component of pricing then take a look at the [Azure Pricing Details](#).

If you have implemented the CAT team's advice and taken advantage of autoscaling in a BizTalk Server Enterprise environment then you would have an environment that caters for your peak demands yet you are only paying for what you use. This is in contrast to a traditional BizTalk Server Enterprise environment on premise in which the instant you need scale beyond 4 cores, regardless whether you scale up or scale out you are going to incur the full cost of another 4 core license.

Also keep in mind that there are a lot of hidden costs to implementing a BizTalk Server environment from a developer's perspective such as hardware provisioning, storage, virtualization, maintenance, DR etc. Some of these costs could be mitigated by hosting your BizTalk Server environment in the cloud. There are many infographics such as the below floating around various social media sites comparing hidden costs of projects on premise vs. in the cloud. This document won't speak to the numbers or the scale of the difference between the two (something which can only be truly

evaluated with time) but hopefully it expresses to you what the promise of the cloud is.

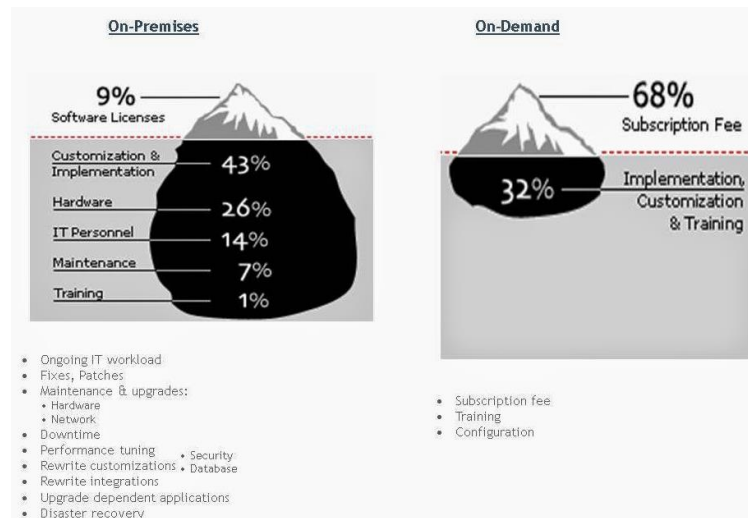


Figure 6 – Hidden costs of On-Premises vs. Cloud Implementations

8 BizTalk Server Performance in Azure

Tweaking your BizTalk Server environment to get the maximum performance is definitely more art than science, mainly due to fact that each BizTalk Server environment would have many variants when it comes to underlying infrastructure (this would of course not be the case in Azure which has a limited set of infrastructure configurations so perhaps this will become more of a science with time). Luckily it is an art that has been pretty well explored and documented such as in this [optimization guide](#). The majority of the documented optimizations still apply, but there are some additional concepts that need to be kept in mind when creating a BizTalk Server environment in Azure that is being used for anything more than development or testing purposes.

The first and most obvious performance consideration when creating a BizTalk Server environment in Azure is which region you should create your environment within. You will find that based on where you reside you will experience different levels of latency when interacting with Azure components across various regions. Choosing an appropriate region will allow you to RDP into your BizTalk VMs more responsively, and will mean that external clients that interact with your BizTalk environment will experience less latency as well.

Microsoft's cloud business model relies heavily on economies of scale and consistency. It logically follows that the hardware that they employ must be relatively low cost, and not necessarily the highest spec except in specific scenarios which demand high spec hardware (such as for the largest VM sizes). Articles such as [this one](#) discuss the types of CPUs that Azure employs and provides some benchmark reports that show the relatively poor performance of CPUs on most VM sizes in Azure compared to the common CPUs that would be chosen for on premise deployments. Thus you will find that you will not get similar CPU performance out of a standard A3 BizTalk Server VM on Azure compared to a typical 4 Core/8GB Ram setup on premise using a traditional deployment model. Keep this in mind when you size your BizTalk

runtime servers and do consider scaling out with autoscaling enabled rather than only following the more traditional scaling up model.

Another thing to consider is the performance of your SQL Server box especially when it comes to IOPS (Input/output Operations per Second) which is a major performance constraint if you leave the SQL box in its default deployment state. By default when you spin up a SQL box from the gallery image it will provision all databases on the local C: volume. This is not a good practice since the C: volume on Azure VMs is specifically geared towards booting up the operating system quickly.

You would also typically want to avoid using the D: volume on Azure VMs because it will get flushed anytime the server gets shutdown as it is non-persistent and performance is not guaranteed. An exception to this guideline is for the D-Series VMs (released in September 2014) which use SSDs for their D: drives, delivering up to 7000 IOPS which is vastly superior to the 500 IOPS delivered by non-SSD disks. Just like on other Azure VMs, the D: volume on the D-Series VMs is also non-persistent so there are very limited use cases in which it might make sense to use this VM configuration in a BizTalk Server topology. Hopefully Microsoft will soon make SSD disks available for use as VM data disks which would open up a realm of possibilities when trying to implement high performance environments.

A potential use case for D-Series VMs in a BizTalk Server topology is when running BizTalk Server 2013 R2 in Azure with SQL Server 2014 hosting the BizTalk databases, with the TempDB on the SSD based D: volume. This only works for SQL Server 2014 since it will rebuild the TempDB upon server restarts, and thus enables you to take advantage of the increased IOPS delivered by the SSDs. You can read more about the D-Series VMs and benefits for SQL Server 2014 [here](#).

You'll want to ensure that you cater for the required IOPS on volumes on which you'll place your SQL data files, and you will definitely want to move the data files to newly provisioned disks other than the C: and D: volumes (with the possible exception of the TempDB in SQL Server 2014 on D-Series VMs as previously discussed). You may need to consider striping disks to get the required level of IOPS on each volume, keeping in mind that each Azure Disk provides a maximum of 500 IOPS (300 IOPS on basic VM sizes). The choice of the number of data disks you choose to associate with your VM will also drive your SQL VM size as each VM size has a maximum number of data disks. Attaching additional data disks and striping them is discussed in [section 8.4](#) of this document.

You might want to set the path to the temporary directory (%TEMP%) for the BizTalk host instance service accounts onto one of your newly created higher performing volumes. This could potentially be helpful in cases in which you are processing large messages through pipelines and maps which might rely on file persistence in temporary folders rather than on memory. This might even be a use case for the D-Series VMs but chances are they will be overkill for this requirement.

Keep in mind when it comes to disk performance that if you created your Azure storage account prior to June 7th 2012 then that means your disks will be based on 1st generation Azure hardware, and thus won't perform as well as newer hardware. If this applies to you then do consider creating a new storage account rather than using your existing one.

Another well-known performance issue that you will want to know about is that Azure disks that have not been accessed for a while will undergo what is called the "warm-up" effect (see [here](#) for more details). This might be a good reason to keep your SQL

Server VM running at all times. If you shut down your BizTalk and SQL Servers during periods of inactivity, then consider starting up your SQL Server at least 30 minutes prior to starting up your BizTalk Servers.

SQL server performance is critical to a well performing BizTalk deployment. No amount of autoscaling of your BizTalk Server runtime servers will help you increase your throughput if the SQL database is your bottleneck. You must consider that BizTalk runtime resource is variable given autoscaling while SQL resource is static (you can always scale it up further but not without shutting the VM down), making provisioning the SQL layer appropriately very important. There are a lot of important performance considerations to be heeded for SQL Server and the guidelines provided in [this article](#) and [this white paper](#) are helpful for tuning your SQL VMs for a BizTalk based topology. Autoscaling is discussed further in [section 8.6](#) of this document.

9 Step by Step Guide to Setting up BizTalk Server in Azure

9.1 VNets – the backbone of your BizTalk IAAS topology

As previously discussed, for any BizTalk Server topologies that include more than one Azure VM you will need to create a VNet so that the VMs can communicate with each other privately. The typical BizTalk Server topology would at the very least require a separate Domain Controller VM (typically two at the minimum for a production environment, except in the scenario where you are extending an on premise DC), a SQL Server VM, and a BizTalk Server VM (if not multiple). Since we would not want for them to be communicating with each other over the public internet (this would not be viable) we would instead want to create an internal network within Azure so that these servers could communicate with each other using private IP addresses. Creating a VNet is the way to go about doing this.

Prior to around May 2014 there was only one kind of VNet which is an Affinity Group based virtual network. An Affinity Group is a logical grouping of cloud components which are ensured to be in close proximity to each other (not just within the same geographic region but the actual server racks would be physically close to each other as well) in order to keep latency to a minimum. In order to create a VNet prior to May 2014 you would first have to create an Affinity Group and then when creating the VNet you would choose to associate it with an Affinity Group such that any VMs which were then associated to the VNet would also be associated with the Affinity Group, thus being in close proximity to each other.

In May 2014 a new concept called [Regional VNets was announced](#) which allowed you to create VNets which would span an entire region rather than being limited to a specific area within a given data centre. On the face of it this actually sounds like a step backwards since it means that your cloud components are no longer guaranteed to be in close proximity to each other but rather could be anywhere within a given region. However this move enabled a whole range of new features including mixing the higher spec A8 and A9 size VMs with lower spec VMs in a single VNet, reserved IP addresses, and internal load balancing to name a few. Affinity Group based VNets are being phased out so this document will focus on Regional VNets only.

It is no longer possible to create an Affinity Group based VM through the Azure Portal directly; all VNets created through the portal default to being Regional VNets. Creating a VNet is as simple as clicking on the New button in the Azure Portal, choosing Network Service -> Virtual Network -> Custom Create (quick create is also

an option which provides less flexibility during the creation process however you can always configure the VNet further after it is provisioned).

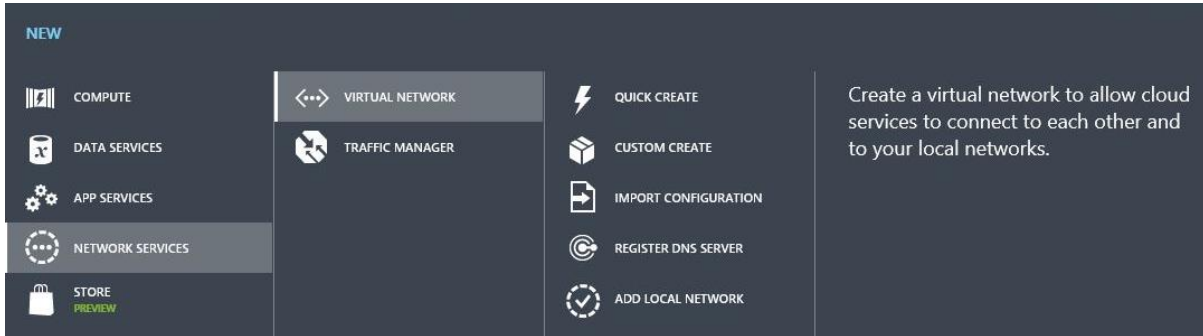


Figure 7 – Create a VNet

While stepping through the Wizard steps you will need to choose an appropriate name for your VNet, associate it with an Azure subscription and region (note that all VMs that will be part of this VNet will also be associated with this region), and then setup your VNet address spaces. In the past it was very important to setup a separate address space for your Domain Controller/DNS servers vs. the rest of your VMs, as there was no way to guarantee that your DC would retain the same IP address every time it restarted since there was no such thing as a reserved IP address in Azure. The below screenshot illustrates how I have setup separate address spaces as discussed above.

ADDRESS SPACE	STARTING IP	CIDR (ADDRESS COUNT)	USABLE ADDRESS RANGE
10.0.0.0/8	10.0.0.0	/8 (16777...	10.0.0.4 - 10.255.255.254
SUBNETS			
DCSubnet	10.0.0.0	/16 (65531)	10.0.0.4 - 10.0.255.254
VMSubnet	10.1.0.0	/16 (65531)	10.1.0.4 - 10.1.255.254
add subnet			
add address space			

Figure 8 – VNets and Subnets

However this has recently changed and it is now possible to provision VMs with a reserved IP (read more about this is [section 8.3](#)), removing the need for a separate address space for your DCs. This is certainly a more elegant approach.

Setting up a VNet using PowerShell is easy enough and is documented very well in this guide (note that the config file used in this guide is for affinity-group based VNets but you can easily switch this over to a Regional VNet by replacing the AffinityGroup attribute in the XML config file with a Location attribute containing your region name).

9.2 So what does a Cloud Service have to do with VMs?

Spinning up a VM via the Azure Portal is a simple task, but something that those unfamiliar with Azure might not appreciate is why the provisioning process requires your VM to be associated with a Cloud Service. Put simply, a Cloud Service is a

collection of cloud instances (VMs or worker roles) that are hidden behind a single public IP address.

A very important consideration when creating your VMs is whether they will be placed within a single or separate Cloud Services. You could consider to at the very least create a separate Cloud Service for your DC/DNS server (potentially creating multiple DC/DNS servers for high availability), for your SQL Server, and for your BizTalk Server(s). The reason for this is that you can then expose an RDP endpoint (more on this below) to one of the VMs contained in each of these Cloud Services on the default port 3389 which is handy since lots of corporate firewalls will block outbound RDP connections on non-standard port numbers.

A common mistake that one can make is to provide the same name for a Cloud Service and its contained VM without really giving it much thought. It is best to name a Cloud Service such that it conveys the role of the VMs contained within, especially considering that a Cloud Service is not limited to a single VM. For example you might decide to spin up a BizTalk VM called JC-BT2013VMA within a Cloud Service named JC-BizTalk2013, which caters for you to add more VMs to the same service in the future without any confusion. Note that your Cloud Service name must be globally unique while you are free to name your VMs as you see fit.

Creating a Cloud Service with PowerShell is well document in [this article](#) (specifically in the createAzureService function).

One final note is that if you want to guarantee that Azure will assign a sticky public IP address to your Cloud Service then you will need to employ the Reserved IP Address feature as detailed in [this article](#). Do note that there is an additional fee if you want to use this service as detailed in [this article](#). Also take note that this Reserved IP Address corresponds to a Cloud Service rather than a single VM within the Cloud Service, and thus any endpoints exposed by this Cloud Service (see [section 8.7](#) for more about this) would use this IP address. This feature could be especially handy if you want to guarantee that your outbound traffic always originates from the same IP address, enabling the targets of your traffic to apply IP address based ACLs.

9.3 Creating the VMs

Assuming you are setting up a topology which includes more than a single VM you will want to create a VM/Cloud Service for your domain controller (you might want to consider multiple DC VMs to cater for high availability), a VM/Cloud Service for your SQL Server, and at least one VM/Cloud Service for your BizTalk runtime servers.

The DC VM can be created by clicking on the new button in the Azure Portal, selecting Compute -> Virtual Machine -> From Gallery, and then selecting the appropriate Windows Server image. You will want to ensure that you select your Virtual Network rather than a region or an Affinity Group, and that you place this VM in the appropriate DC subnet to guarantee its IP address as previously discussed. You could alternatively make use of the Static Internal IP Address (DIP) feature to reserve an IP address for the DC VM if the containing VNet is a regional VNet as per [this article](#). You will need to choose an appropriate size for the VM as well, keeping in mind that different sizes incur different costs (the [Azure Pricing Calculator](#) is a good resource to calculate predicted costs). You would then have to setup the DC / DNS services on this VM.

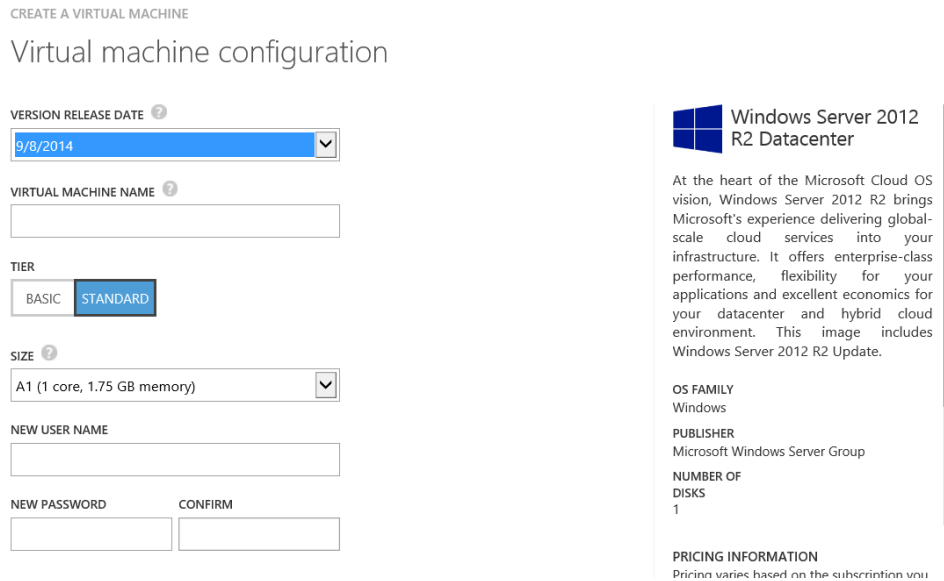


Figure 9 – Provisioning an Azure VM from the gallery

Likewise the SQL VM can be created by clicking on the new button in the Azure Portal, selecting Compute -> Virtual Machine -> From Gallery and then selecting the appropriate edition of SQL 2012 (you could choose SQL 2014 if you're dealing with BizTalk Server 2013 R2), keeping in mind that Enterprise edition is more expensive than Standard edition, and aside from real-time aggregated (RTA) BAM views has little else extra to offer for BizTalk Server in Azure. Once again you will need to choose an appropriate size for the VM, keeping in mind that the size of the VM also determines how many additional disks you can attach to the VM. You will want to ensure that you select your Virtual Network rather than a region or an Affinity Group, and that you place this VM in the appropriate subnet.

Finally you will want to provision your BizTalk VMs by clicking on the new button in the Azure Portal, selecting Compute -> Virtual Machine -> From Gallery and then selecting the appropriate edition of BizTalk Server. If you want to take advantage of high-availability, autoscaling, and load-balancing then you need to choose the Enterprise edition. Once again you will need to choose an appropriate size for the VMs. You will want to ensure that you select your Virtual Network rather than a region or an Affinity Group, and that you place the VMs in the appropriate subnet. You will need to configure BizTalk on each of the VMs once they have been provisioned, or alternately you can use the BizTalk Provisioning Tool to automate the experience as detailed [here](#).

Something else to keep in mind when sizing your VMs is that by default Azure only allows you to provision VMs up to a total of 20 cores. If you need more cores available then you will need to raise a billing support incident with Microsoft and advise them what the new threshold should be. Microsoft appear to be very responsive to such requests and past experience shows that you might have your threshold raised within hours.

The BizTalk community has already sprung to action and documented ways to automate the provisioning of a BizTalk environment in Azure using PowerShell as detailed in [this article](#).

Keep in mind that at any stage if you find you have made a mistake when creating a VM (for example you have assigned it to the wrong Cloud Service) that you can always delete the VM without deleting the actual data disk, and then recreate it. To do this you would stop the VM, select it in the Azure Portal and click the delete button,

and select the “Keep the attached disks option” which will delete the VM without deleting the disk. You can then choose to create a new VM from the gallery, click on the “My Disks” option and you should see the disk associated with the deleted VM there (it might take up to 30 minutes for the disk to show up in this list after deleting the VM).

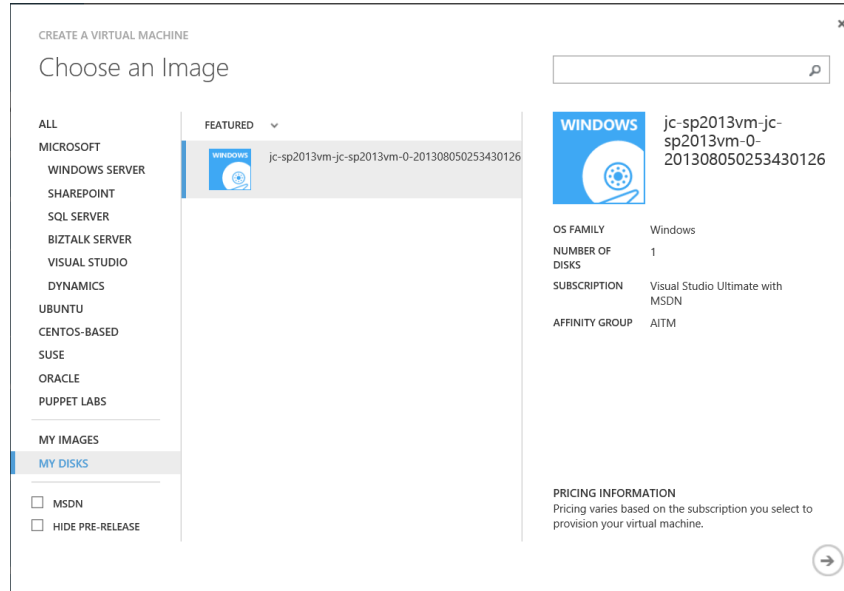


Figure 10 – Creating a VM from an existing Disk

Also note that you can create a VM from one of your existing VHDs as well, which makes good sense when you are migrating on premise VMs into the cloud. You can read more about this [here](#).

9.4 Provisioning additional SQL data disks

When you configure BizTalk Server and the appropriate SQL databases that BizTalk relies on, these databases will all be created on the C: of the SQL Server by default (unless you changed the default data/log file locations as per [this article](#)). This is really not ideal because the C: volume that is automatically created when you provision a VM is really specced to cater for fast OS loads and isn't geared to serve high performance IOs that will be required.

You can easily provision additional data disks for your SQL VM by highlighting the VM in the Azure Portal, clicking the Attach button and selecting attach empty disk.

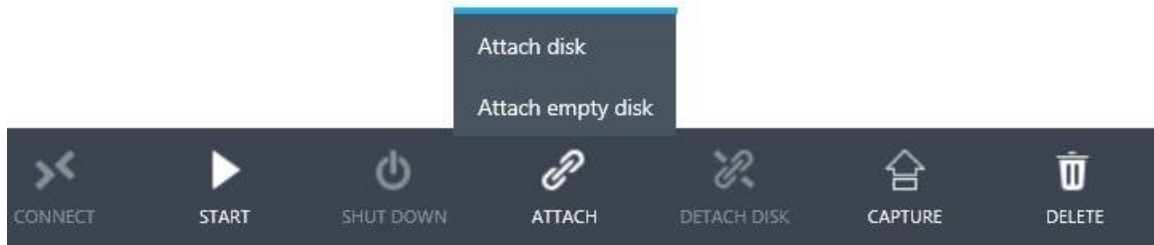


Figure 11 – Attach empty disk

The additional disks would then be available within the VM to assign to new volumes. In Windows Server 2012 (there would be similar steps for other OS's) you would have to open the Server Manager, choose Local Server -> File and Storage Services -> Disks, select the unassigned disk and assign it to a specified drive letter. You can follow these steps to assign as many disks to the server up to the given maximum for the VM size.

Keep in mind that each Azure data disk has a limit of 500 IOPS so you will need to consider this when you plan how many disks you are going to add. You can also stripe multiple data disks together into a single volume to increase IOPS further if required. The steps to stripe multiple disks together are thoroughly detailed in [this article](#).

Once you've created the desired volumes you should move the SQL data and log files (including for the master and TempDB databases) to these new volumes and setup default rules as to which volumes should be used for newly created databases.

9.5 Availability Sets – High Availability in Azure

Along comes the big question. What kind of SLAs can you expect when you're running BizTalk Server on Azure? The answer might be somewhat disappointing to you, at least based on Azure's current capabilities. Microsoft will give you a 99.95% uptime SLA if you have two VMs in an Availability Set, their exact wording follows - **"For all Internet facing Virtual Machines that have two or more instances deployed in the same Availability Set, we guarantee you will have external connectivity at least 99.95% of the time."**

There's a few caveats to this SLA statement. The SQL layer in a BizTalk Server environment can't take advantage of the SQL Server Always-On capability, which is currently the only supported high availability solution for SQL Server in the cloud, so you pretty much get no uptime guarantee on the SQL layer of BizTalk Server as of now. Availability and DR solutions for SQL Server running in Azure have been detailed in [this MSDN post, but none of these are currently applicable in a BizTalk Server environment](#).

If you're using BizTalk Server standard edition then you can't have multiple BizTalk runtime servers in a given group, thus there isn't an applicable scenario in which you could have more than a single BizTalk VM instance in the same Availability Set (the lack of high availability is the same if you run BizTalk Server Standard edition on premise). You would need to use the BizTalk Server Enterprise edition with multiple VMs in an Availability Set in order to qualify for the 99.95% uptime guarantee on the runtime server layer, and even then the SQL layer would be exempt from this SLA until/unless something changes in the future.

What's an Availability Set anyways? You might recall the concept of Affinity Groups which ensure that your cloud instances are within close proximity to each other. An unintended side effect of this (or if you're not using Affinity Groups then plain old bad luck) is that your instances might get too close, and that planned or unplanned outages could end up affecting all your instances rather than just one, thus blowing that 99.95% uptime guarantee out of the window. An Availability Set ensures that your VMs are protected from exactly this scenario. There are two ways in which an Availability Set achieves this, by ensuring your VMs are placed in separate Update Domains (UDs, sometimes also referred to as Upgrade Domains) and Fault Domains (FDs). These are explained in brief below but if you want more information do read [this article](#).

When you place your VMs into an Availability Set, Azure will by default spread them out across up to 5 different UD's (if you have more than 5 VMs in the Availability Set then you will have multiple VMs in the same UD beyond the 5th VM). When Microsoft is performing planned maintenance or upgrades to your VMs they will do so one UD at a time, ensuring that any downtime does not entirely tear down your application as long as each layer is in an Availability Set with multiple instances. Do note that there is no guarantee of the order in which maintenance will be carried out on your UD's.

Likewise, placing VMs into an Availability Set will spread them out across 2 FDs. An FD guarantees segregation of physical hardware that your VMs will depend on, such as power or network switches. VMs in separate FDs will not rely on the same hardware thus ensuring that during unplanned outages due to hardware failures you will not have all your instances affected.

You can create an Availability Set when you provision a VM or post-creation when configuring a VM. When prompted to choose an Availability Set, choose the option to create one for your first VM, and give it a name. When you provision/configure the next VM in the same Cloud Service choose the already created Availability Set and you now have multiple VMs in the Availability Set.

One thing of note is that VMs are not permanently assigned to UD/FDs, but rather these appear to be assigned when you spin up a VM from a deallocated state, thus Azure can ensure that as you spin up multiple VMs they are spread across UD/FDs appropriately. If you visit the instances tab within Cloud Services in the Azure Portal you will see what UD/FDs your VMs are assigned to, and you'll also notice that any VMs that are currently deallocated will not have any assigned UD/FDs.

jc-bt2013vm

DASHBOARD MONITOR SCALE INSTANCES LINKED RESOURCES CERTIFICATES

PROVISIONING One or more role instances are being provisioned.
4 Instances: 2 Starting (Provisioning), 2 Starting

NAME	STATUS	SIZE	UPDATE DOMAIN	FAULT DOMAIN
jc-bt2013vm	* Starting (Provisioning)	Standard_A3	0	0
jc-bt2013vma	* Starting	Standard_A3	1	1
jc-bt2013vmb	* Starting (Provisioning)	Standard_A3	2	0
jc-bt2013vmc	* Starting	Standard_A3	3	1

Figure 12 – Update and Fault Domains

[This article](#) shows how VMs can be associated with an Availability Set during or after provisioning through the use of PowerShell (note that the Availability Set doesn't have to be explicitly created, it will get automatically created the first time it is referred to for a given Cloud Service).

9.6 Autoscaling – just how much BizTalk do you really need?

One truly compelling feature of Azure IAAS is that you can choose to automatically spin up or deallocate your VMs in a given Availability Set based on your current throughput requirements; this feature is called autoscaling. Autoscaling would most likely be best suited to production or performance lab environments for BizTalk Server environments in which CPU on the runtime servers is a bottleneck (typical in applications that make heavy use of orchestration or message debatching) during peak loads, and the amount of load that the server has to handle varies greatly throughout the day.

You have a bit of flexibility in how you configure autoscaling, with the ability to scale automatically based on the CPU usage on currently running VMs, given the current time of the day or day of the week, or a combination of the two. There is also a third option in which your scaling can be based on the number of messages in a Service Bus queue, however this is probably not relevant for BizTalk Server environments unless

all communications are handled through a single queue which is unlikely. While autoscaling also has the benefit of making more RAM available to your BizTalk group, there is currently no way to configure autoscaling to scale up your environment based on available RAM.

You also have the ability to determine how many VMs are added/deallocated to your topology in a single scaling action, what time lapses will be respected between evaluations, and what the minimum and maximum number of instances Azure will scale up or down to. Hopefully Microsoft will keep adding more in the way of flexibility in the future.

By leveraging autoscaling features you could easily setup separate schedules during peak hours whereby you scale online servers up and down based on CPU utilization to cater for peaks and troughs in demand, and potentially have the servers automatically switch off during night time/weekend if there is no requirement for them to be available.

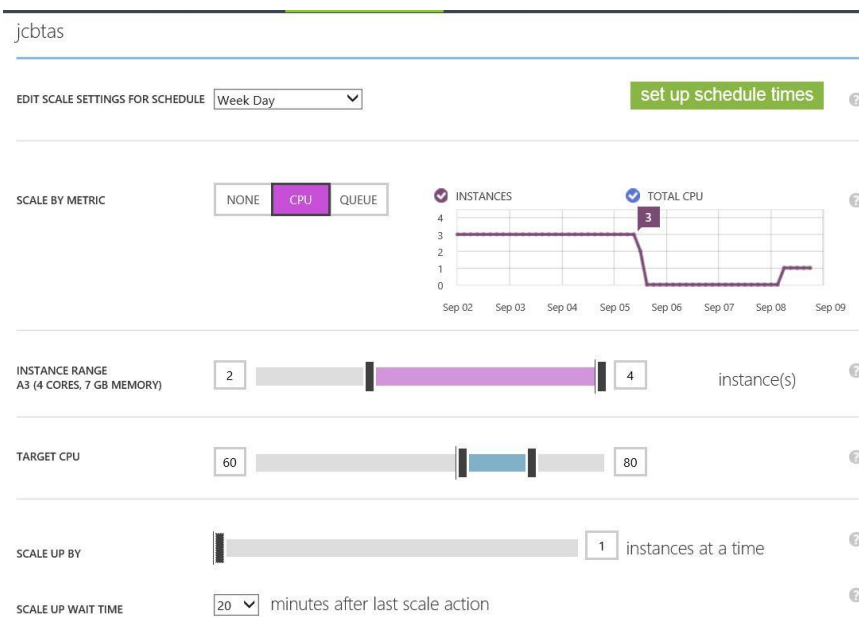


Figure 13 – Autoscaling

Something to keep in mind with autoscaling is that all the VMs in your Availability Set must be the exact same size, so you can't mix low spec and high spec VMs together.

For those who might be a bit nervous about traceability regarding when a VM gets provisioned or deallocated automatically Azure does provide you with operation logs which you can access via the dashboard of your Cloud Service within the portal.

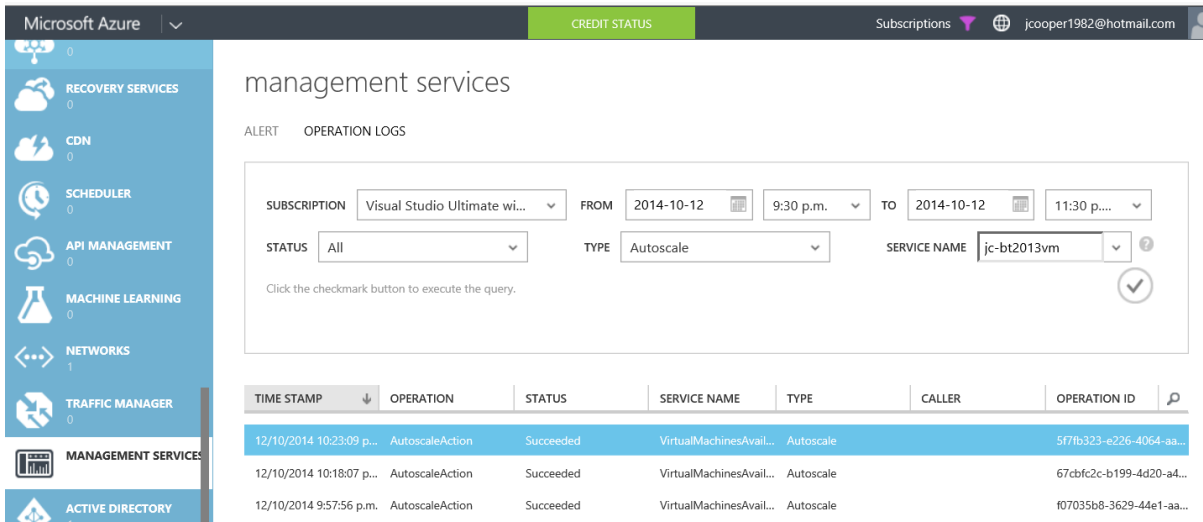


Figure 14 – Autoscaling logs

Also note that autoscaling based on CPU usage is based on averages taken over a time period controlled by a property called the TimeWindow which defaults to 45 minutes. Thus it might take some time before you see your first scale action take place. You can adjust the TimeWindow property above or below the default 45 minutes via PowerShell, however be aware that there could be problems if you set it to less than 25 minutes (see [this article](#) for more information).

Note that you can't use autoscaling to turn on and off VMs if they are not in Availability Sets, or if they are the only VM in an Availability Set. If you want to automate starting and stopping VMs in such a scenario then you should look into the Azure Automation feature (see [here](#) for more information).

Another note if you are taking advantage of autoscaling for BizTalk runtime servers is that if some of your BizTalk runtime servers are currently turned off you might find that when you browse to the platform settings tab in the BizTalk Administration Console the MMC will become unresponsive as it can't see all the servers. This is a limitation of the BizTalk Administration Console which keeps trying to poll the status of all host instances on all servers until it encounters a timeout. A monitoring tool such as BizTalk 360 could be used to fill this gap.

A final thing to keep in mind if you're taking advantage of autoscaling is to ensure that you have set the startup type property on the underlying services for BizTalk host instances to "Automatic (Delayed Start)" to ensure that they always start up automatically when the VM gets spun up as per [this article](#).

Observed results on a load test run against a BizTalk Server environment taking advantage of Azure autoscaling can be read about in [section 9.3](#).

9.7 Endpoints and Load Balancing

You will recall from earlier discussions that by virtue of creating a VNet and adding our VMs to the VNet, our VMs are able to communicate with each other internally using private IP addresses over internal ports. However, exposing these ports to the internet is a whole different story, especially given that multiple VMs behind a given Cloud Service will all share the same public IP address.

By default when you provision a VM in Azure two endpoints are exposed which are PowerShell and RDP. You can choose to expose these endpoints to the public on any unused port number (remember this is an unused port number on the Cloud Service,

not the VM) of your choosing. You can also choose to expose additional internal ports of your choosing.

If you want to specifically expose an HTTP endpoint from a single BizTalk runtime server you would choose to expose port 80 (assuming that your HTTP services on the given VM are bound to port 80) over whatever public port number you want. In the below screenshot the internal port 80 is exposed via the public port 80 as well.

jc-bt360vm

DASHBOARD MONITOR ENDPOINTS CONFIGURE

NAME	PROTOCOL	PUBLIC PORT	PRIVATE PORT
HTTP	TCP	80	80
PowerShell	TCP	5986	5986
Remote Desktop	TCP	3389	3389

Figure 15 – Cloud Service Endpoints

This works well enough when there is only a single VM in a Cloud Service, or there is only one VM which you want to expose these specific ports from in your Cloud Service, but you will of course have to choose non-standard port numbers if you want to expose these ports from each individual VM in a Cloud Service in a non-load balanced fashion. Exposing an endpoint is an easy task which you can do when provisioning a VM or post-creation through configuration. Simply visit the Endpoints tab under your given VM's configuration in the Azure Portal, choose Add, and specify the private and public port numbers and relevant protocols.

It is more likely, especially for protocols such as HTTP and HTTPS that you will want to load balance this traffic to all (or at least a subset) of your BizTalk runtime servers. This makes all the more sense when you are implementing a topology that takes advantage of multiple runtime servers and autoscaling, in which case you will have no idea which VMs are going to be active at any given time and thus you want traffic to be directed to whichever VMs are currently active. Implementing a load balanced endpoint is a simple enough task; when choosing to add an endpoint to a given VM within a Cloud Service ensure you tick the option "Create a load balanced set" and you will be taken to an additional configuration screen which allows you to configure the load balanced endpoint.

When configuring the load balanced endpoint you must specify a probe protocol, path, port, interval in seconds, and number of probes. A probe is similar to a heartbeat, and is simply a relative URL that Azure will poll (exercising an HTTP Get command) on each VM in the load-balanced set using the specified interval. If the HTTP Get returns a 200 success HTTP code then the VM is deemed to be active in the load balanced set. If the probe URL returns non-200 HTTP codes consecutively (the number of consecutive non-200 responses must match the "Number of probes" configuration value before a failure is raised) then the VM is no longer considered to be active in the load balanced set.

Figure 16 – Load Balanced Endpoints

Consider setting the probe URL to a BizTalk based WCF Service over HTTP since port 443 might result in certificate errors as the probe URL will be based on the machine name rather than the Cloud Service name...documentation for the Azure Traffic Manager suggests that HTTPS probes will only check that a certificate exists rather than checking for its validity but there is no explicit documentation regarding this for load balanced Cloud Service endpoints. This ensures that an HTTP 200 status code will only be returned if IIS and the Enterprise Single Sign-On service at the very least are running on your VM and connectivity to the backend SQL databases is successful. Setting the probe URL to a non-BizTalk resource might mean that if BizTalk isn't currently running on one of the VM it will still be entered into the load balanced set since the resource has no dependency on BizTalk services, and thus would result in your service consumers getting some avoidable failure responses. Do note that if you are planning on exposing the port used for probes to the public then you should take measures to protect the probe receive location from being abused as it could potentially be used to inject messages into your BizTalk environment. Consider containing the IIS application corresponding to the probe receive location within a separate website which binds HTTP traffic to a port that you don't plan on exposing to the public.

One other compelling feature of Azure endpoints is that you can setup ACL rules against each endpoint which either restricts requests from certain IP address ranges, or only allows requests from certain IP address ranges (you can read more about ACLs [here](#)).

Keep in mind that the aforementioned load balanced endpoints are only applicable for external traffic that is hitting the public Cloud Service IP address (you can read more about the Azure load balancer [here](#)). Internal load balancing, which caters for load balancing traffic that originated from within your VNet is a relatively new feature to Azure. This feature is bound to be compelling if you have applications that want to

consume services exposed by BizTalk Server which are also deployed within your Azure VNet. You can read more about internal load balancing [here](#).

One final note is that Microsoft have recently released a new feature called Instance Level Public IP Addresses (PIP) which enables you to assign a public IP address directly to a VM rather than to a Cloud Service. This public IP address would be in addition to the Cloud Service IP address rather than service as a replacement. If you believe that this feature will be useful in your topologies then you can read more about it in [this article](#).

9.8 Cloud Service Certificates and HTTPS Endpoints

Microsoft is kind enough to automatically create a self-signed certificate issued to your Cloud Service when you provision a VM, and they even automatically add this certificate into IIS to make it easy for you to bind it to your HTTPS port on your IIS websites. For production environments you will typically want to purchase a trusted certificate issued to your Cloud Service (you could even get a custom domain name for your Cloud Service, see more [here](#)), however for development and test environments self-signed certificates could suffice.

Having these pre-created self-signed certificates is especially handy if you want to expose WCF services from your BizTalk group with transport security enabled since your clients (at least external clients) will be communicating with your Cloud Service address rather than with your VM directly. A caveat with this though is that when you add more VMs to the Cloud Service you'll notice that each VM will have a separate self-signed certificate with different thumbprints generated, and they have non-exportable private keys so you can't share them across multiple VMs.

Having separate certificates per VM doesn't work when you are exposing a Cloud Service HTTPS endpoint corresponding to your BizTalk VMs in a load balanced fashion. You can instead use the MAKECERT command line tool to generate a self-signed certificate that is issued to your Cloud Service as per instructions from [this article](#). The below example command line would generate a self-signed certificate for use on HTTPS endpoints on a cloud service named jc-bt2013vm.cloudapp.net, with an exportable private key, in the Personal certificate store at the Local Machine level.

```
Makecert -r -pe -n CN="jc-bt2013vm.cloudapp.net" -b 05/10/2014 -e 12/22/2018 -eku 1.3.6.1.5.5.7.3.1 -ss my -sr localmachine -sky exchange -sp "Microsoft RSA SChannel Cryptographic Provider" -sy 12
```

You will need to export the certificate to a PFX file along with the private key in order to bind it to an IIS website. To do this you will need to open the certificates MMC on the VM on which the certificate was generated by clicking on the Windows Start button, typing MMC.exe, and choosing to add a snap-in of type Certificates for your Computer Account. If you browse to the Personal Store you will see the generated certificate. Right click on it, choose Tasks -> Export and follow the export prompts to generate a PFX file ensuring that you choose to include the private key (you will need to supply a password when you do this).

To import the certificate onto your VMs you will need to open IIS on each VM, click on the parent folder (same name as your VM name) and choose server certificates, choose import and select your relevant certificate and specify the password. You can then bind HTTPS traffic to the relevant port for your website in IIS and associate the binding with the aforementioned certificate on all the VMs in the load-balanced set.

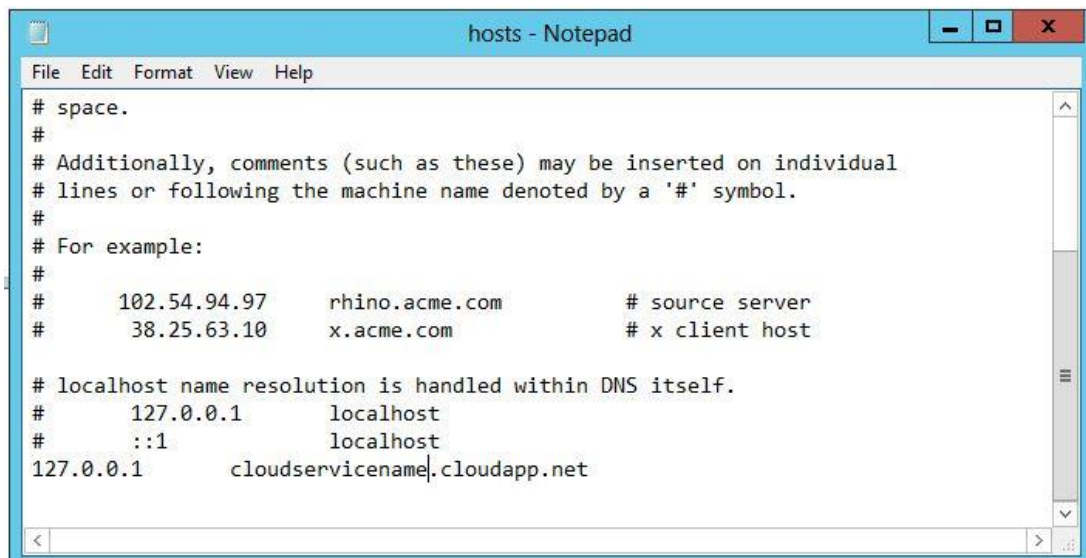
You will probably want to remove the automatically generated self-signed certificate from the IIS certificates list in order to prevent confusion.

An alternative strategy is to use a third party load balancer that supports SSL Offload. This would make certificate management much easier and also reduces CPU load on BizTalk host instances.

9.9 Getting rid of WS-A Address Filter Mismatch errors

Something else worth ensuring is that WCF-WSHTTP services (or any WCF binding that implements WS-Addressing) that you expose don't get upset because of address filter mismatch errors (this is a very common problem, just try googling it if you haven't encountered it yourself) since the To header sent by clients will mention the Cloud Service DNS name rather than the VMs DNS name.

One option is to edit your hosts file (typically in c:\windows\system32\drivers\etc\) such that your Cloud Service address is linked to the server's local IP address as below. The VM will now consider the Cloud Service name to be referencing itself, thus no address filter mismatch error will be encountered.



```

File Edit Format View Help
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com       # source server
#       38.25.63.10      x.acme.com           # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1        localhost
#       ::1              localhost
127.0.0.1       cloudservicename.cloudapp.net

```

Figure 17 – Hosts file for Cloud Service DNS mapping

An alternative method would be to implement a custom behaviour (as described in [this article](#)) to ignore WS-Addressing To headers altogether.

10 Appendix

10.1 Monitoring your BizTalk Server Environments in Azure

Running a BizTalk Server environment in the cloud doesn't mean you have to compromise on monitoring requirements; all the popular monitoring tools that cover BizTalk Server environments on premise will also work in Azure. Do also consider whether you need to build high availability into your monitoring tool to ensure that Azure maintenance or hardware outages do not result in your environment not being monitored.

For example the [BizTalk360](#) product could be made highly available by following instructions detailed in [this article](#), with the additional step of placing the BizTalk 360 runtime servers within an Availability Set in a single Cloud Service. You would then

want to expose a load balanced endpoint (preferably HTTPS) from this Cloud Service so that you could access the monitoring UI remotely.

10.2 Exploring WSFC in Azure

If you have decided to implement WSFC in spite of it not being officially supported then you will probably want to refer to [this guide](#) (the steps detailed in this article have not been verified from the perspective of a BizTalk environment, approach with caution) which walks you through setting up a failover cluster study lab in Azure.

What makes the future look a bit brighter for clustering scenarios is the introduction of the new Azure File Services feature (currently in preview). Previously it was not possible to have a shared disk that multiple VMs could attach to, however the Azure File Services feature will make this possible. Have a read of [this guide](#) to get started with File Services. Note that each disk can only provide a maximum of 1000 IOPS so evaluate carefully whether these will suit your requirements.

10.3 Demonstrating Autoscaling in Action

Below is a detailing of an Azure autoscaling in action. A BizTalk Server Enterprise environment was created in Azure which included an A3 sized Domain Controller, an A4 sized SQL Server, and 4 A2 sized BizTalk VMs which were all part of the same BizTalk group.

Many performance optimization were made on the SQL layer, including (but not limited to) striping additional volumes for data and log files to provide for a high level of IOPS, as well as creating multiple message box databases.

The BizTalk VMs were all contained within a single Availability Set in a Cloud Service, and had a load balanced HTTPS endpoint with an HTTP based probe, that ensured that BizTalk Server must be running on the given VM if it was to be considered active in the load balanced endpoint.

A BizTalk application was deployed to each of these VMs which involved receiving messages on a one-way WCF-WSHttp receive location with a receive pipeline that validates the message which then gets published to the BizTalk message box resulting in a synchronous response being returned to the client. The message will then go through to a reasonably complex orchestration and finally gets routed to a WCF-SQL send port on which a map is executed, the resulting message being written to a SQL database. The application is reasonably CPU hungry by nature, with the orchestration host instances being the primary consumer of CPU. The host settings for the isolated host were tweaked to ensure that the ingestion of messages would not get throttled due to queue sizes in order to ensure that message ingestion maintained priority.

An autoscaling profile was then created against the Availability Set through the Azure Portal as below. Note that because the portal was used to create the autoscaling profile, the TimeWindow configuration property was left at its default of 45 and thus it would take up to 45 minutes before an autoscaling action would be observed.



Figure 18 – Autoscaling options used in the demonstration

The autoscaling profile was enabled and all VMs except for one was shut down to establish the test case. A SOAPUI load test was then started at 21:25 which called on the WCF service with a very heavy load which would definitely overwhelm the CPU on an individual server, and in fact would overwhelm the CPU on all 4 BizTalk VMs if they were active. The goal of this test was to observe autoscaling in action and to note what level of throughput would be observed from SOAPUI based on the number of VMs currently active, and how long after a VM was made active before throughput would increase. The results observed are below.

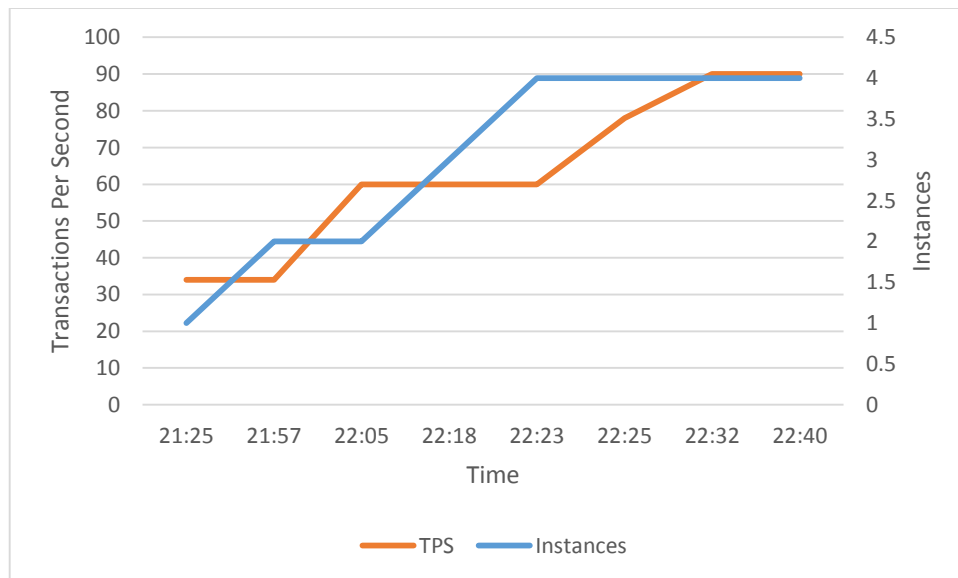


Figure 19 – Load test results with Autoscaling enabled

Of interest is that it takes quite a while for the first autoscaling action to take place, and there is a bit of lead time observed before an increase in throughput is seen, attributed to the VM starting up along with all the BizTalk service on the VM and finally for the load balanced endpoint to recognize the VM being available. The second

autoscaling action happens soon after with a similar lead time before an increase in throughput is observed, however of interest is that even before the increase in throughput was observed the third and final autoscaling action took place.

10.4 Glossary of Acronyms

Acronym	Full Form
ACL	Access Control List
BAM	Business Activity Monitoring
CAPEX	Capital Expenditure
CAT	Customer Advisory Team
CPU	Central Processing Unit
DB2	Database 2 (IBM)
DC	Domain Controller
DIP	Static Internal IP Address
DNS	Domain Name Server
DR	Disaster Recovery
EA	Enterprise Agreement
FD	Fault Domain
FTP	File Transfer Protocol
GA	Generally Available
HTTP	Hypertext Transfer Protocol
IAAS	Infrastructure As A Service
IO	Input/output
IOPS	Input/output Operations Per Second
IP	Internet Protocol
IT Ops	Information Technology Operations
MABS	Microsoft Azure Service Bus
MSDN	Microsoft Developer Network
OPEX	Operational Expenditure

OS	Operating System
PAAS	Platform As A Service
PIP	Instance Level Public IP Address
POC	Proof of Concept
POP3	Post Office Protocol 3
RDP	Remote Desktop Protocol
RTA	Real-time aggregated
SA	Software Assurance
SLA	Service Level Agreement
UD	Update/Upgrade Domain
URL	Universal Resource Locator (internet link)
VHD	Virtual Hard Drive
VM	Virtual Machine
VNet	Azure Virtual Network
VPN	Virtual Private Network
WCF	Windows Communication Foundation
WS-A	Web Service Addressing
WSFC	Windows Server Failover Clustering

Figure 20 – Glossary of Acronyms

11 Acknowledgements

I would like to take this opportunity to thank all the reviewers who have provided invaluable feedback that has ensured that this document is technically accurate and readable. Each reviewer has taken time out of their busy schedules to pour over this relatively long document with a fine-tooth comb, in some cases over multiple versions.

I would also like to especially acknowledge my employers at [Datacom New Zealand](#), who have allowed me time to perform some R&D on this topic as well as to write this document during office hours, and for embracing a community spirit by allowing this paper to be made publicly available. I would also like to encourage my fantastic colleagues who have encouraged me through the long process of writing this White Paper.

A special thanks goes out to the team at [BizTalk360](#) who are graciously hosting and publicizing this article, and are always actively working towards promoting BizTalk Server.

I have to thank the BizTalk community at large for being a constant source of new information for BizTalk newbies and gurus alike. The inspiration for this document was the leadership on show from the MVP crowd that presented at the BizTalk Summit 2014 in Sydney which I was lucky enough to attend. It just goes to show that the sharing of information is a two way street as it encourages others to share more as well.

I would like to extend a huge thanks to my wife Jeska and my 9-month old baby Isla who have both been extremely patient with me while I have taken the time out to research and write this White Paper and have supported me every step of the way (in Isla's case I had to provide distractions). I'd also like to thank my parents for having the foresight to purchase a trusty old x386 for me when I was 5 years old, which I quickly proceeded to destroy with multiple viruses and fix up again, thus starting a lifelong interest in IT.