Microsoft Azure - Starter Kits for Partners

Architecture

Development and Test Enviroment

Last Update: March 2016





**MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.**

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The descriptions of other companies’ products in this document, if any, are provided only as a convenience to you. Any such references should not be considered an endorsement or support by Microsoft. Microsoft cannot guarantee their accuracy, and the products may change over time. Also, the descriptions are intended as brief highlights to aid understanding, rather than as thorough coverage. For authoritative descriptions of these products, please consult their respective manufacturers.

© 2014 Microsoft Corporation. All rights reserved. Any use or distribution of these materials without express authorization of Microsoft Corp. is strictly prohibited.

Microsoft and Windows are either registered trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

[Overview 4](#_Toc446688992)

[This Starter Kit Selected Architecture 4](#_Toc446688993)

[Account & Subscription Management for Dev & Test 4](#_Toc446688994)

[Choosing a Compute Service 5](#_Toc446688995)

[Network and Domain Integration 5](#_Toc446688996)

[Dev & Test Infrastructure Deployment 7](#_Toc446688997)

[Application Deployment 9](#_Toc446688998)

[Additional Considerations for Web Workloads 9](#_Toc446688999)

[High Availability 9](#_Toc446689000)

[Session affinity 11](#_Toc446689001)

[TFS on Azure IaaS 11](#_Toc446689002)

[Azure DevTest Lab 12](#_Toc446689003)

Architecture for Azure Development and Test Environment

# Overview

The purpose of this document is to complement the lab document to include a broader discussion of concepts relevant to setting up a development and test environments in Azure Virtual Machines service.

# This Starter Kit Selected Architecture

The scenario below is illustrated in the Cost Calculator and implemented in the Deployment Guidance (Hand on Labs). However, you may make changes in the architecture and topology as appropriate.



The rest of this document will discuss the other architecture aspects possible for this scenario.

# Account & Subscription Management for Dev & Test

For development and test scenarios Microsoft Partners should use [the Azure MSDN benefit](http://azure.microsoft.com/en-us/pricing/member-offers/msdn-benefits/). The virtual machine pricing in this type of subscription is much lower than for any other subscription type. Also free azure credit is given each month (as opposed to Azure Free Trial subscription) which is essential for projects that will run over several months. Please refer to the Azure Cost Guidance document in this kit for more detail.

Once the subscription is created, the subscription owner can add co-administrators to allow them to administer resources such as virtual machines, storage and networking. Any Microsoft account can be added as co-administrator, however, only Organizational accounts from Windows Active Directory “homed” to the subscription [are allowed](http://msdn.microsoft.com/library/azure/dn629581.aspx).

The subscription owner or co-administrators can also generate and upload [Management certificates](http://msdn.microsoft.com/en-us/library/azure/gg551722.aspx) to the subscription. They can distribute them to other team members that can then manage the subscription resources using Visual Studio or PowerShell. The management certificates can be easily removed from the subscription when no longer required.

In general, we recommend that Organizational accounts are configured and used with the Azure subscription management. New PowerShell commands that use Azure Resource Manager require you to provide organizational account and do not work with Microsoft accounts.

*Note: Partners with existing Enterprise Agreements should note that you must not add the Microsoft Account of the Azure MSDN Subscription Owner as an Account Owner in Enterprise Agreement otherwise your Azure MSDN subscription consumption will not benefit from the lower VM pricing and be billed against your Enterprise Agreement.*

# Choosing a Compute Service

Microsoft Azure Platform offers three compute services: Azure Virtual Machines, Azure Web Apps and Azure Cloud Services (Web and Worker roles). Azure Virtual Machines offer image state persistency as well as the choice of the operating system (Windows Server or LINUX) and therefore are suitable for the majority of the applications.

Web workloads (websites and web services), however, can also be deployed to Azure Web Apps and the Azure Cloud Services (Web and Worker roles). Due the nature of these services, more detailed technical analysis of the application in question is required to determine hosting feasibility such as access to the underlying application server configuration, operating system dependencies, access to domain protected resources, etc.

Because Azure Virtual Machines [offer full control](http://azure.microsoft.com/en-us/documentation/articles/choose-web-site-cloud-service-vm/) over the environment, the rest of the discussion here focuses on this service for the application hosting.

# Network and Domain Integration

Many dev. & test workloads require domain environment. This can be achieved by integrating at the network level using site to site [Azure Virtual Network](http://azure.microsoft.com/en-us/services/virtual-network/) and make use of the existing domain controller. This configuration requires that you configure a VPN Device in your intranet and configure Azure Virtual Network Gateway. The advantage of using Azure VPN is that it creates a direct connection between the development machine in your intranet and Azure Virtual Machines and you can troubleshoot and debug them using the same tools you would use for on-premises applications. You can also connect tools like SQL Server Management Studio to manage the database directly. Note that the Azure virtual machines are not exposed to the Internet traffic (do not have VM endpoints configured) and that all network communication happens through the VPN channel established



**Figure 1**. Dev & Test environment integrated using Azure site to site (s2s) and point to site (p2s) virtual network.

Additional option to having full network integration is to configure point to site Azure virtual network. In that scenario a developer (perhaps a contractor) can connect to the VM on Azure virtual network through the VPN client software after appropriate certificates to secure the connection are configured in the Gateway.

If the network integration between the Azure platform and company intranet does not exist, the developer can still debug code in the individual virtual machines by enabling the [Remote Debugger Virtual machine extension](http://msdn.microsoft.com/en-us/library/azure/ff683670.aspx). This procedure opens public endpoints in Azure VM to which the developer will connect the debugger over the Internet. Similarly to be able to connect to SQL Server with Management Studio would require public endpoint configuration to allow SQL Server traffic.

For security reasons the remote debugging is automatically disabled after 48 hours but it can be enabled again easily from Visual Studio if required.



**Figure 2**. Dev & Test environment with no network integration. Virtual Machines have been configured with debugging extension that also enables required endpoints for the VMs. Developers connect through the Internet, Azure Load Balancer and configured endpoints to individual VMs.

Finally, a cloud-only dev & test environment is possible, where the development machine is a virtual machine running in the Azure platform. The Azure MSDN subscription includes pre-build Windows 8 image with Visual Studio pre-installed. In that scenario the developer would use RDP connection to the VM but that option has a disadvantage of increased latency and bandwidth consumption.

# Dev & Test Infrastructure Deployment

The decision about the integration of Azure virtual network with the intranet determines the need for a VPN device, creation of the VPN Gateway and use of on premises DNS for name resolution. To complete the network design it is important to carry out further configuration such as IP ranges and network segmentation, DNS configuration etc. This detail will depend on the application. In the Figure below as an example the cloud only virtual network is created assigned IP range has been divided between a frontend and a backend:



**Figure 3**. IP address and network segmentation in Azure Virtual Network with two sub-networks. The virtual machines are assigned dynamic IP addresses from the assigned IP range

With the network integration and design in place you can proceed to virtual machines to the environment (note that you cannot modify your network after virtual machines are deployed onto it so it is essential to include this in your planning).

The infrastructure required for the dev & test environment (virtual network, storage, virtual machines and any additional Azure services can be provisioned through the [PowerShell](http://azure.microsoft.com/en-us/documentation/articles/install-configure-powershell/) scripts or [cross platform command line interface](http://azure.microsoft.com/en-us/documentation/articles/xplat-cli/). Examples of scripts are available on [Azure script center](http://azure.microsoft.com/en-us/documentation/scripts/).

The MSDN Azure subscription contains a number of pre-configured images that can be used as a starting point or you can upload your own image. After the initial configuration of the image it can be snapshotted and [generalized](http://azure.microsoft.com/en-us/documentation/articles/virtual-machines-capture-image-windows-server/) so that additional virtual machine instances can be created from them and used with scripts.

For web applications, Visual Studio 2013 offers developers possibility to deploy the application to new or existing virtual machine and generates and adds to the solution [PowerShell scripts](http://msdn.microsoft.com/en-us/library/dn642480.aspx) containing instructions for the creation, configuration of the VM and deployment of the application package through web deploy. This script can be modified as required and used to re-deploy the infrastructure again.

Currently the Visual Studio 2013 generated script will skip configuration for already existing resources, i.e.: each resource in the infrastructure is handled individually. Because of this limitation Azure Resource Manager has been created to it can manage and deploy a group of resources as a logical unit. You interact with the Resource Manager through a different set of [PowerShell commands](http://azure.microsoft.com/en-us/documentation/articles/powershell-azure-resource-manager/) and supply it with the template file that describes individual resources and dependencies between them. The set of templates in likely to expand to cover additional services and to include virtual machines as well.

# Application Deployment

The application deployment varies with the application type. Web applications can be deployed using [WebDeploy](http://technet.microsoft.com/en-us/library/dd569058(v=ws.10).aspx) tool and Visual Studio 2013 Update 3 allows the developer to create a virtual machine with IIS, WebDeploy and Remote Debugging enabled. This functionality implemented in Azure virtual machine through the Virtual Machine [extensions](http://azure.microsoft.com/blog/2014/04/11/vm-agent-and-extensions-part-1/): Windows Azure WebDeploy Extension for DevTest and Remote Debugging. They [can also be added](http://azure.microsoft.com/blog/2014/08/04/announcing-release-of-visual-studio-2013-update-3-and-azure-sdk-2-4/) to existing virtual machine instances.

Visual Studio can build, package and deploy the application to the virtual machine as one automated workflow or you can build the Web Deploy Package only and then use the [Publish-WebApplicationVM](http://msdn.microsoft.com/en-us/library/dn689112.aspx) PowerShell script to deploy it later.

Furthermore, it is possible to integrate the process of the infrastructure creation and application deployment with Visual Studio Online (or Team Foundation Server running on-premises) build by running PowerShell script as a pre or post compilation step of the process.

In case of web farms on Azure Virtual Machines, unlike is the case with PaaS services (Azure Web Apps or Azure Cloud services) you have to take care or ensuring that the application is deployed to all servers in a web farm. You can achieve this by configuring the IIS web farm to use an operating system Task that runs periodically and uses web deploy [to synchronize the changes](http://michaelwasham.com/2012/08/13/publishing-and-synchronizing-web-farms-using-windows-azure-virtual-machines/) across the web farm.

# Additional Considerations for Web Workloads

One of the advantages of the Azure platform is the availability of compute resources on demand that makes it easy to carry out tests of your application when it is scaled out. In this section we discuss additional, more advanced considerations when working with web applications deployed to more than one virtual machine.

## High Availability

Issues regarding high availability can be of interest in Dev & Test environments when the developer team performs application tests to ensure it will run correctly in a scaled out, highly available environment. Indeed, Azure platform only offers [Service Level Agreement](http://azure.microsoft.com/en-us/support/legal/sla/) (SLA) for Azure virtual machines if they are configured for high availability. The reason is that the platform, which relies on virtualization, requires periodic [host updates](http://blogs.technet.com/b/markrussinovich/archive/2012/08/22/3515679.aspx) (security, reliability and performance updates) that may cause the virtual machine (guest OS) to be rebooted. To achieve high availability regardless of the host updates it is necessary to assign two or more virtual machines to [Availability Set](http://azure.microsoft.com/en-us/documentation/articles/virtual-machines-manage-availability/). Machines in the same availability set will not reboot all at once, leaving at least one virtual machine operational.

Therefore, the general recommendation is to place each application tier into separate Availability Set, for example, web server machines and database server machines would be placed into separate sets.



**Figure 4**. Machines in the web tier and database tier assignment to the Availability Sets.

Ideally, virtual machines should be assigned to the Availability set at creation time because adding a virtual machine to an availability set will require an OS restart.

For the web tier you could then create in each virtual machine a [load balanced endpoint](http://azure.microsoft.com/en-us/documentation/articles/virtual-machines-set-up-endpoints/) to allow HTTP/S traffic a through the Azure load balancer (which would open a public port to the internet) and into virtual machine (listening on a private port). The load balanced set configuration ensures that the load balancer will distribute traffic to all virtual machines in the set.



Figure 5. Endpoints on individual virtual machines are added to the load balanced set.

The database tier would require additional configuration to setup a high availability mechanism of your choice and depends on the specific database technology. For [SQL Server high availability on Azure](http://msdn.microsoft.com/en-us/library/jj870962.aspx) you can use either [Database mirroring](http://msdn.microsoft.com/en-us/library/jj870961.aspx) (available for Standard edition) or [Always On Availability Groups](http://msdn.microsoft.com/en-us/library/dn249504.aspx) (available for the Enterprise edition).

Depending on your choice of the mechanism you may need to reconsider your network design as more infrastructure may have to be deployed (Active Directory, witness server, etc.)

## Session affinity

[Azure Load Balancer](https://azure.microsoft.com/en-us/documentation/articles/load-balancer-overview/) performs load balancing at Layer 4 of the OSI model (i.e. TCP or UDP packet level) and distributes incoming traffic uniformly across load balanced endpoint set using a hash function. It has no knowledge of higher level protocols or objects (HTTP/S, cookies, etc.). SSL connections are terminated at the respective host machine, not the load balancer.

This type of load balancing ensures that while the client maintains an active TCP/IP connection it will be directed to the same host. However, if the session is idle the underlying TCP/IP connection will be terminated ([idle timeout is configurable](http://azure.microsoft.com/blog/2014/08/04/announcing-release-of-visual-studio-2013-update-3-and-azure-sdk-2-4/) between 4 to 30 minutes). When the client makes new HTTP requires it will be directed to a potentially different host. This means that web applications that rely on the affinity of the HTTP session to the server (e.g.: because they store the session in the machine memory rather than out of process) will not work correctly.

In general, for scalability and reliability reasons it is recommended that if possible applications do not rely on session affinity (e.g.: store the session state in SQL Server or Caching Service). For applications that cannot be reengineered or have specific requirements for session affinity a possible solution for deployments to Azure virtual machines is to deploy [IIS Application Request Routing](http://www.iis.net/learn/extensions/configuring-application-request-routing-(arr)/configure-3-tier-deployment-architecture-using-application-request-routing) (IIS ARR) between the Azure Load Balancer and the application servers.

Note: Azure Websites service by default uses IIS ARR and is configured to use session affinity. Applications can opt-in to [disable session affinity](http://azure.microsoft.com/blog/2013/11/18/disabling-arrs-instance-affinity-in-windows-azure-web-sites/) when deployed to Azure websites through the custom header setting in the application web.config file.

# TFS on Azure IaaS

The **Visual Studio ALM Ranger** has created a solution that delivers practical and scenario based guidance for the implementation of Team Foundation Server (TFS), including TFS on Azure IaaS. They guide you through the decisions whether to have one or more Team Foundation Servers, one or more Team Project Collections, one or more Team Projects and one or more Teams, based on scenarios and implications of each decision. In addition we cover disaster recovery planning with a focus on avoidance, frequently asked questions and a collection of real-world reference stories. The solution includes practical guidance on TFS on Azure IaaS planning and implementation scenarios.

<https://vsarplanningguide.codeplex.com/>

# Azure DevTest Lab

Azure DevTest Lab is a service that helps developers and testers quickly create environments in Azure while minimizing waste and controlling cost. You can test the latest version of your application by quickly provisioning Windows and Linux environments using reusable templates and artifacts. Easily integrate your deployment pipeline with DevTest Lab to provision on-demand environments. Scale up your load testing by provisioning multiple test agents, and create pre-provisioned environments for training and demos.

The following list contains key DevTest Lab concepts and definitions:

**Artifacts** are used to deploy and configure your application after a VM is provisioned. Artifacts can be:

* Tools that you want to install on the VM - such as agents, Fiddler, and Visual Studio.
* Actions that you want to run on the VM - such as cloning a repo.
* Applications that you want to test.

Artifacts are Azure Resource Manager (ARM) based JSON files that contain instructions to perform deployment and apply configuration. You can read more about ARM in the [Azure Resource Manager overview](https://azure.microsoft.com/en-us/documentation/articles/resource-group-overview/).

**Artifact Repositories** are git repositories where artifacts are checked in. Same artifact repositories can be added to multiple labs in your organization enabling reuse and sharing.

**Base** is a VM image with all the tools and settings preinstalled and configured to quickly create a VM. You can provision a VM by picking an existing base and adding an artifact to install your test agent. You can then save the provisioned VM as a base so that the base can be used without having to reinstall the test agent for each provisioning of the VM.

**Caps** is a mechanism to minimize waste in your lab. For example, you can set a cap to restrict the number of VMs that can be created per user, or in a lab.

**Policies** help in controlling cost in your lab. For example, you can create a policy to automatically shut down VMs based on a defined schedule.