[DOCUMENT TITLE]

Mohamed Reda AbdelAal

[COMPANY NAME] [Company address]

Table of Contents

Application State 16
Application State 16
Stateless Design 16
Scalability 16
Why Patterns? 17
Why Patterns? 17
Azure Cloud Application Design and Implementation Guidance 17
MSDN Cloud Design Patterns 18
MSDN Cloud Design Patterns 18
Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications 19
Infographic 19
Performance Concerns with Cloud Applications 19
Performance Concerns with Cloud Applications 19
Valet Key Pattern 19
Problem: Serving secure data from distributed applications 19
Solution: Data services and temporary access tokens 20
Valet Key Pattern 20
EXAMPLE Implementation in Azure 21
Distributing Application Data Across Multiple Instances 22
Distributing Application Data Across Multiple Instances 22
Sharding Pattern 23
Problem: Hosting large volumes of data in a traditional single-instance store 23
Solution: Partitioning Data Horizontally Across Many Nodes 24
Sharding Pattern 24
EXAMPLE Implementation in Azure 25
What is Resiliency? 26
What is Resiliency? 27
Transient Errors 27
Transient Errors 27
Retry Pattern 27
Problem: Intermittent Errors with Cloud Services 27
Solution: Application logic to retry requests that have temporarily failed 28
Retry Pattern 28
Example Implementation in Azure 29
Types of Retry Patterns 29

Retry Policy 29
Policy Components 29
Escalating Wait 30
Asynchronous Messaging 31
Asynchronous Messaging 31
Message Queues 31
Message Queues 31
Competing Consumers Pattern 32
Problem: Handling variable quantities of requests 32
Solution: Asynchronous messaging with variable quantities of message producers and consumers 32
Competing Consumers Pattern 32
Example Implementation in Azure 33
Caching Data for a Distributed Application 34
Why Cache? 34
Caching Data for a Distributed Application 34
Cache-Aside Pattern 35
Problem: Cached data consistency 35
Solution: Read/Write-Through Caching 35
Cache-Aside Pattern 36
Example Implementation in Azure 36
App Services and Web Apps 37
Azure App Service 37
Web Apps 38
App Service Plans 38
Scaling Web Apps 39
Scaling Web Apps 39
Auto-scale 39
Example 40
Deploying Web Apps 40
Deploying Web Apps 40
Package Deployment 41
Web Deploy Packaging and Deployment 42
Web Deploy 42
Using Web Deploy with Visual Studio and ASP.NET 43
Debugging and Monitoring Web Apps 44

Remote Debugging 44
Viewing Logs 46
Managing Deployment Credentials 47
Managing Deployment Credentials 47
Site Extensions 50
Site Extensions 50
Custom Site Extensions 52
Deployment using Local Git 53
Azure Web Sites Deployment using KUDU 53
Continuous Deployment 53
Continuous Deployment 53
Web Jobs 54
Web Jobs 54
Continuous WebJobs 55
Back-up and Restore Web Apps 56
Back-up and Restore Web Apps 56
Performing a Backup 57
Restoring from a Backup 58
Using Traffic Manager with Web Apps 58
Traffic Manager 58
Using Traffic Manager with Web Apps 59
Introducing App Service Environments (ASE) 60
Introducing App Service Environments (ASE) 60
Networking in ASE 61
Introducing Azure SQL Database 61
Introducing Azure SQL Database 61
Azure SQL Database Tiers 62
Azure SQL Database Architecture 64
Azure SQL Database Architecture 64
Client Layer 66
Azure SQL Database Elastic Scale 66
Azure SQL Database Elastic Scale 66
Logical Layout 68
Migrations to Azure SQL Database 69
Migrations to Azure SQL Database 69
SQL Data Platform 70

Azure SQL Database v12 70
Azure SQL Database v12 70
Customer Scenario 71
Who is the customer? 71
What does the customer already have? 71
Customer Goal 72
What is the customer's goal? 72
Solution Considerations 73
What does the customer need? 73
What things worry the customer? 73
Scale & Performance 73
Business Continuity 73
Tool Familiarity 73
Connectivity 74
Management 74
Security 74
Common Implementations 74
What are some of the available Azure services? 74
Multi-Channel Marketing Application 74
Line of Business Application in Infrastructure Services 75
Call to Action 76
Design the Solution 76
Web App & SQL Case Study 77
Sample Solution 77
Azure Resource Manager 77
Azure Resource Manager 77
Interacting with Resource Manager 78
Resource Groups in the Portal 79
Resource Groups 80
Resource Groups 80
Resource Group Deployments 81
Tags 81
Tags 81
Introducing Templates 83
RESOURCE GROUP TEMPLATES 83
COMMUNITY TEMPLATES 85

Advantages of Using Templates 85
ADVANTAGES OF USING TEMPLATES 86
ARM Template References 87
ARM Template References 87
Azure Storage 88
Azure Storage 88
Replication 88
Architecture 89
Types of Storage 90
Types of Storage 90
Blobs 90
Two Types 91
Tables 91
Features 91
Queues 91
Functions 91
Features 91
Azure Storage Blobs 91
Storage Blobs 92
Block Blobs 92
Page Blobs 92
Containers 92
REST API 93
Azure Storage Queues 93
Azure Storage Queues 93
Queue Service Components 94
Storage Queue Message Handling 94
Storage Queue Message Handling 94
Azure Storage Tables 96
Azure Storage Tables 96
Storage Table Components 96
Table Partitioning 97
Azure Files 98
Azure Files 98
File Components 99
StorSimple 100

StorSimple ... 100
Features ... 101
Data Tiering ... 102
Security in Azure Storage ... 103
Container Security ... 103
Shared Access Signatures ... 104
Stored Access Policies ... 105
Best Practices ... 106
Introducing Mobile Apps ... 106
Introducing Mobile Apps ... 106
Client SDKs ... 107
Architecture ... 109
Tables ... 109
.NET Mobile Apps ... 110
.NET Mobile Apps ... 110
Data Objects ... 111
Controllers ... 111
JavaScript Mobile Apps ... 112
JavaScript Mobile Apps ... 112
Table Operations ... 113
Custom API ... 113
Introducing No-SQL ... 114
Introducing No-SQL ... 114
NoSQL Storage Mechanisms ... 115
Downloads and transcripts ... 115
Document Databases ... 115
DocumentDB ... 116
What is Azure DocumentDB? ... 116
DocumentDB ... 116
DocumentDB Features ... 117
SQL ... 117
Consistency Levels ... 118
DocumentDB Resources ... 119
DocumentDB REsources ... 119
Resource Model ... 120
Addressing Resources ... 122

Programmatic Elements in DocumentDB 123
Programmatic Elements 123
SQL Query 123
Transactions and JavaScript Execution 124
MongoDB 124
MongoDB 124
MySQL 124
MySQL 124
HBase 125
HBase 125
Use Cases 125
Service Bus Overview 126
Service Bus 126
Namespaces 126
Service Bus Notification Hubs 127
Service Bus Notification Hubs 127
Notification Hubs and Mobile Services 128
Notification Hubs and Mobile Services 128
Mobile App Push Notification Flow 129
Filtering Notification Recipients 130
Filtering Notification Recipients 130
Comparing Service Bus Queues to Storage Queues 131
Service Bus Queue 131
Storage Queue 132
Customer Scenario 132
Who is the customer? 132
What does the customer already have? 132
Customer Goal 133
What is the customer's goal? 133
Solution Considerations 134
What does the customer need? 134
What things worry the customer? 134
Common Implementations 134
Azure Mobile Apps 134
Call to Action 135
Design the Solution 135

Sample Solution 136
Active Directory 136
Active Directory 136
Active Directory Integrations 136
Azure Active Directory (AD) 137
Azure Active Directory (AD) 137
Managing Azure Active Directory 137
Azure Active Directory Endpoints 138
Sync Active Directory to Azure AD 138
Sync Active Directory Identities to Azure AD 138
Directory Sync 139
Sync Options 140
Identity Sync 140
Password Sync 140
Password Sync with Writeback 140
Azure Active Directory Single-Sign On 140
Federated Identity with Azure Active Directory 140
Azure Active Directory Single-Sign ON 141
Using Third-Party Identity Providers 142
External Users 142
Azure Active Directory B2C 142
Azure Active Directory B2C 142
Access Control List 143
Access Control List 143
Network ACLs and load balanced sets 144
Network Security Groups 144
Network Security Groups 144
Managing Network Security Groups 145
Default Network Security Group Rules 145
Inbound 145
Outbound 146
Using Network Security Groups to Imply Connectivity Rules 146
Using Network Security Groups in a Subnet 146
Windows Server Firewall 148
Windows Server Firewall 148
Network Security Groups and Windows Firewall 148

Azure Subscriptions 148
Azure Subscriptions 149
Azure Accounts 149
Subscription User Types 151
Subscription User Types 151
Role-Based Access Control (RBAC) 152
Role-Based Access Control (RBAC) 152
Role 152
Role Assignment 153
Resource Scope 153
RBAC and Resource Groups 154
RBAC and Resource Groups 154
Scoping to Resource Groups 155
Storage Security 156
Storage Security 156
Anonymous Access 157
Shared Access Signatures and Stored Access Policies 157
Shared Access Signatures 157
Valet-Key Pattern using Shared Access Signatures 159
Stored Access Policies 160
Express Route 161
ExpressRoute 161
Provider Types 162
Exchange providers (EXPs) 163
Network service providers (NSPs) 163
Site-to-Site 163
Site-to-Site 163
Site-to-Site Connectivity 164
Point-to-Site 164
Point-to-Site 164
Point-to-Site Connectivity 165
Mixing and Matching the Networking Options 165
Combining Site-to-Site and Point-to-Site Connectivity 165
Combining ExpressRoute and Site-to-Site Connectivity 166
Virtual Network to Virtual Network 167
Virtual Network to Virtual Network Connectivity 167

Connecting Across Cloud Providers 168
Amazon Web Services (AWS) 168
Affinity Groups 169
Affinity Groups 169
Regional Virtual Networks 169
Cloud Services 170
Azure Compute Options 170
Cloud Services 171
Cloud Services and Virtual Networks 171
Virtual Machines 172
Virtual Machine Connectivity 172
Availability Sets 173
App Service 174
App Service 174
App Service Connectivity 175
Virtual Networks 175
Hybrid Connections 176
Network Load Balancing (Internal, External, etc.) 177
Network (External) Load Balancer 177
Layer-4 Load Balancer, Hash based distribution 178
Internal Load Balancer 178
Designing a "Lights-Out" Deployment Using Azure Resource Manager 179
"Lights-Out", Immutable Deployment 179
"Lights Out" Deployment 180
Immutable Deployment 180
Designing a "Lights-Out" Deployment Using Azure Resource Manager 180
Fault/Update Domains 182
Fault/Update Domains 182
Domains and System Updates/Failover 183
Fault/Update Domains and System Updates/Failover 183
Virtual Machine Sizes and Relation to Available Features 183
Extended Virtual Machine Features By Size 183
A8/A9 183
A10/A11 184
D-Series 184
G-Series 184

Customer Scenario 184
Who is the customer? 184
What does the customer already have? 184
Customer Goal 185
What is the customer's goal? 185
Solution Considerations 185
What does the customer need? 185
What things worry the customer? 186
Common Implementations 186
What are some of the available Azure services? 186
Azure Virtual Networks 186
Site-to-Site and Point-to-Site Hybrid Connectivity 187
Azure ExpressRoute Hybrid Connectivity 188
Additional References 189
Call to Action 190
Design the Solution 190
Networking Case Study 191
Sample Solution 191
Introducing High-Performance Computing (HPC) 191
Introducing High-Performance Computing (HPC) 191
Remote Direct Memory Access 191
HPC using Azure Virtual Machines 192
HPC Pack using Azure Virtual Machines 192
RDMA on Linux Virtual Machines in Azure 193
Azure Services for HPC (Batch) 193
Azure Batch 193
Scaling out Parallel Workloads 193
Cloud Services Worker Roles 194
Cloud Services Worker Roles 194
Advantages to Asynchronous Messaging/Processing 196
Web App WebJobs 197
Web App WebJobs 197
Asynchronous Messaging/Processing with WebJobs 198
Implementing a Basic WebJob 198
Implementing a Basic WebJob 199
ASP.NET MVC 199

Model-View-Controller (MVC) Pattern 199
ASP.NET MVC 200
Features 201
ASP.NET Web API 201
ASP.NET Ecosystem 202
ASP.NET Ecosystem 202
ASP.NET Identity 202
Other Azure Services 203
Azure Services 203
Azure Services 203
Machine Learning 203
Machine Learning 204
HDInsight 205
HDInsight 205
Search 206
Search 206
DocumentDB 207
DocumentDB 207
Media Services 208
Media Services 208
Azure Active Directory 209
Azure Active Directory 209
Microsoft Monitoring Solutions 210
Monitoring Application Solutions 210
Benefits of Monitoring 210
Flexibility 210
Bug Tracking 211
Behavior/Performance 211
Existing Infrastructure Monitoring Solutions 211
System Center 211
Existing Application Monitoring SOlutions 211
Microsoft Monitoring Agent 211
Third-Party 212
Application Monitoring using Application Insights 212
Application Monitoring using Application Insights 212
Application Insights Features 212

Monitoring Exceptional Scenarios using Application Insights 214
Client/Server Events 215
Telemetry Data 215
Using the Application Insights Dashboard 216
Using the Application Insights Dashboard 216
Infrastructure Monitoring using Operational Insights 216
Operational Insights 216
Extended Features 218
Connectivity to System Center 218
Centralized Machine Data/Log Management 218
Solutions 219
Using the Operational Insights Dashboard 219
Using the Operational Insights Dashboard 219
Availability Groups and Update Domains for Patching 219
Azure Host OS Patching 219
Cloud Service Patching Workflow 220
SLA Considerations 220
Updating Compute Instances 221
Updating PaaS Instances 221
Why Separate Stores for Settings and Code? 222
How do separate stores work? 222
Azure's Architecure: Discussing Host Updates 223
Azure's Architecure: Discussing Host Updates 223
PowerShell 223
Windows PowerShell 223
PowerShell ISE 224
Azure PowerShell Module 225
PowerShell Modules 225
Script Modules 225
Azure PowerShell Module 225
Using the Azure PowerShell Module 226
Creating a Redis Cache Instance using the Service Management mode 226
Creating a Web App and SQL Database Resource Group using the Resource Manager mode 227
PowerShell Desired State Configuration (DSC) 227
PowerShell Desired State Configuration (DSC) 227

Using PowerShell DSC 228
Configuration Management using PowerShell 228
Configuration Management using PowerShell 228
PowerShell DSC and Linux 229
PowerShell DSC and Linux 229
Using PowerShell DSC in Linux 230
PowerShell DSC and Azure Resource Manager 230
PowerShell DSC and Azure Resource Manager 230
Azure Resource Manager DSC Extension 231
DSC and ARM templates with Jeffrey Snover 232
Configuration Management Tooling 232
Configuration Management Utilities 232
Puppet 233
Chef 233
Hyper-V Replica 234
Hyper-V Replica 234
Scenarios 234
Head office and branch office 235
Cloud service provider 235
Windows Server Backup 235
Windows Server Backup 235
Backup and Restore using Windows Server Backup 235
Usage 235
Backup and Restore using Windows Server Backup 236
System Center Data Protection Manager 236
Data deduplication 236
System Center Data Protection Manager 237
Backup Using Data Protection Manager 237
Azure Site Recovery 239
Azure Site Recovery 239
Orchestration using Azure Site Recovery 240
Business Continuity 240
Recovery Plans 241
Executing Recovery Plans 242
Executing Recovery Plans 242
Azure Backup 242

Azure Backup Overview 242
Azure Backup 242
Azure Backup Scenarios 243
StorSimple 244
StorSimple 244
StorSimple: A Hybrid Cloud Storage Solution 244
StorSimple Storage Management 244
Thin provisioning 245
Deduplication and compression 245
Customer Scenario 245
Who is the customer? 245
What does the customer already have? 245
Customer Goal 246
What is the customer's goal? 246
Solution Considerations 246
What does the customer need? 246
What things worry the customer? 247
Common Implementations 247
Backup 247
Virtual Networks 248
VNET Connectivity 248
Call to Action 249
Design the Solution 249
Business Continuity Case Study 249
Sample Solution 249

Application State

Bookmark this page

Application State

Applications often need to store information about transactions, users or even values for variables. At any point in time, the information stored in memory is commonly referred to as an application's state. Over the years, state has grown beyond the simple storage of variable values in memory and now encompasses the storage of complex databases on disk, sophisticated queues in memory or many other storage locations. State represents a single point of stored information that can be accessed by various programs, various threads within a program or even various networked machines.

Application state can easily become a risk when designing your cloud applications for scale. For example, an application that manages user accounts can store information about the currently logged in user to local memory. If this application is scaled out, this creates instance affinity where subsequent requests are required to be routed to the same instance. Design services to be stateless to avoid requiring a series of requests from an application to always be routed to the same instance of a service. Server-side session state management typically requires client affinity (routing each request to the same server instance), which affects the ability of the system to scale.

Another example where state could be a deterrent to cloud scale would be a cloud application where the result of a calculation is stored in a database located on disk. This information needs to be accessed by any user so you can infer that all users are required to access the same instance or machine. If this application was scaled out, there would need to be additional configuration or code written to synchronize the databases on each instance's disk.

Stateless Design

Bookmark this page

Scalability

Scaling allows applications to react to variable load by increasing and decreasing the number of instances of roles, queues, and other services they use. However, the application must be designed with this in mind. For example, the application and the services it uses must be stateless to allow requests to be routed to any instance, and so that the addition or removal of specific instances does not adversely impact current users. Ideally, you should design clients to be stateless with respect to the servers that it uses. However, if the application must maintain session state, store sensitive data or large volumes of per-client data in a distributed storage mechanism such as Azure SQL Database, DocumentDB or HBase so that all instances of the application can access the same data store. Where possible, ensure that the application does not require affinity so that requests can be routed to any instance, and the number of instances is irrelevant. This also avoids the overhead of storing, retrieving, and maintaining state information on the server.

For these reasons, it's common to see application state abstracted away from individual instances. Session state for web application can be stored in state servers that are shared amongst all of the instances. Databases can be stored in a separate virtual machine so that it is universally accessible and the relational data is the same no matter what front-end server instance you access. The applications themselves can be redesigned to be stateless to that requests can be made to any instance indiscriminately and all concepts of state are stored on external machines so that the state is "shared" between all instances.

A truly stateless application can theoretically be horizontally scaled infinitely.

Why Patterns?

Bookmark this page

Why Patterns?

Many common problem areas in Computer Science have been explored and experienced by a wide variety of professionals. As with any discipline, best practices have been conceptualized, proven and prescribed as the most effective or efficient ways to solve common problems. Many professionals rely on these best practices so that they can work more efficiently and focus on obstacles unique to their actual problem space. In software engineering, these best practices are commonly referred to as design patterns.

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is [instead] a description or template for how to solve a problem that can be used in many different situations.[1]

There are many different patterns that already exist in the industry. In this course, we will focus on patterns that are Domain-specific to the cloud. While examples of these patterns may be implemented in Azure or using the .NET framework, most cloud patterns are language-agnostic in their design and can be implemented universally across a wide variety of cloud providers, programming frameworks or operating systems.

---

[1] "Software Design Pattern." Wikipedia. Wikimedia Foundation, 10 June 2015. Web. 11 June 2015

Azure Cloud Application Design and Implementation Guidance

https://github.com/mspnp/azure-guidance

Designing and implementing applications for the cloud brings a unique set of challenges due to the remoteness of the infrastructure and the very nature of distributed services.

Azure provides a comprehensive platform and infrastructure for hosting large-scale web applications and cloud services. However, to be successful, you need to understand how to use the features that Azure provides to support your systems correctly. The purpose of this site is to provide architectural guidance to enable you to build and deploy world-class systems using Azure.

This collection of guidance documents focuses on the essential aspects of architecting systems to make optimal use of Azure, and summarize best practice for building cloud solutions. This collection of documents is hosted on GitHub and is open to contributions from the community.

For more information , you can see:
Azure: https://aka.ms/edx-dev205bx-az34
.NET framework: https://aka.ms/edx-dev205bx-net

MSDN Cloud Design Patterns

Bookmark this page

MSDN Cloud Design Patterns

The patterns & practices team at Microsoft has collected twenty-four design patterns that are relevant when designing the architecture of a cloud application. Each pattern includes a brief discussion of the benefits, considerations and implementation of each pattern. The collection of patterns is not meant to be comprehensive and is instead focused on the most popular design patterns for cloud applications.

The guide consists of a collection of web pages each individually focused on a pattern. The pages are broken down into sections describing the problem domain, a high-level technical solution, how this patterns solves the problem, considerations when using the pattern, an example implementing the pattern and when the pattern is suitable. The patterns included in this guide can be used in many ways including:

- Implement the sample code as a starting point for your own cloud application
- Use the Context, Problem and Solution sections of a pattern's web page as discussion points during an architectural design session
- Use the Example section of a pattern's web page to teach colleagues about a design pattern
- Use the Issues and Considerations section of a pattern's web page to identify common issues in your problem space

The remainder of this module will focus on a subset of patterns from the Cloud Design Patterns documentation and explain why they are important to understand and how they relate to the decisions you will make as an Azure architect.

Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications

https://msdn.microsoft.com/en-us/library/dn568099.aspx

Infographic

http://azure.microsoft.com/en-us/documentation/infographics/cloud-design-patterns/

Zoom-in to see individual patterns.

Performance Concerns with Cloud Applications

Bookmark this page

Performance Concerns with Cloud Applications

Cloud applications typically encounter variable workloads and peaks in activity. Predicting these, especially in a multi-tenant scenario, is almost impossible. Applications should be able to scale out (within limits) to meet peaks in demand and avoid service degradation for any users. Applications should be equally elegant in their scale in operations when demand decreases to minimize cost and resource sprawl.

Scalability is ability of a system either to handle increases in load without impact on performance or for the available resources to be readily increased. Performance is an indication of the responsiveness of a system to execute any action within a given time interval, and this metric is directly affected by scalability. Resource scalability concerns not just compute instances, but other elements such as data storage, messaging infrastructure, and more.

Constantly monitoring performance and scaling a system to adapt to fluctuating workloads to meet capacity targets and optimize operational cost can be a labor-intensive process. It may not be feasible to perform these tasks manually. It is important to use autoscaling features or services to handle both the monitoring of your application's performance and the implementation of scale actions.

Valet Key Pattern

Bookmark this page

Problem: Serving secure data from distributed applications

Client programs and web browsers often need to read and write files or data streams to and from an application’s storage. Typically, the application will handle the movement of the data—either by fetching it from storage and streaming it to the client, or by reading the uploaded stream from the client and storing it in the data store. However, this approach absorbs valuable resources such as compute, memory, and bandwidth.

Data stores have the capability to handle upload and download of data directly, without requiring the application to perform any processing to move this data, but this typically requires the client to have access to the security credentials for the store. While this can be a useful technique to minimize data transfer costs and the requirement to scale out the application, and to maximize performance, it means that the application is no longer able to manage the security of the data. Once the client has a connection to the data store for direct access, the application cannot act as the gatekeeper. It is no longer in control of the process and cannot prevent subsequent uploads or downloads from the data store.

This is not a realistic approach in modern distributed systems that may need to serve untrusted clients. Instead, **applications must be able to securely control access to data in a granular way, but still reduce the load on the server by setting up this connection and then allowing the client to communicate directly with the data store to perform the required read or write operations**.

Solution: Data services and temporary access tokens

To resolve the problem of controlling access to a data store where the store itself cannot manage authentication and authorization of clients, one typical solution is to restrict access to the data store's public connection and provide the client with a key or token that the data store itself can validate.

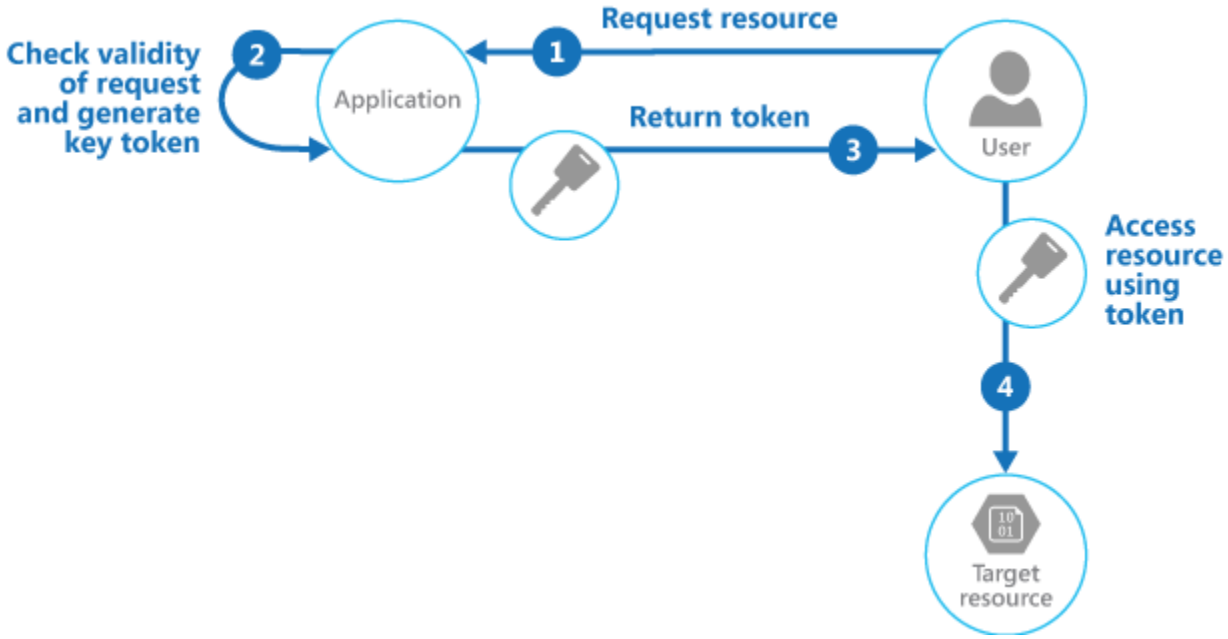
This key or token is usually referred to as a *valet key*. It provides time-limited access to specific resources and allows only predefined operations such as reading and writing to storage or queues, or uploading and downloading in a web browser. Applications can create and issue valet keys to client devices and web browsers quickly and easily, **allowing clients to perform the required operations without requiring the application to directly handle the data transfer**. This removes the processing overhead, and the consequent impact on performance and scalability, from the application and the server.

The client uses this token to access a specific resource in the data store for only a specific period, and with specific restrictions on access permissions.

Valet Key Pattern

1. The client (device, application, etc.) requests a resource. This could be a multimedia file, a record or other stored entity.
2. The application checks the validity of the request. This may include authentication, authorization and ensuring any requirements are met. Once the request is validated, the application generates a key token that gives the client temporary and minimal access to the target resource.
3. The application return the token as the response to the original request from the client.

4. The client accesses the storage resource directly using the token. The storage resource validates the token and can return the resource directly to the client.



EXAMPLE Implementation in Azure

Microsoft Azure supports Shared Access Signatures (SAS) on Azure storage for granular access control to data in blobs, tables, and queues, and for Service Bus queues and topics. An SAS token can be configured to provide specific access rights such as read, write, update, and delete to a specific table; a key range within a table; a queue; a blob; or a blob container. The validity can be a specified time period or with no time limit.

SCENARIO: Distributed image gallery application where users can share images and a mobile client exists. The mobile application connects to a Mobile Services instance and images are stored in an Storage account. The container containing the images is protected against all anonymous access.

1. The mobile application requests an image from a specific user's gallery. The image is a .PNG file.
2. The server application checks that the user is authenticated and valid. The application additionally checks that the user can access the image according to it's business logic. Once validated, the application will go to Azure Storage (where the images are stored) and request a Storage Access Signature (SAS) token granting temporary (timed) access to the individual blob in Azure Storage.
3. The server application responds to the mobile application's request with an HTTP status code indicating success, the URL of the image in Azure Storage and a SAS token.

4. The mobile application can now access the image by appending the SAS token to the end of the URL. This SAS token grants the mobile application temporary access to the image in Azure Storage.

Valet Key Visio File

For more information , you can see:
Microsoft Azure: https://aka.ms/edx-dev205bx-az34
Azure storage: https://aka.ms/edx-dev205bx-az01

Distributing Application Data Across Multiple Instances

Bookmark this page

Distributing Application Data Across Multiple Instances

Most cloud applications and services store and retrieve data as part of their operations. The design of the data stores that an application uses can have a significant bearing on the performance, throughput, and scalability of a system. One technique that is commonly applied in large-scale systems is to divide the data into separate partitions.

Partitioning refers to the physical separation of data for scale. Modern data stores understand that data may be spread across many different instances as the size of the sum data for the application can be larger than any individual physical store can handle in an efficient manner.

Partitioning can improve performance, availability and scalability. Any data access operation will only occur on a smaller subset (volume) of data which in turn ensures that the operation will be more efficient when compared to a query over the entire superset of data for your application. Individual databases can hit physical limits which are also overcome when portioning data across many different individual databases. Partitioning also spreads your data across multiple nodes which can individually be replicated or scaled. When a node is unavailable or being maintained, the application can use replica nodes.

Sharding Pattern

Problem: Hosting large volumes of data in a traditional single-instance store

A data store hosted by a single server may be subject to the following limitations:

- **Storage space**. A data store for a large-scale cloud application may be expected to contain a huge volume of data that could increase significantly over time. A server typically provides only a finite amount of disk storage, but it may be possible to replace existing disks with larger ones, or add further disks to a machine as data volumes grow. However, the system will eventually reach a hard limit whereby it is not possible to easily increase the storage capacity on a given server.
- **Computing resources**. A cloud application may be required to support a large number of concurrent users, each of which run queries that retrieve information from the data store. A single server hosting the data store may not be able to provide the necessary computing power to support this load, resulting in extended response times for users and frequent failures as applications attempting to store and retrieve data time out. It may be possible to add memory or upgrade processors, but the system will reach a limit when it is not possible to increase the compute resources any further.
- **Network bandwidth**. Ultimately, the performance of a data store running on a single server is governed by the rate at which the server can receive requests and send replies. It is possible that the volume of network traffic might exceed the capacity of the network used to connect to the server, resulting in failed requests.
- **Geography**. It may be necessary to store data generated by specific users in the same region as those users for legal, compliance, or performance reasons, or to reduce latency of data access. If the users are dispersed across different countries or regions, it may not be possible to store the entire data for the application in a single data store.

Scaling vertically by adding more disk capacity, processing power, memory, and network connections may postpone the effects of some of these limitations, but it is likely to be only a temporary solution. A commercial cloud application capable of supporting large numbers of users and high volumes of data must be able to scale almost indefinitely, so vertical scaling is not necessarily the best solution.

Solution: Partitioning Data Horizontally Across Many Nodes

Divide the data store into horizontal partitions or *shards*. Each shard has the same schema, but holds its own distinct subset of the data. A shard is a data store in its own right (it can contain the data for many entities of different types), running on a server acting as a storage node.

Sharding physically organizes the data. When an application stores and retrieves data, the sharding logic directs the application to the appropriate shard. This sharding logic may be implemented as part of the data access code in the application, or it could be implemented by the data storage system if it transparently supports sharding.

Abstracting the physical location of the data in the sharding logic provides a high level of control over which shards contain which data, and enables data to migrate between shards without reworking the business logic of an application should the data in the shards need to be redistributed later (for example, if the shards become unbalanced). The tradeoff is the additional data access overhead required in determining the location of each data item as it is retrieved.

To ensure optimal performance and scalability, it is important to split the data in a way that is appropriate for the types of queries the application performs. In many cases, it is unlikely that the sharding scheme will exactly match the requirements of every query. For example, in a multi-tenant system an application may need to retrieve tenant data by using the tenant ID, but it may also need to look up this data based on some other attribute such as the tenant's name or location. To handle these situations, implement a sharding strategy with a shard key that supports the most commonly performed queries.

Sharding Pattern

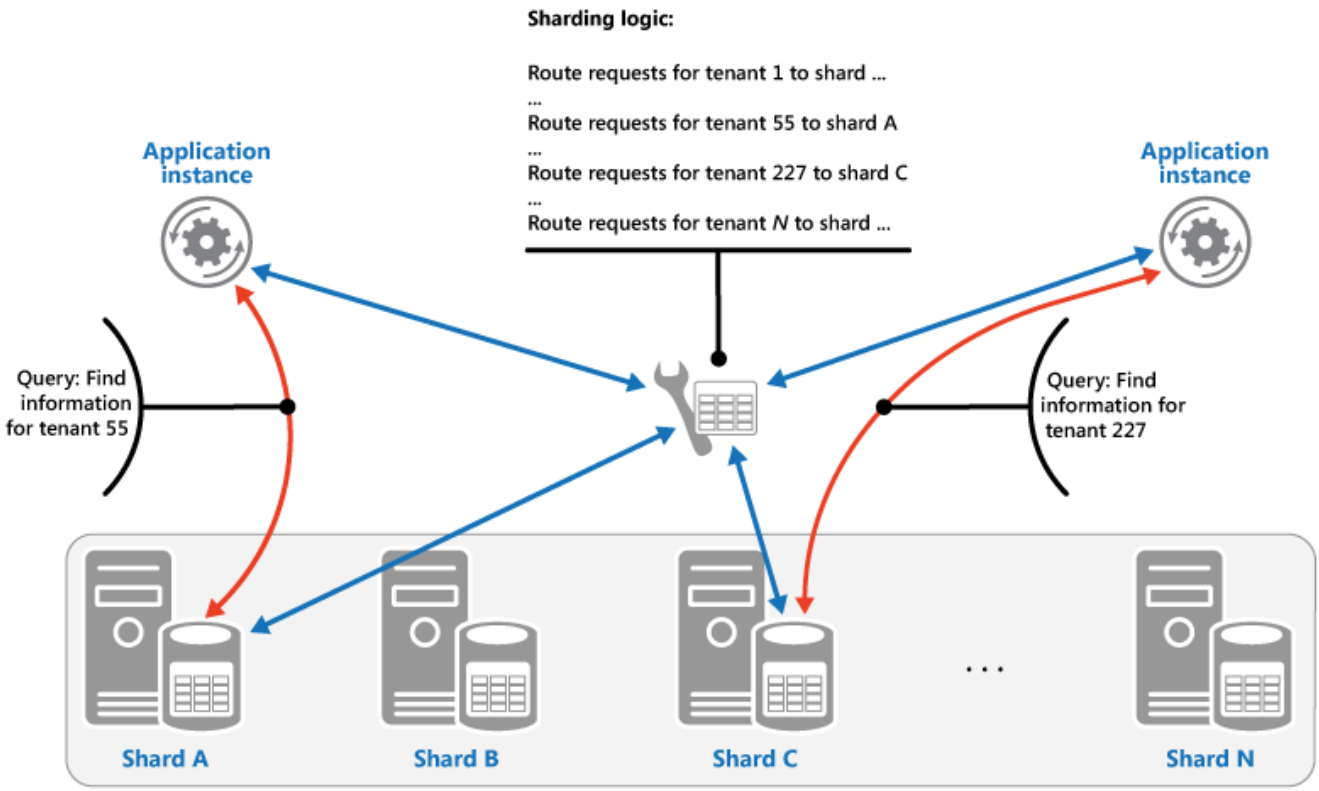
There are various strategies for sharding. This unit will focus on the **lookup** strategy.

Lookup Strategy Implementation

A map is implemented that contains lookup data mapped by shard key. With a multi-tenant application, data using the same shard key will be stored in an identical shard. In this example, the tenant ID is the shard key.

1. An application instance will make a request to the map for the shard which contains tenant #55. The map will return "Shard A".

2. The application instance will then make a request directly to the database at "Shard A" for records related to tenant #55.
3. A new application instance will make a request to the map for tenant #227 and will receive a response of "Shard C".
4. The new application instance makes a request directly to the database at "Shard C" for records related to tenant #227.
5. As new tenants are added and more space is necessary, new shards can be added to the map. Tenant IDs can than be associated with the new shards.



EXAMPLE Implementation in Azure

Microsoft Azure supports Sharding for the Azure SQL Database either by manually creating shards or using the Elastic Scale automated functionality. This example assumes that you are using the Elastic Scale feature.

SCENARIO: Your music catalog application uses elastic scale to spread the large amount of data across many partitions. Each partition is an instance of Azure SQL Database. The artist's unique ID number is used as the shard key.

1. A shard map manager is created and shards are added to the manager. ID ranges for are associated to each shard so that artists are evenly spread across shards (databases). The first three shards are configured in the following manner:

Shard Name	ID range
DataStore_I	[(0-1000)]
DataStore_II	[(1001-2000)]
DataStore_III	[(2001-3000)]
DataStore_IV	[(3001-4000)]

2. There are more shards with similar ranges The application has approximately 40 shards to start.

3. Your application needs a list of albums for an artist. The application uses the shard map manager to get a connection string to the database associated with the shard. The application can now query that database directly for all albums with the artist ID.

4. One of the artists in a shard is getting very popular. This artist has an ID of 1245 and is affecting the resources of other artists in the same shard. Elastic Scale uses the "Split" feature to create two new shards from the original DataStore_II shard. The new DataStore_II shard has a disjoined ID range of [(1001-1244),(1246-2000)]. This means that all IDs from 1001-2000 except 1245 are included in this range. The new DataStore_XLI shard has an ID range of just [1245].

5. The application already uses the smallest database size for two of it's shards, DataStore_IV and DataStore_III. Even with this small size, the databases are not fully utilized. Elastic Scale uses the "Merge" feature to create a new shard from these two shards. The new DataStore_XLII shard has a continuous ID range of [(2001-4000)].

   For more information , you can see:
   Azure SQL Database: https://aka.ms/edx-dev205bx-asd

What is Resiliency?

Bookmark this page

What is Resiliency?

In cloud hosting, many applications are multi-tenant, use shared services, compete for resources and/or bandwidth, communicate over the Internet, and run on commodity hardware. This means there is an increased likelihood that both transient and more permanent faults will arise. The very nature of the internet indicates that it is very likely that latency alone can cause faults before you factor in many of the other mentioned potential reasons for transient faults.

Resiliency is the ability of a system to gracefully handle and recover from failures. Detecting failures, and recovering quickly and efficiently, is necessary to maintain resiliency. Your application must be minimally scalable to a couple of instances so that recovery can occur in case a single or small set of instances encounter an outage. Because of this, understanding scalability is important prior to thinking about and designing for resiliency.

Transient Errors

Bookmark this page

Transient Errors

When an application uses an external service, there is a risk of errors occurring. Temporary conditions related to service infrastructure or network latency can cause unforeseen faults. Services can be intermittent in their performance and cloud applications must be designed with this in mind.

Error conditions that are resolved simply be retrying your request (typically a few milliseconds later) are referred to as transient faults. If your application does not consider transient faults in your design or implementation, these errors can be accidentally handled as fatal exceptions and your application can crash or close rather than attempt the request again.

Outside of unforeseen faults, explicit throttling by a service provider can also cause an error that your application may see as fatal. For example, if your application accesses a storage service which is throttled to 1,000 transactions a second at a rate that is beyond the target, your application may be throttled. An HTTP status message or soft exception (if you are using an SDK) will be returned that indicates that you are throttled and to wait before issuing another request. If your application is not designed with this in mind, your code may handle this as a fatal exception and crash the application rather than wait and continue the processing later.

Retry Pattern

Bookmark this page

Problem: Intermittent Errors with Cloud Services

An application that communicates with elements running in the cloud must be sensitive to the transient faults that can occur in this environment. Such faults include the

momentary loss of network connectivity to components and services, the temporary unavailability of a service, or timeouts that arise when a service is busy.

These faults are typically self-correcting, and if the action that triggered a fault is repeated after a suitable delay it is likely to be successful. For example, a database service that is processing a large number of concurrent requests may implement a throttling strategy that temporarily rejects any further requests until its workload has eased. An application attempting to access the database may fail to connect, but if it tries again after a suitable delay it may succeed.

Solution: Application logic to retry requests that have temporarily failed

In the cloud, transient faults are not uncommon and an application should be designed to handle them elegantly and transparently, minimizing the effects that such faults might have on the business tasks that the application is performing.

If an application detects a failure when it attempts to send a request to a remote service, it can handle the failure by retrying the application logic after a short wait. For the more common transient failures, the period between retries should be chosen so as to spread requests from multiple instances of the application as evenly as possible. This can reduce the chance of a busy service continuing to be overloaded. If many instances of an application are continually bombarding a service with retry requests, it may take the service longer to recover.

If the request still fails, the application can wait again and make another attempt. There should be a limit on attempts to avoid sending endless requests to a service that may actually be completely inoperable. All code that access the remote service should be implemented using a retry policy such as the one described here.

Retry Pattern

1. Application makes an HTTP requests to an external service. The request fails and the service host responds with HTTP status code 429 (Too Many Requests). This typically indicates that your requests are throttled.
2. The application waits for a short time period and tries again. This request fails again with the same HTTP status code.
3. The application waits for the same time period and then tries another time. The request succeeds with HTTP status code 200 (OK). This indicates that the retry logic can end and the request was successful.

Example Implementation in Azure

Many Azure client libraries already implement the Retry pattern. If you are using the Azure SDK for your preferred language, retry policies are implemented in a transparent manner so that you do not have to change the logic in your existing application.

Types of Retry Patterns

Bookmark this page

Retry Policy

It is not simply enough to retry your application's logic on an endless loop. Typically there is a substantial amount of logic involved when implementing the Retry Pattern. This logic is normally referred to as the **Retry Policy**.

Different Retry Policies can be used for each service or request you may make. Retry policies can be tuned and modified as your learn more about your application and the hosted service's behavior.

Policy Components

A policy can consist of components that deal with many scenarios. The first major component deals with reading an HTTP response status code and determining an appropriate action

- If the status code indicates a fatal error or that the service call would remain unsuccessful if the request is repeated, than the retry policy will not take effect and requests will not be repeated. Examples include:
- **[401 - Unauthorized]** This authorization exception indicates that the application has provided invalid credentials. Even if repeated, this exception is likely to remain the same.
- **[400 - Bad Request]** This exception indicates that the application has provided a request that does not meet the syntax for a valid request. This typically means that the application code must be changed to only issue valid requests.

- **[404 - Not Found]** This exception indicates that the request was made to a resource that could not be found. This is typically due to a malformed URL and would not be resolved by repeating the request.
- The status code can indicate a common failure due to connectivity or due to the hosted service being busy. These errors can typically be solved by waiting a short period and then making additional requests.
- **[503 - Service Unavailable]** This exception can occur when the service is unavailable due to any number of issues. Many times this exception occurs because the service is overloaded.
- **[429 - Too Many Requests]** This exception indicates that your application is being throttled. You should wait a short period before issuing additional requests.
- **[408 - Request Timeout]** This exception indicates that your client application did not issue a requests within a certain time period. You can typically repeat this request safely.
- The status code can also be an unusual code that you have not planned for. In this case you should retry your request just like you would with a common failure.
- **[520 - Unknown Error]** This exception is returned when there is not another status that accurately describes the issue.

Escalating Wait

A highly aggressive retry policy with minimal delay between attempts, and a large number of retries, could further degrade a busy service that is running close to or at capacity. This retry policy could also affect the responsiveness of the application if it is continually attempting to perform a failing operation rather than doing useful work. For the more common transient failures, the period between retries should be chosen so as to spread requests from multiple instances of the application as evenly as possible. This can reduce the chance of a busy service continuing to be overloaded. If many instances of an application are continually bombarding a service with retry requests, it may take the service longer to recover.

If the request still fails, the application can wait for a further period and make another attempt. If necessary, this process can be repeated with increasing delays between retry attempts until some maximum number of requests have been attempted and failed. The delay time can be increased incrementally, or a timing strategy such as exponential back-off can be used, depending on the nature of the failure and the likelihood that it will be corrected during this time.

If a request still fails after a significant number of retries, it may be better for the application to prevent further requests going to the same resource for a period and simply report a failure immediately.

Asynchronous Messaging

Bookmark this page

Asynchronous Messaging

Many software curriculums teach that separating your application into smaller modules will make it easier to manage the application in the long term. Modules can be swapped, modified and updated without having to update the entire application. Partitioning your workloads into modules also carries another benefit of allowing each logic center in your application to scale in isolation.

If you have a web application that allows people to upload images for processing, your image processing module can become CPU intensive and easily account for the majority of your CPU time and disk usage. By separating the image processing module out to another distinct server (or set of servers), you can scale this module in isolation without having to modify, scale or change the module that serves the web pages. It then becomes very important to figure out how to communicate between these modules.

Messaging is a key strategy employed in many distributed environments such as the cloud. It enables applications and services to communicate and cooperate, and can help to build scalable and resilient solutions. Messaging supports asynchronous operations, enabling you to decouple a process that consumes a service from the process that implements the service.

Message Queues

Bookmark this page

Message Queues

Asynchronous messaging in the cloud is usually implemented by using message queues. Regardless of the technology used to implement them, most message queues support three fundamental operations:

- A sender can post a message to the queue.
- A receiver can retrieve a message from the queue (the message is removed from the queue).
- A receiver can examine (or *peek*) the next available message in the queue (the message is not removed from the queue).

Conceptually, you can think of a message queue as a buffer that supports send and receive operations. A sender constructs a message in an agreed format, and posts the message to a queue. A receiver retrieves the message from the queue and processes it. If a receiver attempts to retrieve a message from an empty queue, the receiver may be blocked until a new message arrives on that queue. Many message queues enable a receiver to query the current length of a queue, or peek to see whether one or messages are available, enabling the receiver to avoid being blocked if the queue is empty.

Many modern queue systems also support transactions, visibility and leasing on top of the standard queue operations.

Competing Consumers Pattern

Bookmark this page

Problem: Handling variable quantities of requests

An application running in the cloud may be expected to handle a large number of requests. The number of requests could vary significantly over time for many reasons. A sudden burst in user activity or aggregated requests coming from multiple tenants may cause unpredictable workload. At peak hours a system might need to process many hundreds of requests per second, while at other times the number could be very small. Additionally, the nature of the work performed to handle these requests might be highly variable.

Using a single instance of the consumer service might cause that instance to become flooded with requests or the messaging system may be overloaded by an influx of messages coming from the application.

Solution: Asynchronous messaging with variable quantities of message producers and consumers

Rather than process each request synchronously, a common technique is for the application to pass them through a messaging system to another service (a *consumer* service) that handles them asynchronously. This strategy helps to ensure that the business logic in the application is not blocked while the requests are being processed.

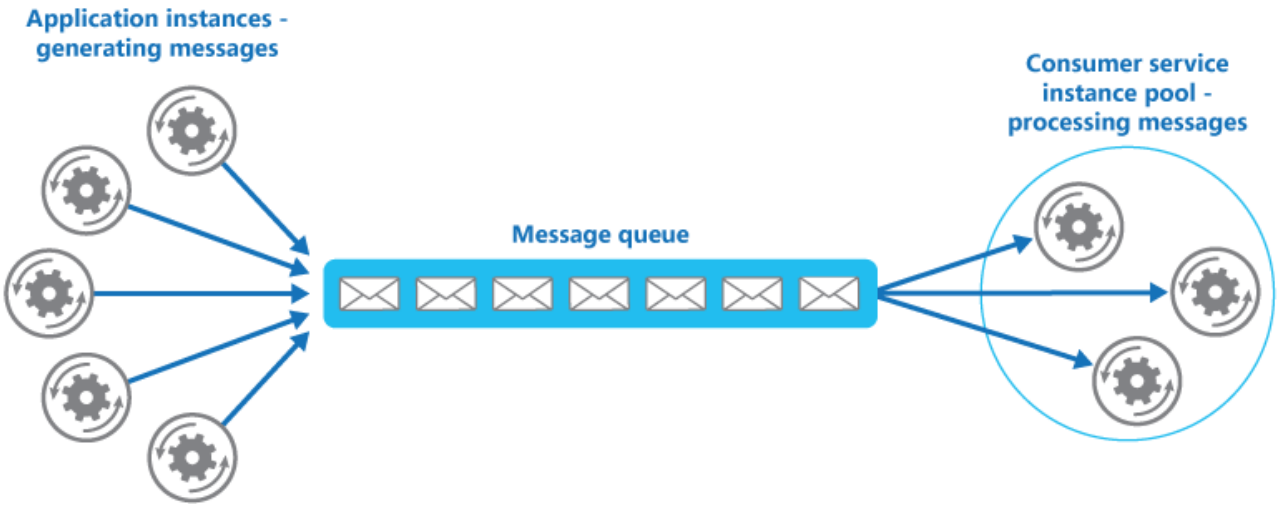
A message queue can be used to implement the communication channel between the application and the instances of the consumer service. To handle fluctuating workloads, the system can run multiple instances of the consumer service. The application posts requests in the form of messages to the queue, and the consumer service instances receive messages from the queue and process them. This approach enables the same pool of consumer service instances to handle messages from any instance of the application.

Competing Consumers Pattern

1. Application instances (**producers**) generate messages to be placed in the queue.

2. Service instances (**consumers**) poll the queue to see if any messages are waiting to be processed.
3. The service instance that receives the message will process the message and then flag the message as processed in the queue

- If the service instance fails to process the message, the queue will eventually make the message available to other instances after a period of time.



Example Implementation in Azure

In Azure, the competing consumers pattern can be easily implemented using either Storage Queues or Service Bus Queues. The queues are compared and contrasted in this MSDN article: https://msdn.microsoft.com/en-us/library/azure/hh767287.aspx

SCENARIO: An Azure API App is hosted to collect telemetry data from various Windows 10 IoT devices (Rasperry Pi II). An Azure Storage Queue is used to communicate between the API App which uses Node.js and Express for the front-end and a processing module in Ruby that collects the telemetry data and performs complex calculations based on the data.

1. The IoT device sends an HTTP request to the distributed Azure API App instances with temperature, barometric pressure and humidity information collected throughout the day.
2. The individual application instance that receives the request uses an HTTP request to add a message to the Storage Queue with the weather metadata included as part of the message body.
3. The consumer services uses the Azure SDK for Ruby to poll the queue to see if any messages are available.
4. One of the consumer services' instances will receive the previously generated message and can now process the message. If the message has been successfully processed, the

instance will use the SDK to mark the message as deleted in the queue. If the instance crashes or times out, the queue message will eventually be available to other instances after a visibility timeout period has elapsed.

For more information , you can see:
Azure: https://aka.ms/edx-dev205bx-az34
Azure SDK: https://aka.ms/edx-dev205bx-az02
Storage Queues: https://aka.ms/edx-dev205bx-az03
Service Bus: https://aka.ms/edx-dev205bx-az04
Azure API App: https://aka.ms/edx-dev205bx-az05
Windows 10 IoT: https://aka.ms/edx-dev205bx-wiot

Caching Data for a Distributed Application

Bookmark this page

Why Cache?

Caching is a common technique that aims to improve the performance and scalability of a system by temporarily copying frequently accessed data to fast storage located close to the application. Caching is most effective when an application instance repeatedly reads the same data, especially if the original data store is slow relative to the speed of the cache, is subject to a high level of contention, or is far away when network latency can cause access to be slow.

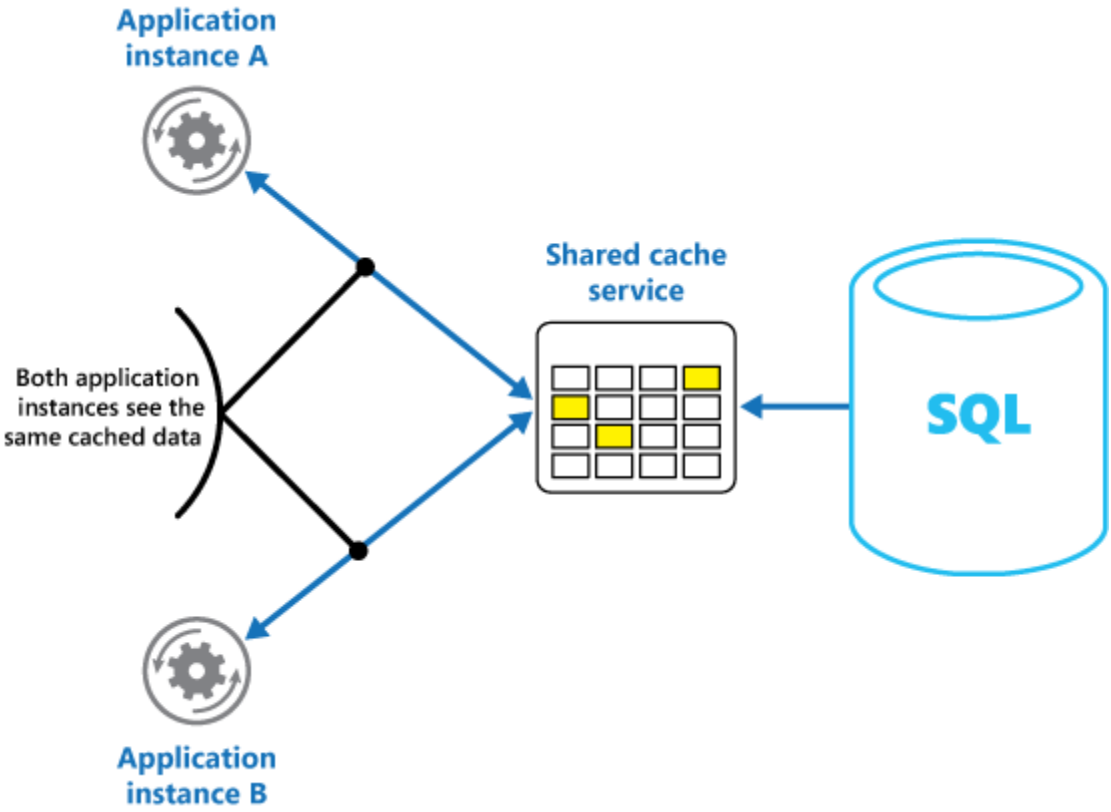
There are two primary types of cache:

- In-memory cache
- Shared cache

Caching Data for a Distributed Application

While in-memory cache can be far superior to not having a cache at all, in-memory cache introduces complexities when you try to scale an application. For example, a web application for an online game may store leaderboard information in cache. If you scale the web application to three different instance, you will essentially have three different caches. Each cache is not guaranteed to have the same information which can cause perceived problems if the user refreshes the page and the load balancer sends the user to a different application instance. You also lose some of the performance benefits of caching when each application instance must separately pull data from the slower data store and store it in it's own in-memory cache.

Using a shared cache can help to alleviate the concern that data may differ in each cache, as can occur with in-memory caching. Shared caching ensures that different application instances see the same view of cached data by locating the cache in a separate location, typically hosted as part of a separate service. An important benefit of using the shared caching approach is the scalability it can provide. An application instance simply sends a request to the cache service regardless of how many instances you have of your application.



Shared cache is typically slower than in-memory cache as the cache is hosted as a service external to your application instance.

Cache-Aside Pattern

Bookmark this page

Problem: Cached data consistency

Applications use a cache to optimize repeated access to information held in a data store. However, it is usually impractical to expect that cached data will always be completely consistent with the data in the data store. Applications developers should consider a strategy that helps to ensure that the data in the cache is up to date as far as possible, but can also detect and handle situations that arise when the data in the cache has become stale.

Solution: Read/Write-Through Caching

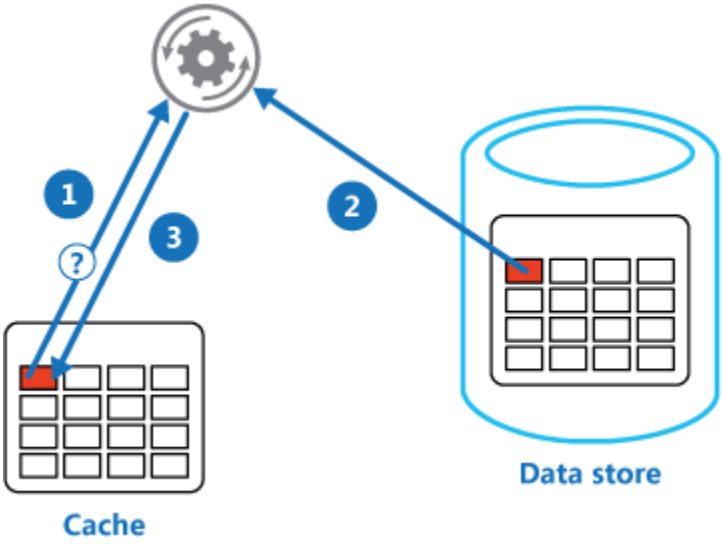
Many commercial caching systems provide read-through and write-through/write-behind operations. In these systems, an application retrieves data by referencing the cache. If the data is not in the cache, it is transparently retrieved from the data store and added to the cache. Any modifications to data held in the cache are automatically written back to the data store as well.

For caches that do not provide this functionality, it is the responsibility of the applications that use the cache to maintain the data in the cache. An application can emulate the functionality of read-through caching by implementing the cache-aside

strategy. This strategy effectively loads data into the cache on demand if it's not already available in the cache.

Cache-Aside Pattern

1. Determine whether the item is currently stored in cache.
2. If the item is not currently stored in ache, retrieve the item from the source data store.
3. Store a copy of the item in the cache.



Example Implementation in Azure

This pattern can be used with a variety of cache mechanisms. In Azure, the most popular cache mechanism is Azure Redis Cache. Redis Cache is a key-value store that is wildly popular for caching in many web applications. You can learn more about Redis Cache here: http://redis.io/

SCENARIO: Your online game web application shows the featured player on the homepage. Since the homepage is accessed often, it is important to cache this value. Azure Redis Cache is used for the cache and Document DB is used for the data store. The featured player is typically updated using a separate application where the admins can specify a new featured player.

1. When the web application loads for the first time it makes a request (**GET**) to the Redis Cache instance for the name of the featured player using the key: **player:featured**
2. The cache will return a **nil** value indicating that there is not a corresponding value stored in the cache for this key.
3. The web application can then go to the Document DB data store and gets the name of the featured player.

4. The name of the featured player will be stored in the Redis Cache using a request (**SET**) and the key **player:featured**.
5. The web application returns the name of the featured player and displays it on the homepage.
6. Subsequent requests for the home page will use the cached version of the value as Redis Cache will successfully return a value for the name of the featured player.
7. If an administrator updates the featured player, the web application will replace the value in both Redis Cache and Document DB to maintain consistency.

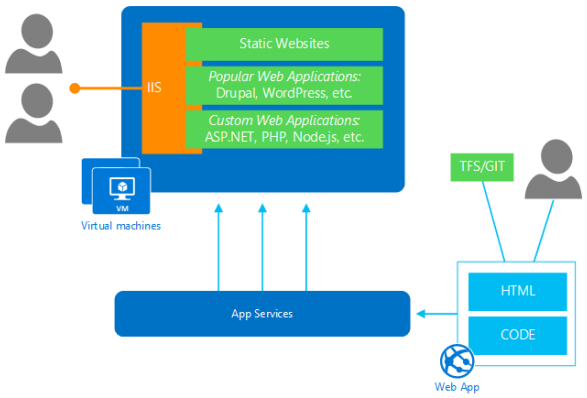
   For more information , you can see:
   Azure Redis Cache: https://aka.ms/edx-dev205bx-az33
   Azure Document DB: https://aka.ms/edx-dev205bx-az06

App Services and Web Apps

Bookmark this page

Azure App Service

Azure App Service is the integration of multiple components to support connectivity between application components that communicate with many different client devices, browsers or B2B endpoints. App service is a fully-managed Platform-as-a-Service (PaaS) offering with minimal administrative overhead and tight integration with Azure's existing DevOps tooling.



The Azure App Service consists of the following services:

- **Web Apps**: Web application hosting that is scalable to your resource needs and compatible with enterprise virtual networks.
- **Mobile Apps**: Dynamic mobile application data hosting integrated with authentication, push notifications and custom endpoints.

- **API Apps**: API application hosting that uses open-source frameworks for API metadata and connectivity.
- **Logic Apps**: Automated business process workflows that can integrate existing SaaS applications or your custom API Apps from on-premise or the cloud.

Web Apps

Azure Web Apps are a managed hosting platform for web applications with minimal friction or configuration. Without any extra customization, a Web App instance can host web applications written in many languages including:

- ASP.NET
- Ruby
- Python
- Java

Web Apps also support extended features that will be discussed in this module including:

- Extensions
- Live Debugging and Live Log Tracing
- Continous Deployment
- WebDeploy Deployment
- Local Git Repository
- Web Jobs
- Backup/Restore
- Autoscale

App Service Plans

App Service Plans represent an assignment of features and capacity for multiple services within App Service. Each Mobile, Logic, Web or API App must be associated with an App Service Plan which dictates both the features available, capacity available and billing for the services. The App Service Plans are grouped into the following tiers:

- Free
- Shared
- Basic (reserved VM)
- Standard (reserved VM)

The tier for an App Service Plan can be changed at any time.

For more information , you can see:
Azure App Service: https://aka.ms/edx-dev205bx-az07
Azure Web Apps: https://aka.ms/edx-dev205bx-az08
ASP.NET: https://aka.ms/edx-dev205bx-asp

Scaling Web Apps

Bookmark this page

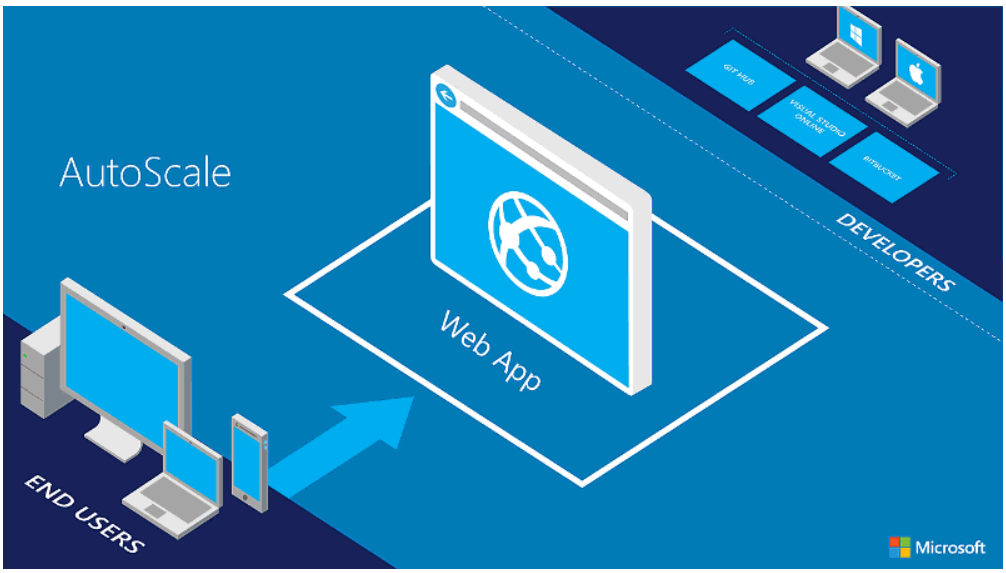
Scaling Web Apps

Web Apps can be scaled in both the horizontal and vertical directions:

- **Horizontal**: Web Apps can be scaled to multiple instances by either manually changing the instance count or by using an autoscale algorithm in the App Service Plan. By default a Standard App Service Plan has 2 instances implemented using autoscale and a metric that measures CPU Percentage.
- **Vertical**: Web Apps can be scaled to have more memory or CPU time by change the App Service Plan tier. For example, the Standard S1 App Service Plan has 1 reserved CPU core and 1.75 GB of RAM. If you need 4 cores and 7 GB of RAM you can scale up to the Standard S3 Service Plan.

Auto-scale

Auto-scale allows your Web App to respond to changes in user demand and scale your application up or down appropriately to gracefully handle the additional load.



By default you can manually scale the number of instances for your Web App by setting the **Instance** count. You can use auto-scale by first defining a schedule. For each

schedule you can configure metrics to be measured and scale actions that should occur based on the metric.

For example, you may create a schedule that encompasses the time from Thursday evening to Sunday morning. You have determined by monitoring your application that this is the highest point of usage for your Web App. Within this schedule you can create a metric to measure any of the following metrics:

- HTTP Queue Length
- Disk Queue Length
- Data In/Out (Bandwidth)
- Memory Percentage
- CPU Percentage

Example

You start by setting the Instance Range to a minimum of 4 and a maximum of 9. Using the CPU Percentage metric, you have determined that if the average CPU percentage is greater than 85% for a duration of 5 minutes that you want to add two more instances to your set of current instances. You have also specified that if the CPU Percentage is less than or equal to 65% for a duration of 15 minutes that you want to remove one instance at a time from the set of current instances. You can also specify a "Cool Down" period where there is wait after any scale operation before any other scale operations can occur.

Using a combination of schedules and metrics you can configure a wide possibility of auto-scale scenarios.

Deploying Web Apps

Deploying Web Apps

Since the Web App service is fully managed, you must use one of the supported methods to deploy web applications to a Web App instance. These methods include:

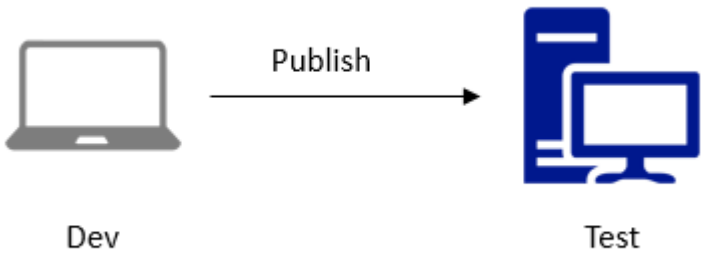
- FTP
- WebDeploy
- Manual Upload to **wwwroot** folder
- Visual Studio Monaco Site Extension
- Continous Deployment from Source Control or Storage Providers
- GitHub

- Visual Studio Online
- BitBucket
- DropBox
- External Git Repository
- Local Git Repository
- Mercurial

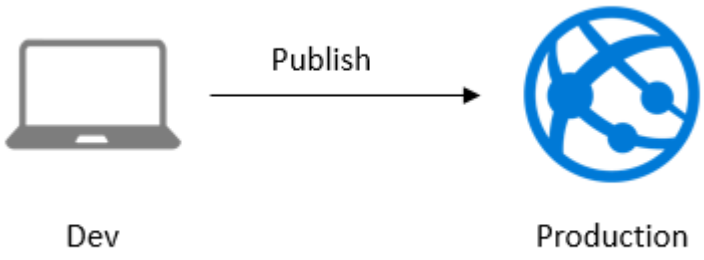
Throughout this module we will discuss many of the various options for deployment

Package Deployment

Many developers still deploy applications directly to their environments. This type of deployment typically requires administrative access to a server.



This deployment technique can work fine for test and staging environments but typically causes issues when deploying to production. In order to deploy to a production instance, this requires the development lead or application manager to have direct administrative access to the target web server.

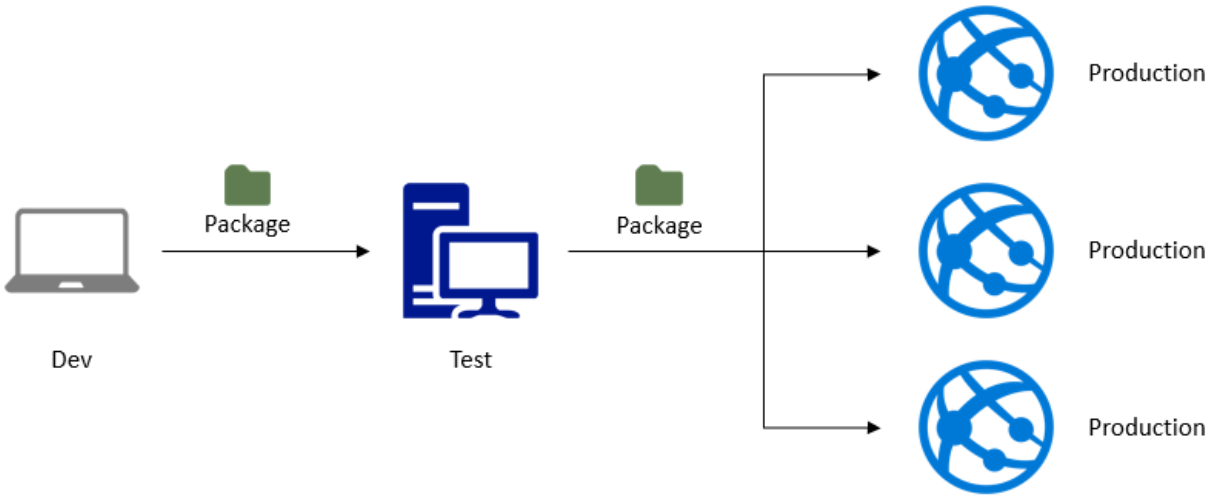


In most organizations, you do not want anyone to have direct administrative access to any other your cloud resources other than the service administrators. To accomplish this you can use a package deployment. A package is created of the web application in the development environment. The developer team can then hand the package off to

the operations team who will install the package on either the test environment or production environments.

WebDeploy is one of the most popular package formats available in Internet Information Services (IIS) environments. You can learn more about WebDeploy here: http://www.iis.net/downloads/microsoft/web-deploy.

In lieu of an operations team, a remote endpoint can be created in an on premise environment that allows anyone with the appropriate credentials (typically username and password) to deploy a WebDeploy package directly. This endpoint can be advantageous because it allows individuals in your organization to deploy a package without having direct administrative or Remote Desktop Protocol (RDP) access to the web server. Azure Web Apps is a managed service with no option for administrative or direct access to the web server hosting your web application[s]. Web Apps supports the WebDeploy remote deployment endpoint out-of-the-box so that you can easily deploy WebDeploy packages to an Azure Web App instance.



For more information , you can see:
WebDeploy: https://aka.ms/edx-dev205bx-wd
Visual Studio: https://aka.ms/edx-dev205bx-vs
Internet Information Services: https://aka.ms/edx-dev205bx-iis

Web Deploy Packaging and Deployment

Bookmark this page

Web Deploy

Web Deploy allows you to package configuration and content of your installed Web applications, including databases, and use the packages for storage or redeployment. These packages can be deployed using IIS Manager, Visual Studio, PowerShell or a wide variety of IDEs without requiring administrative privileges to the destination server. Web Deploy packages can also contain advanced features such as Access

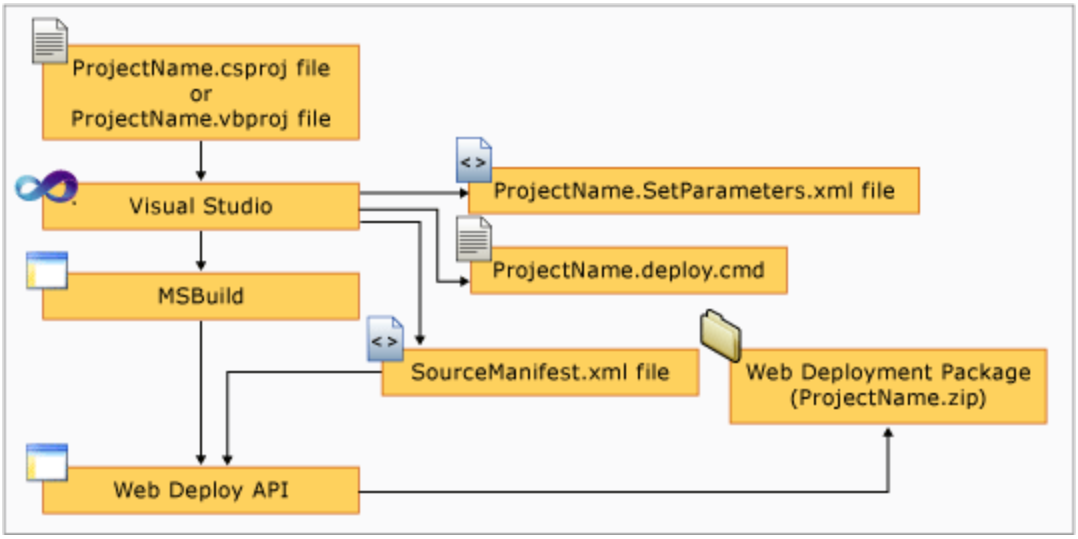
Control Lists (ACL) configurations, registry modifications or Global Assembly Cache (GAC) installations

Web Deploy is used in contexts outside of Azure. Web Deploy is used on premise by administrators to move applications from one IIS server to another (Ex. Test to Staging) or to distribute a web application to nodes of a server farm. Web Deploy packages are also used with the Microsoft Web Platform Installer (Web PI) to install applications from the Web Application Gallery. Web Deploy can be installed on your lcocal machine by using Web PI and it is included as part of the Visual Studio installer.

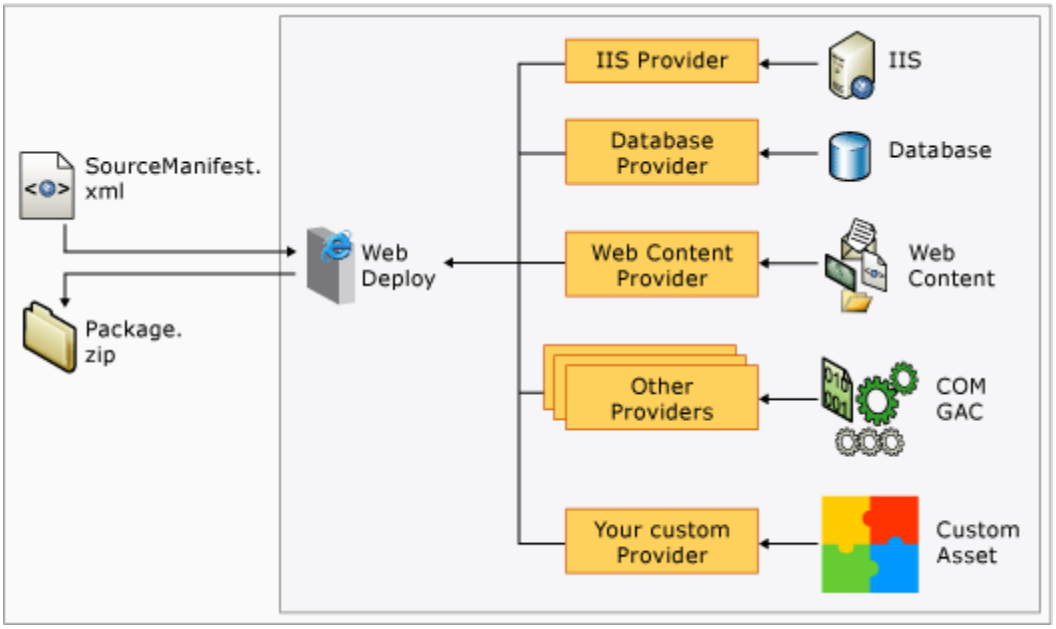
Using Web Deploy with Visual Studio and ASP.NET

https://msdn.microsoft.com/en-us/library/dd394698(v=vs.110).aspx

Visual Studio uses the Web Deploy API to create a Web Deployment Package for you to use in your deployments.



This package contains a comprehensive representation of your application including application content (binaries), IIS configuration settings, database configuration, COM/GAC modifications and even custom components.



For more information , you can see:
IIS Manager: https://aka.ms/edx-dev205bx-iis02
PowerShell: https://aka.ms/edx-dev205bx-ps
Microsoft Web Platform Installer: https://aka.ms/edx-dev205bx-wpi

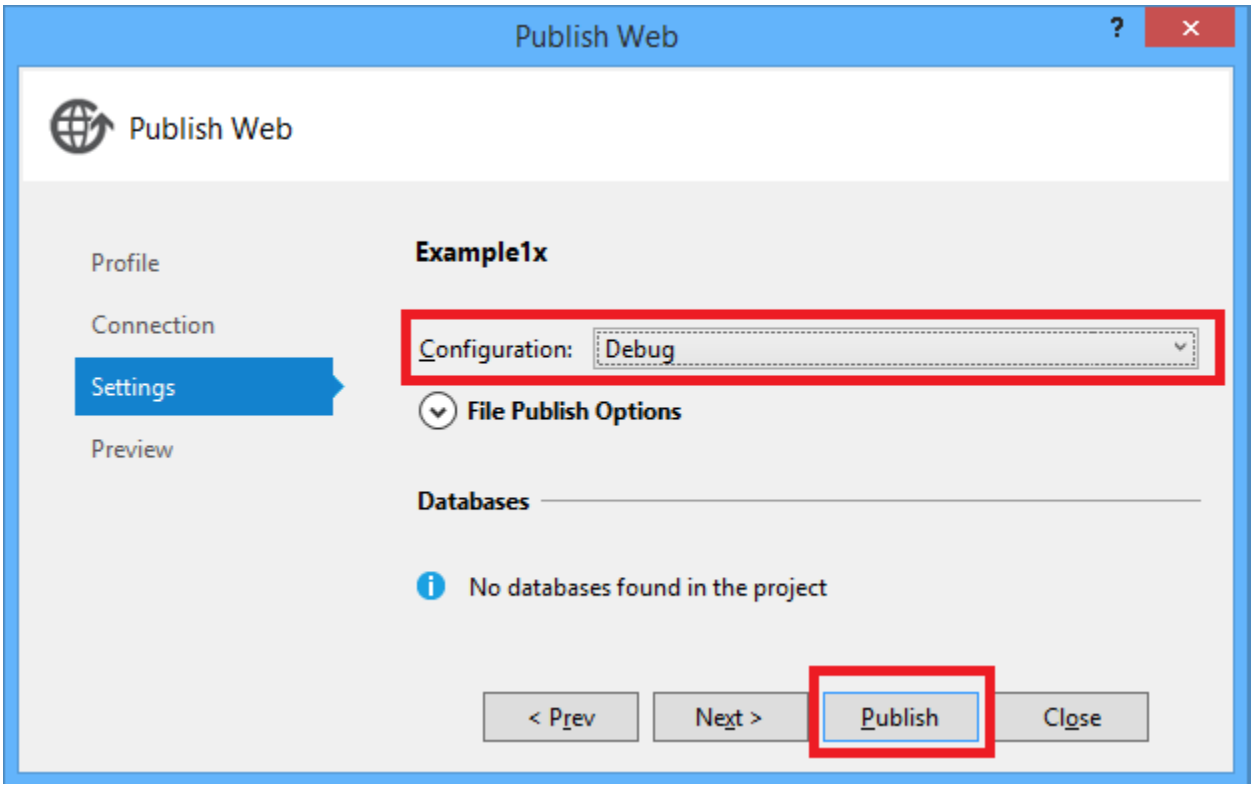
Debugging and Monitoring Web Apps

Bookmark this page

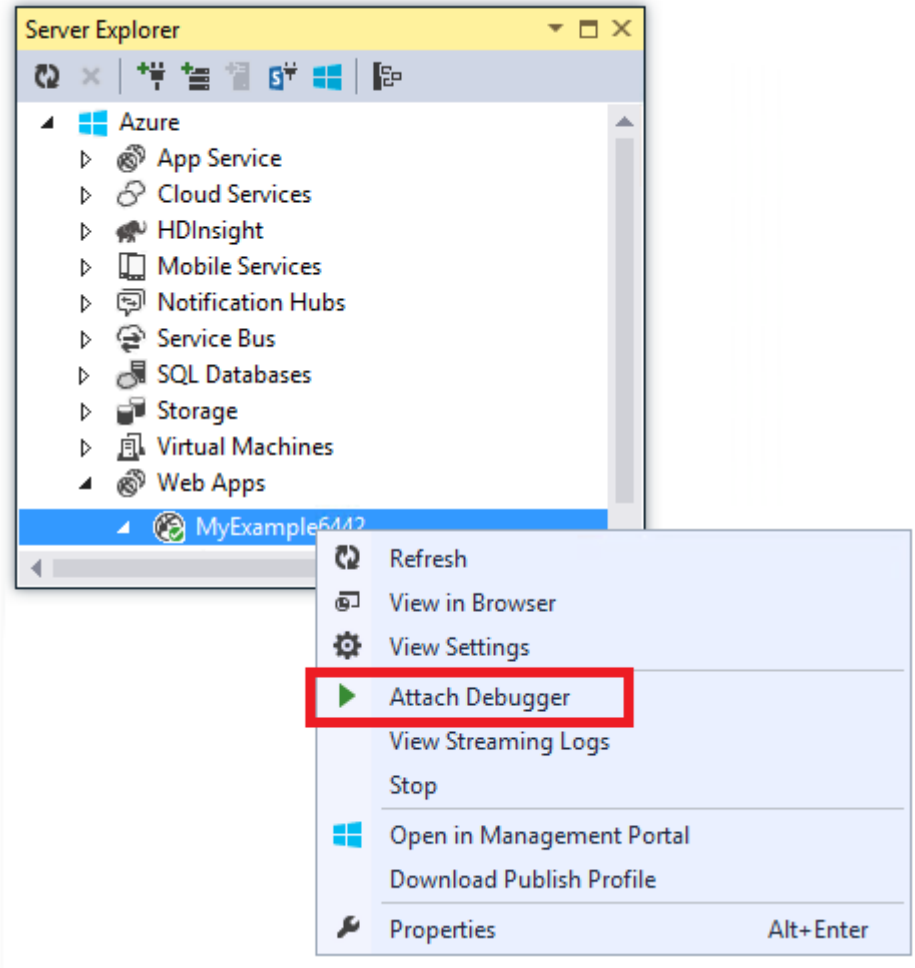
Remote Debugging

Both web applications and web jobs in Azure Web Apps can be remotely debugged using Visual Studio. You can set breakpoints, manipulate memory directly, step through code, and even change the code path much like applications that you are debugging locally.

Remote debugging can occur on web applications even if they were built with a build configuration that include Release optimizations and excludes symbols. Much like local applications, the debug experience would be subpar if you choose to debug an application with an optimized build. For this reason, many choose to use a Debug build of their web application when remotely debugging. To do this, you must first publish a version of your web application using the **Debug** build configuration instead of the **Release** configuration.



Once you have published the debug build of your web application, you can use the Server Explorer in Visual Studio to remotely attach the debugger to your Web App instance.



The debugger will then attach to an instance of your Web App and you can debug your application using live breakpoints and other Visual Studio debug features. It is important to note that a build configuration that includes symbols and is not optimized will have increased overhead reducing application performance as a whole. Once you are finished debugging, it is generally recommended to publish a release build of your web application.

BEST PRACTICE: To avoid having to republish your application, you can always create a new Web App (or a new Web App slot) and publish the debug configuration directly to that isolated instance.

Viewing Logs

Web applications running in the Azure App Service environment can create a wide variety of logs including:

- **Application tracing logs**: The application creates these logs directly by using an in-memory trace class.
- **Web server logs**: The web server creates a log entry for every HTTP request to the web app.

- **Detailed error message logs**: The web server creates an HTML page with some additional information for failed HTTP requests (those that result in status code 400 or greater).
- **Failed request tracing logs**: The web server creates an XML file with detailed tracing information for failed HTTP requests. The web server also provides an XSL file to format the XML in a browser.

  Logging does have an impact on performance as it causes some overhead. For this reason, Web Apps in Azure can be configured to enable or disable each of these types of logs. For Application logs specifically, you can refine further by only allowing logs of certain severity to be stored.

  Log files are stored in the file system of your Web App instance and can be accessed using FTP. You can alternatively replicate your logs to an Azure Storage account for long-term storage.

  BEST PRACTICE: It is typically preferable to replicate logs to a Storage account as logs are only retained in the file system for a short duration. If you do replicate logs to a Storage account, Storage Blobs are ideal as they can store large quantities of logs that can be downloaded as a complete set with minimal performance impact. No-SQL stores such as Storage Tables are not as performant when you need to download all files across partitions.

  For more information , you can see:
  Azure Web Apps: https://aka.ms/edx-dev205bx-az08
  Visual Studio: https://aka.ms/edx-dev205bx-vs
  Azure App Service: https://aka.ms/edx-dev205bx-az07

Managing Deployment Credentials

Bookmark this page

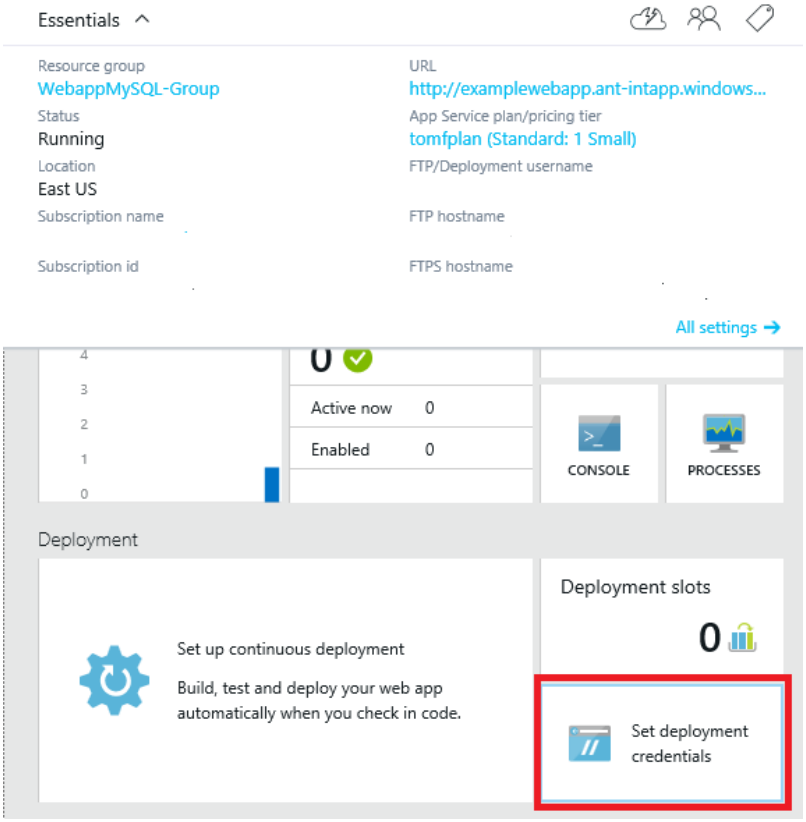
Managing Deployment Credentials

When deploying using FTP or the local Git repository, you must first select your deployment credentials. This can be accomplished using the Portal.

1. Navigate to the blade for your Web App instance.


RUNNING
myexample28587
WEBSITE
Settings
Browse
Start
Stop
Swap
Restart
Delete
Get publish profile
Essentials
Resource group
Default-Web-WestUS
Status
Running
Location
West US
Subscription name
Windows Azure MSDN - Visual Studio Ulti...
Subscription id
aeb4ae60-b7cb-4f3d-966d-fa43b6607f30
URL
http://myexample28587.azurewebsites.net
Web hosting plan/pricing tier
Default0 (Free)
FTP/Deployment username
MyExample28587\td15426
FTP hostname
ftp://waws-prod-bay-003.ftp.azurewebsite...
FTPS hostname
ftps://waws-prod-bay-003.ftp.azurewebsit...
All settings
Monitoring
Requests and errors
20
15
10
5
0
06 PM
THU 05
06 AM
12 PM
REQUESTS
16
HTTP SERVER ERRORS
0
Analytics
Answers and insights to improve your website—and your business.
02 PM
02:30
Application monitoring
MYEXAMPLE28587
APPLICATION
RESPONSE TIME
ERROR RATE
Click here to collect insights into the performance of your .NET applications.
To run webtests, upgrade to a basic or standard...
Usage
File System Storage
DEFAULT0
13.83%
Quotas
DEFAULT0
CPU Time 3m/1h
Data Out 11MB/165MB
Memory ... 176MB/1GB
Scale
DEFAULT0
Autoscale Off
Instances 1
Sites 7
Estimated spend
MYEXAMPLE28587
$0.00
Pricing tier
DEFAULT0
F1
Shared Infrastructure

2. Locate the **Deployment** section and then located the **Deployment Credentials** tile.



3. Click the Deployment Credentials tile and then set a username and password combination. Once saved you can immediately use these credentials to deploy code to the Web App instance using either FTP or a local Git repository.

Site Extensions

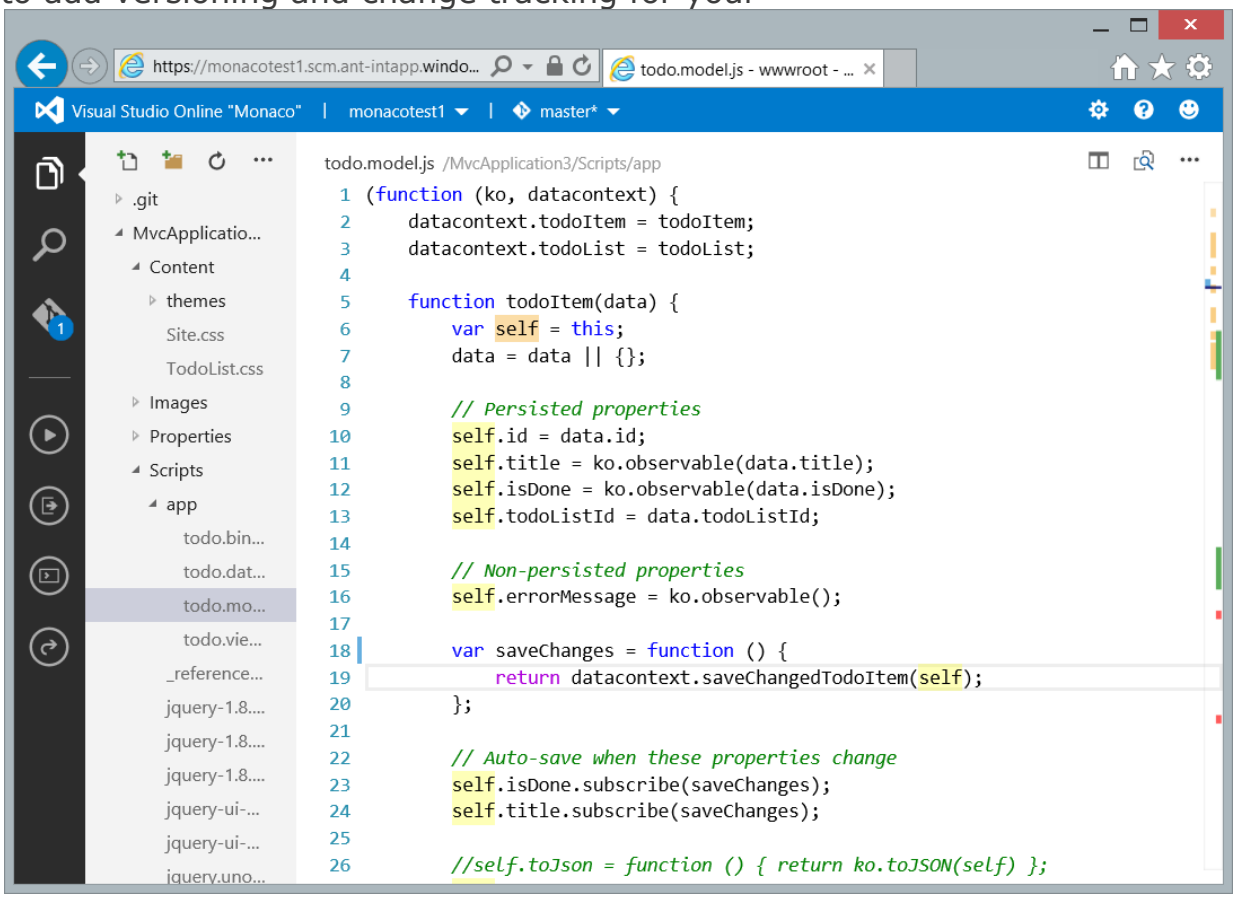
Bookmark this page

Site Extensions

Each Azure Web App instance contains a management endpoint that is hosted by KUDU to allow for a wide variety of extensibility scenarios. These scenarios range from adding new features, providing unique editing experiences to features such as log browsing and image optimization. There are a variety of site extensions that are available in the Azure Marketplace including:

- **Visual Studio Monaco**: Visual Studio Monaco is a source code editor designed to work entirely within the browser. Visual Studio Monaco allows you to edit your web application's source files directly in the live web application. You can also create a Git

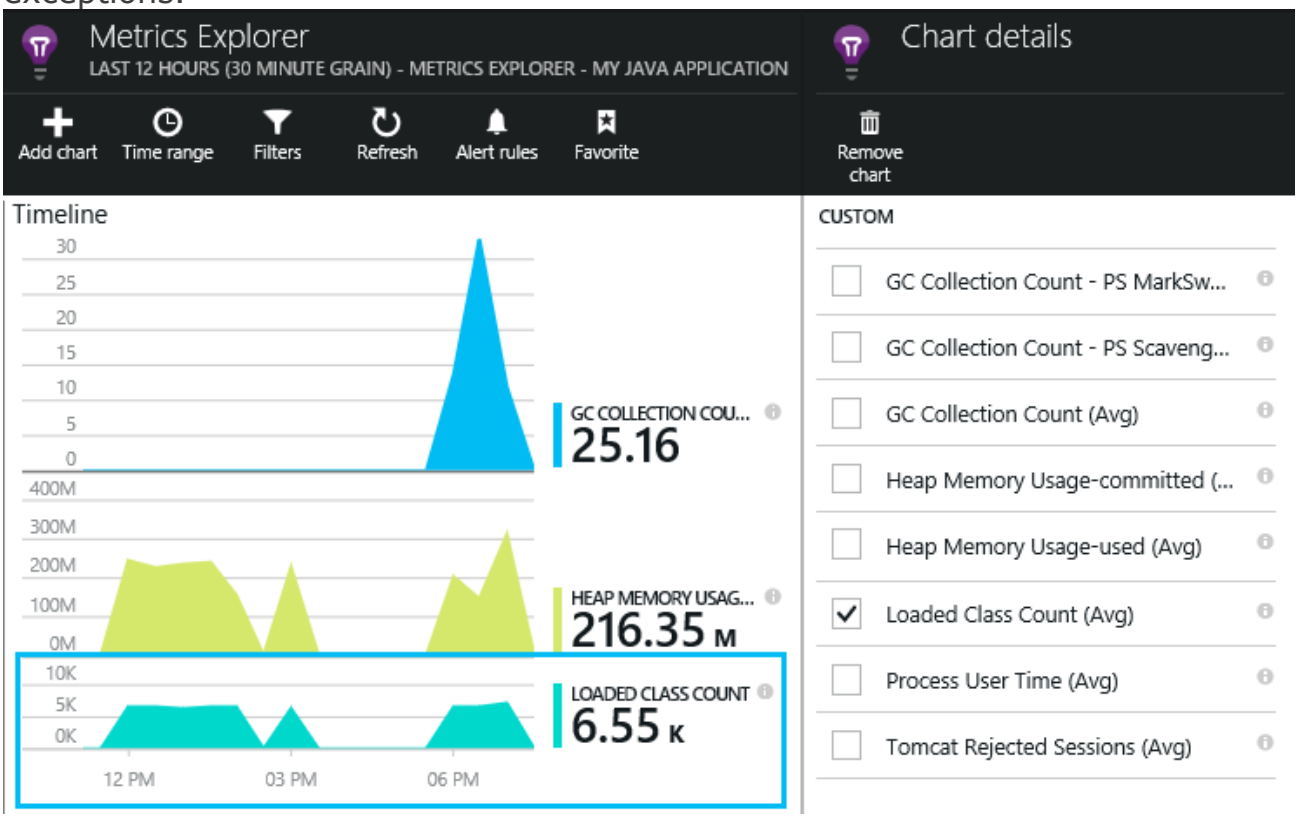
repo to add versioning and change tracking for your



files.

- **Application Insights**: This extension enables monitoring for your web application with the Azure Application Insights service. Application Insights monitors your application for analytical data such as browser, usage and page views. Application Insights can also monitor your application at a lower level exposing useful information such as application

exceptions.



- **Gulp**: This extension enables the Gulp build engine. Gulp is an open-source build engine based on Node.js that has a rich ecosystem of plugins that can support a wide variety of build scenarios.
- **Go Lang**: This extension enables support for Go Lang in your Azure Web App instance.

Custom Site Extensions

As an organization, you can create your own site extensions for your Azure Web Apps. This is very useful if you would like to implement custom functionality that is repeated across a wide variety of Web Apps. By abstracting the repeated functionality out, you can simplify the complexity and surface area of each individual web application.

Site extensions exist in the KUDU web application. This can be located either using the portal or using a URL in the following format:

https://[name of your web app].scm.azurewebsites.net

This web application is referred to as the KUDU dashboard or SCM site and can be used to view logs, manage extensions and access the local console. A site extension is a combination of a custom web application and an XDT file that is deployed to this site. The extensions components are listed below:

- The XDT file is an XML transformation file which is used to modify the applicationHost.config file in the SCM site. Extensions may require you to configure this

file with changes such as connection strings, framework versions or application configuration settings.

- The web application is where the actual log for the extension resides. This can be any web application that you would typically deploy to a Web App instance.

Once created you can deploy your Site Extension in one of two ways:

- **Private Extension**: The extension can be installed directly to your SCM site in the **SiteExtensions** folder. This can be accomplished using either the KUDU dashboard's web application or using FTP.
- **Site Extension Gallery**: An extension can optionally be packaged using the NuGet format. Once packaged, you can submit the package to the public gallery so that the extension is widely available.

Deployment using Local Git

Bookmark this page

Azure Web Sites Deployment using KUDU

Channel 9: https://channel9.msdn.com/Shows/Azure-Friday/What-is-Kudu-Azure-Web-Sites-Deployment-with-David-Ebbo

Azure App Service supports continuous deployment to a Web App by using a local Git repository. The Git repository is "local" to your Web App instance. You can publish directly to the Web App from your local computer simply by cloning the Git repository. Once you have made your changes, you can commit the code and push directly to the repository that is local to the Web App. On any commits, the Web App creates a new deployment and using the latest build of your code.

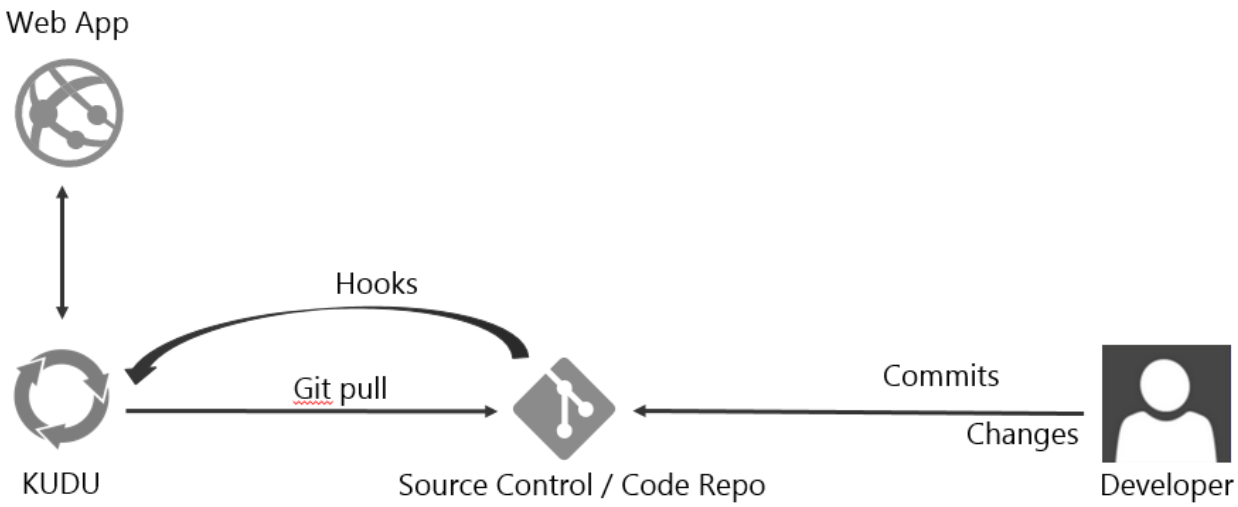
Continuous Deployment

Bookmark this page

Continuous Deployment

The Azure App Service supports continuous deployment to Web Apps from source code control and repository tools such as Git or TFS. External sites that host repositories such as GitHub and Bitbucket are also well supported.

In this diagram we show how KUDU handles continuous deployment from a GitHub repo. This diagram is generic enough to use with many Git installations.



1. In the Web App's configuration, you can set up continuous deployment (CD) from a repo. You must first configure your GitHub account to allow access to the Azure App Service.
2. You can then choose GitHub as a CD source and the appropriate repository and branch. Once configured, an initial deployment will occur from the code that exists in your branch. The App Service will figure out which build engine to use based on the code in your repo.
3. KUDU then configures callbacks for the various GitHub hooks. This allows KUDU to tell GitHub to call certain HTTP endpoints when a commit is made. KUDU can then handle a GitHub commit as an **event**.
4. As a developer you can make commits to your local repo and then push those commits to GitHub.
5. Once the commits are pushed to GitHub, the callback HTTP endpoints are then called and KUDU is notified in this way that a new commit is ready for it to pull. This saves KUDU from having to poll your repo constantly.
6. KUDU will then pull the latest code from your repository, build and then deploy it to your Web App instance[s]. KUDU can handle deployments to as many instances as you have running.

Web Jobs

Bookmark this page

Web Jobs

In the Web Apps service, you can upload and run various types of executable files such as:

- cmd

- bat
- exe (.nET)
- ps1 (PowerShell)
- sh
- php
- py (Python)
- js (Node)
- jar

These programs run as WebJobs. Visual Studio Tooling is available for WebJobs but they can be created in the IDE or environment of your choice. These WebJobs can either be ran continuously or as a chron job. The differences are below:

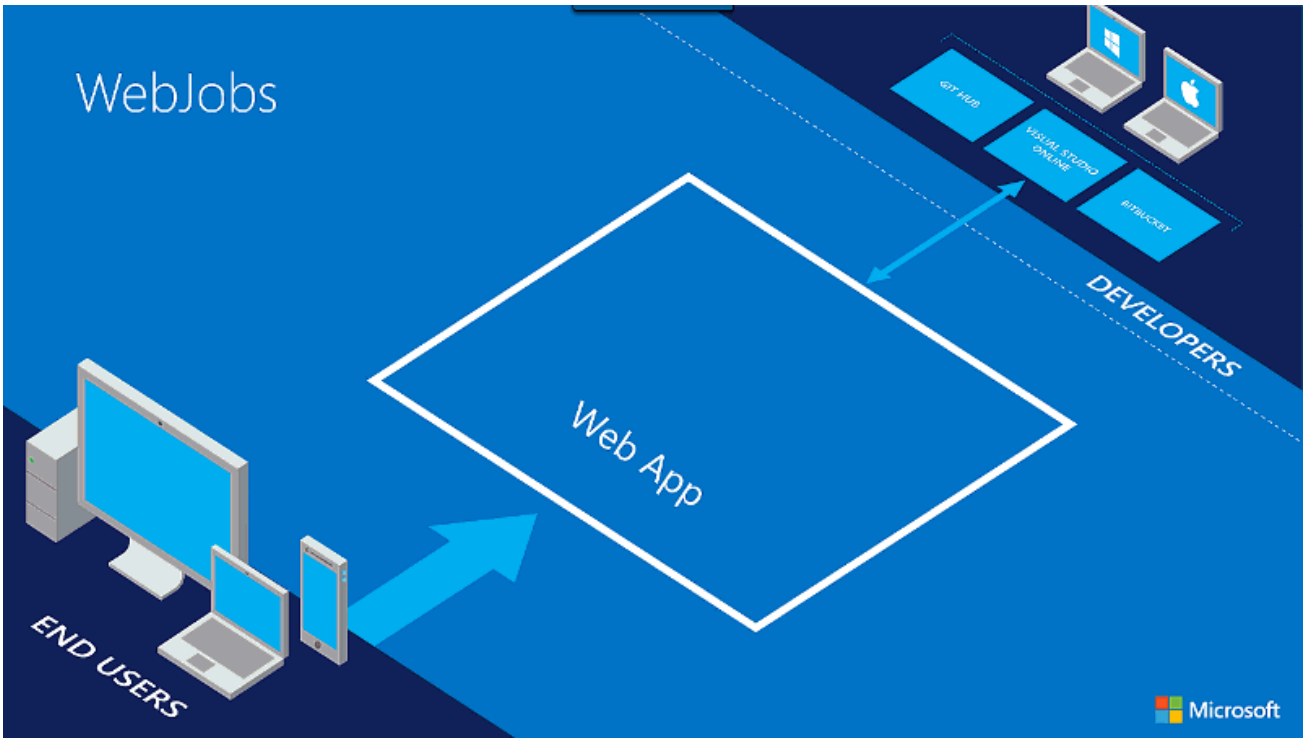
- **Continuous**: A continuous WebJob runs indefinitely and listens for external events. Based on these external events, the WebJob can take action. These continuousjobs can either use the WebJobs SDK to integrate with Azure service events or it can poll a third-party resource such as a queue. When ran, the WebJob is expected to run and block infinitely.
- **Scheduled**: A scheduled WebJob can either be ran manually or on a provided schedule. When ran, the WebJob is expected to perform some functionality and then exit.

Continuous WebJobs

Continuous WebJobs can either poll an external resource or use the SDK to listen to service events.

- **Polling**: In this scenario, the WebJob runs in an infinite loop The loop uses the following seuquence:

1. Polls an external storage mechanism (such as a queue) to see if new messages are available.
2. If messages are available, gets the next message and processes the message.
3. Once complete, use a Sleep function to pause execution temporarily (this prevents throttling from the queue service provider).



- **WebJobs SDK**: The WebJobs SDK ships with a variety of helper method attributes that allows you to bind a method to an Azure service event. Currently, the SDK is only available for .NET WebJobs. To use the SDK, you simply create a new WebJob in Visual Studio. The default implementation blocks the Main method indefinitely (equivalent of infinite loop). You then create methods that are bound to event in Azure services.

```
public static void ProcessQueueMessage([QueueTrigger("webjobsqueue")]] string input) { }
```

The above method binds to a Storage Queue named **webjobsqueue**. The connection string to the Storage account is stored in the **App.config** file. The method can now use the body of the queue message by leveraging the **input** parameter. SDK methods are also available for ServiceBus messages and Storage Blobs.

For more information , you can see:
.NET: https://aka.ms/edx-dev205bx-ne

Back-up and Restore Web Apps

Bookmark this page

Back-up and Restore Web Apps

The Azure Web Apps service can manage both manual or automatic backups. At any time, you can restore your web app to a previous backup or create a new web app from an existing backup. When backed up, Web Apps store the following information:

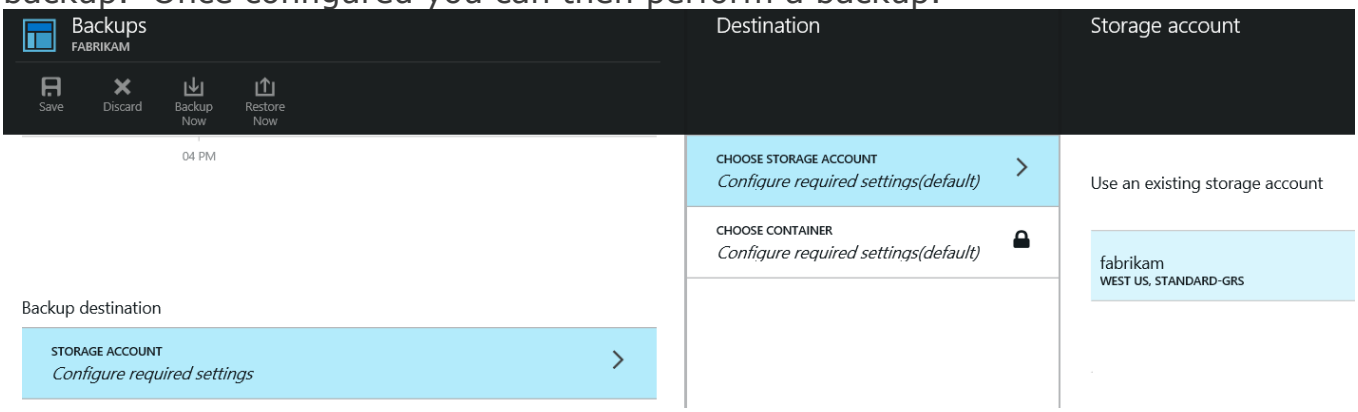
- Configuration settings for Web App instance

- File content on Web App instance
- Connected SQL Server or MySQL database instances.

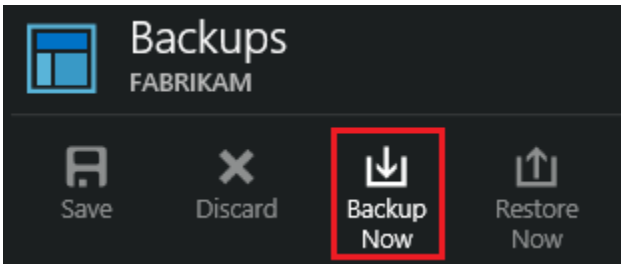
  Backups are made to an Azure Storage account as a blob within a specified container. The Storage Account must be part of the same subscription as the Web App instance.

Performing a Backup

Backups can be made either manually or automatically. Backups must first be configured. This is done by selecting a Storage Account and then a container. Once selected, you will have the option to include any SQL or MySQL databases as part of the backup. Once configured you can then perform a backup.

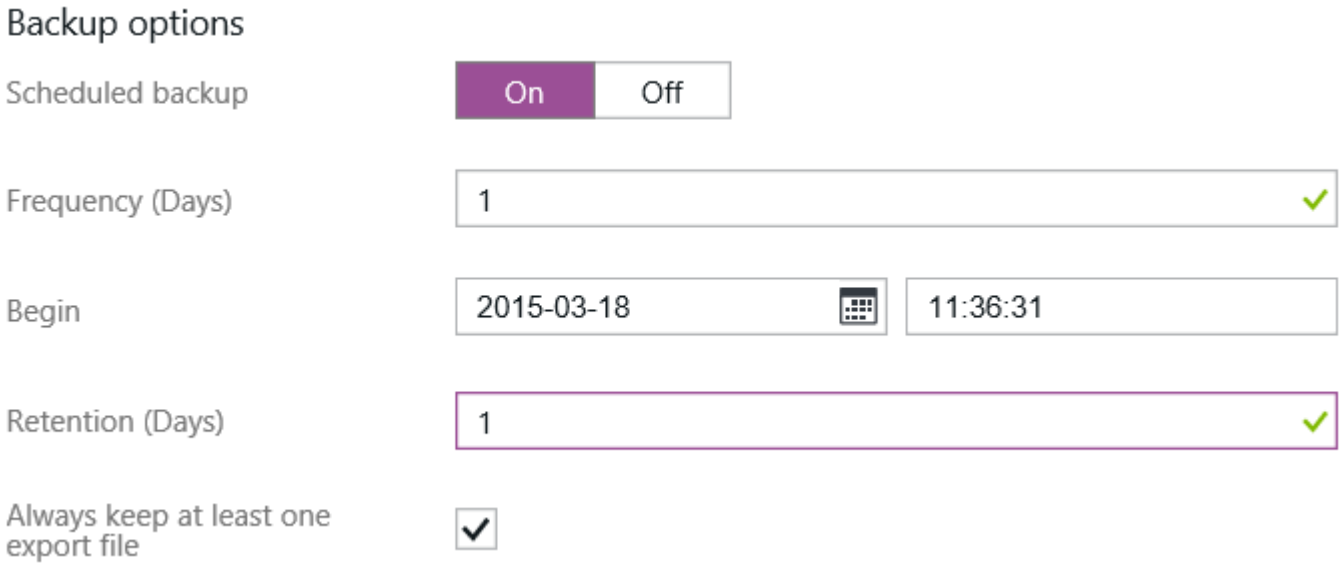


- **Manual**: A manual backup is made immediately once you click the **Backup Now** button.



- **Automatic**: A backup schedule can be configured so that backups occur in an unattended fashion. You first specify the frequencies and the start date. You will also have the option to specify a retention policy for backups. This retention policy dictates

how long a backup stays in your storage account before it is cleaned up.



BEST PRACTICE: Frequent backups that include databases can easily become costly. You may want to decrease the frequency of any backups that includes databases or backup the database separately using another service or feature.

Restoring from a Backup

The Web Apps Restore feature lets you restore your web app on-demand to a previous state, or create a new web app based on one of your original web app's backups. Creating a new web app that runs in parallel to the latest version can be useful for A/B testing.

The Web Apps Restore feature, is available on the same **Backups** blade that was used for the original backup. You can also access the Storage account to download or delete any previous backups. Finally, the restore option allows you to deploy the restored web app to a new Web App instance.

For more information , you can see:
Azure Web Apps service: https://aka.ms/edx-dev205bx-az07
Web Apps: https://aka.ms/edx-dev205bx-az08

Using Traffic Manager with Web Apps

Bookmark this page

Traffic Manager

Microsoft Azure Traffic Manager allows you to control the distribution of user traffic to your specified endpoints, which can include Azure cloud services, websites, and other endpoints. Traffic Manager works by applying an intelligent policy engine to Domain Name System (DNS) queries for the domain names of your Internet resources. Your

Azure cloud services or websites can be running in different datacenters across the world.

Traffic Manager is very flexible because it allows you to mix various endpoints behind the same DNS name. Traffic Manager can be used in a variety of scenarios but most use cases fall in the following scenarios:

- **Failover**: Traffic Manager can poll to determine if and endpoint is online or offline. The endpoints are then ordered in a priority list. By default, traffic routes to the first endpoint. If the first endpoint is down, traffic routes to the next endpoint (2) in the list. Traffic Manager will route requests to the endpoint that is the highest in the priority list and is still online. Using this method, you can have Traffic Manager route traffic to primary or backup datacenters/services for a simple failover scenario.
- **Geography**: Traffic Manager uses an Internet Latency Table to determine the endpoint that is "closest" to the client that is making a request. Using this method, an application can be hosted in West Europe and West US. A user from Denmark can reasonably expect to be served by the endpoint residing in the West Europe datacenter and should experience lower latency and higher responsiveness.
- **Distribution**: Traffic Manager can distribute traffic in a near-random way to distribute traffic evenly across a set of endpoints. If a specific endpoint is down, the traffic is distributed evenly across the remaining endpoints. The distribution can optionally be weighted so that certain endpoints receive more requests than others. The weighted distribution is especially useful if you want to distribute a small subset of your traffic to a hot disaster recovery site that is using smaller service tiers but keep the majority of your traffic to a primary site that is using larger service tiers.

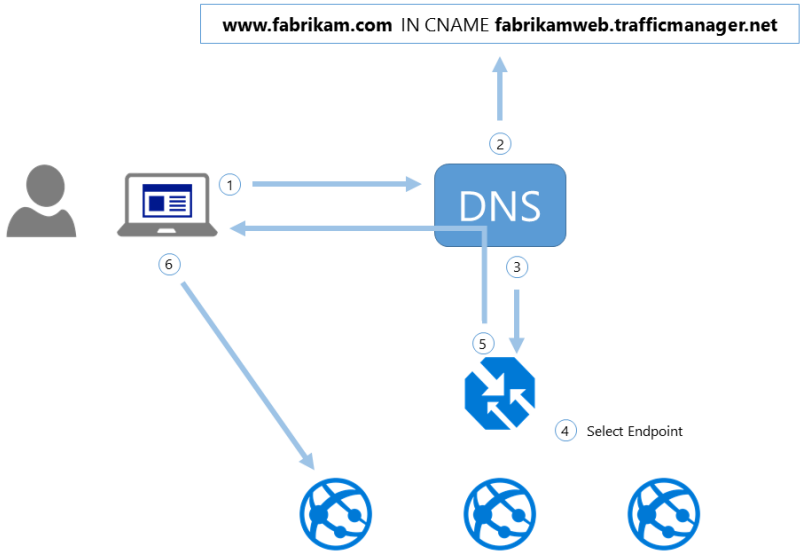
Using Traffic Manager with Web Apps

Traffic Manager can be integrated with Web Apps easily. When you configure a Traffic Manager profile, the settings that you specify provide Traffic Manager with the information needed to determine which endpoint should service the request based on a DNS query. No actual endpoint traffic routes through Traffic Manager. The below diagram shows how Traffic Manager directs users to one of a set of endpoints.

1. **User traffic to company domain name**: The client requests information using the company domain name. The goal is to resolve a DNS name to an IP address. Company domains must be reserved through normal Internet domain name registrations that are maintained outside of Traffic Manager. In Figure 1, the example company domain is *www.fabrikam.com*.
2. **Company domain name to Traffic Manager domain name**: The DNS resource record for the company domain points to a Traffic Manager domain name maintained in Azure Traffic Manager. This is achieved by using a CNAME resource record that maps the

company domain name to the Traffic Manager domain name. In the example, the Traffic Manager domain name is *fabrikamweb.trafficmanager.net*.

3. **Traffic Manager domain name and profile**: The Traffic Manager domain name is part of the Traffic Manager profile. The user's DNS server sends a new DNS query for the Traffic Manager domain name (in our example, *fabrikamweb.trafficmanager.net*), which is received by the Traffic Manager DNS name servers.
4. **Traffic Manager profile rules processed**: Traffic Manager uses the specified load balancing method and monitoring status to determine which Azure or other endpoint should service the request.
5. **Endpoint domain name sent to user**: Traffic Manager returns a CNAME record that maps the Traffic Manager domain name to the domain name of the endpoint. The user's DNS server resolves the endpoint domain name to its IP address and sends it to the user.
6. **User calls the endpoint**: The user calls the returned endpoint directly using its IP address.



For more information , you can see:
Microsoft Azure Traffic Manager: https://aka.ms/edx-dev205bx-az09
Azure cloud services: https://aka.ms/edx-dev205bx-az10

Introducing App Service Environments (ASE)

Bookmark this page

Introducing App Service Environments (ASE)

App Services are useful because they separate many of the hosting and management concerns for your web application and allow you to focus on your application's functionality and configuration. There are some scenarios, however, where you require a bit more control.

For example, your organization may require you to make sure that all of the virtual machines hosting your web applications do not allow any outbound requests. This is a common scenario when implementing a web solution that must be PCI compliant. With App Services, you can't access or modify the configuration of the virtual machines hosting your applications. You do not have any mechanism to implement this requirement.

To implement scenarios where you require more control, you can use the App Service Enviornment (ASE) service in Azure. ASE allows you to configure network access and isolation for your applications. ASE also allows you to scale using pools of instances far beyond the limits of a regular App Service plan instance. Finally, ASE instances are dedicated to your application alone. You retain much of the convenience of using App Services such as automated scaling, instance management and load balancing when using ASE but you gain more control.

Networking in ASE

With ASE, you can configure network access using the same concepts and paradigms that you use in Virtual Machines. Environments are created within a subnet in an Azure Virtual Network. You can use Network Security Groups to restrict network communications to the subnet where the Environment resides. You can also using a protected connection to connect your Virtual Network to corporate resources so that your ASE instance can securely use the resources.

For more information , you can see:
Virtual Machines: https://aka.ms/edx-dev205bx-az11
Azure Virtual Network: https://aka.ms/edx-dev205bx-az12

Introducing Azure SQL Database

Bookmark this page

Introducing Azure SQL Database

Azure SQL Database (SQL DB) is a relational database service in the cloud based on the Microsoft SQL Server engine, with built-in, mission-critical capabilities. SQL DB delivers predictable performance, scalability, business continuity, data protection, and near-zero administration to cloud developers and solution architects. SQL DB is highly compatible with existing tooling because the service implements a Tabular Data Stream (TDS) protocol endpoint that is used by applications such as:

- Visual Studio

- SQL Server Management Studio
- LINQPad

SQL DB allows you to scale up or scale out to thousands of databases, with predictable performance that can be dialed up or down as needed. There are built-in data protection features such as auditing, restore and geo-replication of data allowing you to replicate data to an Azure region of your choice and implement a geographic disaster recovery plan through configuration. The latest version of SQL DB (v12) offers the highest compatibility currently with the SQL Server standalone engine.

Azure SQL Database Tiers

SQL Database is available in Basic, Standard, and Premium service tiers which support light-weight to heavy-weight database workloads, so you can move across or blend the tiers together for various application designs. These tiers offer predictable performance, flexible business continuity options, and streamlined billing. As your workload changes over time, you can change tiers for your database to modify the performance characteristics of your database instance.

Service Tier	Common App Pattern	Transactional Perf. Objective
Basic	Small databases with a single operation at a given point in time	Reliability per hour
Standard	Workgroup and cloud applications with multiple concurrent transactions	Reliability per minute
Premium	Mission-critical, high transactional volume with many concurrent users	Reliability per second

Each Service Tier is broken down into various Performance Levels. These levels offer a more granular way to manage billing and performance for an individual database instance. The current performance characteristics for SQL DB (v12) are listed below. Databases on other versions may have different performance characteristics.

Service Tier/Performance Level	DTU	MAX DB Size	Max Concurrent Requests	Max Concurrent Logins	MaxSessions	Benchmark Transaction Rate	Predictability
Basic	5	2 GB	30	30	300	16,600 transactions per hour	Good
Standard/S0	10	250 GB	60	60	600	521 transactions per minute	Better
Standard/S1	20	250 GB	90	90	900	934 transactions per minute	Better
Standard/S2	50	250 GB	120	120	1,200	2,570 transactions per minute	Better
Standard/S3	100	250 GB	200	200	2,400	5,100 transactions per minute	Better
Premium/P1	125	500 GB	200	200	2,400	105 transactions per second	Best

Premium/P2	250	500 GB	400	400	4,800	228 transactions per second	Best
Premium/P3	1000	500 GB	1,600	1,600	19,200	735 transactions per second	Best

The above table contained some unique metrics used to compare database in SQL DB. They are listed below:

- **Database Throughput Unit (DTU):** DTUs provide a way to describe the relative capacity of a performance level of Basic, Standard, and Premium databases. DTUs are based on a blended measure of CPU, memory, reads, and writes. As DTUs increase, the power offered by the performance level increases. For example, a performance level with 5 DTUs has five times more power than a performance level with 1 DTU. A maximum DTU quota applies to each server.
- **Benchmark Transaction Rate:** Transaction rate is the metric produced by using a benchmark. This metric is reported in transactions per unit-of-time, counting all transaction types. The benchmark is the combination of what is considered a "normal" transaction and "normal" sized record. As with all benchmarks, a benchmark is useful in relative comparisons but does not necessarily represent the types of records your application may have or the types of queries your application may perform. You can easily see more or less transactions per second/minute/hour with many different database schemas.
- **Predictability:** Consistency of response time is an indication of performance predictability. A database which achieves a more stringent response time constraint delivers more predictable performance.

For more information , you can see:
Azure SQL Database: https://aka.ms/edx-dev205bx-asd
Visual Studio: https://aka.ms/edx-dev205bx-vs
SQL Server Management Studio: https://aka.ms/edx-dev205bx-ssms

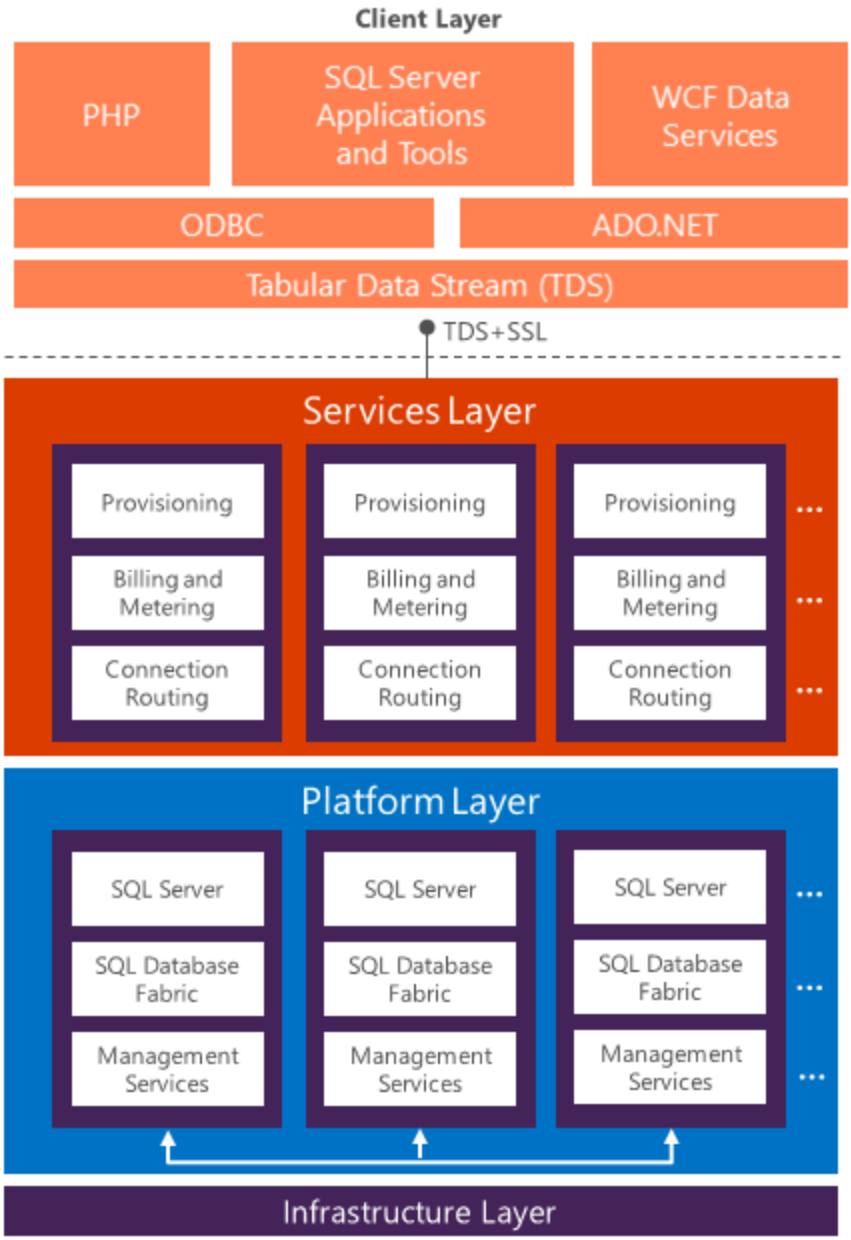
Azure SQL Database Architecture

Bookmark this page

Azure SQL Database Architecture

Behind the scenes, the Azure SQL Database service is separated into tiers with varying sets of responsibility. These tiers are listed below:

1. **Client Layer**: This layer is composed of the tools that you can use to connect to Azure SQL Database at it's TDS endpoint. This layer is used by applications to communicate directly with SQL Database.
2. **Services Layer**: This layer is a gateway between the client layer and the Platform layer
3. **Platform Layer**: This layer includes physical servers and services that support the Services layer and actually implements the database service.
4. **Infrastructure Layer**: This layer is the layer where Azure's fabric controller and hypervisor manages the physical hardware and operating systems.



Client Layer

Azure SQL Database data is transferred using tabular data stream (TDS) over a secure sockets layer (SSL). This is a commonly used protocol and already supported by a wide variety of existing tools. Because of this, you rarely have to change your existing DBA workflow when working with this service. Additionally, the Azure SQL Database instances can be managed directly using the Azure SDKs, REST APIs, Portals or IDE tooling. This gives you a much wider range of methods you can use to manage both databases and data in Azure.

Azure SQL Database Elastic Scale

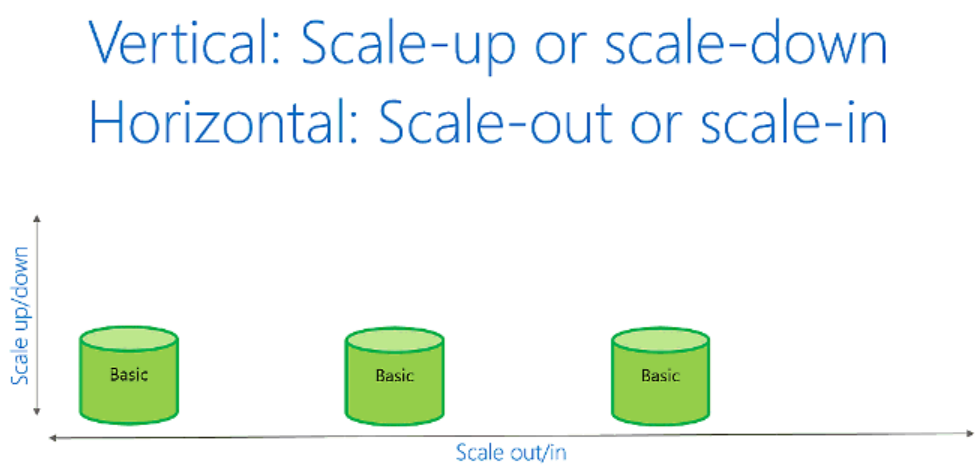
Bookmark this page

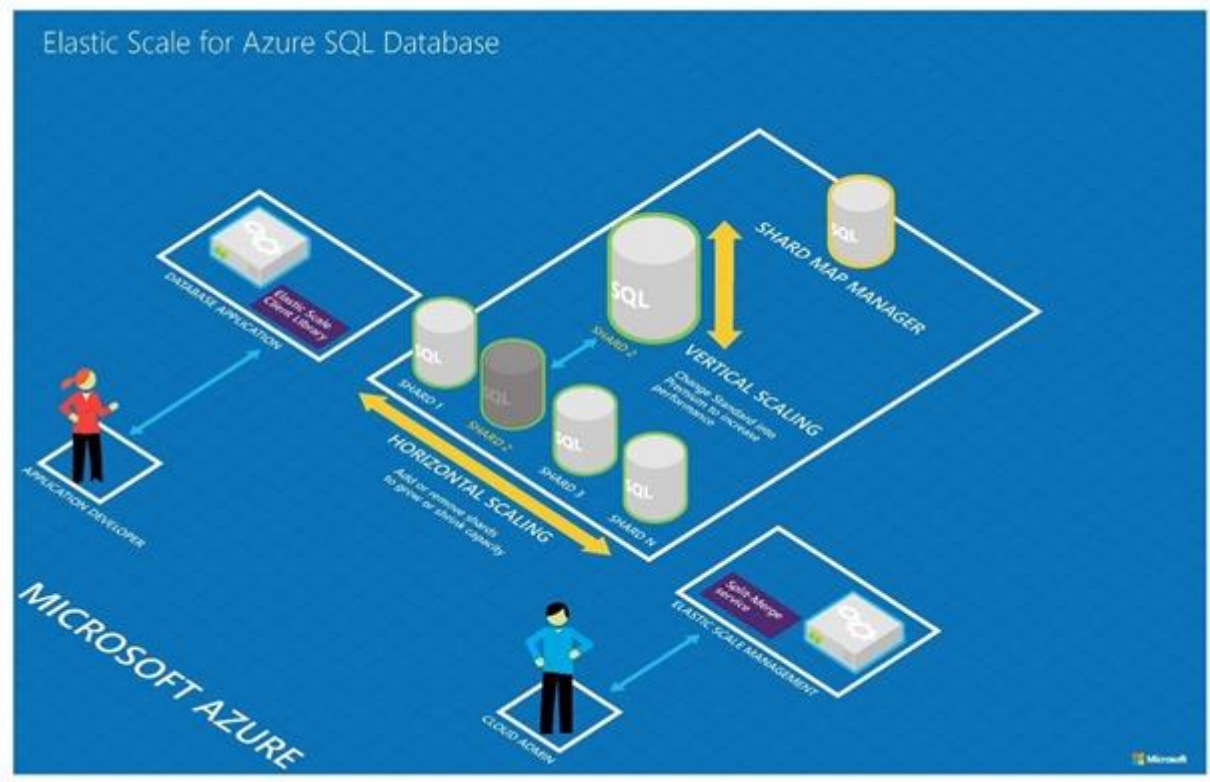
Azure SQL Database Elastic Scale

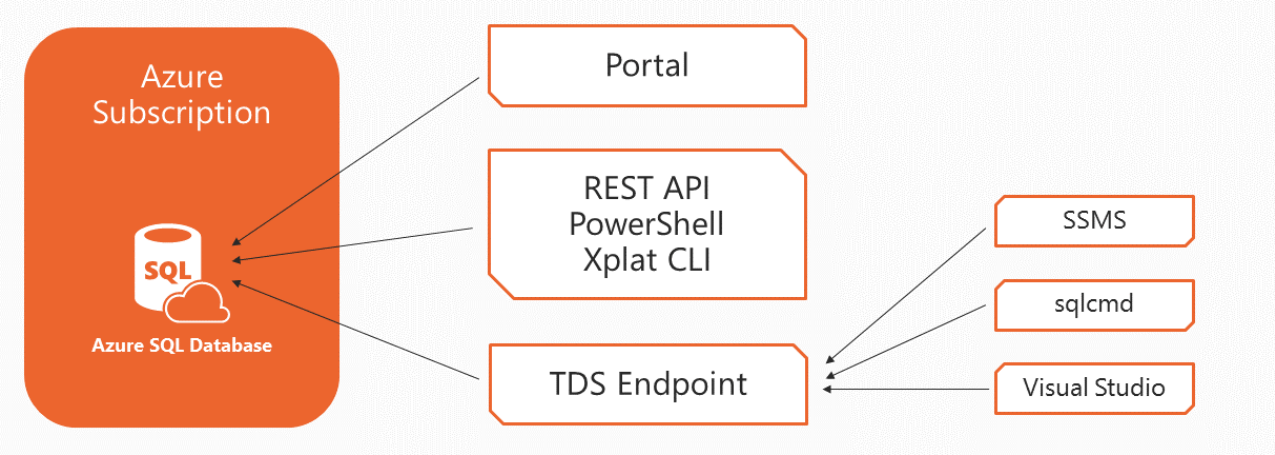
The new Elastic Scale capabilities simplify the process of scaling out (and in) a cloud application's data tier by streamlining development and management. Elastic Scale is composed of two main parts:

- An Elastic Scale library for client applications to configure shards and access shards.
- The Elastic Scale features in Azure SQL Database that implements the any changes requested by your application.

Elastic Scale implements the database scaling strategy known as sharding. As a developer, you can establish a "contract" that defines a shard key and how shards should be partitioned across a collection of databases. The application, using the SDK, can then automatically direct transactions to the appropriate database (shard), perform queries across multiple shards or modify the service tier for existing shards. Elastic Scale also enables coordinated data movement between shards to split or merge ranges of data among different databases and satisfy common scenarios such as pulling a busy tenant into its own shard. The Split-Merge service is provided through a downloadable package that customers can deploy as an Azure cloud service into their own subscription.



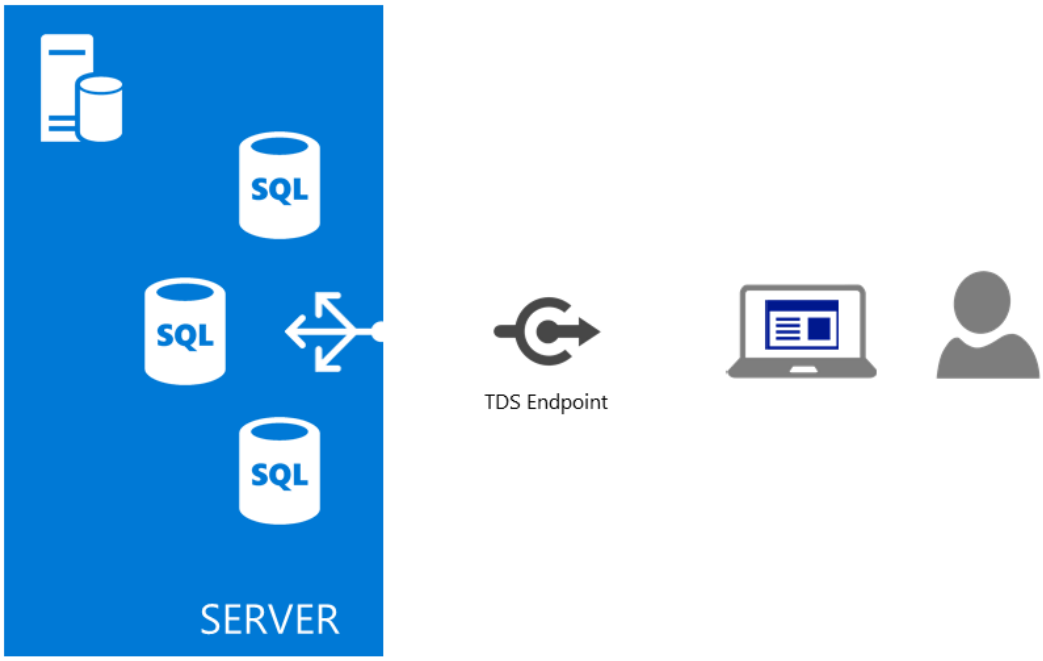




Logical Layout

The Azure SQL Database service is organized logically into **databases** and **servers**.

- **Server**: The server is a logical container for database instances. The TDS Endpoint and user accounts exists at this level. In order to connect to a SQL Database instance, one must use the name of the server in the following fashion: [server name].database.windows.net.
- **Database**: The database instance is where the schema and data is stored. Once connected to the server using the TDS endpoint, you can either specify a database in a connection string or access multiple databases using a management tool.



For more information , you can see:
Azure SDK: https://aka.ms/edx-dev205bx-az02

Migrations to Azure SQL Database

Bookmark this page

Migrations to Azure SQL Database

Migrating databases from on-premise to Azure SQL Database can range from very simple scripts to complex migrations using transformation tools. Below are some examples of methods that you can use to migrate data to Azure:

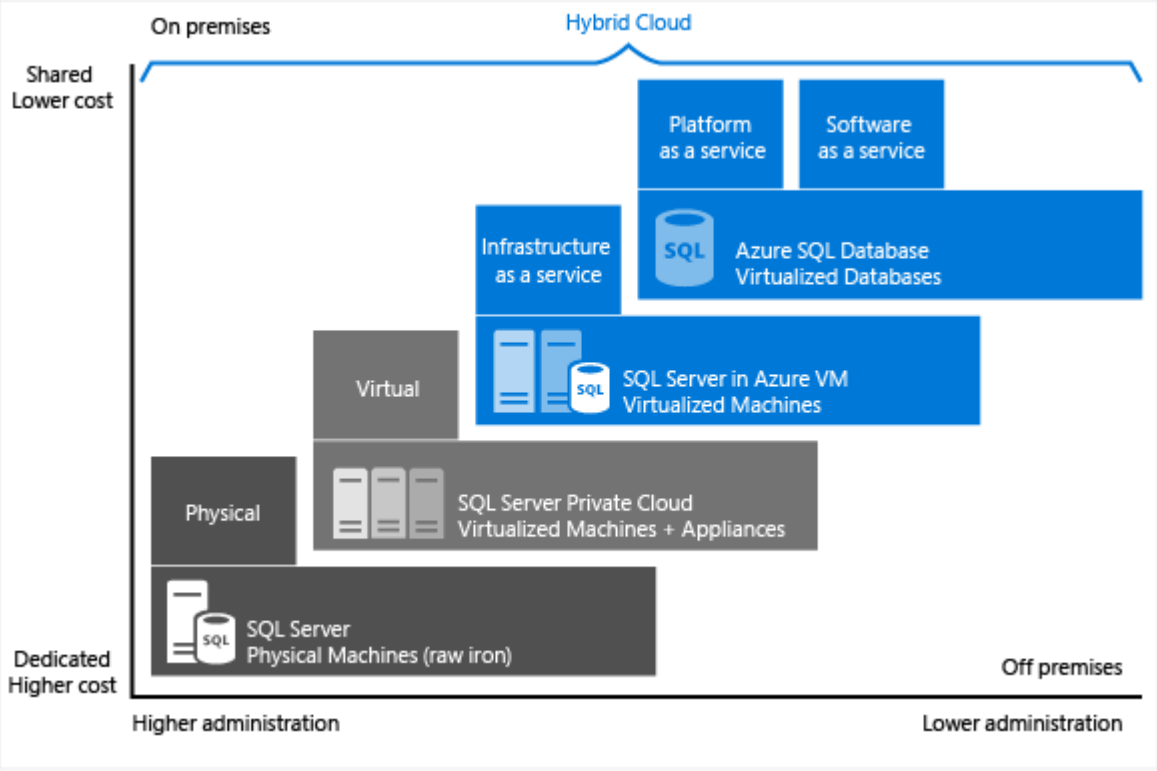
- **Azure Websites Migration Assistant**: The Azure Websites Migration Assistant is a tool that analyzes your existing web application to ensure that it is compatible with the Azure Websites platform. This is done by generating a Readiness Assessment for the IIS web site that you select as the source for your deployment. Then the tool creates target resources in Azure and then performs the migration automatically. The application is available to install from http://www.movemetothecloud.net. You select the installer based on your existing website hosting.
- **SQL Database Migration Wizard**: The SQL Database Migration Wizard is a standalone tool available for download (source and binaries) from https://sqlazuremw.codeplex.com. SQL Database Migration Wizard (SQLAzureMW) is designed to help you migrate your SQL Server 2005 (or greater) databases to Microsoft Azure SQL Database. The migration wizard will analyze your source database for compatibility issues and allow you to fully or partially migrate your database schema and data to Azure SQL Database. Using SQLAzureMW, you can specify an explicit list of the database's objects to be migrated. This can be useful for scenarios where an application

shares a database with other tenant applications and you wish to migrate specific tables or schemas.

- **SQL Server Integration Services (SSIS)**: SSIS can be used to create a workflow that transforms data from one schema to another across various databases. This is ideal in a scenario where you existing database is using features not yet available in Azure SQL Database. is an ideal as an ETL solution for migrating the most complex SQL databases from your on premise infrastructure to an Azure SQL Database instance.
- **Data-Tier Application Import and Export**: BACPAC files can be exported from an existing SQL Server Standalone database using the Export Data-Tier Application wizard. The file can then be uploaded to Blob Storage and eventually imported into an Azure SQL Databases instance.

For more information , you can see:
SQL Server 2005: https://aka.ms/edx-dev205bx-sql

SQL Data Platform



Azure SQL Database v12

Bookmark this page

Azure SQL Database v12

The SQL Database team at Microsoft has been working for years on a vision to bring parity between many of the features in SQL Server and Azure SQL Database. Because of this effort, many of the new features introduced in SQL Server 2016 are already

available in Azure SQL Database. To enable these features, you must create an instance of Azure SQL Database using the version titled **v12**.

Here's a sample of some of the new features:

- **JSON Support**: Queries that produce JSON results and operate over JSON data are now supported. You can use T-SQL to save a JSON data as a rowset in SQL.
- **Change Tracking:** Change tracking allows you to keep track of which rows have changed in your database without the need to add new columns such as watermarks and timestamps.
- **Ranking Functions**: You can create a function to rank results based on values in each row returned by your query. The ranked value can be returned as part of the result set to create a comprehensive search experience.
- **SELECT...INTO:** You can select data using a query and insert the result of that query into a different table.
- **Full-Text Search:** You can search the entire text of columns in queries using a full-text index.

Customer Scenario

Bookmark this page

Who is the customer?

Founded in 1954, Adventure Works Cycles has grown from a boutique manufacturer of high-quality bicycles and parts into one of the world's largest makers of premium race and commuter bicycles. The Adventure Works mission has remained the same: to passionately pursue advanced, innovative technologies that help cyclists of all abilities find more enjoyment in the sport. Adventure Works Cycles Company manufactures and sells bicycles, bicycle parts, and bicycle accessories under the Adventure Works Cycles and Tailspin brands worldwide.

During the global recession of 2009, most of the company's IT infrastructure was located in the company's Provo, Utah, headquarters, but Adventure Works also had a sizable third-party colocation datacenter, costing US$30,000 to $40,000 a month, and other servers scattered around the United States.

Adventure Works knows that its datacenters are filled with dozens of smaller web servers and databases that run on underutilized hardware, and it has customer data scattered in multiple places. Faced with the prospect of a very large capital expenditure owing to the fact that the vast majority of their servers are now due for a hardware refresh, Adventure Works is looking for other options that eliminate the costly and high risk hardware refresh cycles.

What does the customer already have?

In reviewing all of Adventure Works' web applications, it was determined that there were three archetypical database backed web applications present:

- **Product Catalog:** Resellers and consumers in North America, Asia and Europe access the product catalog via a website, the data for which is currently stored in SQL Server. Currently, both web site and database are hosted in its Utah datacenter. The database is 50GB in size, and is not expected to grow past 100GB.
- **Inventory:** The inventory web application is a mission critical system primarily used by operations running in North America. Currently, both web site and SQL Server database are hosted in its Utah datacenter. This is Adventure Works' largest database at 3 TB in size, but it is not expected to grow beyond 5 TB.
- **Departmental:** These web applications support the regional offices only. Both web site and database are hosted in the Utah data center. While individually these have web applications have low demands and data sizes in the 500MB-1GB range, Adventure Works has 100's of SQL Server databases supporting the numerous web apps, and fully expects new databases to be created to support departmental efforts.

Most of Adventure Works developers are already trained in Microsoft tools as all of Adventure Works' web applications and services are built using .NET. Additionally, they have already deployed Active Directory and are currently using a replicated directory in support of departments within each regional office.

For more information , you can see:
SQL Server: https://aka.ms/edx-dev205bx-sql01
SQL Server database: https://aka.ms/edx-dev205bx-sql02
.NET: https://aka.ms/edx-dev205bx-ne

Customer Goal

Bookmark this page

What is the customer's goal?

Not only are all the servers expensive to acquire and maintain, but scaling the infrastructure takes significant time. "During and after a high-profile race, there's a great deal of interest in our product, and our websites receive a lot of hits," says Hayley Leigh, Manager of Solution Development for Adventure Works Cycles. "It is difficult to scale our web hosting environment fast enough, and consumers and resellers could experience slow response times and even downtime." Another challenge: Adventure Works conducts weekly server hardware maintenance, which causes downtime for some of its global offices.

Adventure Works wants to move many consumer-facing websites, enterprise databases, and enterprise web services to Azure. "By using Microsoft global datacenters, we're able to move infrastructure for key applications and websites closer to the people who use them," Leigh says. A big problem for Adventure Works is resellers and consumers in Japan and China have to use applications that run in a Utah datacenter, and because of the distance, encounter performance problems. Adventure Works would like to resolve this without the difficulty, expense and time requirements incurred by setting up infrastructure on the other side of the world using Adventure Works-owned servers.

For more information , you can see:
Azure: https://aka.ms/edx-dev205bx-az34

Solution Considerations

Bookmark this page

What does the customer need?

- Improve the performance for customers and resellers accessing its websites around the world
- Support for easily provisioning resources to meet bursts of demand
- Consolidate and improve the utilization of website and database hosting resources
- For their most demanding databases, they want to migrate without having to re-architect their database structure or make large changes the application.
- Avoid downtime, particularly that caused by web and database server patching
- Leverage familiarity with Microsoft tools

What things worry the customer?

Scale & Performance

- I do not want to have to make code changes (or re-deploy) in order to change the scale of a website.
- I hear Azure Web Sites is only useful for websites with small amounts of traffic; will it really support the heavy traffic we receive?
- We would prefer to avoid performing a database migration (e.g., to another server) in order to scale the throughput of our database.
- We have heard SQL Database does not provide consistent performance, is this true?

Business Continuity

- How can we certain our data will survive in the event of a catastrophe in a certain part of the world?
- We need to be able to recover from mistakes made by administrators that accidentally delete production data (we know they happen, we would love an “undo”).
- Do we need to have multiple web server instances for each property to have a high SLA?

Tool Familiarity

- Will we need to learn new tools to develop for Azure Websites and SQL Database?
- What about diagnosing problems? Are there new tools we need purchase and learn?

Connectivity

- Some of our enterprise web services need to access data and other services located on-premises, is this supported?
- How can we ensure we are delivering the lowest latency possible to our website visitors?
- We need to ensure that if we have multiple web servers backing a given website, that no one web server gets all the traffic.

Management

- We would prefer not to have to manage patching of web servers and databases.
- With all of our websites and databases around the world, how do we keep tabs on which is up and which is down and which is struggling?
- We need a simple solution to schedule and automate backup of the website and database.

Security

- Is it possible to allow our visitors to use a mix of legacy and modern browsers and still provide for secure transactions?
- What does Azure offer to help us with auditing access to our web servers and databases?
- Our staff is accustomed to accustomed to a single sign-on experience — will this still be possible?

Common Implementations

What are some of the available Azure services?

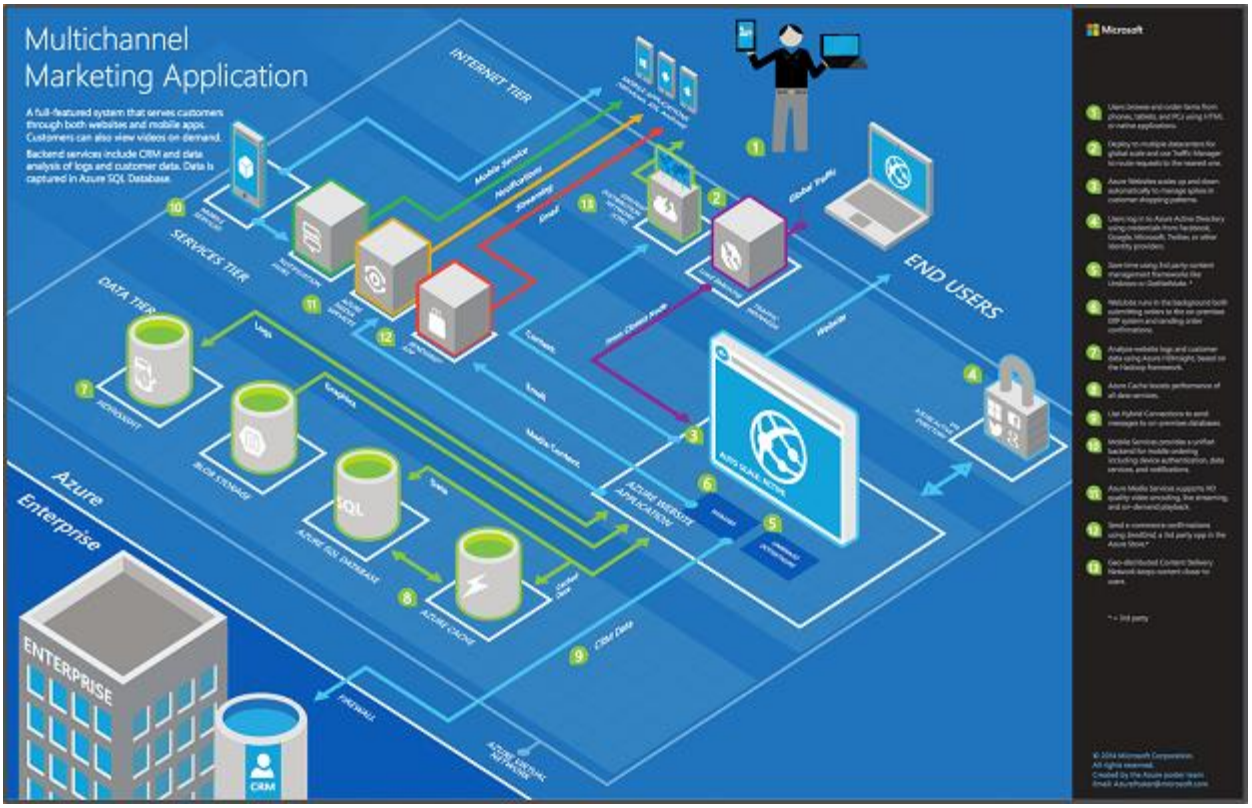
Consider some of the scenarios that Microsoft Azure cloud services can be used with:

- Azure SQL Database can be used to host scalable, elastic, highly-available, geo-redundant databases
- Azure Blob Storage provides a resilient and secure storage medium for all types of applications, as well as infrastructure virtual machines hosted on Azure.
- Azure Web Apps offers a lightweight, easy-to-manage platform for .NET developers to deploy their web front-ends and web services to.
- Azure Virtual Networks can be connected to on-premises networks, effectively extending the existing corporate network into the cloud.

Multi-Channel Marketing Application

http://download.microsoft.com/download/2/C/8/2C8EB75F-AC45-4A79-8A63-C1800C098792/MS_Arch_Multichan_3D_pdf.pdf

Zoom-in to see more detail.

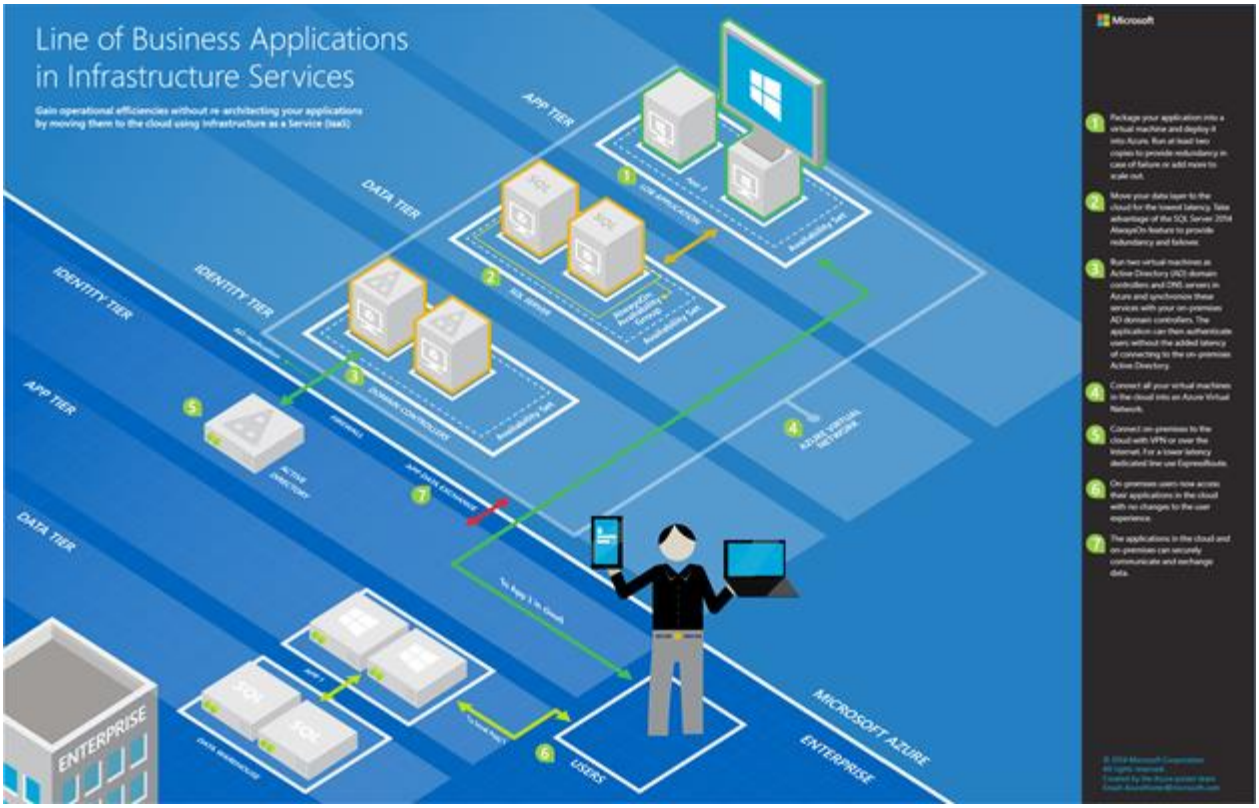


The need to enable modern business applications is critical. According to **Gartner**, a quarter of external application implementation spending will go toward supporting social, mobile, cloud, and data analytics trends. Gartner expects that more than half of the effort will go toward enhancing the functionality of existing applications. The Multichannel Marketing Application blueprint shown above illustrates how to extend an existing Customer Relationship Management (CRM) system with a new online marketing campaign.

Line of Business Application in Infrastructure Services

http://download.microsoft.com/download/2/C/8/2C8EB75F-AC45-4A79-8A63-C1800C098792/MS_Arch_LOB_App_3D_pdf.pdf

Zoom-in to see more detail.



Enterprises are running out of space, literally, and need a way to support all of the new requirements for compute, network, and storage. As an architect, you will want to look for quick wins to help ease your organization into the cloud. To extend your organization's LOB applications to the cloud, first configure an Azure Virtual Network that connects with your on-premises network. Then, move your application, data, and identity tiers as Azure virtual machines within the network. Azure provides Active Directory replication with Windows Server for authentication. SQL Server provides AlwaysOn Availability Groups for high availability within the data tier. Virtual machines for your application tier and other tiers are grouped within availability sets that tell Azure to place the virtual machines in separate racks to minimize the impact if a host rack fails.

For more information , you can see:
Microsoft Azure cloud services: https://aka.ms/edx-dev205bx-az10
Azure SQL Database: https://aka.ms/edx-dev205bx-asd
Azure Blob Storage: https://aka.ms/edx-dev205bx-az14
Azure Web Apps: https://aka.ms/edx-dev205bx-az08
Azure Virtual Network: https://aka.ms/edx-dev205bx-az12
virtual machines: https://aka.ms/edx-dev205bx-az11

Call to Action

Bookmark this page

Design the Solution

You will now design a potential solution for **Adventure Works Cycles**. Prior to completing this exercise, please ensure that you have read all of the previous units in this case study. Particularly, make sure that you have read and you understand the **Customer Scenario** and **Solution Considerations**.

In this exercise, you will tackle a few primary tasks.

1. You will need to determine who you should present this solution to. Who is the target customer audience (stakeholder)? Who are the decision makers? You will answer these questions below in the problem sections.
2. You will need to determine what you intend to address with your solution. What customer business needs do you need to address with your solution? How will you address each business need? You will provide a brief explanation below in the problem sections.
3. You will need to create a diagram for your proposed solution. This can be done with Visio or with any image editor of your choice. The ideal output is either a high-resolution PNG file or a PDF document. You will upload this diagram in the final section below.

Once you have completed these three tasks, create a new post in the solution forum, attach an image file to the post for your diagram and your response to prompts #1 & #2.

Web App & SQL Case Study

Topic: Case Studies / Module 2: Web App & SQL Case Study

Show Discussion

Sample Solution

You are highly encouraged to try and come up with your own solution for this case study and post that solution in the forums. Once you have completed that, you can review the sample solution linked below and compare it against your own solution.

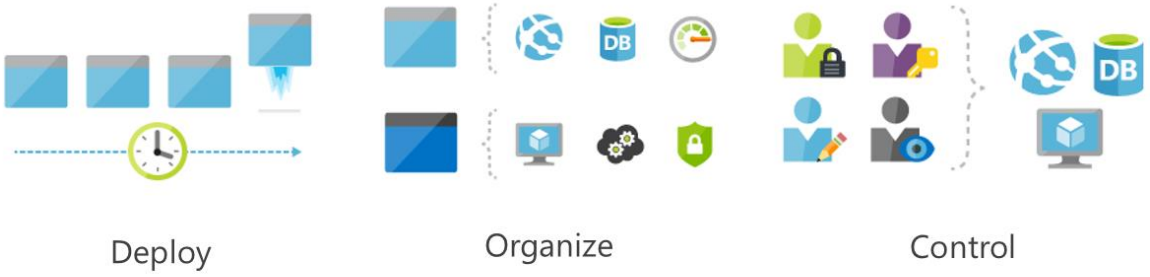
Sample Solution

For more information , you can see:
Visio: https://aka.ms/edx-dev205bx-vp

Azure Resource Manager

Bookmark this page

Azure Resource Manager



Azure Resource Manager introduces an entirely new way of thinking about your Azure resources. Instead of creating and managing individual resources, you begin by imagining a complex service, such as a blog, a photo gallery, a SharePoint portal, or a wiki. You use a template -- a resource model of the service -- to create a resource group with the resources that you need to support the service. Then, you can manage and deploy that resource group as a logical unit. There are three primary concepts in Resource Manager:

- **Resource**: A resource is simply a single service instance in Azure. Most services in Azure can be represented as a resource. For example, a **Web App** instance is a resource. An **App Service Plan** is also a resource. Even a **SQL Database** instance is a resource.
- **Resource Group**: A resource group is a logical grouping of resources. For example, a Resource Group where you would deploy a VM compute instance may be composed of a **Network Interface Card (NIC)**, a **Virtual Machine**, a **Virtual Network**, and a **Public IP Address**.
- **Resource Group Template**: A resource group template is a **JSON** file that allows you to declaratively describe a set of resources. These resources can then be added to a new or existing resource group. For example, a template could contain the configuration necessary to create 2 **API App** instances, a **Mobile App** instance and a **Document DB** instance.

Each of these three concepts will be explored further in later units.

Interacting with Resource Manager

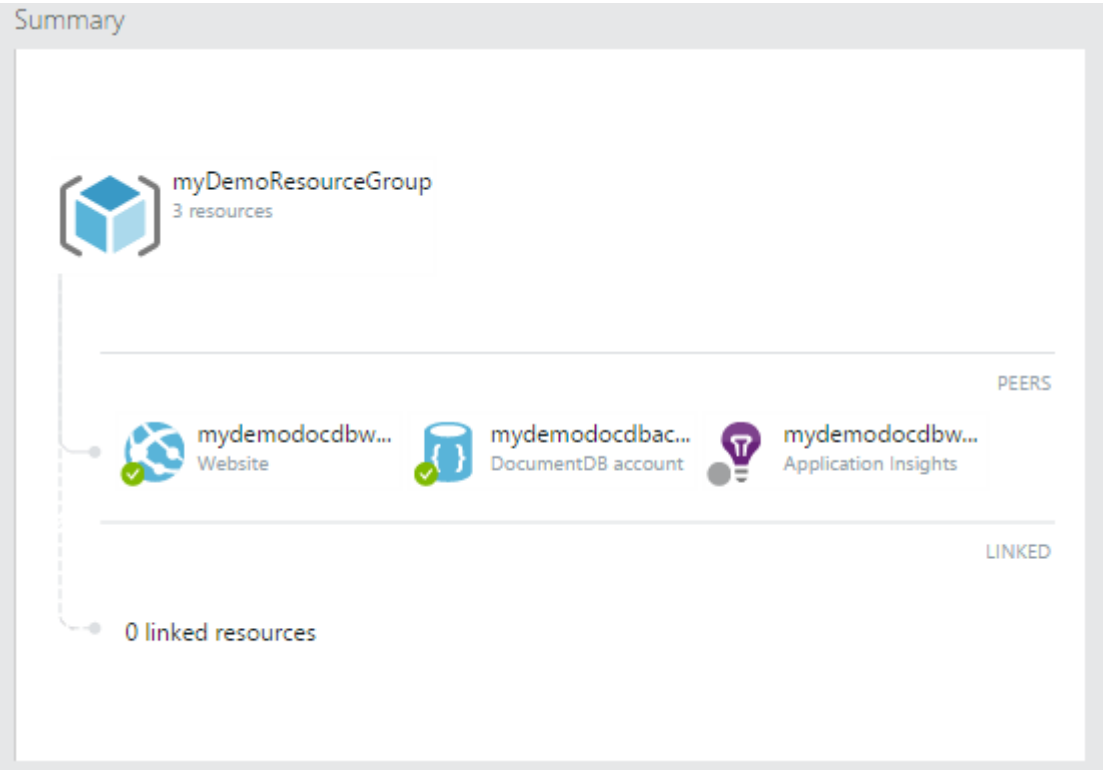
Resource Manager can be used in a variety of different ways including:

- **PowerShell**: There are PowerShell cmdLets already available to allow you to manage your services in the context of resources and resource groups.
- **Cross Platform Command-Line Interface:** This CLI allows you to manage your Azure resources from many different operating systems.
- **Client Libraries**: There are client libraries already available for various programming frameworks/languages to create resources and resource groups in Azure.
- **Visual Studio**: Visual Studio 2015 ships with a Resource Manager project type that allows you to create a resource group template by either manually modifying JSON (with schema intellisense) or use scaffolding to automatically update your JSON template.
- **Portal template deployment**: In the portal, you can use the **Template Deployment** option in the Marketplace to deploy a Resource Group from a template.
- **REST API**: All of the above options use the REST API to create your resources and resource groups. If you prefer to create resources without a library, you can always use the REST API directly. The REST API is available here: https://msdn.microsoft.com/en-us/library/azure/dn790568.aspx

For more information , you can see:
Azure: https://aka.ms/edx-dev205bx-az34
SQL Database: https://aka.ms/edx-dev205bx-asd
virtual machines: https://aka.ms/edx-dev205bx-az11
Virtual Network: https://aka.ms/edx-dev205bx-az12
Document DB: https://aka.ms/edx-dev205bx-az06
Visual Studio 2015: https://aka.ms/edx-dev205bx-vs01

Resource Groups in the Portal

In the Azure Preview Portal, Resource Groups are visually show in the Summary tile. In this tile, you can see an icon to navigate to the entire resource group, and another list of a subset of the peer resources in the group. If you click on the resource group icon, you will see a list of all of the resources in the group.



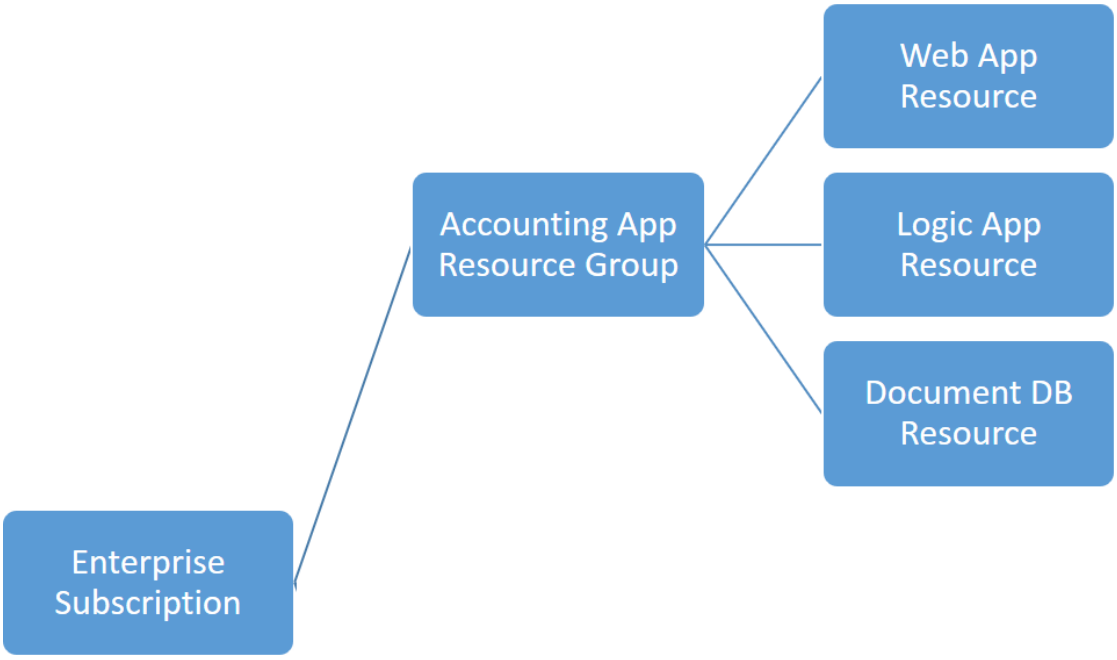
Resource Groups

Bookmark this page

Resource Groups

Resource Groups are at their simplest a container for multiple resources. There are a couple of small rules for resource groups.

- Resources can only exist in one resource group
- Resource Groups cannot be renamed
- Resource Groups can have resources of many different types (services)
- Resource Groups can have resources from many different regions.



Resource Group Deployments

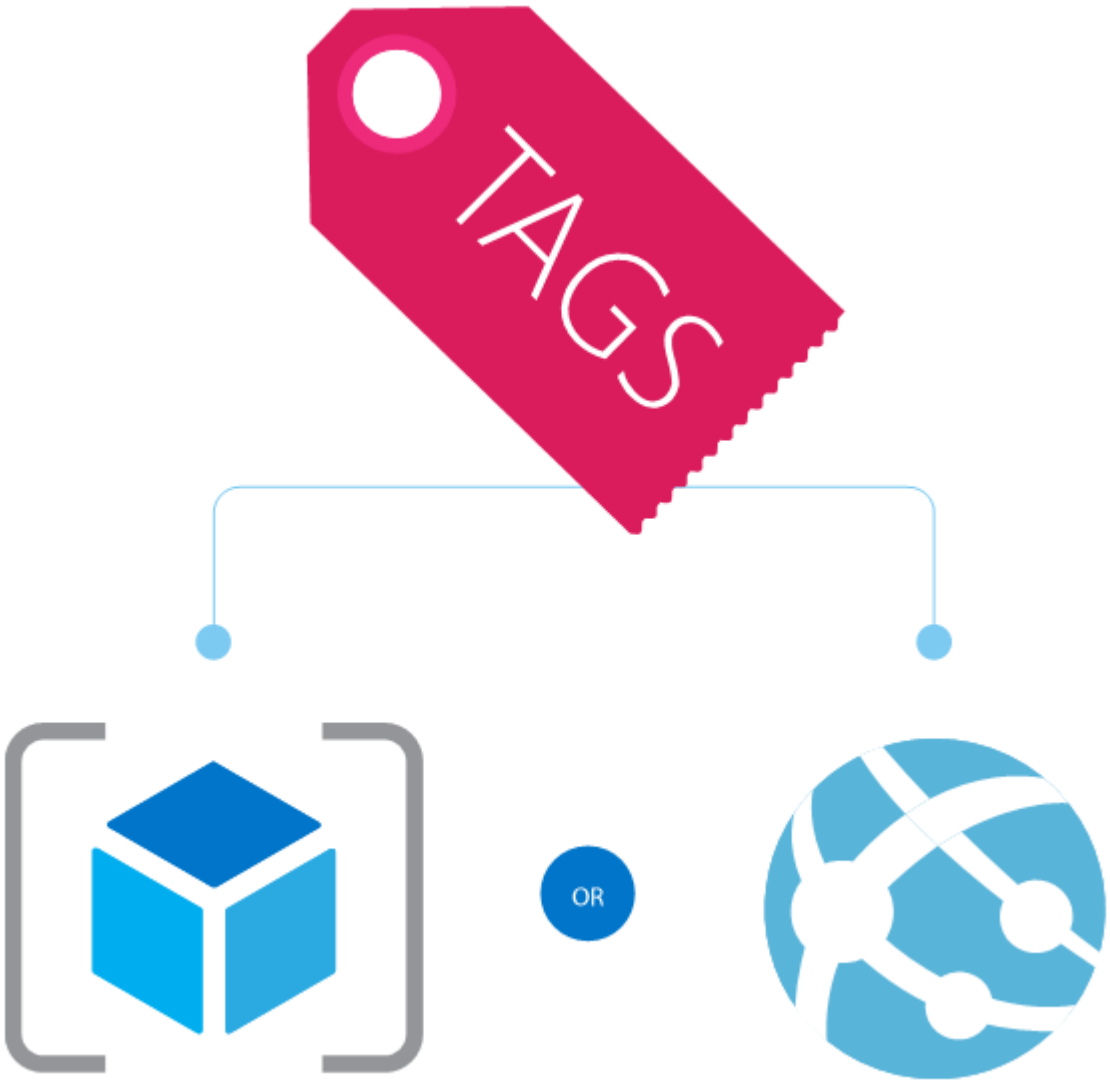
Resources can be deployed to any new or existing resource group. Deployment of resources to a resource group becomes a job where you can track the template execution. If deployment fails, the output of the job can describe why the deployment failed. Whether the deployment is a single resource to a group or a template to a group, you can use the information to fix any errors and redeploy. Deployments are **incremental**; if a resource group contains 2 web apps and you decide to deploy a third, the existing web apps will not be removed. Currently, immutable deployments are not supported in a resource group. To implement an immutable deployment, you must create a new resource group.

Tags

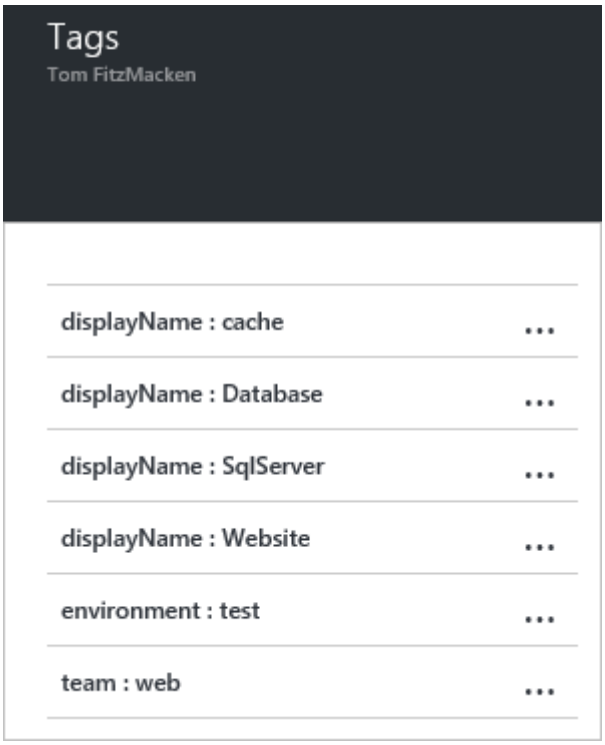
Bookmark this page

Tags

You can organize resources in Azure using subscriptions, their names or resource groups. While all of these may work, these concepts really don't scale when you are managing hundreds or thousands of resources. In scenarios, like this you need the ability to apply metadata to a resource and search for resources using the same metadata.



Using the Azure Resource Manager model, we can apply **Tags** to various resources. Each tag is a combination of a **Name** and a **Value**. These tags allow you to apply multiple name+value pairs to a resource. You can then search for resources by name tags or specific values.



For example, you may have a tag applied to all of your VMs with the name **environment** and a value corresponding to the environment for that VM. Examples could include **test**, **dev** and **prod**. Now you can search for all VMs in your Azure subscription that has a tag named **environment** with a value of **prod**.

Introducing Templates

Bookmark this page

RESOURCE GROUP TEMPLATES

Most applications that are designed to run in Microsoft Azure use a combination of resources (such as a database server, database, or website) to perform as designed. An Azure Resource Manager template makes it possible for you to deploy and manage these resources together by using a JSON description of the resources and associated deployment parameters.

The JSON file that describes the resources typically has a structure such as this:

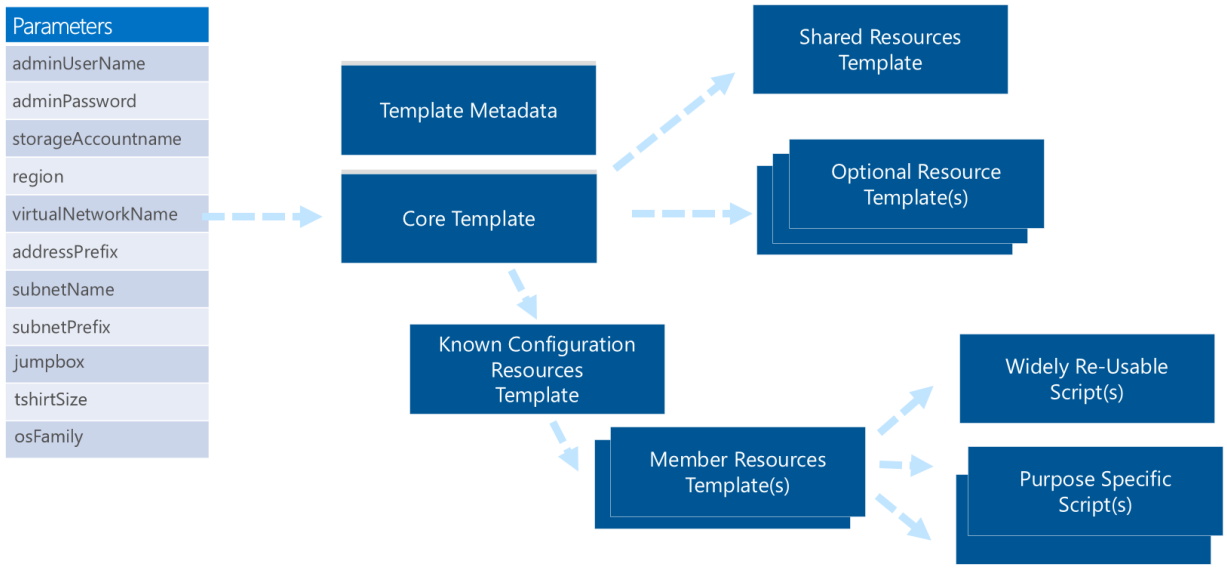
```
{

  "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",

  "contentVersion": "1.0.0.0",
```

```
  "parameters": {
    "location": {
      "type": "string",
      "allowedValues": ["East US", "West US", "West Europe", "East Asia",
"South East Asia"],
      "metadata": {
        "description": "Location to deploy to"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Compute/availabilitySets",
      "name": "availabilitySet1",
      "apiVersion": "2015-05-01-preview",
      "location": "[parameters('location')]",
      "properties": {
      }
    }
  ]
}
```

This template has a parameter that is required. The parameter is a string value that must be one of the values in the provided list of valid values. This template will create an availability set and place it in the datacenter you specify using the location parameter.

If you use a client library (SDK) or the portal to deploy resources with this template, you will specify the parameter directly. If you use PowerShell or the REST API, you will likely provide a parameters JSON file to specify your values for the parameters. An example is below:

```
{
    "location": {
        "value": "West US"
    }
}
```



COMMUNITY TEMPLATES

The Azure Resource Manager team has put together a GitHub repo of the most common scenarios where you can use an Azure Resource Manager template located here:

https://github.com/Azure/azure-quickstart-templates

For more information , you can see:
Microsoft Azure: https://aka.ms/edx-dev205bx-az34
PowerShell: https://aka.ms/edx-dev205bx-ps

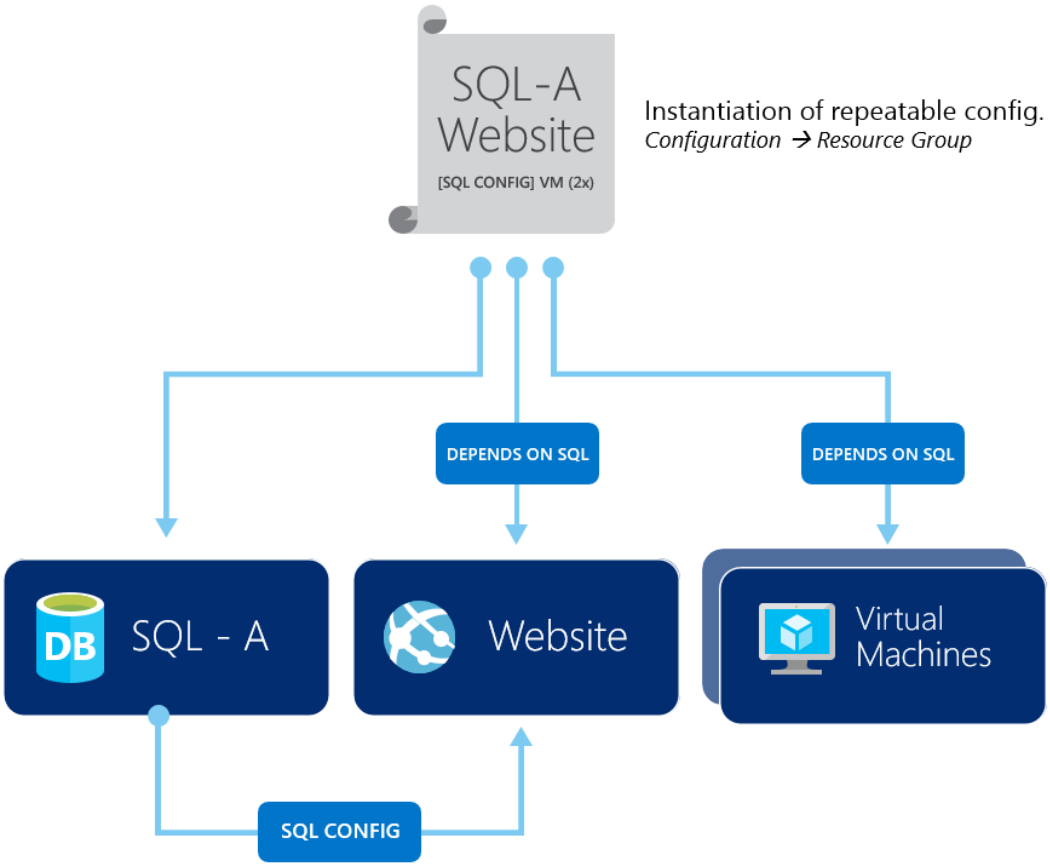
Advantages of Using Templates

Bookmark this page

ADVANTAGES OF USING TEMPLATES

Templates are generally preferred to manually deploying resources for quite a few reasons:

- A template can ensure idempotency. If you deploy an identical template to multiple resource groups, they would functionally be the same.
- A template can simplify orchestration as you only need to deploy the template to deploy all of your resources. Normally this would take multiple operations.
- A template allows you to configure multiple resources simulatenously and use variables/parameters/functions to create dependencies between resources. For example you can require that a VM is created before a Web App because you need the VM's public IP address for one of the Web App's settings. Another example is to require a Storage account is created before a VM so that you can place the VHDs in that storage account.
- A template is a JSON file so it can be compared, managed using a source control provider and used as part of any continuous integration process
- Templates can parameterize input and output values so they can be reused across many different scenarios. Templates can also be nested so you can reuse smaller templates as part of a larger orchestration.



ARM Template References

Bookmark this page

ARM Template References

When authoring ARM Templates, there are three incredible resources available on http://azure.microsoft.comthat can help you master the art of declarative templating:

https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/	This article introduces some of the major concepts associated with authoring templates for ARM.
https://azure.microsoft.com/en-us/documentation/articles/resource-group-template-functions/	This article lists many of the special functions available in ARM templates that you can use to make templates more useful and easier to author.

https://azure.microsoft.com/en-us/documentation/articles/best-practices-resource-manager-design-templates/	This article goes through some proven design practices when authoring ARM templates.

Azure Storage

Bookmark this page

Azure Storage

Azure Storage is massively scalable, so you can store and process hundreds of terabytes of data to support the big data scenarios required by scientific, financial analysis, and media applications. You can also store the small amounts of data required for a small business website as billing is calculated by usage, not capacity. Storage uses an auto-partitioning system that automatically load-balances your data based on traffic. Since storage is elastic and decoupled from your application, you can focus on your workload while relying on Storage's elastic capabilities to scale to meet demand for your applications.

All Storage services can be accessed using a REST API. Client libraries are also available for popular languages and platforms such as:

- .NET
- Java/Android
- Node.js
- PHP
- Ruby
- Python
- PowerShell

Finally, all resources in Storage can be protected from anonymous access and can be used in the Valet-Key pattern configuration discussed in previous modules.

Replication

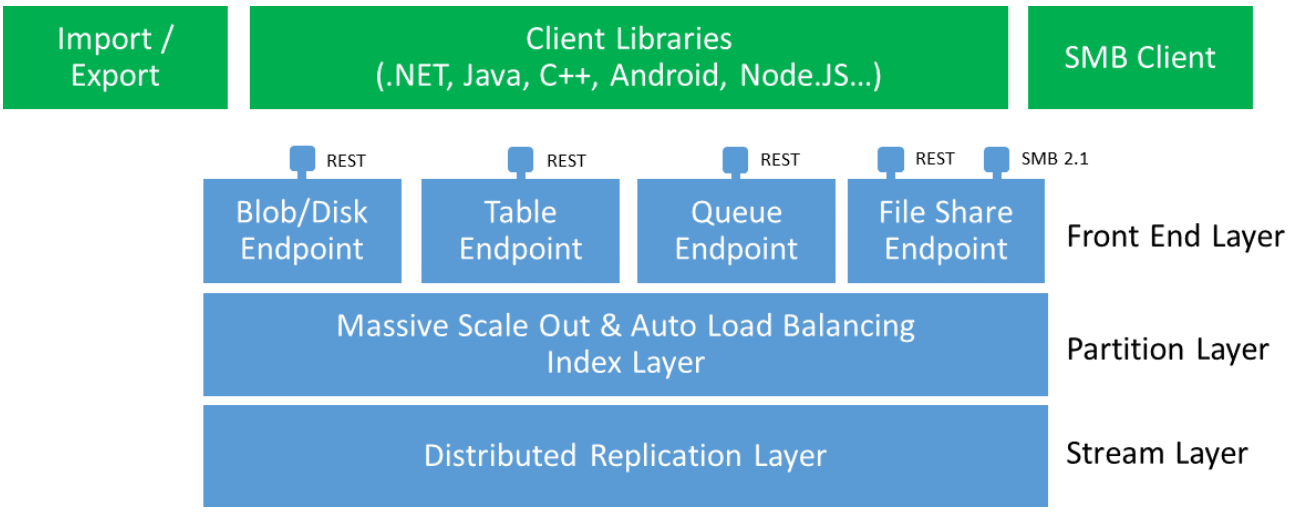
The data in your Microsoft Azure storage account is always replicated to ensure durability and high availability. At a minimum, your data is stored in triplicate. You may also choose extended replication options for scenarios where you require your data to be replicated across geography.

- **Locally redundant storage (LRS).** Locally redundant storage maintains three copies of your data. LRS is replicated three times within a single facility in a single region. LRS protects your data from normal hardware failures, but not from the failure of a single facility. LRS is the minimum amount of replication.
- **Zone-redundant storage (ZRS).** Zone-redundant storage maintains three copies of your data. ZRS is replicated three times across two to three facilities, either within a single region or across two regions, providing higher durability than LRS. ZRS ensures that your data is durable within a single region.
- **Geo-redundant storage (GRS)**. Geo-redundant storage is enabled for your storage account by default when you create it. GRS maintains six copies of your data. With GRS, your data is replicated three times within the primary region, and is also replicated three times in a secondary region hundreds of miles away from the primary region, providing the highest level of durability. In the event of a failure at the primary region, Azure Storage will failover to the secondary region. GRS ensures that your data is durable in two separate regions.
- **Read access geo-redundant storage (RA-GRS)**. Read access geo-redundant storage replicates your data to a secondary geographic location, and also provides read access to your data in the secondary location. Read-access geo-redundant storage allows you to access your data from either the primary or the secondary location, in the event that one location becomes unavailable. RA-GRS is also used in scenarios where reporting and other read-only functions can easily be distributed to the replica instead of the primary therefore spreading application load across multiple instances.

  *NOTE: Geographically distributed replicas receive any replication asynchronously. This means that your replica is **eventually consistent** and could possibly have older data if you access the replica before the replication operation from the primary is complete.*

Architecture

To learn more about Storage architecture and internals, you can read the Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency whitepaper that was released at the 2011 Association for Computing Machinery (ACM) Symposium on Operating Systems Principles.



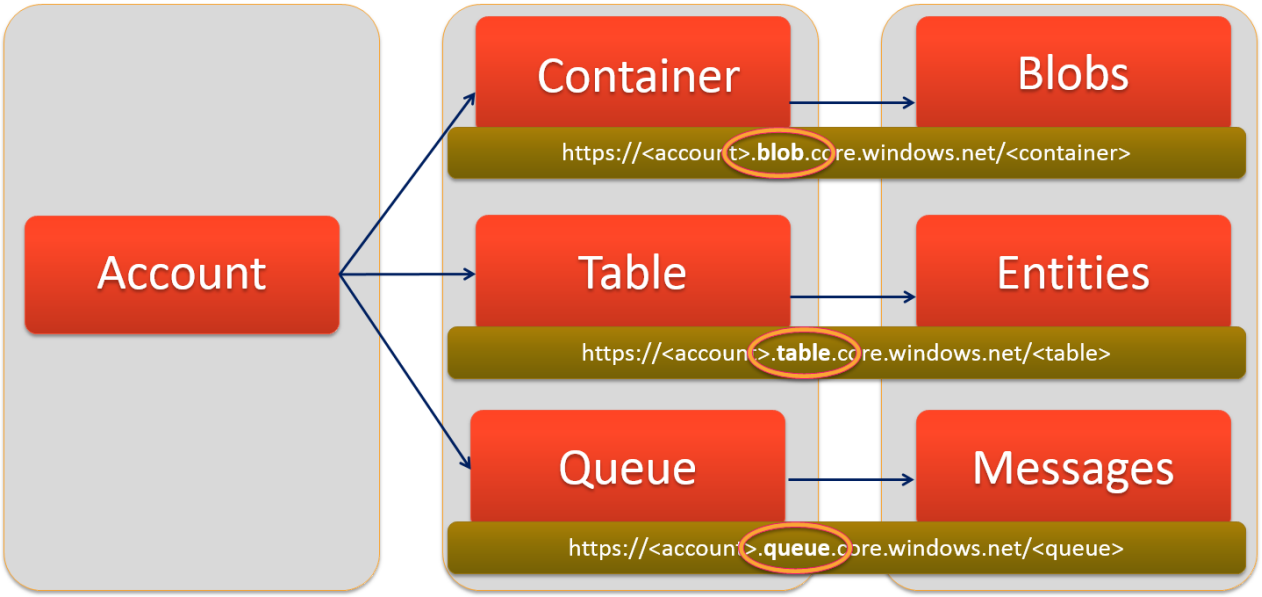
For more information , you can see:
Azure Storage: https://aka.ms/edx-dev205bx-az01
.NET: https://aka.ms/edx-dev205bx-ne
PowerShell: https://aka.ms/edx-dev205bx-ps

Types of Storage

Bookmark this page

Types of Storage

Azure Storage is comprised of three primary types of storage services:



Blobs

"Highly scalable, REST based cloud object store"

Ideal for: Data sharing, Big Data, Backups

Two Types

- **Block Blobs**: Read and write data in blocks. Optimized for sequential IO. Most cost effective Storage. Used for most blobs.
- **Page Blobs**: Sparse files, read/write in 512 byte pages. Optimized for random access. Typically used with virtual hard disks (VHD).

Tables

"Massive auto-scaling NoSQL store"

Ideal for: User, device and service metadata, structured data

Features

- Schema-less entities with strong consistency
- No limits on number of table rows or table size
- Dynamic load balancing of table regions
- Best for Key/value lookups on partition key and row key
- Entity group transactions for atomic batching

Queues

"Reliable messaging system at scale for cloud services"

Ideal for: Data sharing, Big Data, Backups

Functions

- Decouple components and scale them independently
- Scheduling of asynchronous tasks
- Building processes/work flows

Features

- No limits on number of queues or messages
- Message visibility timeout to protect from component issues U
- UpdateMessage to checkpoint progress part way through

Azure Storage Blobs

Bookmark this page

Storage Blobs

Blobs provide a way to store large amounts of unstructured, binary data, such as video, audio, images, etc. In fact, one of the features of blobs is streaming content such as video or audio. There are two types of blob storage available, each provides specific functionality:

Block Blobs

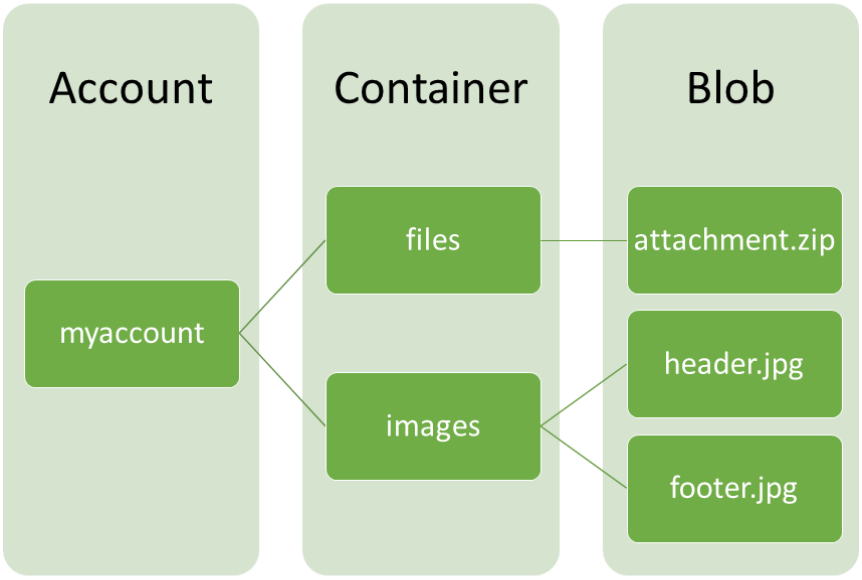
Block blobs are optimized for uploading large files in an efficient manner. Block blobs are composed of sets of blocks that are uniquely identified using a block ID. Block blobs are modified by writing a set of blocks. The set of blocks are then committed using a commit operation and the block IDs. After you upload one or more blocks to your block blob, they are associated with the blob but not committed. You must then commit the set of blocks to the block blob by again using their block IDs. Uncommitted blocks can also optionally be discarded. Block blobs are typically used for most files and multimedia.

Page Blobs

Page blobs are blobs that are composed of individual pages that are optimized for random read-write operations. These individual pages are 512 bytes. When creating a page blob, you must specify the maximum size for the blob. If you need more space for your file, you must create a new page blob. To update the page blob, you must write one or more 512-byte pages. When writing the pages, an offset is required along with a range that within the 512-byte page boundary. When updating a page blob, you can overwrite up to 4 megabytes (MB) of pages. Page blob write operations are immediately committed and overwrite the in-place pages. A page blob is typically used for virtual hard disks and can be up to 1 terabyte (TB).

Containers

A container provides a grouping of a set of blobs. Every blob is organized into a container. All blobs must be in a container as the container forms part of the blob name. A storage account can contain any number of containers, and a container can contain any number of blobs, up to the 500 TB capacity limit of the storage account. Containers also provide a useful way to assign security policies to groups of objects.



A storage account can contain a single root container named $root. This container must be explicitly created as it does not exist by default. A blob stored in the root container may be addressed without referencing the root container name, so that a blob can be addressed at the top level of the storage account hierarchy.

```
https://myaccount.blob.core.windows.net/mywebpage.html
```

REST API

Every blob uploaded to Azure Storage is associated with a relative URI. An extensive REST API for Storage is already available that allows you to manage your Storage Account and individual blobs in a RESTful manner. For blobs, this API has been extended to ensure that it is easy to access a blob by using a simple URL. You can access blobs by using the GET, PUT, POST, or DELETE HTTP methods. You can access blobs by using the following URL format:

https://[account name].blob.core.windows.net/[container name]/[blob name]

In addition to the traditional Create, Read, Update, Delete (CRUD) functionality, the metadata and properties of a blob can be accessed by using the REST API

For more information , you can see:
Azure Storage Blobs: https://aka.ms/edx-dev205bx-az14

Azure Storage Queues

Bookmark this page

Azure Storage Queues

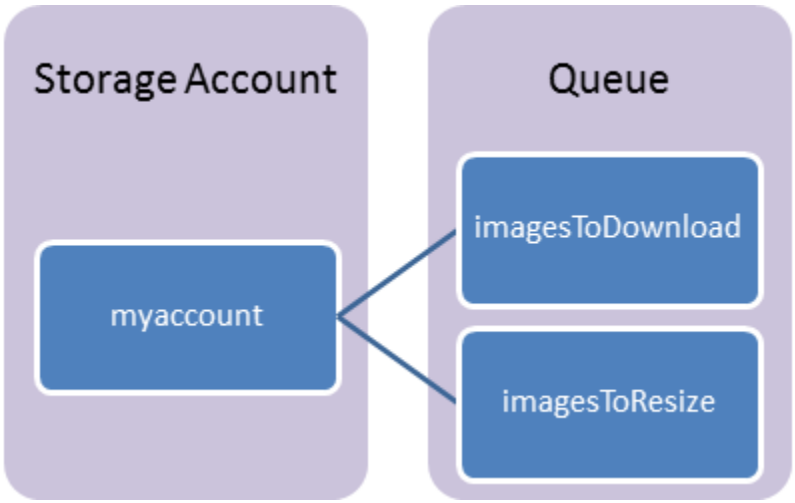
Azure Queue storage is a service for storing large numbers of messages that can be accessed from anywhere in the world via authenticated calls using HTTP or HTTPS. A single queue message can be up to 64 KB in size, and a queue can contain millions of messages, up to the total capacity limit of a storage account. A storage account can contain up to 500 TB of blob, queue, and table data.

Common uses of Queue storage include:

- Creating a backlog of work to process asynchronously
- Passing messages from an Azure Web role to an Azure Worker role

Queue Service Components

The Queue service contains the following components:



- **URL format:** Queues are addressable using the following URL format:
  `http://[account].queue.core.windows.net/<queue>`
  The following URL addresses one of the queues in the diagram:
  `http://[account].queue.core.windows.net/imagesToDownload`
- **Queue:** A queue contains a set of messages. All messages must be in a queue.
- **Message:** A message, in any format, of up to 64KB.

For more information , you can see:
Azure Queue storage: https://aka.ms/edx-dev205bx-az03

Storage Queue Message Handling

Bookmark this page

Storage Queue Message Handling

Queue messages can be managed in a variety of different ways. The following is a list of actions that you can perform on a queue or it's messages:

- **Create/Delete Queue**

  Client SDKs, PowerShell or the REST API can be used to create a new queue instance or remove an existing one.

- **Measure Queue Length**

  You can get an estimate of the number of messages in the queue. Do to race conditions and technical implementation, this count is not guaranteed and should be treated in your application as an approximate queue length.

- **Insert Message into Queue**

  New messages can be added to the queue. These messages have a body that are either a string (in UTF-8 format) or a **byte** array.

- **Retrieve the Next Message**

  A copy of the next available message is retrieved and the message is made **invisible** for a specific duration. During this time, you can process the message. Other queue consumers will not see this message in the queue when they retrieve messages while it is invisible. After a specific amount of time, the invisibility duration will elapse and the message is available to other queue consumers.

- **Extend Message Lease**

  If you need more time to process a retrieved queue message, you can return to the queue and update the invisibility duration for the queue message. This will ensure that the message is not prematurely available to other queue consumers.

- **Peek at the Next Message**

  You can retrieve a copy of the next available message in the queue without making the message invisible to other queue consumers.

- **Update a Message**

  The contents of a retrieved message can be updated in the queue if you need to have a concept of checkpoints or state for queue messages.

- **Delete a Message**

  Once you have completed processing a message, you must delete the message from the queue if you do not wish for it to be processed by any more queue consumers. Otherwise the message invisibility will timeout and the message will be processed again by other queue consumers.

Because of the workflow, queue messages and their processors/consumers must be designed to be idempotent and should not have side effects from processing the same message multiple times. For example, if a queue message contains data that needs to be stored in a SQL database, your processor should check to see if an existing record exists and then update that record or create a new record. Otherwise, you may end up with false duplicate data from your queue processors processing the same message multiple times.

Azure Storage Tables

Bookmark this page

Azure Storage Tables

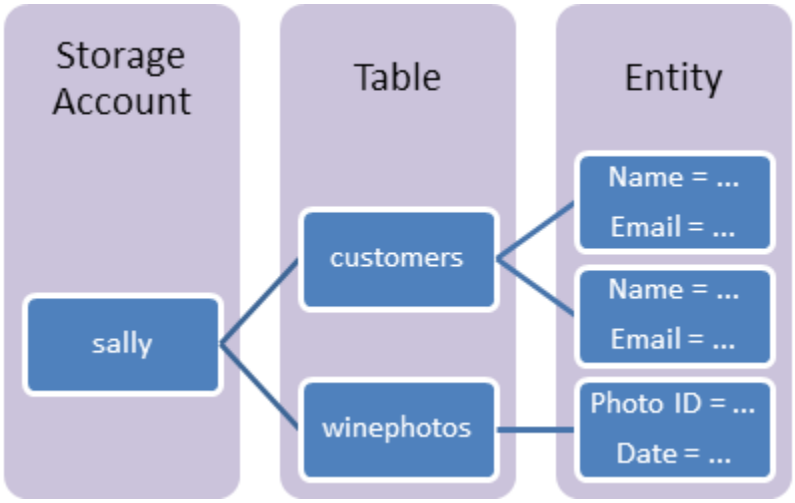
The Azure Table storage service stores large amounts of structured data. The service is a NoSQL datastore which accepts authenticated calls from inside and outside the Azure cloud. Azure tables are ideal for storing structured, non-relational data. Common uses of the Table service include:

- Storing TBs of structured data capable of serving web scale applications
- Storing datasets that don't require complex joins, foreign keys, or stored procedures and can be denormalized for fast access
- Quickly querying data using a clustered index
- Accessing data using the OData protocol and LINQ queries with WCF Data Service .NET Libraries

You can use the Table service to store and query huge sets of structured, non-relational data, and your tables will scale as demand increases.

Storage Table Components

The Table service contains the following components:



- **URL format:** Code addresses tables in an account using this address format:
  `http://<storage account>.table.core.windows.net/<table>`
  You can address Azure tables directly using this address with the OData protocol.
- **Storage Account:** All access to Azure Storage is done through a storage account.
- **Table**: A table is a collection of entities. Tables don't enforce a schema on entities, which means a single table can contain entities that have different sets of properties. The number of tables that a storage account can contain is limited only by the storage account capacity limit.
- **Entity**: An entity is a set of properties, similar to a database row. An entity can be up to 1MB in size.
- **Properties**: A property is a name-value pair. Each entity can include up to 252 properties to store data. Each entity also has 3 system properties that specify a partition key, a row key, and a timestamp. Entities with the same partition key can be queried more quickly, and inserted/updated in atomic operations. An entity's row key is its unique identifier within a partition.

Table Partitioning

Partitions represent a collection of entities with the same PartitionKey values. Partitions are always served from one partition server and each partition server can serve one or more partitions. A partition server has a rate limit of the number of entities it can serve from one partition over time. Specifically, a partition has a scalability target of 500 entities per second. This throughput may be higher during minimal load on the storage node, but it will be throttled down when the node becomes hot or very active.

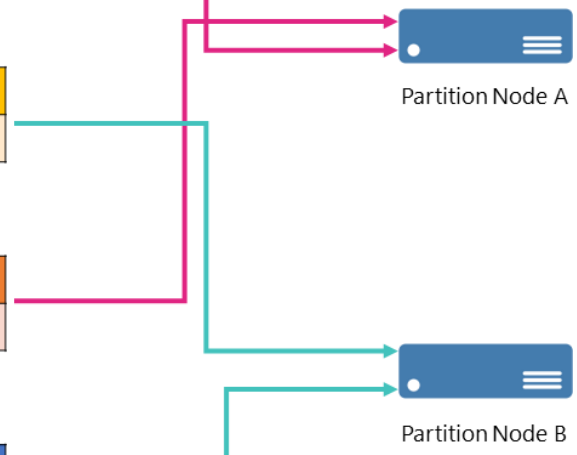
To better illustrate the concept of partitioning, the following figure illustrates a table that contains a small subset of data for users. It presents a conceptual view of partitioning where the PartitionKey contains different values:

Partition Key	Row Key	Partition Node
microsoft.com	bterkaly	A

Partition Key	Row Key	Partition Node
hotmail.com	bterkaly	B

Partition Key	Row Key	Partition Node
microsoft.com	ricardo	A

Partition Key	Row Key	Partition Node
hotmail.com	allen	B



The primary key for an Azure entity consists of the combined PartitionKey and RowKey properties, forming a single clustered index within the table. The PartitionKey and RowKey properties can store up to 1 KB of string values. Empty strings are also permitted; however, null values are not. The clustered index sorts by the PartitionKey in ascending order and then by RowKey in ascending order. The sort order is observed in all query responses.

Because a partition is always served from a single partition server and each partition server can serve one or more partitions, the efficiency of serving entities is correlated with the health of the server. Servers that encounter high traffic for their partitions may not be able to sustain a high throughput. For example, if there are many requests for Partition B, server B may become too hot. To increase the throughput of the server, the storage system load-balances the partitions to other servers. The result is that the traffic is distributed across many other servers. For optimal load balancing of traffic, you should use more partitions, so that the Azure Table service can distribute the partitions to more partition servers.

The PartitionKey values you choose will dictate how a table will be partitioned and the type of queries that can be used. Storage operations, in particular inserts, can also affect your choice of PartitionKey values. The PartitionKey values can range from single values to unique values and also can be composed from multiple values. Entity properties can be composed to form the PartitionKey value. Additionally, the application can compute the value.

Azure Files

Bookmark this page

Azure Files

File storage offers shared storage for applications using the standard **SMB 2.1 protocol**. Microsoft Azure virtual machines and cloud services can share file data across application components via mounted shares, and on-premises applications can access file data in a share via the File storage API.

Applications running in Azure virtual machines or cloud services can mount a File storage share to access file data, just as a desktop application would mount a typical SMB share. Any number of Azure virtual machines or roles can mount and access the File storage share simultaneously.

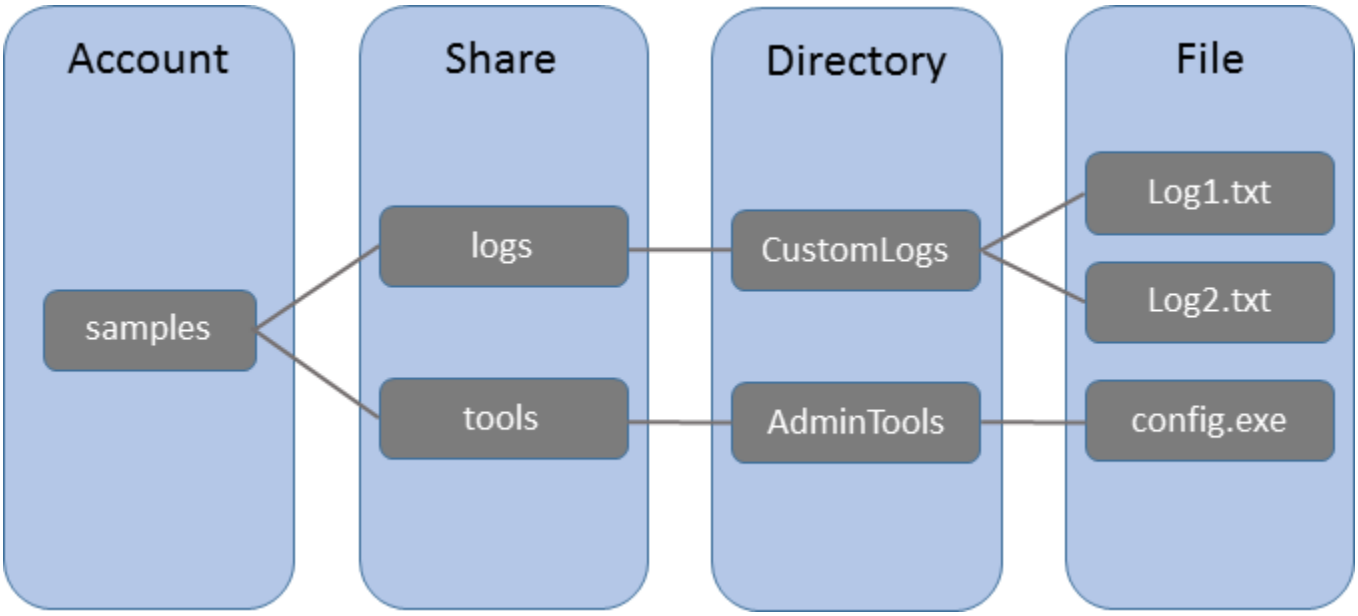
Since a File storage share is a standard SMB 2.1 file share, applications running in Azure can access data in the share via file I/O APIs. Developers can therefore leverage their existing code and skills to migrate existing applications. IT Pros can use PowerShell cmdlets to create, mount, and manage File storage shares as part of the administration of Azure applications. This guide will show examples of both.

Common uses of File storage include:

- Migrating on-premises applications that rely on file shares to run on Azure virtual machines or cloud services, without expensive rewrites
- Storing shared application settings, for example in configuration files
- Storing diagnostic data such as logs, metrics, and crash dumps in a shared location
- Storing tools and utilities needed for developing or administering Azure virtual machines or cloud services

File Components

File storage contains the following components:



- **Storage Account:** All access to Azure Storage is done through a storage account.
- **Share:** A File storage share is an SMB 2.1 file share in Azure. All directories and files must be created in a parent share. An account can contain an unlimited number of shares, and a share can store an unlimited number of files, up to the capacity limits of the storage account.
- **Directory:** An optional hierarchy of directories.
- **File:** A file in the share. A file may be up to 1 TB in size.
- **URL format:** Files are addressable using the following URL format:
  `https://[account].file.core.windows.net/<share>/<directory/directories>/<file>`
  The following example URL could be used to address one of the files in the diagram above:
  `http://[account].file.core.windows.net/logs/CustomLogs/Log1.txt`

  For more information , you can see:
  Microsoft Azure virtual machines: https://aka.ms/edx-dev205bx-az11
  cloud services: https://aka.ms/edx-dev205bx-az10

StorSimple

Bookmark this page

StorSimple

StorSimple is the combination of a service, device and management tools that can create workflows for migrating data to a cloud storage center or back on premise.

The StorSimple device is an on-premises hybrid storage array that provides primary storage and iSCSI access to data stored on it. It manages communication with cloud storage, and helps to ensure the security and confidentiality of all data that is stored on the StorSimple solution. The StorSimple device includes solid state drives (SSDs) and hard disk drives (HDDs), as well as support for clustering and automatic failover. It contains a shared processor, shared storage, and two mirrored controllers.

You can alternatively use StorSimple to create a virtual device that replicates the architecture and capabilities of the actual hybrid storage device. The StorSimple virtual device (also known as the StorSimple Virtual Appliance) runs on a single node in an Azure virtual machine.

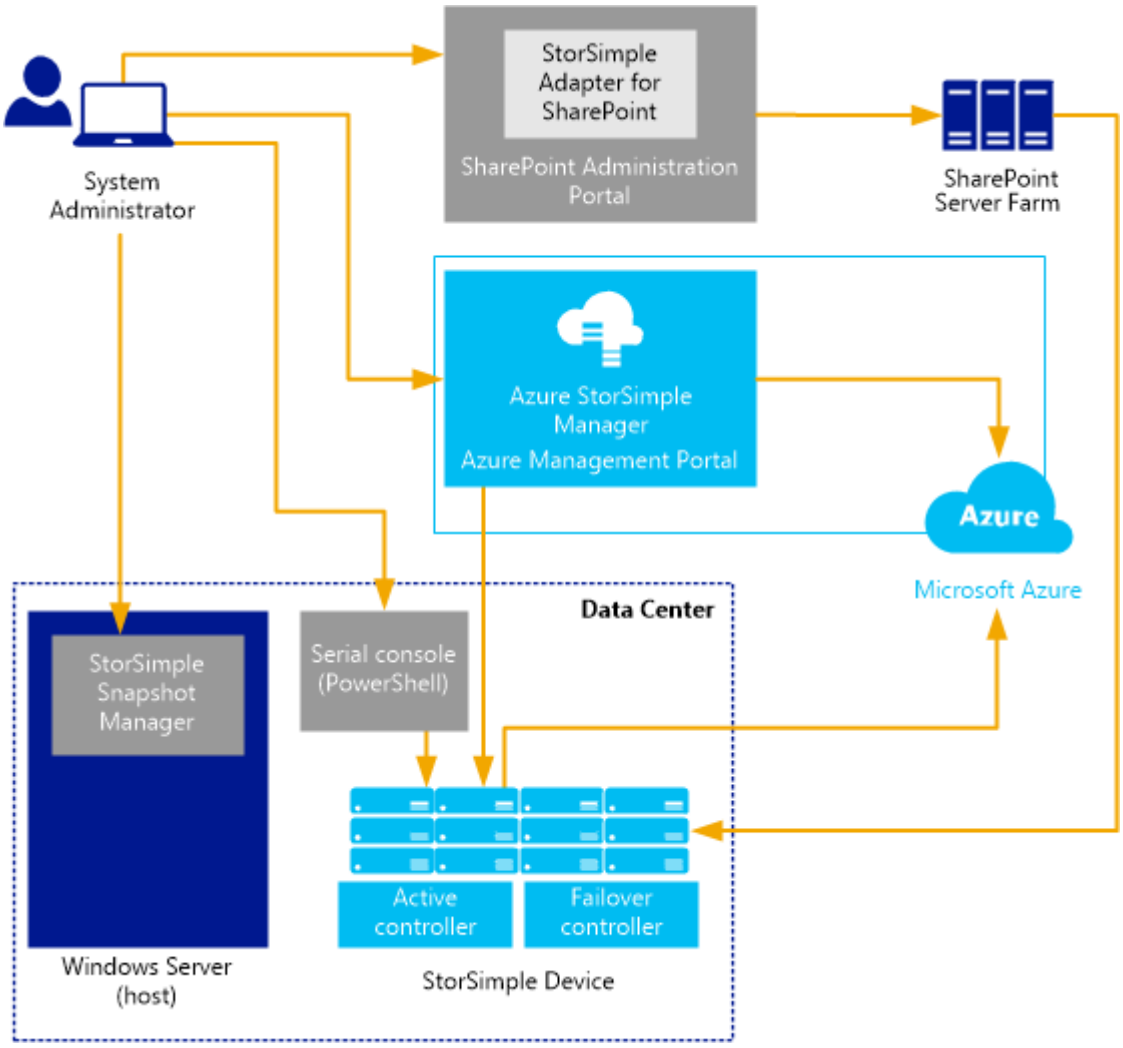
StorSimple provides a web-based user interface (the StorSimple Manager service) that enables you to centrally manage datacenter and cloud storage. You can also use a Windows PowerShell–based, command-line interface that includes dedicated cmdlets for managing your StorSimple device. Finally, you can interact with StorSimple using a

Microsoft Management Console (MMC) snap-in that's used to configure create consistent, point-in-time backup copies of local and cloud data.

Features

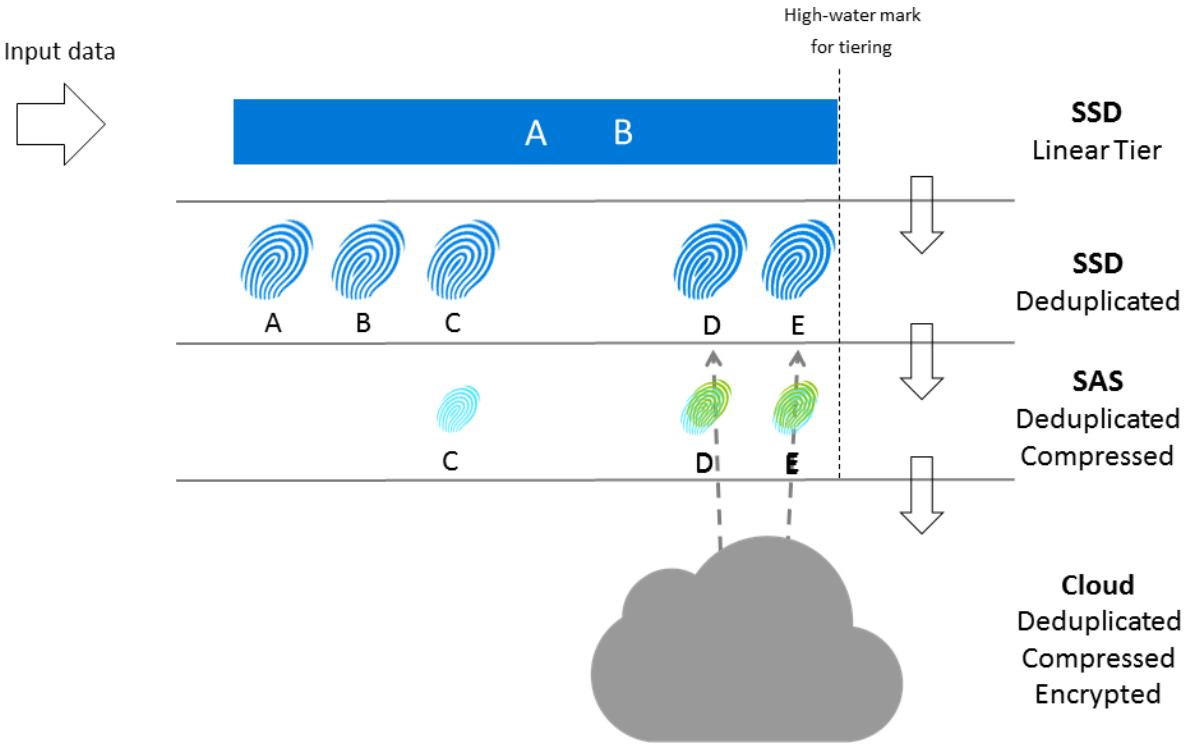
- **Transparent integration** – Microsoft Azure StorSimple uses the Internet Small Computer System Interface (iSCSI) protocol to invisibly link data storage facilities. This ensures that data stored in the cloud, at the datacenter, or on remote servers appears to be stored at a single location.
- **Reduced storage costs** – Microsoft Azure StorSimple allocates sufficient local or cloud storage to meet current demands and extends cloud storage only when necessary. It further reduces storage requirements and expense by eliminating redundant versions of the same data (deduplication) and by using compression.
- **Simplified storage management** – Microsoft Azure StorSimple provides system administration tools that you can use to configure and manage data stored on-premises, on a remote server, and in the cloud. Additionally, you can manage backup and restore functions from a Microsoft Management Console (MMC) snap-in. StorSimple provides a separate, optional interface that you can use to extend StorSimple management and data protection services to content stored on SharePoint servers.
- **Improved disaster recovery and compliance** – Microsoft Azure StorSimple does not require extended recovery time. Instead, it restores data as it is needed. This means normal operations can continue with minimal disruption. Additionally, you can configure policies to specify backup schedules and data retention.
- **Data mobility** – Data uploaded to Microsoft Azure cloud services can be accessed from other sites for recovery and migration purposes. Additionally, you can use StorSimple to configure StorSimple virtual devices on virtual machines (VMs) running in Microsoft Azure. The VMs can then use virtual devices to access stored data for test or recovery purposes.

Example StorSimple implementation. This implementation uses the StorSimple Adapter for SharePoint:



Data Tiering

StorSimple automatically tiers and classifies your data based on how often you access it. Data is constantly being shuffled between tiers as the mechanism learns about your usage patterns.



Data that is most active is stored locally, while less active and inactive data is automatically migrated to the cloud. To enable quick access, StorSimple stores very active data (hot data) on SSDs in the StorSimple device. It stores data that is used occasionally (warm data) on HDDs in the device or on servers at the datacenter. It moves inactive data, backup data, and data retained for archival or compliance purposes to the cloud. StorSimple adjusts and rearranges data and storage assignments as usage patterns change. For example, some information might become less active over time. As it becomes progressively less active, it is migrated from SSD to HDD and then to the cloud. If that same data becomes active again, it is migrated back to the storage device.

Security in Azure Storage

Bookmark this page

Container Security

Typically, only the owner of a storage account can access resources within that account. If your service or application needs to make these resources available to other clients, you have various options available. First, you can make the public access key generally available. This is not typically recommended as this key gives individuals full access to your entire storage account and its management operations. Another, more common option is to manage access for the entire container. This access can be managed using the Public Read Access property of a specific container.

	Anonymous Access			Access With Key
	Enumerate Containers	Enumerate Container Blobs	Read Blob	Read Blob
Container		✓	✓	✓
Blob			✓	✓
Off				✓

The **Public Read Access** property controls what data is available anonymously for your container. You can select the following values for the Public Read Access setting:

- **Container.** Blobs in a container can be enumerated. The container metadata is also accessible. Individual blobs within this container and their properties can also be accessed with this setting.
- **Blob.** Only individual blobs and their properties in this container can be accessed. Blobs are not allowed to be enumerated.
- **Off.** With this setting, enumeration of blobs is not allowed. Individual blobs and their properties are also not accessible. You must use your access keys to access any data about this container or its blobs.

Shared Access Signatures

A shared access signature is a URI that grants restricted access rights to containers, blobs, queues, and tables. You can provide a shared access signature to clients who should not be trusted with your storage account key but to whom you wish to delegate access to certain storage account resources. By distributing a shared access signature URI to these clients, you can grant them access to a resource for a specified period of time, with a specified set of permissions.

A shared access signature can grant any of the following operations to a client that possesses the signature:

- Reading and writing page or block blob content, block lists, properties, and metadata
- Deleting, leasing, and creating a snapshot of a blob
- Listing the blobs within a container
- Adding, removing, updating, and deleting queue messages
- Getting queue metadata, including the message count

- Querying, adding, updating, deleting, and upserting table entities
- Copying to a blob from another blob within the same account

The shared access signature URI query parameters incorporate all of the information necessary to grant controlled access to a storage resource. The URI query parameters specify the time interval over which the shared access signature is valid, the permissions that it grants, the resource that is to be made available, and the signature that the storage services should use to authenticate the request.

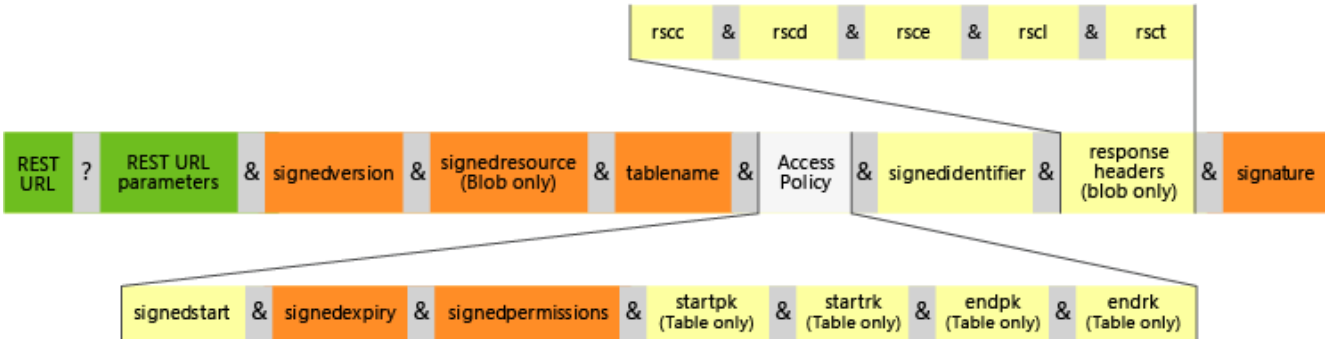
For example, you may wish to have your client application access a resource (container: **pictures**, blob: **profile.jpg**) at this URI:

```
GET https://[account].blob.core.windows.net/pictures/profile.jpg
```

You could give the client application your access key to manage your entire subscription or make the **pictures**container anonymously accessible. Neither of these are ideal solutions so you instead chose to generate a shared access signature. You simply append the generated signature to the end of your URI like thus:

```
GET https://[account].blob.core.windows.net/pictures/profile.jpg?sv=2012-02-12&st=2009-02-09&se=2009-02-10&sr=c&sp=r&si=YWJjZGVmZw%3d%3d&sig=dD80ihBh5jfNpymO5Hg1IdiJIEvHcJpCMiCMnN%2fRnbI%3d
```

The signature is using version 2012-02-12 of the storage API, it allows read access beginning from 02/09/09 to 02/10/09 to the container.



Now the resource is temporarily accessible by this particular client.

Stored Access Policies

Azure SAS also supports server-stored access policies that can be associated with a specific resource such as a table or blob. This feature provides additional control and flexibility compared to application-generated SAS tokens, and should be used whenever possible. **Settings defined in a server-stored policy can be changed and are reflected in the token without requiring a new token to be issued, but settings**

defined in the token itself cannot be changed without issuing a new token.This approach also makes it possible to revoke a valid SAS token before it has expired.

A stored access policy provides an additional level of control over shared access signatures on the server side. Establishing a stored access policy serves to group shared access signatures and to provide additional restrictions for signatures that are bound by the policy. You can use a stored access policy to change the start time, expiry time, or permissions for a signature, or to revoke it after it has been issued.

New shared access signatures are then generated from an existing stored access policy. A maximum of five access policies may be set on a container, table, or queue at any given time. To revoke a stored access policy, you can either delete it, or rename it by changing the signed identifier. Changing the signed identifier breaks the associations between any existing signatures and the stored access policy. Deleting or renaming the stored access policy immediately effects all of the shared access signatures associated with it.

Best Practices

https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-shared-access-signature-part-1/#best-practices-for-using-shared-access-signatures

Introducing Mobile Apps

Bookmark this page

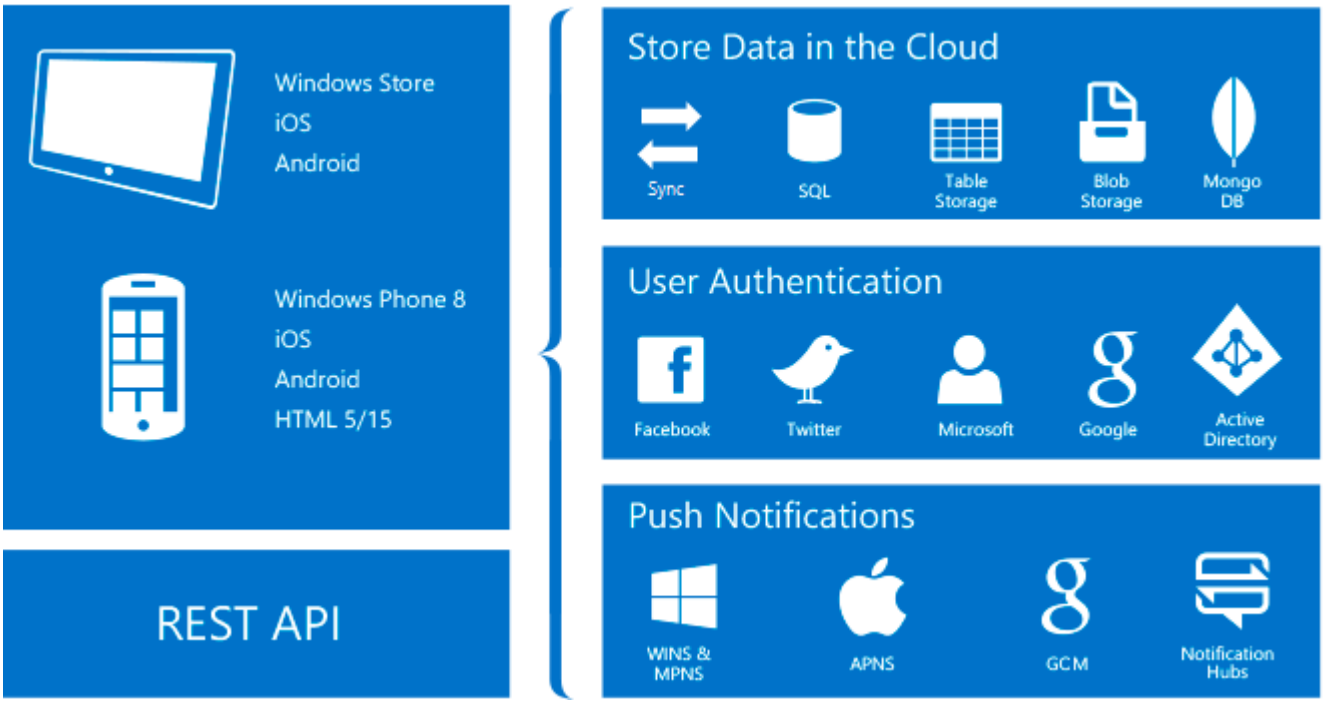
Introducing Mobile Apps

Azure Mobile Apps is a component of Azure App Services offering designed to make it easy to create highly-functional mobile apps using Azure. Mobile Apps brings together a set of Azure services that enable backend capabilities for your apps. Mobile Apps provides the following backend capabilities in Azure to support your apps:

- **Single Sign On** - Select from an ever-growing list of identity providers including Azure Active Directory, Facebook, Google, Twitter, and Microsoft Account, and leverage Mobile Apps to add authentication to your app in minutes.
- **Offline Sync** - Mobile Apps makes it easy for you to build robust and responsive apps that allow employees to work offline when connectivity is not available, and synchronize with your enterprise backend systems when devices comes back online. Offline sync capability is supported on all client platforms and works with any data source including SQL, Table Storage, Mongo, or Document DB, and any SaaS API including Office 365, Salesforce, Dynamics, or on-premises databases.
- **Push Notifications** - Mobile Apps offers a massively scalable mobile push notification engine, Notification Hubs, capable of sending millions of personalized push notifications

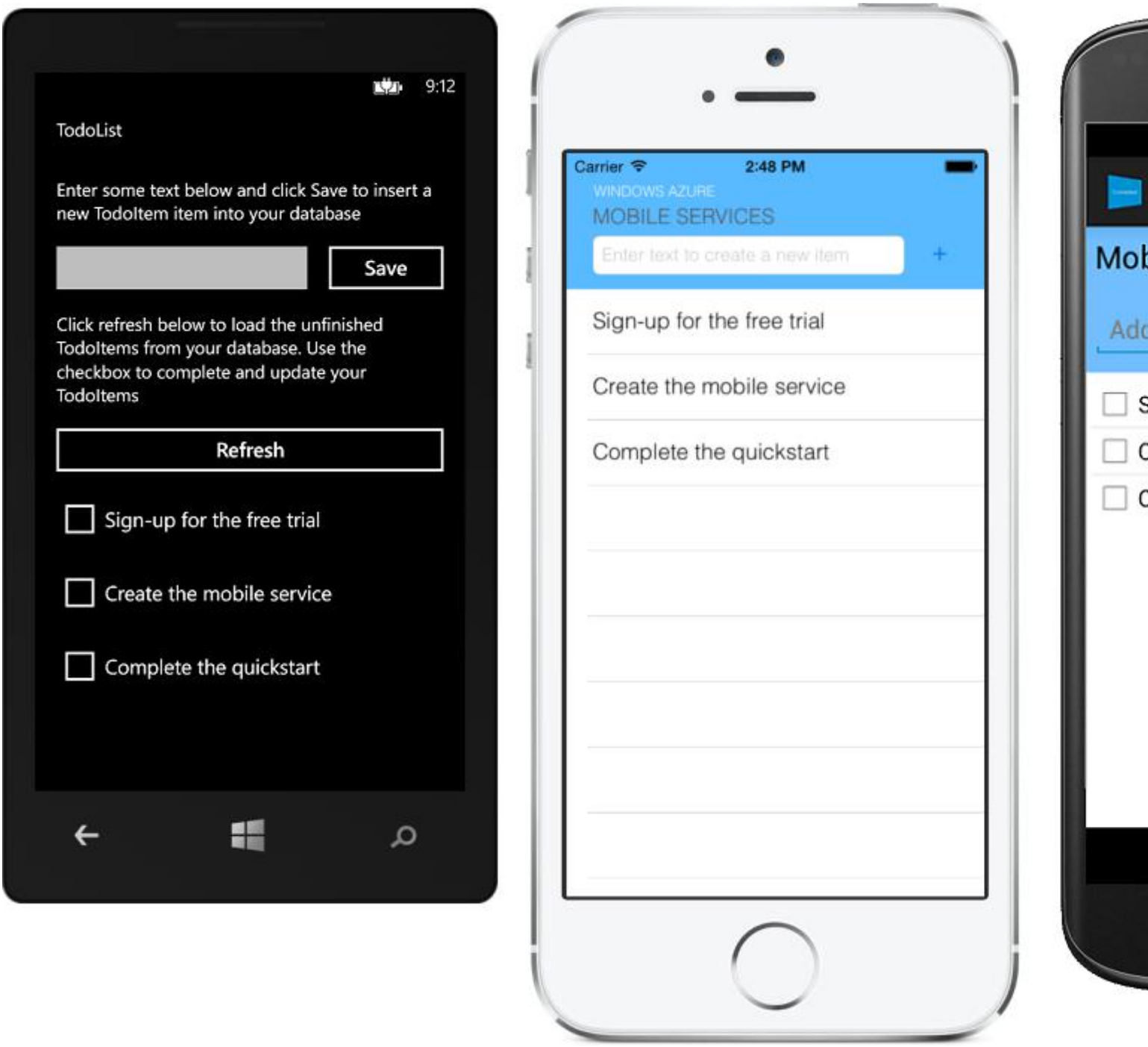
to dynamic segments of audience using iOS, Android, Windows, or Kindle devices within seconds. You can easily hook Notification Hubs to any existing app backend, whether that backend is hosted on-premises or in the cloud.

- **Auto Scaling** - App Service enables you to quickly scale-up or out to handle any incoming customer load. Manually select the number and size of VMs or set up auto-scaling to scale your mobile app backend based on load or schedule.



Client SDKs

Mobile App endpoints are, at their simplest, REST APIs and can be used on a wide variety of platforms and with a wide variety of devices. Client SDKs are available, however, if you like to connect your mobile application to a Mobile App instance for it's backend data.



Mobile client SDKs are available for the following platforms:

- Xamarin Android/IOS
- Android Native
- IOS Native
- Windows Store
- Windows Phone
- .NET

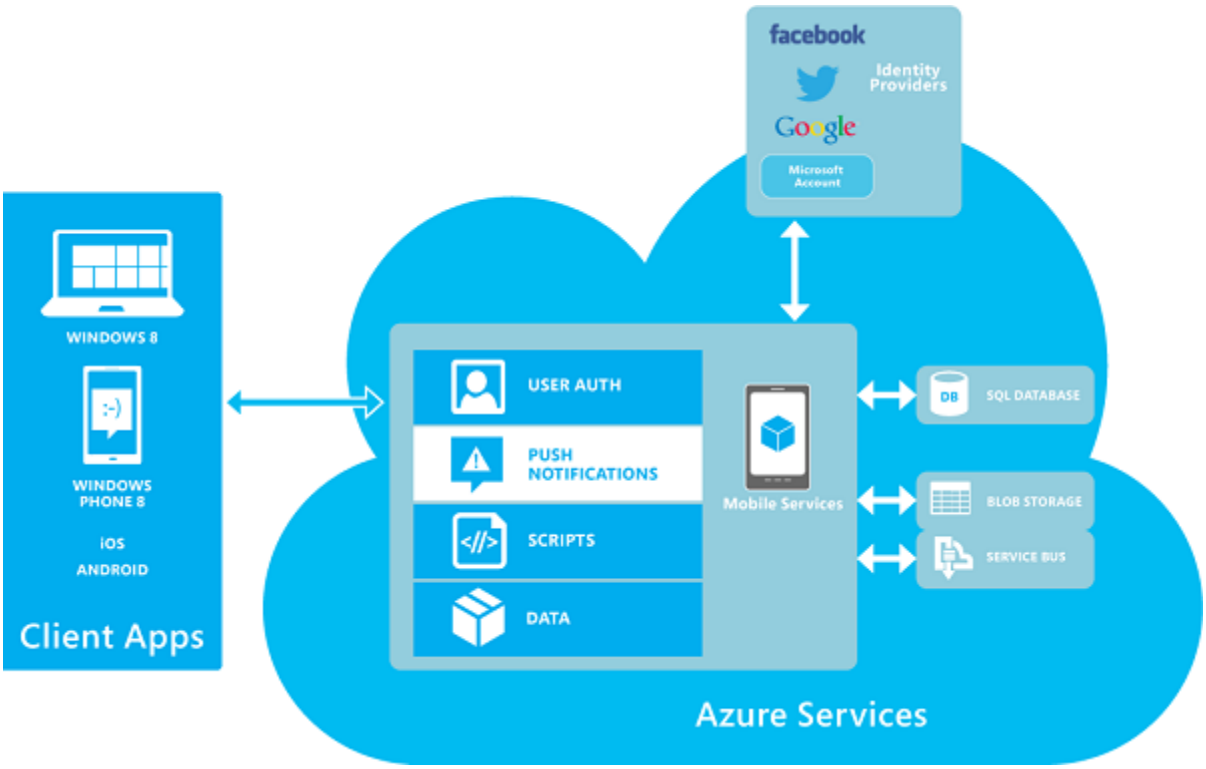
- HTML

Architecture

The following diagram illustrates all of the services that combine together to form the Mobile Apps offering. The focus with Mobile Apps is to integrate many different Azure services and features in an accessible manner. You could deploy a Mobile App for a production application without having to write any custom code or logic. Mobile Apps combines the following Azure services:

- SQL Database
- Storage Blobs
- Azure Active Directory
- Service Bus Notification Hubs

Mobile Apps can be customized in either JavaScript (Node.js) or .NET (Web API). The custom code can either be edited manually or managed using continuous delivery from a source control provider.



Tables

In Microsoft Azure Mobile Services, data is stored in tables, which are maintained in the Microsoft Azure SQL Database that is associated with your mobile service. You can add one or more tables to your mobile service. In a JavaScript backend mobile service, click the **Create** button to add a new table. When using a .NET backend, your data model is defined in your mobile service project in Visual Studio using Entity Framework.

For a data table, you can access it using the following URI:

```
https://[account].azure-mobile.net/tables/[table]
```

GET, PUT, POST and DELETE HTTP operations will directly add, remove or update records in the SQL database. All of this is done without the need for any custom code if you are using a JavaScript backend mobile service. If you are using a .NET backend mobile service, you must create a controller to enable this functionality.

Mobile Services simplifies the process of storing data in a SQL Database. When using a JavaScript backend, you don't need to predefine the schema of tables in your database during development. Mobile Services automatically adds columns to a table based on the data you insert. Dynamic schema is enabled by default. With dynamic schema, you do not need to specify column names or their types. New columns are automatically created based on the data in insert or update requests. When a column with the same name does not already exist, a column is created with the data type inferred based on the property of the received JSON object

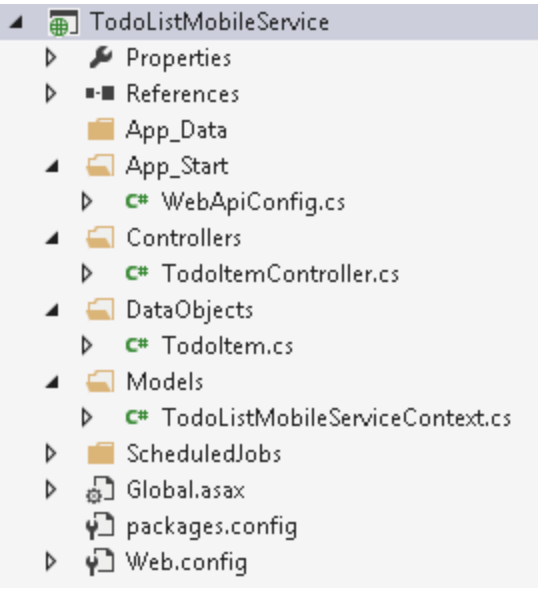
For more information , you can see:
Azure App Service: https://aka.ms/edx-dev205bx-az07
Azure Active Directory: https://aka.ms/edx-dev205bx-azad
Microsoft Account: https://aka.ms/edx-dev205bx-msa
Document DB: https://aka.ms/edx-dev205bx-az06
Office 365: https://aka.ms/edx-dev205bx-o365
Notification Hubs: https://aka.ms/edx-dev205bx-az15
Microsoft Account: https://aka.ms/edx-dev205bx-msa
.NET: https://aka.ms/edx-dev205bx-ne
SQL Database: https://aka.ms/edx-dev205bx-asd
Azure Active Directory: https://aka.ms/edx-dev205bx-azad
Service Bus: https://aka.ms/edx-dev205bx-az04
Microsoft Azure Mobile Services: https://aka.ms/edx-dev205bx-az16
Visual Studio: https://aka.ms/edx-dev205bx-vs

.NET Mobile Apps

Bookmark this page

.NET Mobile Apps

Mobile Apps can be developed using .NET as an alternative to JavaScript. When using this strategy, you must create a Web API project (using Entity Framework) in Visual Studio and deploy that to your Mobile App instance.



A template is available in Visual Studio that will crate a Web API project that is specifically configured for Mobile Apps. This project will have references to the Mobile Apps NuGet libraries. Within this project, you will see two types of classes, **DataObjects** and **Controllers.**

Data Objects

TodoItem represents the data type and maps onto a single database table. The class inherits from EntityData, which means this data object maps onto a mobile service table and can be used as a template type with the table controller, `TableController<TEntityData>`.

```
using Microsoft.WindowsAzure.Mobile.Service;

public class TodoItem : EntityData
{
      public String Text { get; set; }

      public bool Complete { get; set; }
}
```

Controllers

The TodoItemController class is a Web API controller. It's a special kind of Web API controller, since it inherits from TableController, which is designed to work with mobile services tables. The class contains methods that implement a REST api which your mobile service clients call to create, update, read, or delete data. REST is a preferred design pattern for web services and is the design pattern used by mobile services table controllers.

```
...
using Microsoft.WindowsAzure.Mobile.Service;

public class TodoItemController : TableController
```

```
{
     public IQueryable<TodoItem> GetAllTodoItems()
      {
           return Query();
      }

      public async Task<IHttpActionResult> PostTodoItem(TodoItem item)
     {
           TodoItem current=await InsertAsync(item);
           return CreatedAtRoute("Tables", new { id=current.Id }, current);
     }

      public Task DeleteTodoItem(string id)
     {
           return DeleteAsync(id);
     }
}
```

JavaScript Mobile Apps

Bookmark this page

JavaScript Mobile Apps

In a JavaScript backend Mobile App, you can define custom business logic as JavaScript code that's stored and executed on the server. This server script code is ran using Node.js. The signature of the main function in the server script depends on the context of where the script is used. For example, a script to handle GET requests would look like this:

```
function read(query, user, request) {
      request.respond(statusCodes.OK, 'Here is your response');
}
```

Because Mobile Apps uses Node.js on the server, your scripts already have access to the built-in Node.js modules. You can also use source control to define your own modules or add other Node.js modules to your service. Popular Node.js modules (such as Express) can be imported and used in your scripts. Some of the modules that you can use in your scripts include:

- **azure**: Exposes the functionality of the Azure SDK for Node.js.
- **crypto**: Provides crypto functionality of OpenSSL.
- **path**: Contains utilities for working with file paths.
- **querystring**: Contains utilities for working with query strings.
- **request**: Sends HTTP requests to external REST services, such as Twitter and Facebook.
- **sendgrid**: Sends email by using the Sendgrid email service in Azure.
- **url**: Contains utilities to parse and resolve URLs.

- **util**: Contains various utilities, such as string formatting and object type checking.
- **zlib**: Exposes compression functionality, such as gzip and deflate.

Table Operations

A table operation script is a server script that is registered to an operation on a table—insert, read, update, or delete (*del*). The name of the script must match the kind of operation for which it is registered. Only one script can be registered for a given table operation. The script is executed every time that the given operation is invoked by a REST request—for example, when a POST request is received to insert an item into the table.

```
function insert(item, user, request) {
      if (item.text.length > 10) {
             request.respond(statusCodes.BAD_REQUEST, 'Text length must be
less than 10 characters');
      }
      else {
            request.execute();
      }
}
```

For a table named **greeting**, This code is invoked by sending a POST request to the following URL:

```
https://[instance].azure-mobile.net/table/greeting
```

Mobile Apps does not preserve state between script executions. Because a new global context is created every time a script is run, any state variables that are defined in the script are reinitialized. A table script function always takes three arguments.

- The first argument varies depending on the table operation.
- For inserts and updates, it is an **item** object, which is a JSON representation of the row being affected by the operation. This allows you to access column values by name, for example, *item.Owner*, where *Owner* is one of the names in the JSON representation.
- For a delete, it is the **ID** of the record to delete.
- And for a read, it is a **query** object that specifies the rowset to return.
- The second argument is always a user object that represents the user that submitted the request.
- The third argument is always a request object, by which you can control execution of the requested operation and the response that's sent to the client.

Custom API

A custom API is an endpoint in your Mobile App that is accessed by one or more of the standard HTTP methods: GET, POST, PUT, PATCH, DELETE. A separate function export can be defined for each HTTP method supported by the custom API, all in a single script file. The registered script is invoked when a request to the custom API using the given method is received.

When custom API functions are called by the Mobile Apps runtime, both a request and response object are supplied. These objects expose the functionality of the Express.js library, which can be leveraged by your scripts. The following custom API named **hello** is a very simple example that returns *Hello, world!* in response to a POST request:

```
exports.post = function(request, response) {
     response.send(200, "{ message: 'Hello, world!' }");
}
```

The **send** function on the response object returns your desired response to the client. This code is invoked by sending a POST request to the following URL:

```
https://[instance].azure-mobile.net/api/hello
```

For more information , you can see:
Mobile Apps: https://aka.ms/edx-dev205bx-az17
Azure SDK: https://aka.ms/edx-dev205bx-az02

Introducing No-SQL

Bookmark this page

Introducing No-SQL

Many modern application workloads need to store large amounts of data that may not be well structured or even deduplicated. These large amounts of data need to be stored or read in bulk and in a very performant manner. Most traditional relational databases are based on the concepts of **ACID** (atomicity, consistency, isolation and durability) which can be restrictive when trying to solve these problems. ACID concerns are why the storage and retrieval of records in databases such as SQL can become very complicated. The **CAP** theorem states that databases may only excel at two out of three attribtues:

- **Consistency** (all nodes see the same data at the same time)
- **Availability** (a guarantee that every request receives a response about whether it succeeded or failed)
- **Partition tolerance** (the system continues to operate despite arbitrary partitioning due to network failures)

Wikipedia, https://en.wikipedia.org/wiki/CAP_theorem

Relational databases excel in consistency and partition tolerance but they suffer in availability in their efforts to maintain consistency. NoSQL databases are a class of database engines optimized to handle high-performance, scalable workloads with extremely simple storage and retrieving operations. They excel at availability and partition tolerance and often de-emphasizes consistency in favor of performance.

From an application architect perspective, relational databases are ideal for scenarios where you need highly consistent data. NoSQL databases are ideal for scenarios where you need to store and retrieve large amounts of data and you need the transactions to be both resilient and fast.

NoSQL Storage Mechanisms

Downloads and transcripts

Document Databases

There are many types of NoSQL stores. For the next few units, we will focus on **Document Databases**.

A document database is similar in concept to a key/value store except that the values stored are documents. A document is a collection of named fields and values, each of which could be simple scalar items or compound elements such as lists and child documents. The data in the fields in a document can be encoded in a variety of ways, including XML, YAML, JSON, BSON, or even stored as plain text.

The fields in the documents are exposed to the database management system, enabling an application to query and filter data by using the values in these fields.

Row Key	Document
1001	OrderDate: 06/06/2013 OrderItems: ProductID: 2010 Quantity: 2 Cost: 520 ProductID: 4365 Quantity: 1 Cost: 18 OrderTotal: 1058 Customer ID: 99 ShippingAddress: StreetAddress: 999 500th Ave City: Bellevue State: WA ZipCode: 12345
1002	OrderDate: 07/07/2013 OrderItems: ProductID: 1285 Quantity: 1 Cost: 120 OrderTotal: 120 Customer ID: 220 ShippingAddress: StreetAddress: 888 W. Front St City: Boise State: ID ZipCode: 54321

The database in this example is designed to optimize query access to sales order information. A problem that a relational database specialist might have with this example concerns the shipping address that is embedded into the sales order. In a typical relational database, this information would be held in a separate table, enabling it to be modified without having to hunt down all the orders that reference it. However, the relational approach is not necessarily an advantage.

A sales order is an historical document, and the shipping address should indicate where the order was actually sent. If the address of the customer changes at a later date, the shipping address for orders already fulfilled should not be changed (indeed, doing so may break the auditing requirements of the system). Implementing this functionality in a relational database would require maintaining version histories and timestamps for each shipping address, and each query that required the shipping address for an order would be more complicated as a result. The document database is more naturally tuned to this type of information as a typical document contains the entire data for an entity, and queries are simpler as a result.

DocumentDB

Bookmark this page

What is Azure DocumentDB?

Channel 9: https://channel9.msdn.com/Blogs/Windows-Azure/What-is-Azure-DocumentDB

DocumentDB

Modern applications produce, consume and respond quickly to very large volumes of data. These applications evolve very rapidly and so does the underlying data schema. In response to this, developers have increasingly chosen schema-free NoSQL document databases as simple, fast, scalable solutions to store and process data while preserving the ability to quickly iterate over application data models and unstructured data feeds. However, many schema-free databases do not allow for complex queries and transactional processing, making advanced data management difficult.

DocumentDB is a NoSQL document database service designed both as a highly scalable and available document store and higher levels of consistency than traditional NoSQL databases. DocumentDB is designed to consistently fast reads and writes, schema flexibility, and the ability to easily scale a database up and down on demand. DocumentDB enables complex ad hoc queries using a SQL language, supports well defined consistency levels, and offers JavaScript language integrated, multi-document transaction processing using the familiar programming model of stored procedures, triggers, and UDFs.

Individual entities in DocumentDB are stored and manipulated as JSON documents.

designed, built and optimized for JSON

For more information , you can see:
Document DB: https://aka.ms/edx-dev205bx-az06

DocumentDB Features

Bookmark this page

SQL

Microsoft Azure DocumentDB supports querying documents using SQL (Structured Query Language) over hierarchical JSON documents. DocumentDB is truly schema-free. By

virtue of its commitment to the JSON data model directly within the database engine, it provides automatic indexing of JSON documents without requiring explicit schema or creation of secondary indexes.

For example, here is a collection with two documents. They do not have to conform to any specific schema:

```
[
     { "id": "AndersenFamily", "lastName": "Andersen" },
     { "id": "WakefieldFamily", "address": { "state": "NY", "county":
"Manhattan", "city": "NY" } }
]
```

You can query this collection with the following query:

```
SELECT * FROM Families f WHERE f.id = "AndersenFamily"
```

Which will return this result:

```
{ "id": "AndersenFamily", "lastName": "Andersen" }
```

Since DocumentDB SQL works on JSON values, it deals with tree shaped entities instead of rows and columns. Therefore, the language lets you refer to nodes of the tree at any arbitrary depth (**Node1.Node2.Node3…NodeM**), such as **Node2.Address.County**. SQL queries in DocumentDB conform to the ANSI-SQL standards and they require a SELECT clause with optional FROM or WHERE clauses. Advanced operations and keywords are available for more complex queries.

Consistency Levels

DocumentDB offers four well-defined consistency levels with associated performance levels. In most real world scenarios, applications benefit from making fine grained trade-offs between consistency, availability, and latency. This allows application developers to make predictable consistency-availability-latency trade-offs. The four consistency levels are listed below:

- **Strong**: Strong consistency guarantees that a write is only visible after it is committed durably by the majority quorum of replicas. A write is either synchronously committed durably by both the primary and the quorum of secondaries or it is aborted. A read is always acknowledged by the majority read quorum - a client can never see an uncommitted or partial write and is always guaranteed to read the latest acknowledged write.
- **Bounded staleness**: Bounded staleness consistency guarantees the total order of propagation of writes with the possibility that reads lag behind writes by at most K prefixes. The read is always acknowledged by a majority quorum of replicas. The response of a read request specifies its relative freshness (in terms of K).

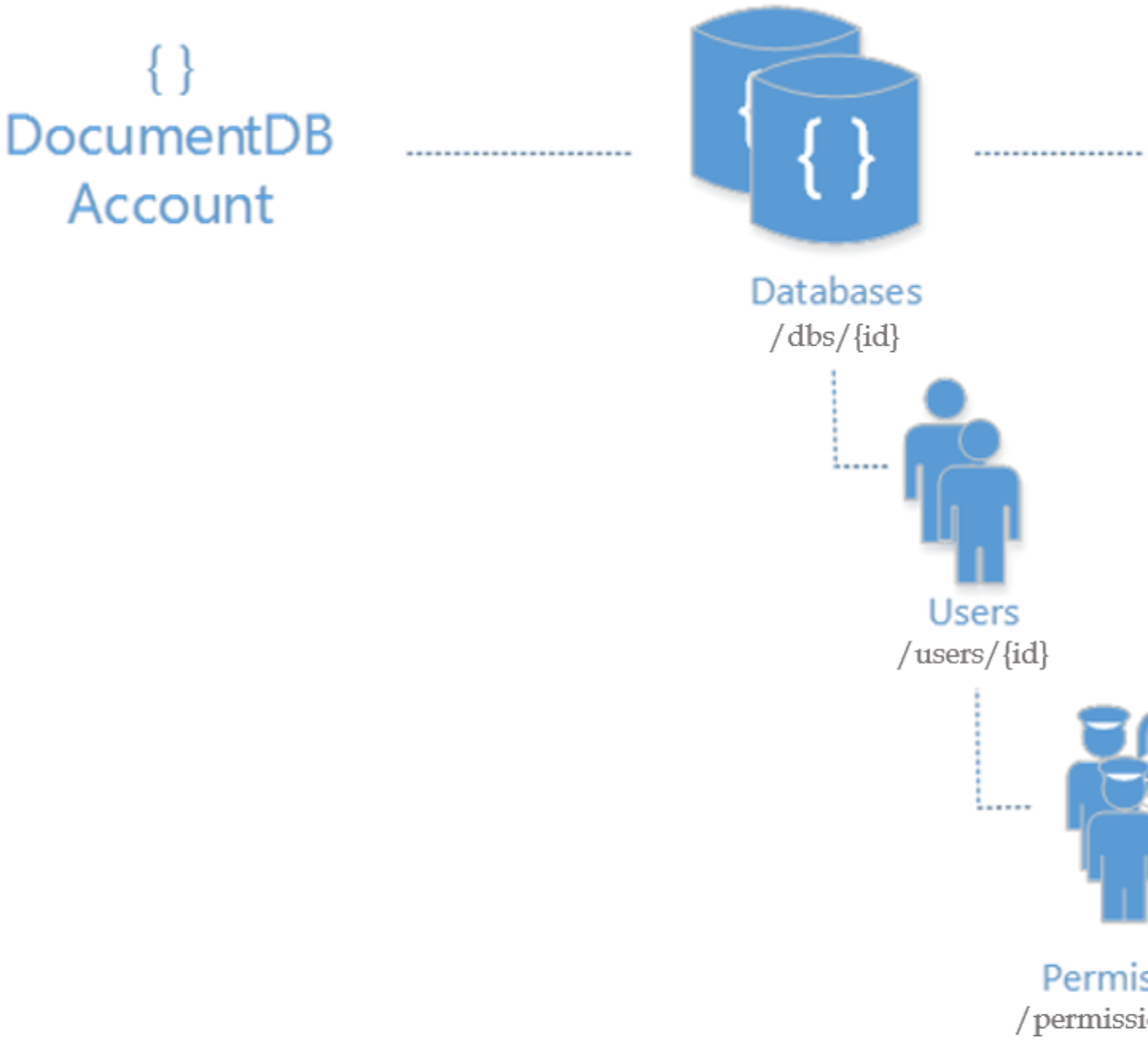
- **Session**: Unlike the global consistency models offered by strong and bounded staleness consistency levels, "session" consistency is tailored for a specific client session. Session consistency is usually sufficient since it provides guaranteed monotonic reads, and writes and ability to read your own writes. A read request for session consistency is issued against a replica that can serve the client requested version (part of the session cookie).
- **Eventual**: Eventual consistency is the weakest form of consistency wherein a client may get the values which are older than the ones it had seen before, over time. In the absence of any further writes, the replicas within the group will eventually converge. The read request is served by any secondary index.

  You can configure a default consistency level on your database account that applies to all the collections (across all of the databases) under your database account. By default, all reads and queries issued against the user defined resources will use the default consistency level specified on the database account. You can always lower the consistency level for an individual request using a request header.

DocumentDB Resources

DocumentDB REsources

The following diagram illustrates the DocumentDB resource model:



Resource Model

DocumentDB **resource model** consists of sets of resources under a database account, each addressable via a logical and stable URI. As et of resources is referred to as a feed.

Resource	Description
Database account	A database account is associated with a set of databases and a fixed amount of blob storage for attachments (prev feature). You can create one or more database accounts using your Azure subscription. Every Standard database a allocated a minimum capacity of one S1 collection.
Database	A database is a logical container of document storage partitioned across collections. It is also a users container.
User	The logical namespace for scoping/partitioning permissions.
Permission	An authorization token associated with a user for authorized access to a specific resource.
Collection	A collection is a container of JSON documents and the associated JavaScript application logic. A collection is a billal entity, where the cost is determined by the performance level associated with the collection. The performance lev S2 and S3) provide 10GB of storage and a fixed amount of throughput.
Stored Procedure	Application logic written in JavaScript which is registered with a collection and transactionally executed within the database engine.
Trigger	Application logic written in JavaScript modeling side effects associated with insert, replace or delete operations.

UDF	A side effect free, application logic written in JavaScript. UDFs enable you to model a custom query operator and t extend the core DocumentDB query language.
Document	User defined (arbitrary) JSON content. By default, no schema needs to be defined nor do secondary indices need t provided for all the documents added to a collection.
(Preview) Attachment	An attachment is a special document containing references and associated metadata for external blob/media. The developer can choose to have the blob managed by DocumentDB or store it with an external blob service provider OneDrive, Dropbox, etc.

System resources have a fixed schema and are also represented as standard JSON. System resources can be accessed, queried and viewed for monitoring or management tasks.

Addressing Resources

Resources are addressed using their unique names within your account. You can access items using the following URIs:

URI	Description
/dbs	Feed of databases in an account
/dbs/[database id]	Individual database
/dbs/[database id]/colls/	Feed of collections in a database
/dbs/[database id]colls/[collection id]	Individual collection

/dbs/[database id]colls/[collection id]/docs	Feed of documents in a collection

Programmatic Elements in DocumentDB

Bookmark this page

Programmatic Elements

You can write application logic to run directly within a transaction inside of the database engine. The application logic can be written entirely in JavaScript and can be modeled as a stored procedure, trigger or a UDF.

- **Stored Procedure:** Application logic written in JavaScript which is registered with a collection and transactionally executed within the database engine.
- **Trigger**: Application logic written in JavaScript modeling side effects associated with insert, replace or delete operations.
- **User Defined Function (UDF)**: A side effect free, application logic written in JavaScript. UDFs enable you to model a custom query operator and thereby extend the core DocumentDB query language.

The JavaScript code within a stored procedure or a trigger can insert, replace, delete, read or query documents within a collection. On the other hand, the JavaScript within a UDF can only perform side effect free computation by enumerating the documents of the result set of the query and produce another result set. Upon registration a stored procedure, trigger, or a UDF is pre-compiled and stored as byte code which gets executed later.

For multi-tenancy, DocumentDB enforces a strict reservation based resource governance. Each stored procedure, trigger or a UDF gets a fixed quantum of operating system resources to do its work. Further, stored procedures, triggers or UDFs cannot link against external JavaScript libraries and are blacklisted if they exceed the resource budgets allocated to them. You can register, unregister stored procedures, triggers or UDFs with a collection by using the REST APIs.

SQL Query

Azure DocumentDB supports querying documents using a SQL language, which is rooted in the JavaScript type system, and expressions with support for rich hierarchical queries. The DocumentDB query language is a simple yet powerful interface to query JSON documents. The language supports a subset of ANSI SQL grammar and adds deep integration of JavaScript object, arrays, object construction, and function invocation. DocumentDB provides its query model without any explicit schema or indexing hints from the developer.

User Defined Functions (UDFs) can be registered with DocumentDB and referenced as part of a SQL query, thereby extending the grammar to support custom application logic. These UDFs are written as JavaScript programs and executed within the database.

For .NET developers, DocumentDB also offers a LINQ query provider as part of the .NET SDK.

Transactions and JavaScript Execution

DocumentDB allows you to write application logic as named programs written entirely in JavaScript. These programs are registered for a collection and can issue database operations on the documents within a given collection. JavaScript can be registered for execution as a trigger, stored procedure or user defined function. Triggers and stored procedures can create, read, update, and delete documents whereas user defined functions execute as part of the query execution logic without write access to the collection.

JavaScript execution within DocumentDB is modeled after the concepts supported by relational database systems, with JavaScript as a modern replacement for T-SQL. All JavaScript logic is executed within an ambient ACID transaction with snapshot isolation. During the course of its execution, if the JavaScript throws an exception, then the entire transaction is aborted.

MongoDB

Bookmark this page

MongoDB

MongoDB is an open source, document-oriented NoSQL database designed for maximum scalability and agility. Unlike traditional relational databases, MongoDB doesn't store data in tables and rows. Rather, it stores BSON (binary serialized object notation) documents, which are binary JSON (JavaScript Object Notation) documents, with dynamic schemas. These BSON documents are stored in collections, which are named groupings of documents. Instead of a SQL query syntax, BSON queries can be made directly in most object-oriented languages.

The MongoDB project website has provided best practices documentation on how to run a MonogoDB instance in Azure: http://docs.mongodb.org/ecosystem/platforms/windows-azure/

MySQL

Bookmark this page

MySQL

Using Windows or Linux virtual machines, you can always install and run MySQL in the Azure environment. ClearDB also provides a managed MySQL instance that you can create from the Azure Marketplace.

To learn more about ClearDB, you can visit their website: http://www.cleardb.com/

For more information , you can see:
Azure: https://aka.ms/edx-dev205bx-az34

HBase

Bookmark this page

HBase

Apache HBase is an open-source, NoSQL database that is built on Hadoop and modeled after Google BigTable. HBase provides random access and strong consistency for large amounts of unstructured and semistructured data in a schemaless database organized by column families.

Data is stored in the rows of a table, and data within a row is grouped by column family. HBase is a schemaless database in the sense that neither the columns nor the type of data stored in them need to be defined before using them. The open-source code scales linearly to handle petabytes of data on thousands of nodes. It can rely on data redundancy, batch processing, and other features that are provided by distributed applications in the Hadoop ecosystem.

HDInsight HBase is offered as a managed cluster that is integrated into the Azure environment. The clusters are configured to store data directly in Azure Blob storage, which provides low latency and increased elasticity in performance and cost choices.

The HDInsight implementation leverages the scale-out architecture of HBase to provide automatic sharding of tables, strong consistency for reads and writes, and automatic failover. Performance is enhanced by in-memory caching for reads and high-throughput streaming for writes. Virtual network provisioning is also available for HDInsight HBase.

Use Cases

The canonical use case for which BigTable (and by extension, HBase) was created was web search. Search engines build indexes that map terms to the web pages that contain them. But there are many other use cases that HBase is suitable for including:

- **Key-value store**

  HBase can be used as a key-value store, and it is suitable for managing message systems. Facebook uses HBase for their messaging system, and it is ideal for storing and managing Internet communications. WebTable uses HBase to search for and manage tables that are extracted from webpages.

- **Sensor data**

  HBase is useful for capturing data that is collected incrementally from various sources. This includes social analytics, time series, keeping interactive dashboards up-to-date with trends and counters, and managing audit log systems. Examples include Bloomberg trader terminal and the Open Time Series Database (OpenTSDB), which stores and provides access to metrics collected about the health of server systems.

- **Real-time query**

  Phoenix is a SQL query engine for Apache HBase. It is accessed as a JDBC driver, and it enables querying and managing HBase tables by using SQL.

- **HBase as a platform**

  Applications can run on top of HBase by using it as a datastore. Examples include Phoenix, OpenTSDB, Kiji, and Titan. Applications can also integrate with HBase. Examples include Hive, Pig, Solr, Storm, Flume, Impala, Spark, Ganglia, and Drill.

Service Bus Overview

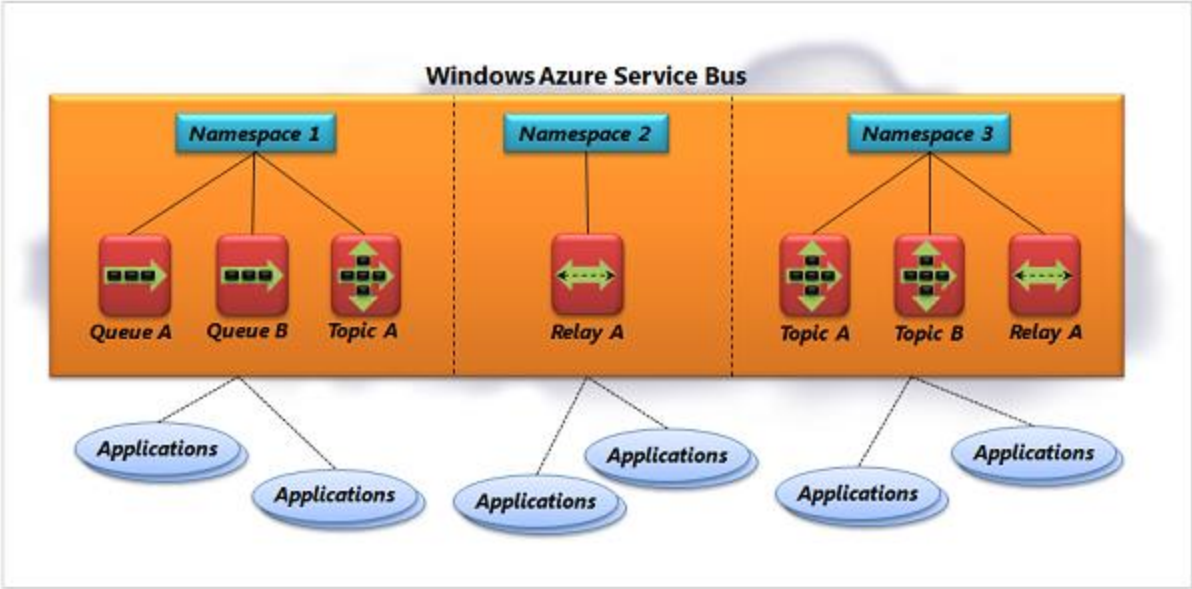
Service Bus

Azure Service Bus provides a hosted, secure, and widely available infrastructure for widespread communication, large-scale event distribution, naming, and service publishing. Service Bus provides connectivity options for Windows Communication Foundation (WCF) and other service endpoints – including REST endpoints -- that would otherwise be difficult or impossible to reach. Endpoints can be located behind network address translation (NAT) boundaries, or bound to frequently-changing, dynamically-assigned IP addresses, or both.

Service Bus provides both "relayed" and "brokered" messaging capabilities. In the relayed messaging pattern, the relay service supports direct one-way messaging, request/response messaging, and peer-to-peer messaging. Brokered messaging provides durable, asynchronous messaging components such as *Queues*, *Topics*, and *Subscriptions*, with features that support publish-subscribe and temporal decoupling: senders and receivers do not have to be online at the same time; the messaging infrastructure reliably stores messages until the receiving party is ready to receive them.

Namespaces

Service Bus services are typically partitioned into namespaces as each namespace provide both a service and security boundary.



When you create a queue, topic, relay, or Event Hub, you give it a name. Combined with whatever you called your namespace, this name creates a unique identifier for the object. Applications can provide this name to Service Bus, then use that queue, topic, relay, or Event Hub to communicate with one another.

To use any of these objects, Windows applications can use Windows Communication Foundation (WCF). For queues, topics, and Event Hubs Windows applications can also use Service Bus-defined messaging APIs. To make these objects easier to use from non-Windows applications, Microsoft provides SDKs for Java, Node.js, and other languages. You can also access queues, topics, and Event Hubs using REST APIs over HTTP.

For more information , you can see:
Azure Service Bus: https://aka.ms/edx-dev205bx-az19
Windows Communication Foundation (WCF): https://aka.ms/edx-dev205bx-wcf
Event Hubs: https://aka.ms/edx-dev205bx-az20
Notification Hubs: https://aka.ms/edx-dev205bx-az15
Mobile Apps: https://aka.ms/edx-dev205bx-az17

Service Bus Notification Hubs

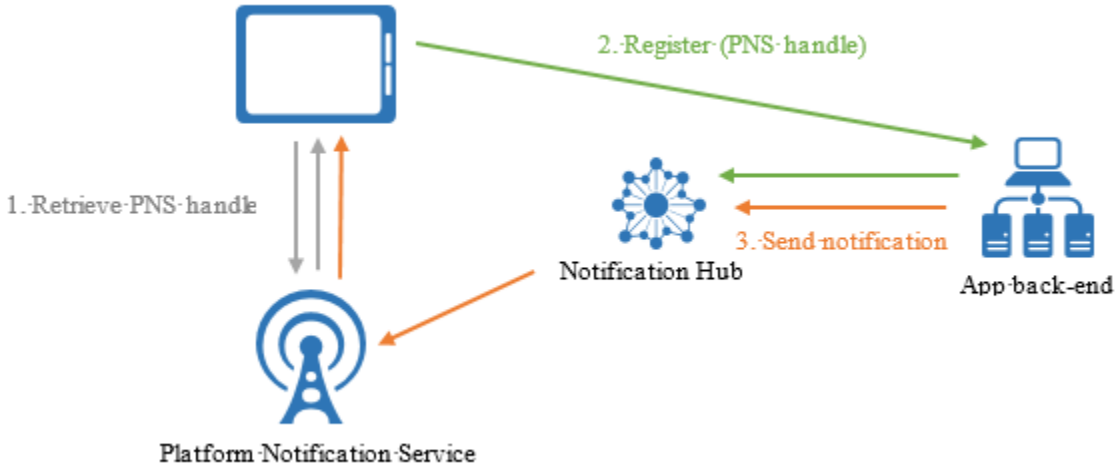
Bookmark this page

Service Bus Notification Hubs

Smartphones and tablets have the ability to "notify" users when an event has occurred whether or not your application is running. Typically, to implement this push functionality, you would require deep experience on all major mobile platforms along with a rich network of servers sending the actual notification payload. Azure Notification Hubs provide an easy-to-use, multiplatform, scaled-out push infrastructure that enables you to send mobile push notifications from any backend (in the cloud or on-premises) to any mobile platform as a managed service.

With Notification Hubs you can easily send cross-platform, personalized push notifications, abstracting the details of the different platform notification systems (PNS). With a single API call, you can target individual users or entire audience segments containing millions of users, across all their devices.

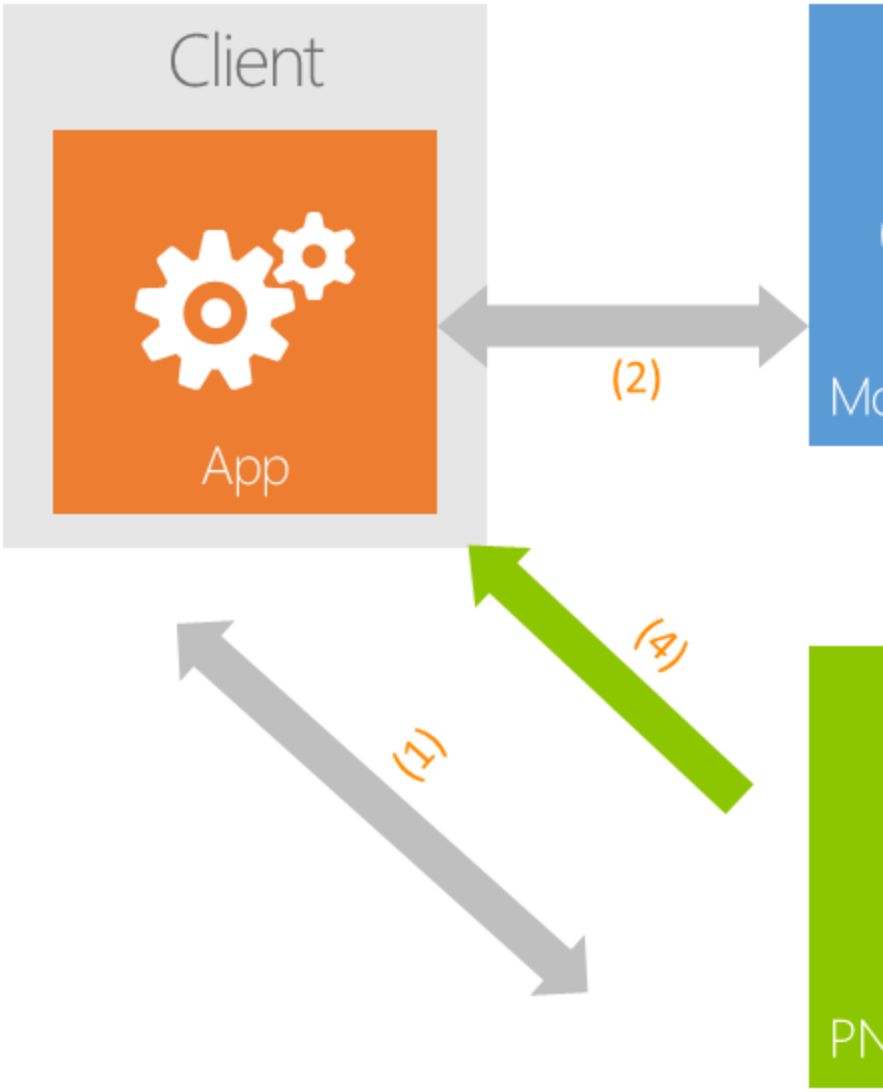
Notification Hubs abstracts the complexity of managing a Platform Notification Service and allows you to send notifications as a simple API call or REST request. Notification Hubs use a full multiplatform, scaled-out push notification infrastructure, and considerably reduce the push-specific code that runs in the app backend. Devices only need to register PNS handles which can greatly simplify device-specific code. Your application backend sends platform-independent messages to specific users or groups.



Notification Hubs and Mobile Services

Bookmark this page

Notification Hubs and Mobile Services



Mobile App Push Notification Flow

Push notifications work the same no matter which client you are working with. Because of this, you can make calls to your Notification Hub directly from a Mobile App. This can be used in scenarios where a client (mobile device) sends a request to process data in the Mobile App and the Mobile App can asynchronously notify the client via Push Notification when the processing is done.

1. Client device (application) registers for push notifications and gets a PNS handle.
2. PNS handle is sent to the Mobile App instance.

3. The Mobile App sends a request to your Notification Hub (PNS in this diagram) to send a push notification to the client device.
4. The client device receives a push notification.

Filtering Notification Recipients

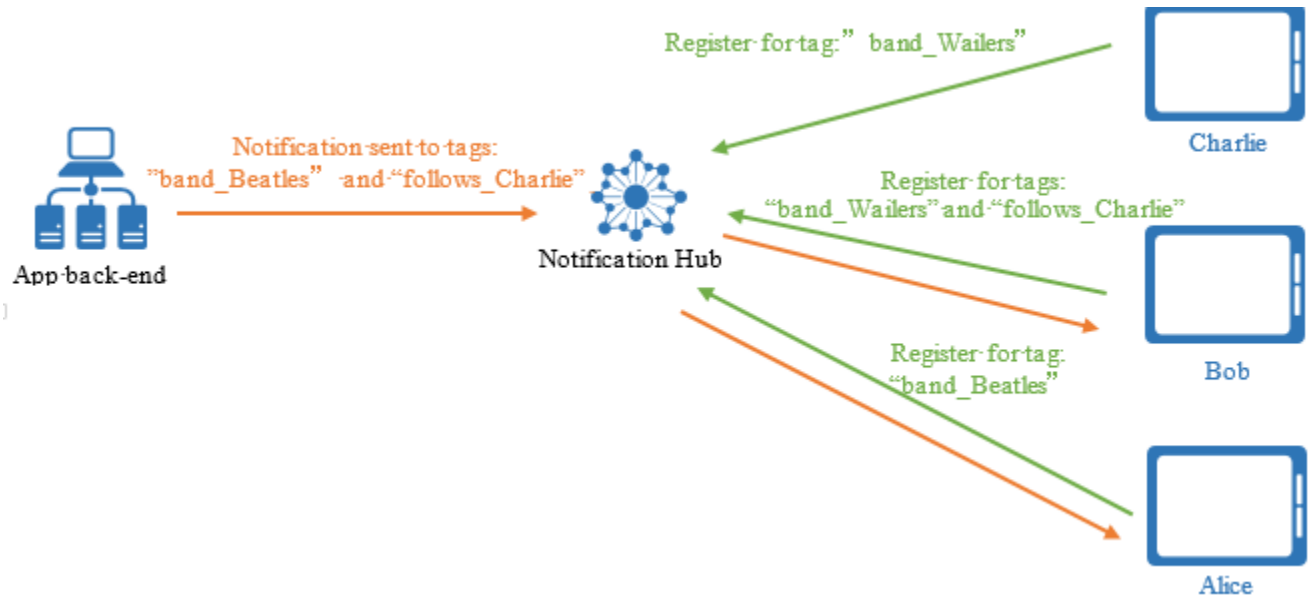
Bookmark this page

Filtering Notification Recipients

When devices register with a notification hub, they can optionally specify tags to be associated with the device. These tags enable you to target specific sets of devices, or more specifically registrations, when sending a push notification through Notification Hubs. The only way to target specific registrations is to associate them with a tag, then target that tag. When the application backend creates a registration, the application can target the notification to all devices by sending a Broadcast notification.

Notifications can be targeted to a specific Tag by specifying the tag when creating the notification. A tag can be any string, up to 120 characters, containing alphanumeric and the following non-alphanumeric characters: '_', '@', '#', '.', ':', '-'. The resulting notification will be sent to all devices registered with the Notification Hub with that tag value included in their collection of registered Tags.

Tags do not have to be pre-provisioned and can refer to multiple app-specific concepts. For example, users of this example application can comment on bands and want to receive toasts, not only for the comments on their favorite bands, but also for all comments from their friends, regardless of the band on which they are commenting. The following picture shows an example of this scenario:

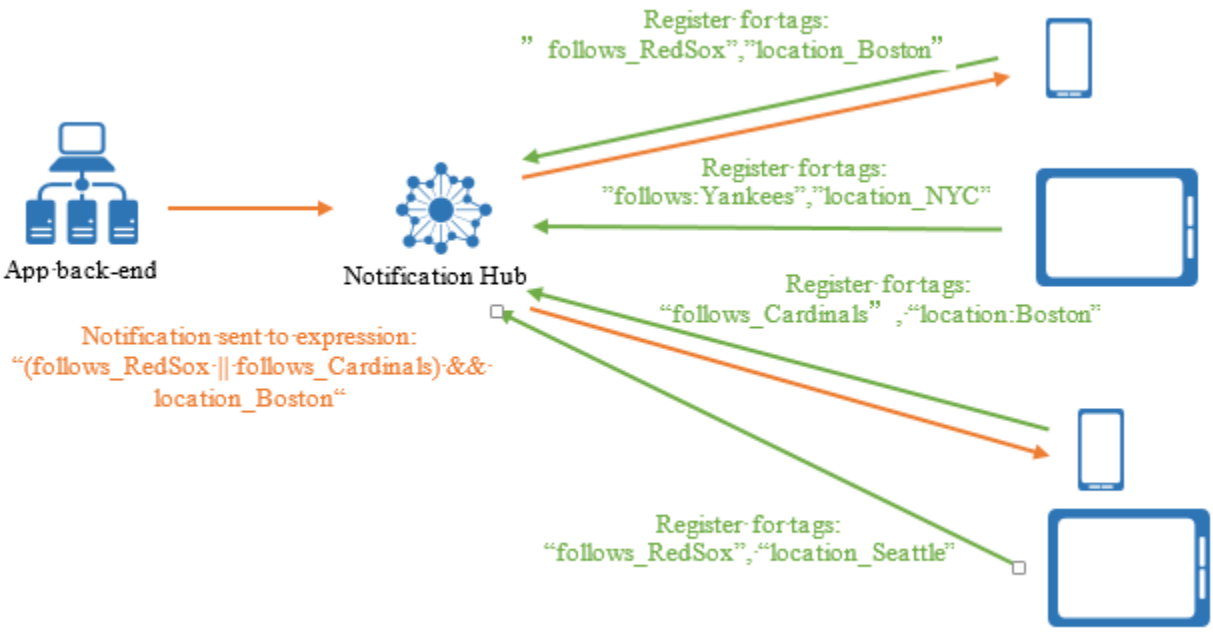


In this picture, Alice is interested in updates for the Beatles, and Bob is interested in updates for the Wailers. Bob is also interested in Charlie's comments, and Charlie is in

interested in the Wailers. When a notification is sent for Charlie's comment on the Beatles, both Alice and Bob receive it.

There are cases in which a notification has to target a set of registrations that is identified not by a single tag, but by a Boolean expression on tags. Consider a sports application that sends a reminder to everyone in Boston about a game between the Red Sox and Cardinals. If the client app registers tags about interest in teams and location, then the notification should be targeted to everyone in Boston who is interested in either the Red Sox or the Cardinals. This condition can be expressed with the following Boolean expression:

```
(follows_RedSox || follows_Cardinals) && location_Boston
```



Tag expressions can contain all Boolean operators, such as AND (&&), OR (||), and NOT (!). They can also contain parentheses. Tag expressions are limited to 20 tags if they contain only ORs; otherwise they are limited to 6 tags.

Comparing Service Bus Queues to Storage Queues

Bookmark this page

https://azure.microsoft.com/en-us/documentation/articles/service-bus-azure-and-service-bus-queues-compared-contrasted/

Service Bus Queue

- Guarantees First-In-First-Out (FIFO) order
- Messages are guaranteed to be delivered at-least-once and at-most-once
- Supports batch send and retrieve

- Supports peek
- Transactions are supported
- Supports long polling (blocking)

Storage Queue

- Ordering is not guaranteed due to visibility timeout
- Messages are guaranteed to be delivered at-least-once
- Supports batch receive
- Supports peek
- Supports different timeout values per message and timeout renewals (leases)

Customer Scenario

Bookmark this page

Who is the customer?

With Corporate Headquarters in Phoenix Arizona, the Crazy Taxi Cab Company has quickly risen to be the premier provider of private low-cost transportation in Arizona. Bucking industry norms, Crazy Taxi Cab Co drivers are company employees who work as a team, rather than independent contractors who essentially compete with each other for fares. The founding partners believed this would allow its drivers to focus on providing a great customer experience as opposed to simply "racing to the finish line". Crazy Taxi has developed a reputation for having fast, reliable, and friendly service, due largely in part to their extensive network of drivers, and their well-executed, albeit radio based dispatching approach.

Crazy Taxi is drowning in success. While dispatchers are reaching out to drivers to pick up customers who have called in, new callers often find themselves on hold, waiting for their calls to be answered. The executives at Crazy Taxi realize that they need to modernize their operation or risk losing business in the future. Their Chief Operating Officer Christopher Giovanni states that "while we function like a well-oiled machine, we're still running on 20th century equipment and we are already seeing signs that this is eroding our advantage over the competition…we need to bring our operation into the 21st century and that starts with FastRide".

What does the customer already have?

Crazy Taxi does not want to manage a more complex of a system than absolutely necessary. They already have three web and app developers, who have built their marketing pages as well as few proof of concept apps for iOS and Android, and one of them is quite savvy with .NET and brings some backend experience to the table.

However, the executives all agree that this one developer cannot deliver the entire backend alone, so they are looking for a solution that provides them a "back end in a box" as they cannot afford to hire any more developers. Additionally, headquarters wants their IT team to have easy visibility into the health of the system, and more importantly that the system can take care of itself. Specifically, they are familiar with the notion of pay-as-you-go billing, and would love a solution that operates at nominal cost during their average daytime load, but on Friday night and Saturday night automatically handles the increased demand, albeit at an additional cost.

For more information , you can see:
.NET: https://aka.ms/edx-dev205bx-ne
Microsoft Account: https://aka.ms/edx-dev205bx-msa
Azure Mobile Apps: https://aka.ms/edx-dev205bx-az17

Customer Goal

Bookmark this page

What is the customer's goal?

FastRide is a software solution that the company has been planning to build that uses the GPS of 4G enabled mobile devices in each Crazy Taxi to monitor the location of each vehicle in the Crazy Taxi fleet. By being able to visualize how their fleet is moving throughout the city in real time, FastRide will allow Crazy Taxi Cab Co. to optimize their driver coverage throughout the city. When a new customer calls in, the telephone dispatcher enters their pickup location and the FastRide system identifies the closest available driver, instead of requiring dispatch to manually poll all of the drivers over the radio. While stationary, a driver accepts a fare using the FastRide app on his mobile device. FastRide will provide the driver with the pick-up details such as location, passenger name and callback phone (if available). Upon arrival the destination, the device will compute the fare automatically based on time in transit and distance driven, all with intent of minimizing driver distraction while driving due to interacting with the device. The fare information should be collected on the device and uploaded to a central database in near real-time. Should the driver enter a pocket without cellular connectivity, the device should store the data offline and sync it as soon as connectivity is restored.

The fare data is associated with each driver, who logs in in thru his or her app using his or her own Microsoft Account, Google, Twitter or Facebook credentials. It is the intent that driver fare data will be used to track the driver's progress against established company goals, which both the company and driver access in the form of reports. Because these reports may grow to involve more complex computations (such as comparing the driver's selected route to similar routes taken by other drivers), these data powering the reports will be computed nightly and made available the following day.

While initially notifications would be sent only to drivers about potential fares they can choose to accept, Crazy Taxi is looking forward to when they can offer their patrons a mobile app that can receive notifications about when the driver is in route, ETA updates

and any messaging from the driver to the customer. They would like to be certain that the system they build on today has the capacity to meet those increased demands in the future.

Solution Considerations

Bookmark this page

What does the customer need?

- A back-end as a service that provides both data storage, API support and push notifications.
- A solution that is quick to implement with a small team that has limited back-end development experience and capacity.
- A back end solution that can be implemented using .NET.
- A solution that is easy to monitor for health and telemetry data.

What things worry the customer?

- Doesn't Azure Mobile Services only work on Windows devices?
- Our development team doesn't know node.js. We had heard mention of Mobile Services, but thought it only supported JavaScript backends.
- Our development team seems to think implementing push notifications using Apple and Android apps directly is easy, but we (as the executives) aren't so sure. How difficult can it be?
- Can't we just build all of our backend using Azure Websites?

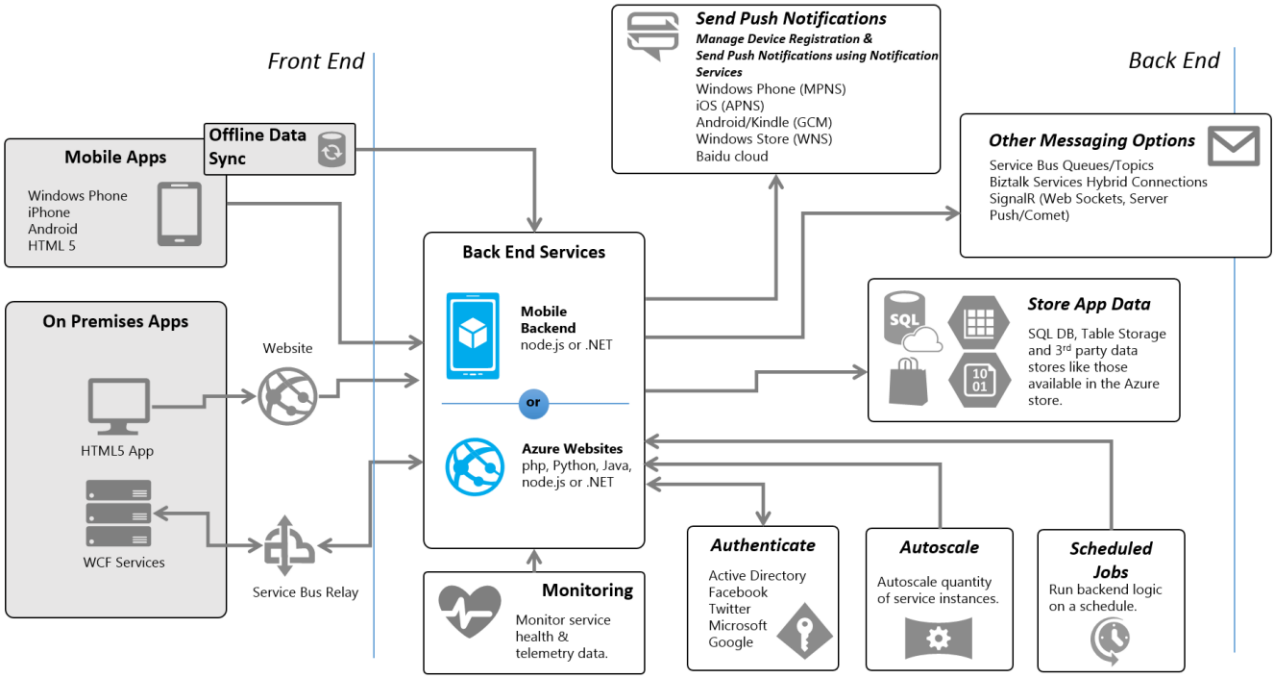
For more information , you can see:
Azure Mobile Services: https://aka.ms/edx-dev205bx-az16

Common Implementations

Bookmark this page

Azure Mobile Apps

Mobile Apps makes it easy to authenticate employees with corporate credentials, support occasionally connected apps, or access data stored on-premises. Build employee apps natively for iOS, Android, Windows, or target cross-platform with Xamarin, PhoneGap, and Sencha.



Call to Action

Bookmark this page

Design the Solution

You will now design a potential solution for **Crazy Taxi Cab Company**. Prior to completing this exercise, please ensure that you have read all of the previous units in this case study. Particularly, make sure that you have read and you understand the **Customer Scenario** and **Solution Considerations**.

In this exercise, you will tackle a few primary tasks.

1. You will need to determine who you should present this solution to. Who is the target customer audience (stakeholder)? Who are the decision makers? You will answer these questions below in the problem sections.
2. You will need to determine what you intend to address with your solution. What customer business needs do you need to address with your solution? How will you address each business need? You will provide a brief explanation below in the problem sections.
3. You will need to create a diagram for your proposed solution. This can be done with Visio or with any image editor of your choice. The ideal output is either a high-resolution PNG file or a PDF document. You will upload this diagram in the final section below.

Once you have completed these three tasks, create a new post in the solution forum, attach an image file to the post for your diagram and your response to prompts #1 & #2.

Sample Solution

You are highly encouraged to try and come up with your own solution for this case study and post that solution in the forums. Once you have completed that, you can review the sample solution linked below and compare it against your own solution.

Sample Solution

Active Directory

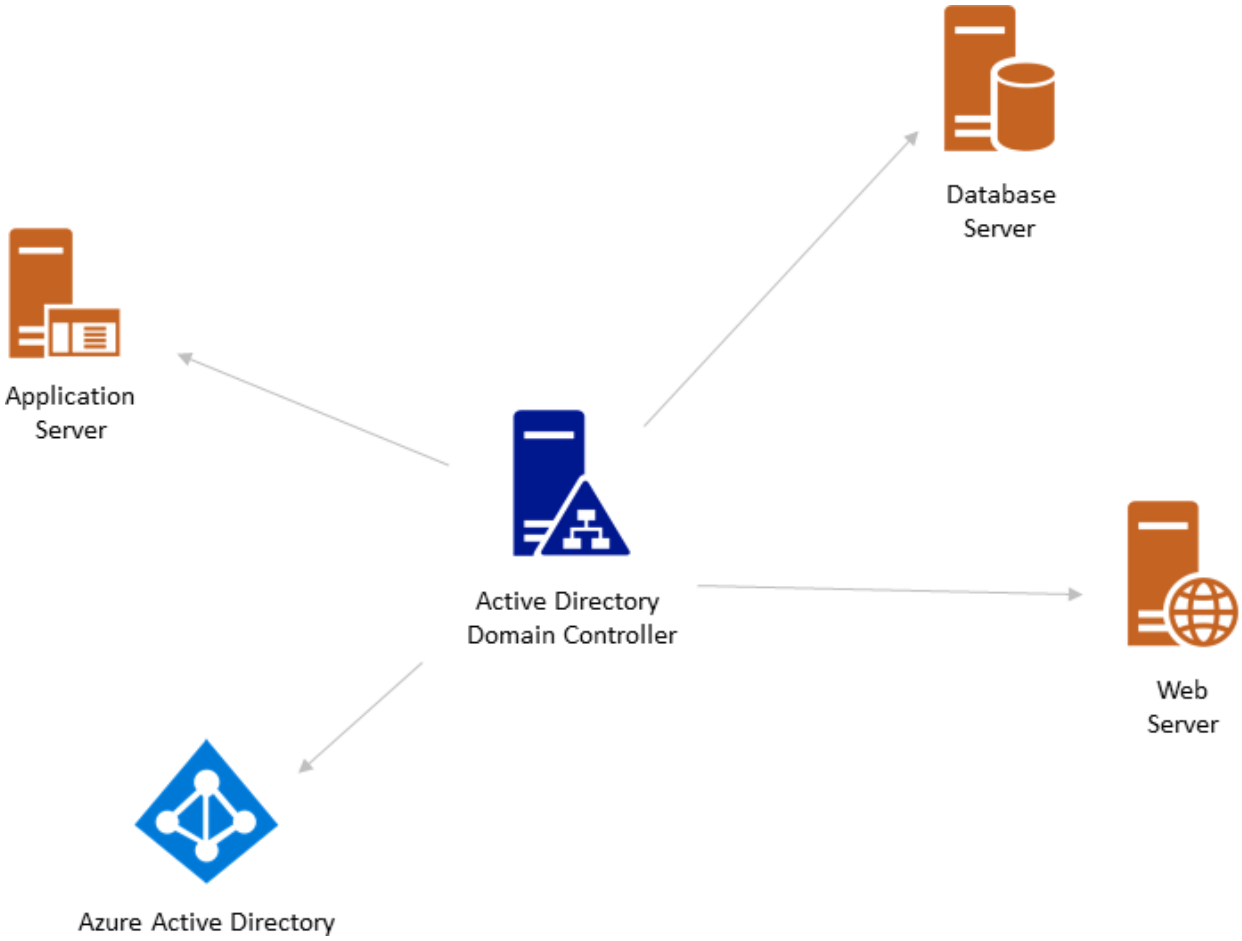
Bookmark this page

Active Directory

Active Directory (AD) is a domain service that stores directory data and manages communication between users and domains, including user logon processes, authentication, and directory searches. An AD domain controller is a server that is running AD Domain Services.

As a database, AD is designed to handle a large number of read and search operations and a significantly smaller number of changes and updates. AD data is hierarchical, replicated, and extensible. Information about a user's identity is stored independently from information about machines. This identity data can be queried, leveraged in your existing applications or managed using custom tools. This identity data is important because AD essentially can be used as an identity provider for your applications.

Active Directory Integrations



Azure Active Directory (AD)

Bookmark this page

Azure Active Directory (AD)

Azure Active Directory (Azure AD) allows businesses to manage identity and access, both in the cloud and on-premises across many different applications and devices. Users can use the same work or school account for single sign-on to any cloud and on-premises web application. Your users can use their favorite devices, including iOS, Mac OS X, Android, and Windows. Your organization can protect sensitive data and applications both on-premises and in the cloud with integrated multi-factor authentication ensuring secure local and remote access. Azure AD extends your on-premises directories so that information workers can use a single organizational account to securely and consistently access their corporate resources. Azure AD also offers comprehensive reports, analytics, and self-service capabilities to reduce costs and enhance security.

Managing Azure Active Directory

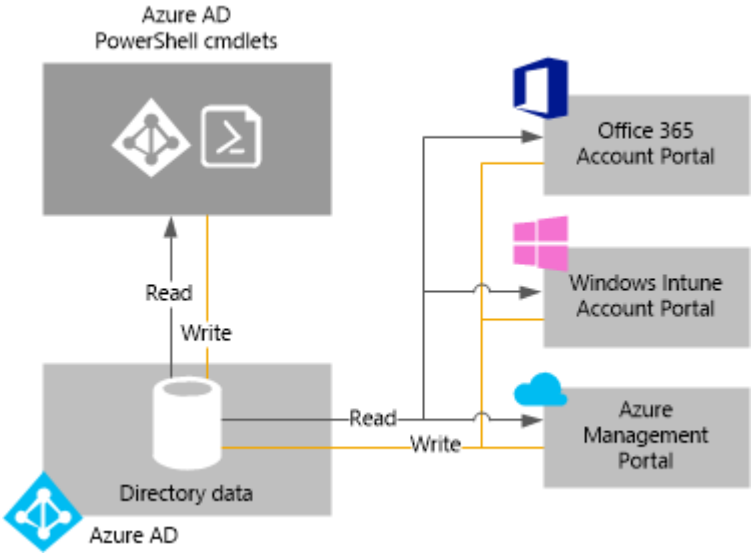
As an administrator of one or more Microsoft cloud service subscriptions, you can either use the Azure Management Portal, the Windows Intune account portal, or the Office 365 Admin Center to manage your organizations directory data. You can also use the downloadable Microsoft Azure Active Directory Module for Windows PowerShell cmdlets to help you manage data stored in Azure AD.

From either of these portals (or cmdlets), you can:

- Create and manage user and group accounts
- Manage related cloud service(s) your organization subscribes to
- Set up on-premises integration with your directory service

The Azure Management Portal, Office 365 Admin Center, Windows Intune account portal and the cmdlets all read from and write to a single shared instance of Azure AD that is associated with your organization's directory. In this way, portals (or cmdlets) act as a front-end interface that pull in and/or modify your directory data.

Azure Active Directory Endpoints

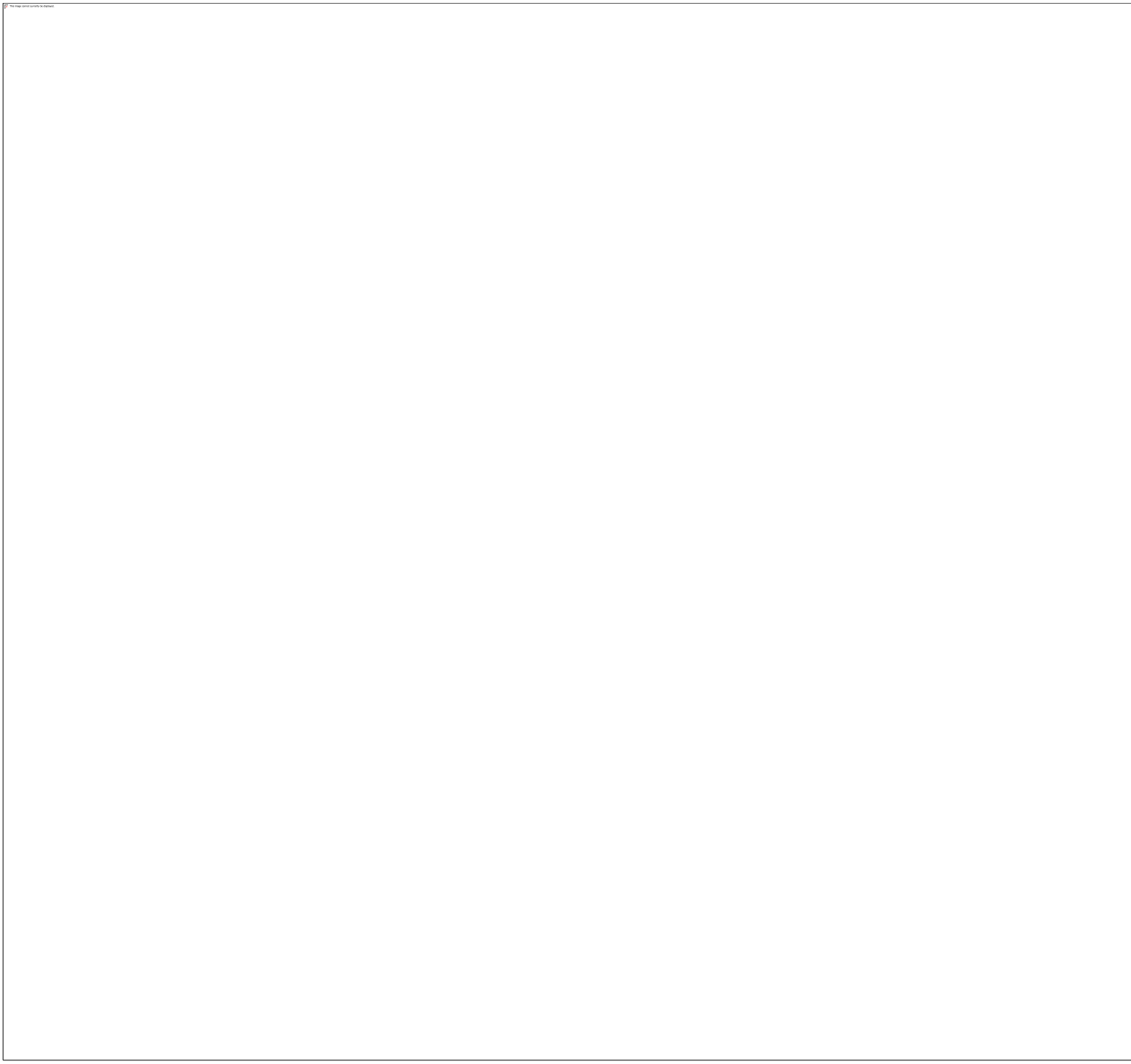


For more information , you can see:
Azure Active Directory: https://aka.ms/edx-dev205bx-azad
Microsoft Azure portal: https://aka.ms/edx-dev205bx-az21
Windows PowerShell: https://aka.ms/edx-dev205bx-ps

Sync Active Directory to Azure AD

Bookmark this page

Sync Active Directory Identities to Azure AD



Directory Sync

If your organization has an on-premises directory service, you can integrate it with your Microsoft Azure Active Directory (Microsoft Azure AD) directory by syncing only identity objects and forcing all authentication to flow through your on-premise directory or syncing both identities and passwords. With directory sync, you can manage the entire lifecycle of
your cloud user and group accounts using your on-premise Active Directory management tools and minimize any impact to your existing workflow.

When password sync is enabled on your directory sync computer, your users will be able to sign into Microsoft cloud services, such as Office 365, Dynamics CRM, and Microsoft Intune, using the same password as they use when logging into your on-premises network. When your users change their passwords in your corporate network, those changes are synchronized to the cloud. To synchronize a password, the Directory Sync tool extracts the user password hash from the on-premises Active Directory. Additional security processing is applied to the password hash before it is synchronized to Azure AD. Identities and passwords are synchronized at different frequencies with passwords synchronized more frequently than other directory data.It is important to note that this feature does not provide a full single sign-on (SSO) solution because there is no token sharing / exchange in the Password Sync based process.

Sync Options

There are three primary ways that you can sync identities with an Azure AD directory.

Identity Sync

In the simplest directory synchronization scenario, user (identity) objects are the only ones synced with Azure AD. Identities can be managed on-premise and these changes will reflect in the Azure AD directory. The users, however, will have different credentials for their cloud and on-premise identities.

Password Sync

In this scenario, the hash value of the password is also synced with the user identity. This allows users to log into off-premise services (such as Office 365, Microsoft Intune, CRM Online) using the same password that they use on-premise. Passwords can be modified on-premise and eventually synced to the Azure AD instance. This offers eventual consistency for passwords.

Password Sync with Writeback

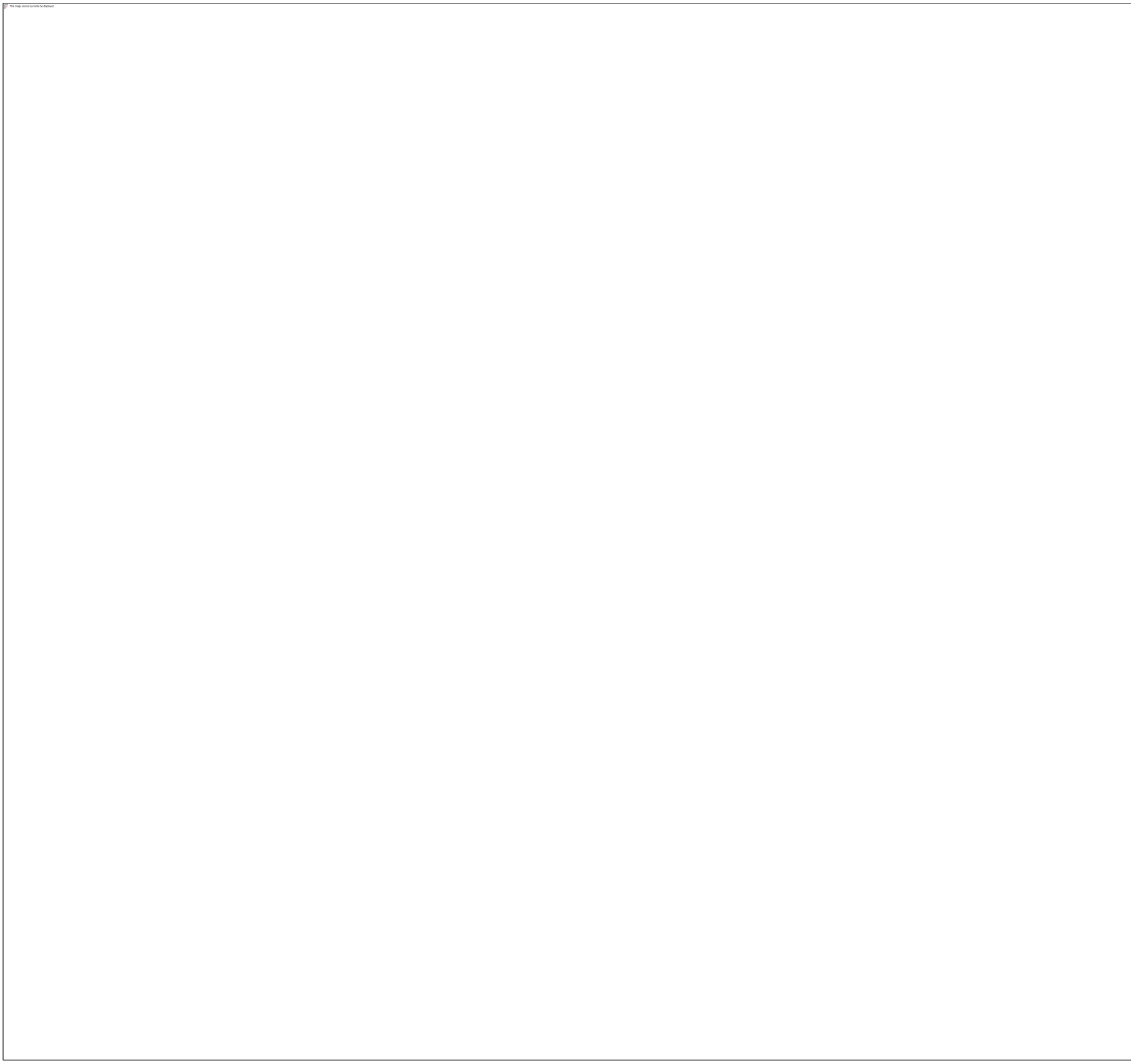
Password writeback is only available for current subscribers of Azure AD Premium. Users can use an online self-service password management portal to reset their password from any location. The passwords are then validated immediately against your existing AD password policies. If validated, this password is then stored as a hash and synced with your enterprise Active Directory instance. The writeback is done using Service Bus relay to avoid creating inbound firewall rules.

For more information , you can see:
Service Bus: https://aka.ms/edx-dev205bx-az04

Azure Active Directory Single-Sign On

Bookmark this page

Federated Identity with Azure Active Directory



Azure Active Directory Single-Sign ON

Single sign-on, also called identity federation, is a hybrid-based directory integration scenario of Azure Active Directory that you can implement when you want to simplify your user's ability to seamlessly access cloud services, such as Office 365 or Microsoft Intune, with their existing Active Directory corporate credentials. Without single sign-on, your users would need to maintain separate user names and passwords for your online and on-premises accounts.

An Secure Token Service (STS) enables identity federation, extending the notion of centralized authentication, authorization, and SSO to Web applications and services

located virtually anywhere, including perimeter networks, partner networks, and the cloud. When you configure an STS to provide single sign-on access with a Microsoft cloud service, you will be creating a federated trust between your on-premises STS and the federated domain you've specified in your Azure AD tenant.

There is a clear benefit to users when you implement single sign-on: it lets them use their corporate credentials to access the cloud service that your company has subscribed to. Users don't have to sign in again and remember multiple passwords.

For more information , you can see:
Microsoft cloud services: https://aka.ms/edx-dev205bx-az10

Using Third-Party Identity Providers

Bookmark this page

External Users

In Azure AD you can also add users to an Azure AD directory from another Azure AD directory or a user with a Microsoft Account. A user can be a member of up to 20 different directories. Users who are added from another directory are external users. External users can collaborate with users who already exist in a directory, such as in a test environment, without requiring them to sign in with new accounts and credentials. External users are authenticated by their home directory when they sign in, and that authentication works for all other directories that they are a member of.

When you add a user from one directory into a new directory, that user is an external user in the new directory. Initially, the display name and user name are copied from the user's "home directory" and stamped onto the external user in the other directory. From then on, those and other properties of the external user object are entirely independent: if you make a change to the user in the home directory, such as changing the user's name, adding a job title, etc. those changes are not propagated to the external user account in the other directory.

The only linkage between the two objects is that the user always authenticates against the home directory or with their Microsoft Account. That's why the option to reset the password or enable multi-factor authentication for an external user account is not available. Currently the authentication policy of the home directory or Microsoft Account is the only one that's evaluated when the user signs in.

For more information , you can see:
Microsoft Account: https://aka.ms/edx-dev205bx-msa

Azure Active Directory B2C

Bookmark this page

Azure Active Directory B2C

While it is simple to create identities in Azure Active Directory, many times you wish to offer the option for your application's consumers to use their existing social identities. The infrastructure and code to create a solution that can handle both Azure AD and social identities can be quite complex.

Azure AD B2C is an identity-as-a-service (IDaaS) solution that handles many aspects of identity for your web & mobile solutions including:

- Allowing users to sign up for an account in a self-service manner
- Giving users the option to create an account using a new user name and password
- Giving users the option to create an account using an existing social identity such as LinkedIn, Twitter, Facebook or Google
- Offering users a self-service profile and password management experience
- Enabling single sign-on (SSO) to all of your applications using a cohesive identity

Using Azure AD B2C, you do not need to write code specifically to manage your user's identity, password, profile and social accounts.

Access Control List

Access Control List

A Network Access Control List (ACL) is a security enhancement available for your Azure deployment. An ACL provides the ability to selectively permit or deny traffic for a virtual machine endpoint. This packet filtering capability provides an additional layer of security. An ACL is an object that contains a list of rules. When you create an ACL and apply it to a Virtual Machine endpoint, packet filtering takes place on the host node of your VM. This means the traffic from remote IP addresses is filtered by the host node for matching ACL rules instead of on your VM. This prevents your VM from spending the CPU cycles on packet filtering.

When a virtual machine is created, a default ACL is put in place to block all incoming traffic. If you create an endpoint (example: port 3389), then the default ACL is modified to allow all inbound traffic for that endpoint. Inbound traffic from any remote subnet is then allowed to that endpoint and no firewall provisioning is required. All other ports are blocked for inbound traffic unless endpoints are created for those ports. Outbound traffic is allowed by default.

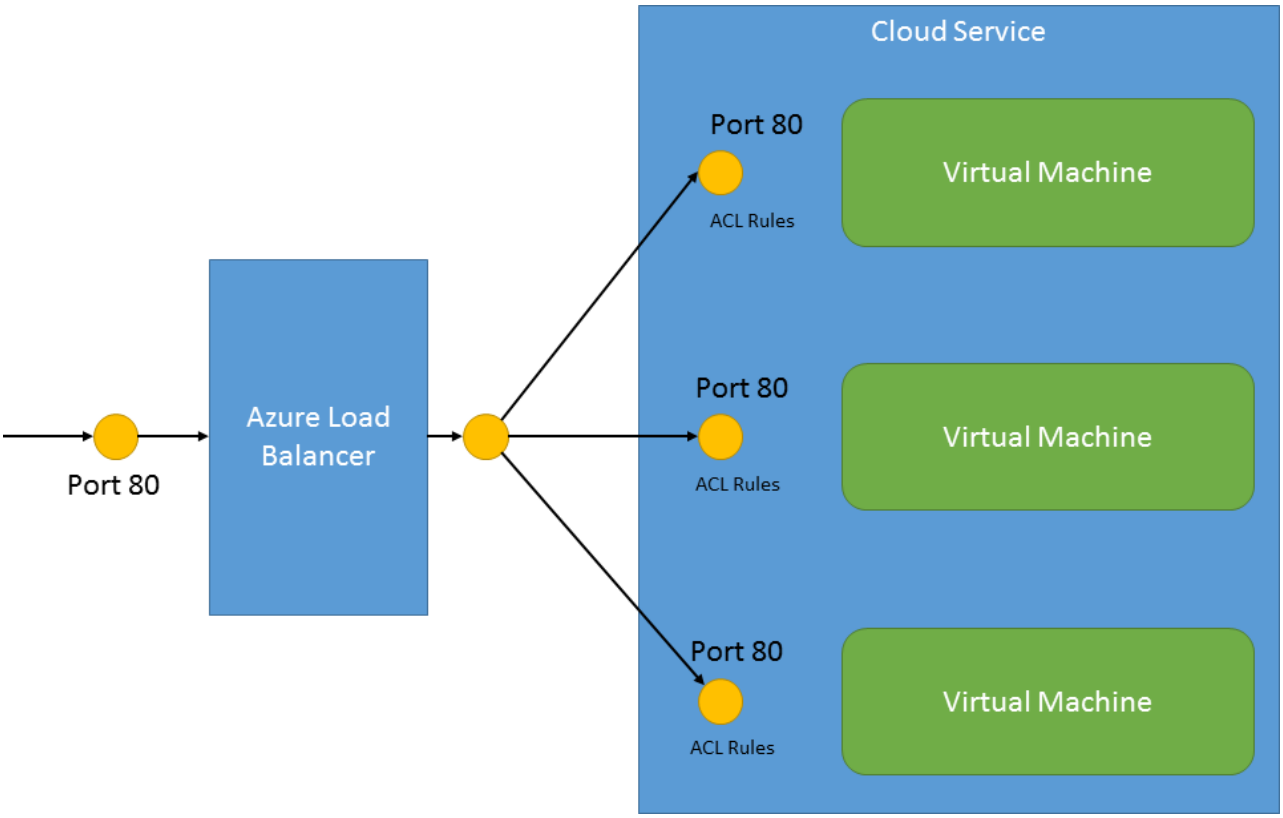
RULE #	REMOTE SUBNET	ENDPOINT	PERMIT/DENY

100	0.0.0.0/0	3389	Permit

You can specify network ACLs for virtual machine endpoints only. You can't specify an ACL for a Virtual Network or a specific subnet contained in a virtual network. In that scenario, you would use a Network Security Group to specify rules for a Virtual Network's subnet.

Network ACLs and load balanced sets

Network ACLs can be specified on a Load balanced set (LB Set) endpoint. If an ACL is specified for a LB Set, the Network ACL is applied to all Virtual Machines in that LB Set. For example, if a LB Set is created with "Port 80" and the LB Set contains 3 VMs, the Network ACL created on endpoint "Port 80" of one VM will automatically apply to the other VMs.



For more information , you can see:
Virtual Network: https://aka.ms/edx-dev205bx-az12

Network Security Groups

Bookmark this page

Network Security Groups

Network security groups are different than endpoint-based ACLs. Endpoint ACLs work only on the public port that is exposed through the input endpoint. An NSG works on one or more VM instances and controls all the traffic that is inbound and outbound.

You can associate an NSG to a VM, or to a subnet within a VNet. When associated with a VM, the NSG applies to all the traffic that is sent and received by the VM instance. When applied to a subnet within your VNet, it applies to all the traffic that is sent and received by ALL the VM instances in the subnet. A VM or subnet can be associated with only 1 NSG, and each NSG can contain up to 200 rules. You can have 100 NSGs per subscription.on the VM.

Managing Network Security Groups

A NSG is a top level object that is associated to your subscription. An NSG contains access control rules that allow or deny traffic to VM instances. The rules of an NSG can be changed at any time, and changes are applied to all associated instances.

A network security group has a Name, is associated to a Region, and has a descriptive label. It contains two types of rules, Inbound and Outbound. The Inbound rules are applied on the incoming packets to a VM and the Outbound rules are applied to the outgoing packets from the VM. The rules are applied at the host where the VM is located. An incoming or outgoing packet has to match an Allow rule for it be permitted, if not it will be dropped.

Rules are processed in the order of priority. For example, a rule with a lower priority number (e.g. 100) is processed before rules with a higher priority numbers (e.g. 200). Once a match is found, no more rules are processed.

Default Network Security Group Rules

An NSG contains default rules. The default rules cannot be deleted, but because they are assigned the lowest priority, they can be overridden by the rules that you create. The default rules describe the default settings recommended by the platform. While connectivity to the Internet is allowed for Outbound direction, it is by default blocked for Inbound direction. There is a default rule to allow Azure’s load balancer (LB) to probe the health of the VM. You can override this rule if the VM or set of VMs under the NSG does not participate in the load balanced set.

Inbound

NAME	PRIORITY	SOURCE IP	SOURCE PORT	DESTINATION IP	DESTINATION PORT	PROTOCOL

ALLOW VNET INBOUND	65000	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*	*
ALLOW AZURE LOAD BALANCER INBOUND	65001	AZURE_LOADBALANCER	*	*	*	*
DENY ALL INBOUND	65500	*	*	*	*	*

Outbound

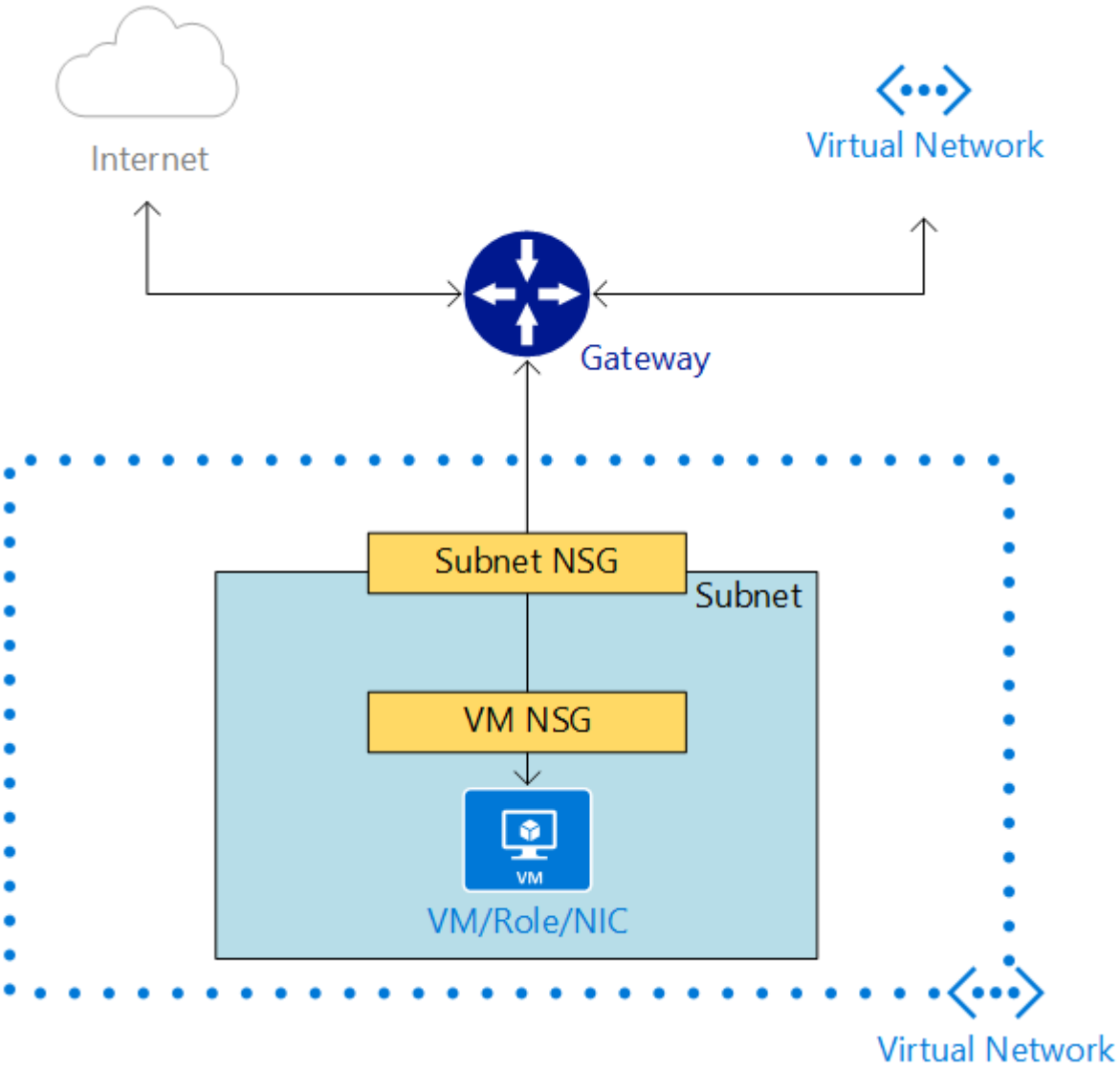
NAME	PRIORITY	SOURCE IP	SOURCE PORT	DESTINATION IP	DESTINATION PORT	PROTOCOL
ALLOW VNET OUTBOUND	65000	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*	*
ALLOW INTERNET OUTBOUND	65001	*	*	INTERNET	*	*
DENY ALL OUTBOUND	65500	*	*	*	*	*

Using Network Security Groups to Imply Connectivity Rules

Bookmark this page

Using Network Security Groups in a Subnet

When an NSG is associated to a subnet, the Network access rules in the NSG are applied to all the VMs in the subnet. Whenever the access rules in the NSG are updated the changes are applied to all Virtual machines in the subnet within minutes.



When an NSG is associated with a VM or subnet, the network access control rules becomes very explicit. The platform will not insert any implicit rule to allow traffic to a particular port. In this case, if you create an endpoint in the VM, you also have to create a rule to allow traffic from the Internet. If you don't do this, the VIP: will not be accessible from outside.

It is possible that you can associate an NSG to a VM and a different NSG to the subnet where the VM resides. This is supported and in this case the VM gets two layers of protection. On the Inbound traffic the packet goes through the access rules specified in the subnet followed by rules in the VM and in the Outbound case it goes through the rules specified in the VM first before going through the rules specified in the subnet, as illustrated in the diagram below.

For more information , you can see:
virtual machines: https://aka.ms/edx-dev205bx-az11

Windows Server Firewall

Bookmark this page

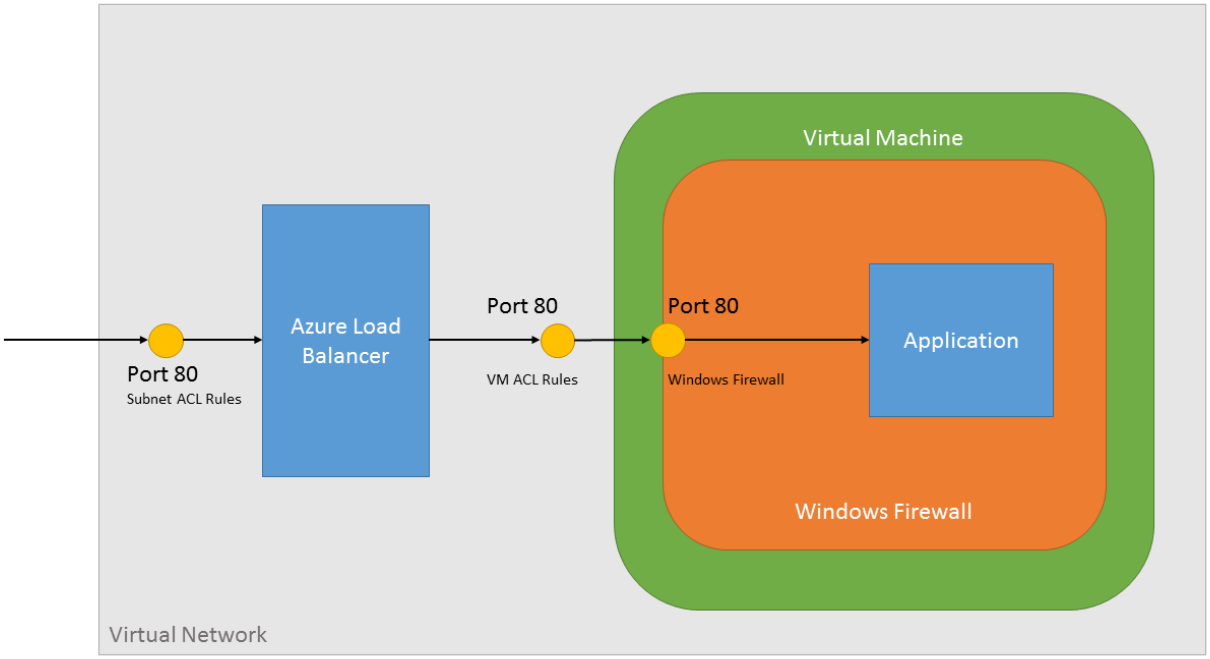
Windows Server Firewall

Windows Server Firewall is an important part of a layered security model. By providing host-based, two-way network traffic filtering for a computer, Windows Firewall with Advanced Security blocks unauthorized network traffic flowing into or out of the local computer.

In order to provide maximum protection for a virtual machine within a virtual network, Windows Firewall can be activated and configured initially to block all inbound connections. Then ports could be opened on an as-needed basis, adding IPsec in order to enforce secure connections and also encrypt those connections.

You can manage these firewall rules individually for each virtual machine or you can use Active Directory group policy to apply firewall rules to multiple machines in a domain. Alternatively, a configuration management utility (such as Chef and Puppet) can be used to manage firewall rules.

Network Security Groups and Windows Firewall



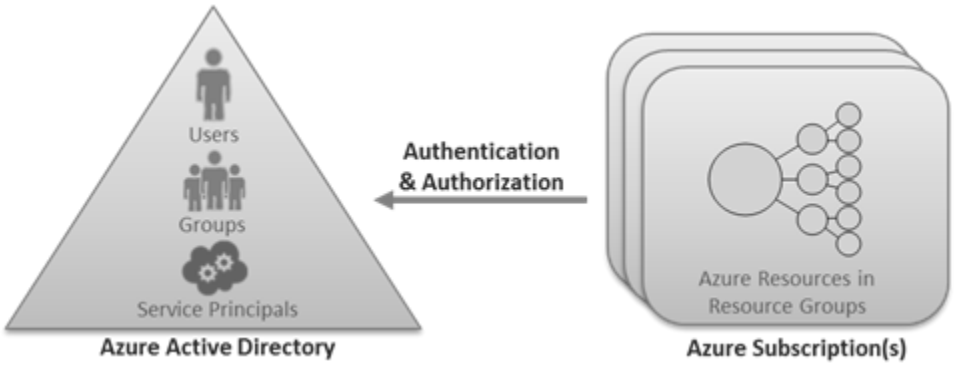
Azure Subscriptions

Bookmark this page

Azure Subscriptions

Subscriptions help you organize access to cloud service resources. They also help you control how resource usage is reported, billed, and paid for. Each subscription can have a different billing and payment setup, so you can have different subscriptions and different plans by department, project, regional office, and so on. Every cloud service belongs to a subscription, and the subscription ID may be required for programmatic operations.

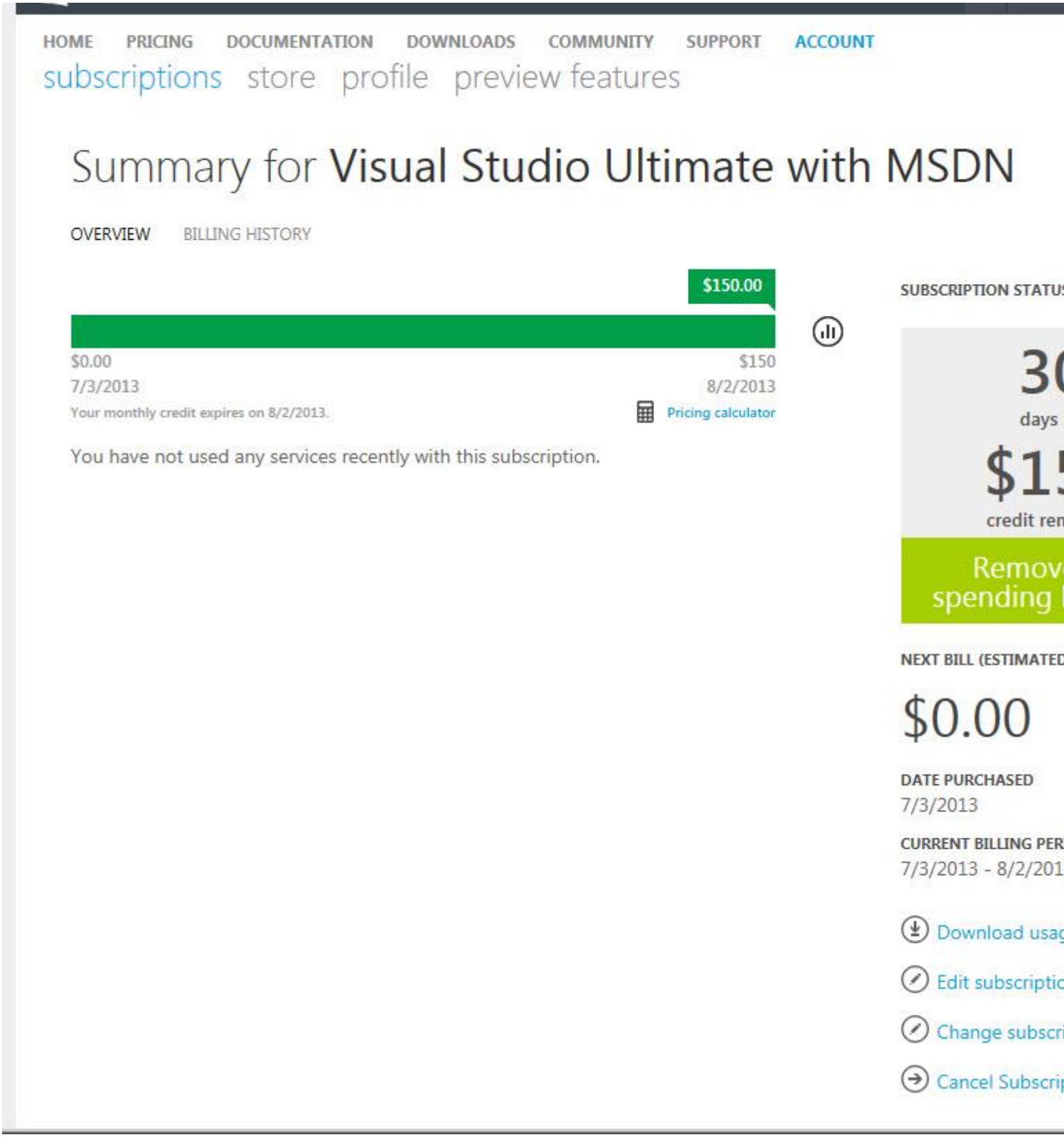
Every Azure subscription is associated with an Azure Active Directory. Users and services that access resources of the subscription using the Microsoft Azure Management Portal or Azure Resource Manager API first need to authenticate with that Azure Active Directory.



Typically to grant a user access to your Azure resources, you would add them to the Azure AD directory associated with your subscription. The user will now have access to all of the resources in your subscription. This is an all-or-nothing operation that may give that user access to more resources than you anticipated.

Azure Accounts

An Azure account determines how Azure usage is reported and who the Account Administrator is. Accounts and subscriptions are created at the Azure Account Center. The person who creates the account is the Account Administrator for all subscriptions created in that account. That person is also the default Service Administrator for the subscription.



Account Administrators using a Microsoft account must log in every 2 years (or more frequently) to keep the account active. Inactive accounts are cancelled and the related subscriptions removed. There are no login requirements if using a work or school account.

For more information , you can see:
Azure Active Directory: https://aka.ms/edx-dev205bx-azad
Microsoft Azure: https://aka.ms/edx-dev205bx-az34
Microsoft Azure portal: https://aka.ms/edx-dev205bx-az21

Subscription User Types

Bookmark this page

Subscription User Types

There are three roles related to Azure accounts and subscriptions:

Administrative role	Limit	Summary
Account Administrator	1 per Azure account	Authorized to access the Account Center (create subscri cancel subscriptions, change billing for a subscription, ch Service Administrator, and more)
Service Administrator	1 per Azure subscription	Authorized to access Azure Management Portal for all subscriptions in the account. By default, same as the Ac Administrator when a subscription is created.
Co-administrator	200 per subscription (in addition to Service Administrator)	Same as Service Administrator, but can't change the association of subscriptions to Azure directories.

The Account Administrator for a subscription is the only person with access to the Account Center. The Account Administrator does not have any other access to services in that subscription; they need to also be the Service Administrator or a co-administrator for that. For security reasons, the Account Administrator for a subscription can only be changed with a call to Azure support. The Account Administrator can easily reassign the Service Administrator for a subscription at the Account Center at any time.

The Service Administrator is the first co-administrator for a subscription. Like other co-administrators, the Service Administrator has management access to cloud resources using the Azure Management Portal, as well as tools like Visual Studio, other SDKs, and command line tools like PowerShell. The Service Administrator can also add and remove other co-administrators.

Additionally, Co-administrators can't delete the Service Administrator from the Azure Management Portal. Only the Account Administrator can change this assignment at the Account Center. The Service Administrator is the only user authorized to change a subscription's association with a directory in the Azure Management Portal.

For more information , you can see:
Visual Studio: https://aka.ms/edx-dev205bx-vs
Windows PowerShell: https://aka.ms/edx-dev205bx-ps

Role-Based Access Control (RBAC)

Bookmark this page

Role-Based Access Control (RBAC)

Azure role-based access control allows you to grant appropriate access to Azure AD users, groups, and services, by assigning roles to them on a subscription or resource group or individual resource level. The assigned role defines the level of access that the users, groups, or services have on the Azure resource.

Role

A role is a collection of actions that can be performed on Azure resources. A user or a service is allowed to perform an action on an Azure resource if they have been assigned a role that contains that action. There are built-in roles that include (but is not limited to):

ROLE NAME	DESCRIPTION
Contributor	Contributors can manage everything except access.
Owner	Owner can manage everything, including access.
Reader	Readers can view everything, but can't make changes.
User Access Administrator	Lets you manage user access to Azure resources.

Virtual Machine Contributor	Lets you manage virtual machines, but not access to them, and not the virtual network or storage accou they're connected to.

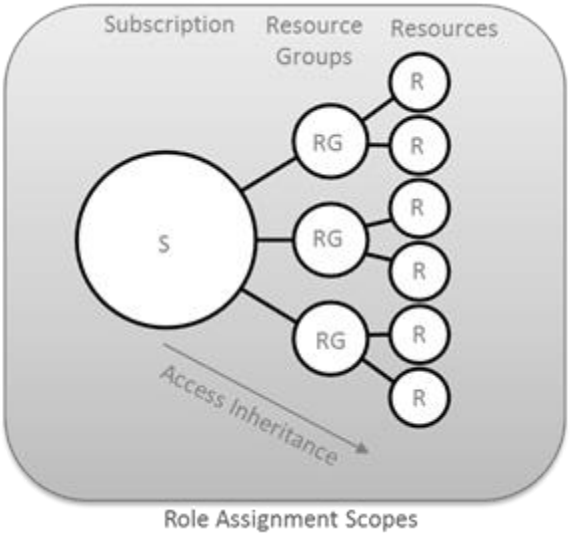
Role Assignment

A role assignment can be created that associates a security principal to a role. The role is further used to grant access to a resource scope. This decoupling allows you to specify that a specific role has access to a resource in your subscription and add/remove security principals from that role in a loosely connected manner. Roles can be assigned to the following types of Azure AD security principals:

- **Users**: roles can be assigned to organizational users that are in the Azure AD with which the Azure subscription is associated. Roles can also be assigned to external Microsoft accounts that exist in the same directory.
- **Groups**: roles can be assigned to Azure AD security groups. A user is automatically granted access to a resource if the user becomes a member of a group that has access. The user also automatically loses access to the resource after getting removed from the group. Managing access via groups by assigning roles to groups and adding users to those groups is the best practice, instead of assigning roles directly to users.
- **Service principals**: service identities are represented as service principals in the directory. They authenticate with Azure AD and securely communicate with one another. Services can be granted access to Azure resources by assigning roles via the Azure module for Windows PowerShell to the Azure AD service principal representing that service.

Resource Scope

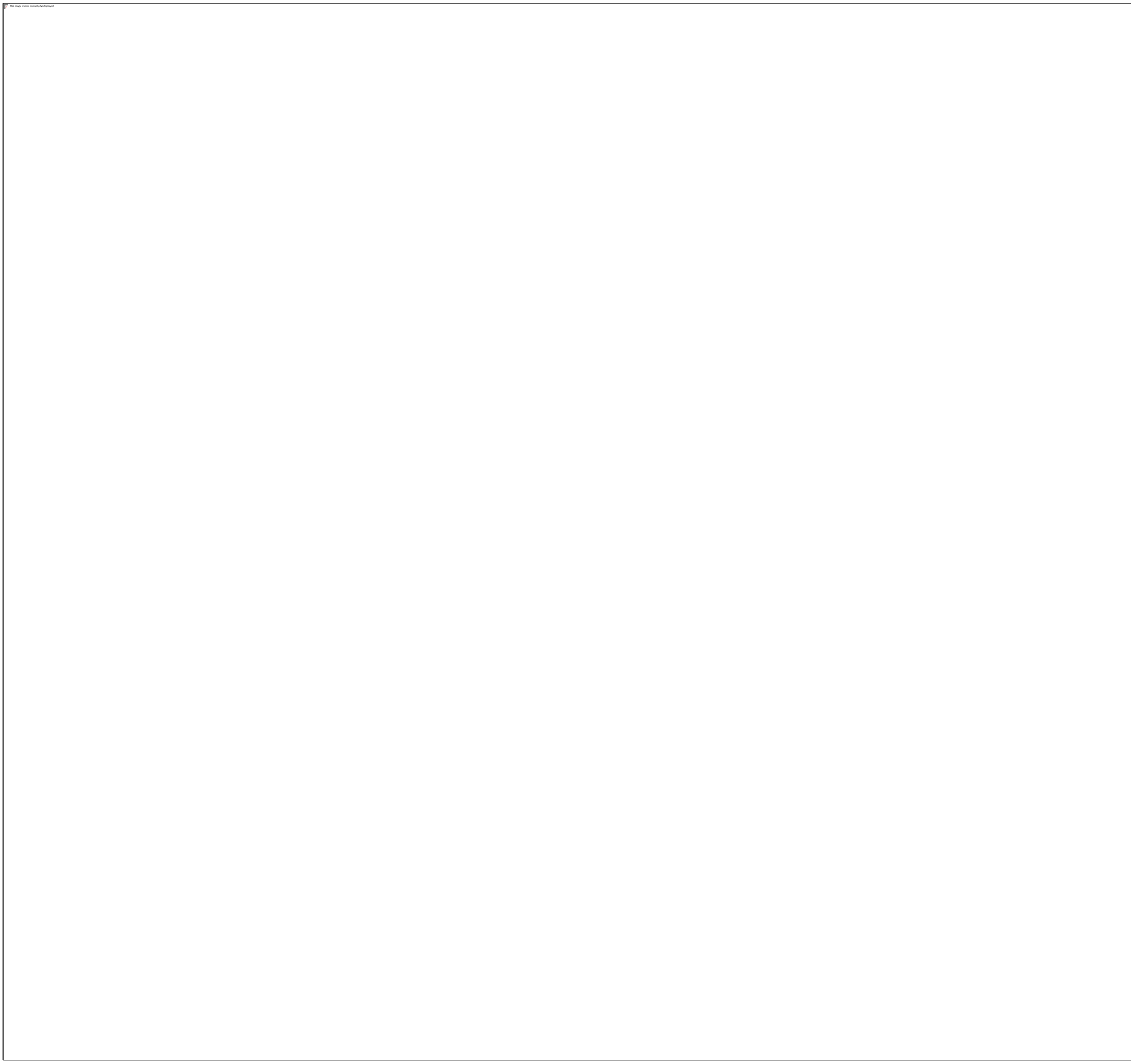
Access does not need to be granted to the entire subscription. Roles can also be assigned for resource groups as well as for individual resources. In Azure RBAC, a resource inherits role assignments from its parent resources. So if a user, group, or service is granted access to only a resource group within a subscription, they will be able to access only that resource group and resources within it, and not the other resources groups within the subscription. As another example, a security group can be added to the Reader role for a resource group, but be added to the Contributor role for a database within that resource group.



RBAC and Resource Groups

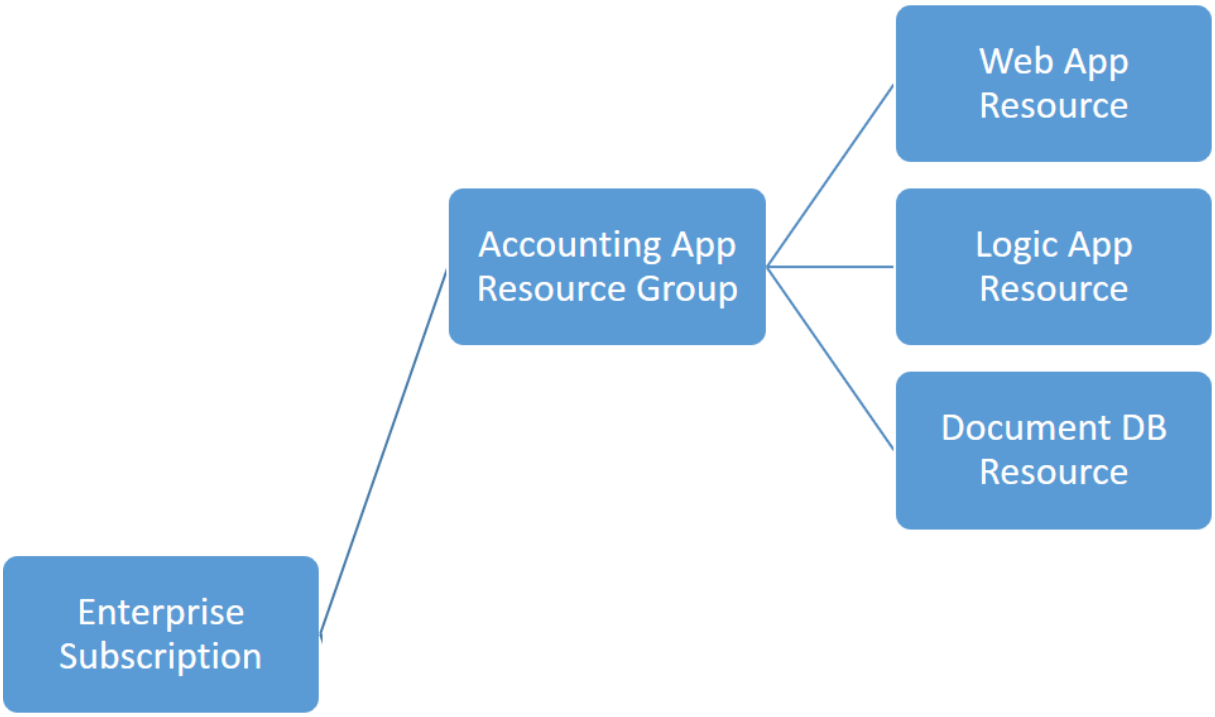
Bookmark this page

RBAC and Resource Groups



Scoping to Resource Groups

While RBAC can easily be used to grant access to an individual resource, it is preferrable to grant access to an entire Resource Group as opposed to indvidiual resources. By scoping to a resource group, you can add/remove and modify resources easily without having to recreate assignments and scopes. You can also give an individual owner or contributor access to a resource group so that they can create, recreate or destroy resources on their own without requiring involvement from the account administrator.



In this example, There is a resoruce group for an Accounting application. This resource group contains multiple resources that are used in the application. By granting an individual owner or contributor access to the resource group, they can configure resources, create new deployments, add new resources or create automation scripts at will without requiring additional administrator assistance or having access to resources in other resource groups.

Storage Security

Bookmark this page

Storage Security

Azure Storage flexibly stores and provides retrieval access for large amounts of unstructured data. By default, only the storage account owner can access resources in the storage account. For the security of your data, every request made against resources in your account must be authenticated. Authentication relies on a Shared Key model. Blobs can also be configured to support anonymous authentication.

Your storage account is assigned two private access keys on creation that are used for authentication. Having two keys ensures that your application remains available when you regularly regenerate the keys as a common security key management practice. While you may access storage services using your key and HTTP, using HTTPS for secure access is highly recommended.

If you do need to allow users controlled access to your storage resources, you can use of the following options:

- You can set a container's permissions to permit anonymous read access to the container and its blobs. This is not allowed for tables or queues.

- You can expose a resource via a shared access signature, which enables you to delegate restricted access to a container, blob, table or queue resource by specifying the interval for which the resources are available and the permissions that a client will have to it.

- You can use a stored access policy to manage shared access signatures for a container or its blobs, for a queue, or for a table. The stored access policy gives you an additional measure of control over your shared access signatures and also provides a straightforward means to revoke them.

Anonymous Access

To give anonymous users read permissions to a container and its blobs, you can set the container permissions to allow public access. Anonymous users can read blobs within a publicly accessible container without authenticating the request. Containers provide the following options for managing container access:

- **Full public read access**: Container and blob data can be read via anonymous request. Clients can enumerate blobs within the container via anonymous request, but cannot enumerate containers within the storage account.
- **Public read access for blobs only**: Blob data within this container can be read via anonymous request, but container data is not available. Clients cannot enumerate blobs within the container via anonymous request.
- **No public read access**: Container and blob data can be read by the account owner only.

If your service requires that you exercise more granular control over blob resources, or if you wish to provide permissions for operations other than read operations, you can use a Shared Access Signature to make a resource accessible to users.

Shared Access Signatures and Stored Access Policies

Shared Access Signatures

A shared access signature is a URI that grants restricted access rights to containers, blobs, queues, and tables for a specific time interval. By providing a client with a shared access signature, you can enable them to access resources in your storage account without sharing your account key with them.

The shared access signature URI query parameters incorporate all of the information necessary to grant controlled access to a storage resource. The URI query parameters

specify the time interval over which the shared access signature is valid, the permissions that it grants, the resource that is to be made available, and the signature that the storage services should use to authenticate the request.

Here is an example of a SAS URI that provides read and write permissions to a blob. The table breaks down each part of the URI to understand how it contributes to the SAS:

https://myaccount.blob.core.windows.net/sascontainer/sasblob.txt?sv=2012-02-12&st=2013-04-29T22%3A18%3A26Z&se=2013-04-30T02%3A23%3A26Z&sr=b&sp=rw&sig=Z%2FRHIX5Xcg0Mq2rqI3OlWTjEg2tYkboXr1P9ZUXDtkk%3D

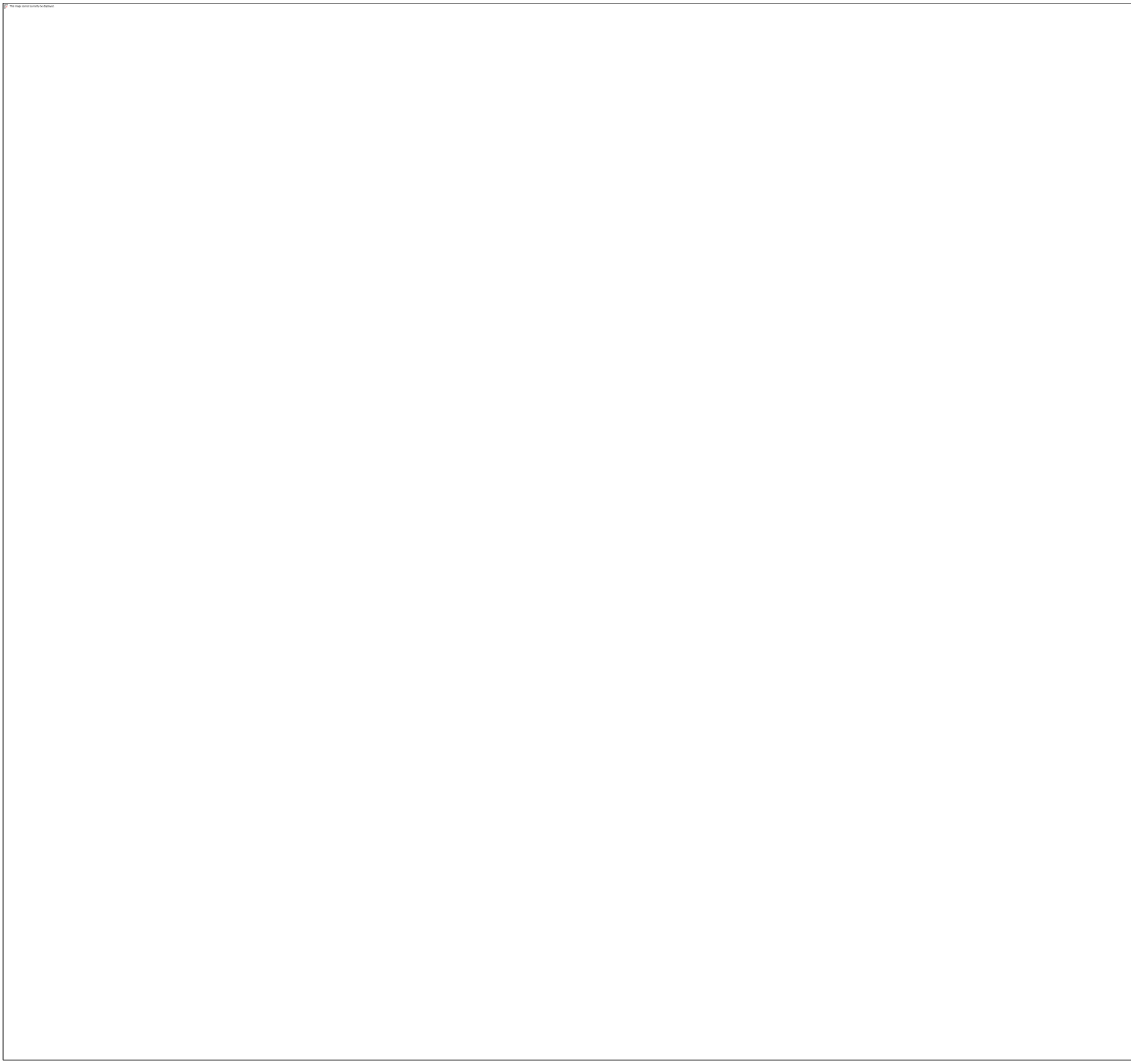
Blob URI	https://myaccount.blob.core.windows.net/sascontainer/sasblob.txt	The address of the blo Note that using HTTPS highly recommended.
Storage services version	sv=2012-02-12	For storage services v 2012-02-12 and later parameter indicates th version to use.
Start time	st=2013-04-29T22%3A18%3A26Z	Specified in an ISO 80 format. If you want th to be valid immediate omit the start time.
Expiry time	se=2013-04-30T02%3A23%3A26Z	Specified in an ISO 80 format.
Resource	sr=b	The resource is a blob
Permissions	sp=rw	The permissions grant the SAS include Read and Write (w).
Signature	sig=Z%2FRHIX5Xcg0Mq2rqI3OlWTjEg2tYkboXr1P9ZUXDtkk%3D	Used to authenticate a to the blob. The signa

		an HMAC computed o string-to-sign and key the SHA256 algorithm then encoded using B encoding.

Valet-Key Pattern using Shared Access Signatures

A common scenario where a SAS is useful is a service where users read and write their own data to your storage account. In a scenario where a storage account stores user data, there are two typical design patterns:

1. Clients upload and download data via a front-end proxy service, which performs authentication. This front-end proxy service has the advantage of allowing validation of business rules, but for large amounts of data or high-volume transactions, creating a service that can scale to match demand may be expensive or difficult.
2. Using the **Valet Key Pattern**, A lightweight service authenticates the client as needed and then generates a SAS. Once the client receives the SAS, they can access storage account resources directly with the permissions defined by the SAS and for the interval allowed by the SAS. The SAS mitigates the need for routing all data through the front-end proxy service.



Stored Access Policies

A shared access signature can take one of two forms:

- **Ad hoc SAS:** When you create an ad hoc SAS, the start time, expiry time, and permissions for the SAS are all specified on the SAS URI (or implied, in the case where start time is omitted). This type of SAS may be created on a container, blob, table, or queue.

- **SAS with stored access policy:** A stored access policy is defined on a resource container - a blob container, table, or queue - and can be used to manage constraints for one or more shared access signatures. When you associate a SAS with a stored access policy, the SAS inherits the constraints - the start time, expiry time, and permissions - defined for the stored access policy.

  The difference between the two forms is important for one key scenario: revocation. A SAS is a URL, so anyone who obtains the SAS can use it, regardless of who requested it to begin with. If a SAS is published publically, it can be used by anyone in the world. Stored access policies give you the option to revoke permissions without having to regenerate the storage account keys. Set the expiration on these to be a very long time (or infinite) and make sure that it is regularly updated to move it farther into the future.

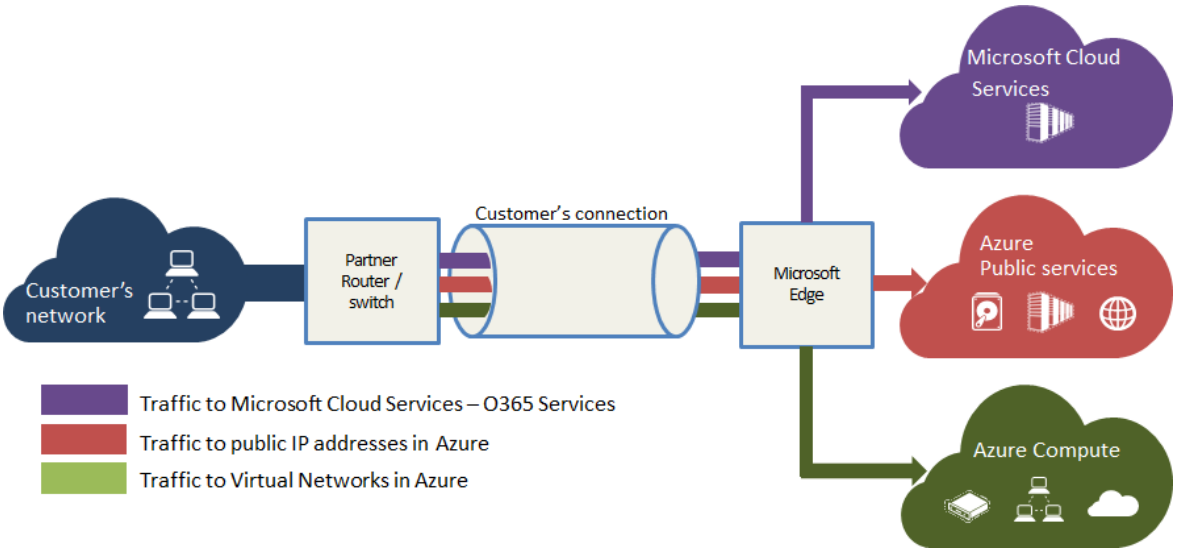
Express Route

ExpressRoute

ExpressRoute offers private connections between Microsoft data centers and your existing infrastructure that's located either on-premise or in a co-location environment. ExpressRoute is offered in two primary ways:

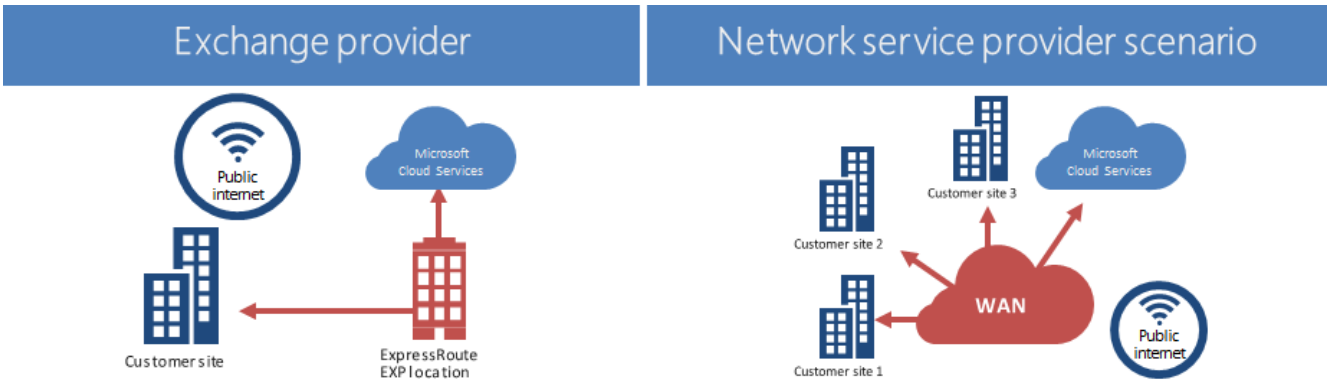
- You can connect to a partner co-location facility or host your hardware in the facility. This facility will have a private connection directly to Azure data centers.
- You can connect directly from your Wide Area Network (WAN) to an Azure data center using a VPN connection (Ex. MPLS) provided by your network service provider.

  ExpressRoute's connections are more secure, faster and more reliable than a typical internet connection. ExpressRoute's private connections do not need to compete with internet traffic and this can yield faster speeds, lower latency and much higher reliability. ExpressRoute does not require existing virtual networks to be reconfigured. ExpressRoute does require that you dedicate a circuit to connect through your connectivity provider.



Provider Types

ExpressRoute providers are classified as Network Service Providers (NSPs) and Exchange providers (EXPs). You can choose to enable one, or both types of connectivity between your WAN and the Microsoft cloud.



	Exchange Provider	Network Service Provider
Typical Connectivity Model	Point-to-point Ethernet links or Connectivity at a cloud exchange	Any-to-any connectivity through a tel
Supported Bandwidths	200 Mbps, 500 Mbps, 1 Gbps and 10 Gbps	10 Mbps, 50 Mbps, 100 Mbps, 500 M Gbps

Routing	BGP sessions directly with customer edge routers	BGP sessions with telco

Exchange providers (EXPs)

Exchange providers offer a direct layer 3 connection with circuit bandwidths from 200 Mbps to 10 Gbps. They offer this in three different ways:

- You can be co-located with the cloud exchanges in the locations Azure offer services in. In such cases you will order redundant connectivity to the cloud exchange.
- You can work with providers to have Ethernet circuits setup between your data centers and Microsoft.
- You can work with your local connectivity provider to acquire redundant connectivity to the closest exchange provider facility and connect to the cloud exchange.

In order to meet the Azure SLA requirements, you are required to have redundant connectivity.

Network service providers (NSPs)

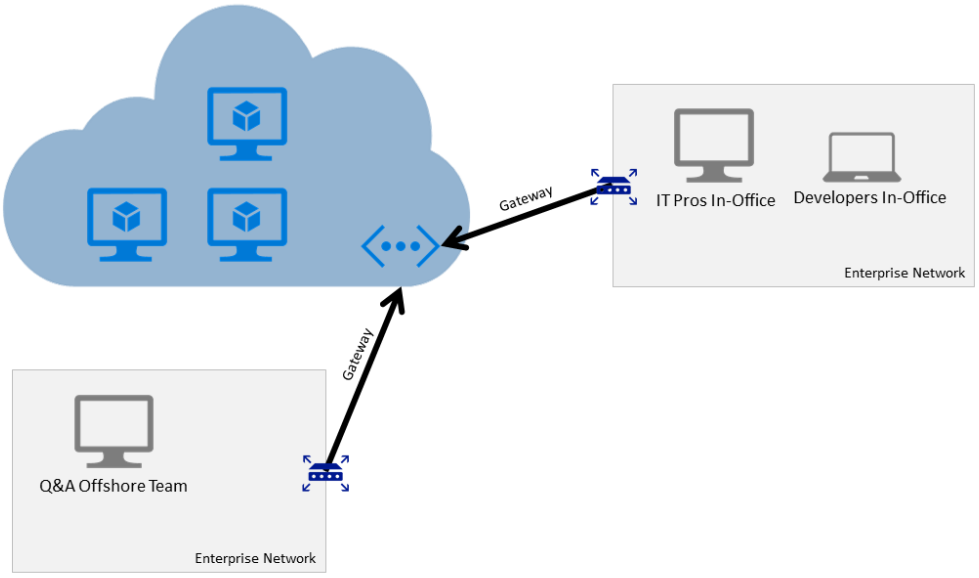
Alternatively, connectivity can be offered between Azure and your Wide Area Network (WAN) with circuit bandwidths from 10 Mbps to 1 Gbps. This offering has the least amount of friction to a current networking setup as it typically does not require the deployment of new hardware or major configuration changes to existing networks.

For more information , you can see:
ExpressRoute: https://aka.ms/edx-dev205bx-az22

Site-to-Site

Bookmark this page

Site-to-Site



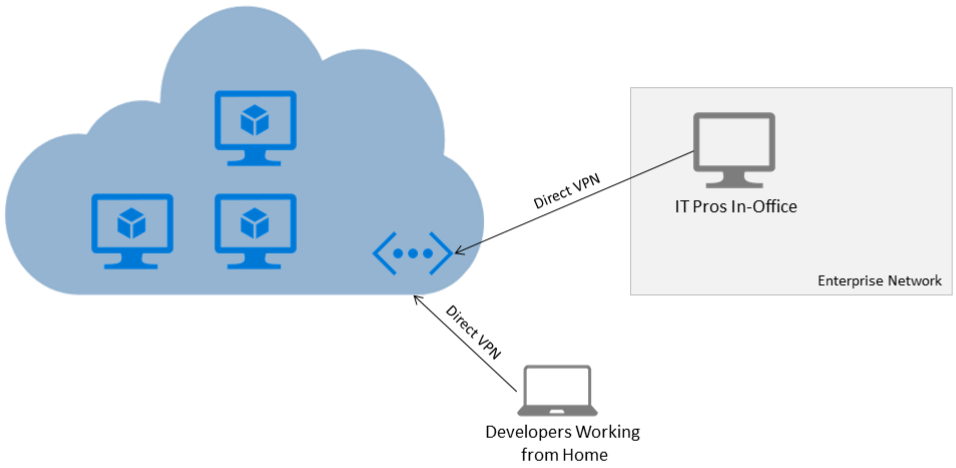
Site-to-Site Connectivity

A site-to-site VPN allows you to create a secure connection between your on-premises site and your virtual network. To create a site-to-site connection, a VPN device that is located on your on-premises network is configured to create a secure connection with the Azure Virtual Network Gateway. Once the connection is created, resources on your local network and resources located in your virtual network can communicate directly and securely. Site-to-site connections do not require you to establish a separate connection for each client computer on your local network to access resources in the virtual network.

Point-to-Site

Bookmark this page

Point-to-Site



Point-to-Site Connectivity

A point-to-site VPN also allows you to create a secure connection to your virtual network. In a point-to-site configuration, the connection is configured individually on each client computer that you want to connect to the virtual network. Point-to-site connections do not require a VPN device. They work by using a VPN client that you install on each client computer. The VPN is established by manually starting the connection from the on-premises client computer. You can also configure the VPN client to automatically restart.

Point-to-site and site-to-site configurations can exist concurrently.

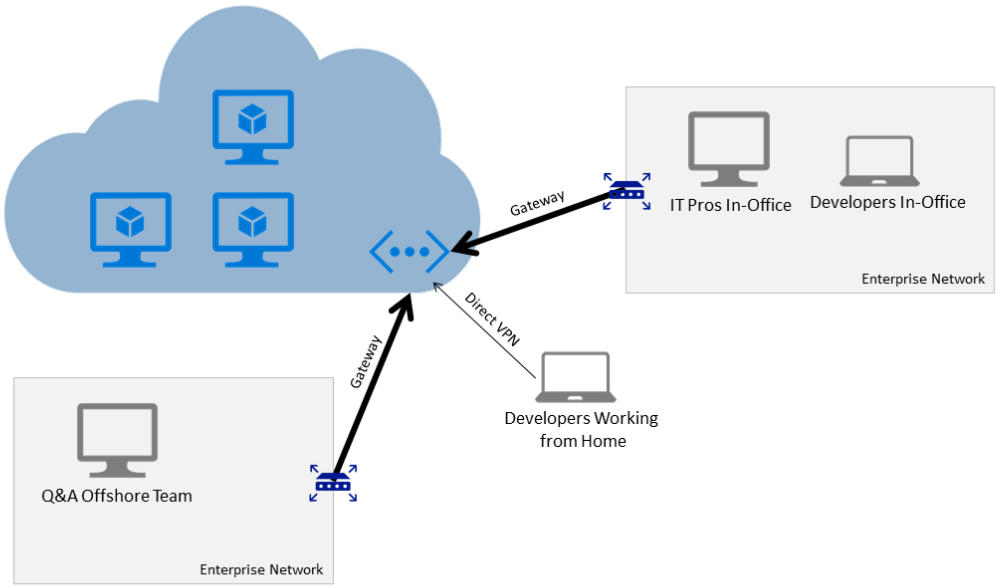
For more information , you can see:
virtual network: https://aka.ms/edx-dev205bx-az12

Mixing and Matching the Networking Options

Bookmark this page

Combining Site-to-Site and Point-to-Site Connectivity

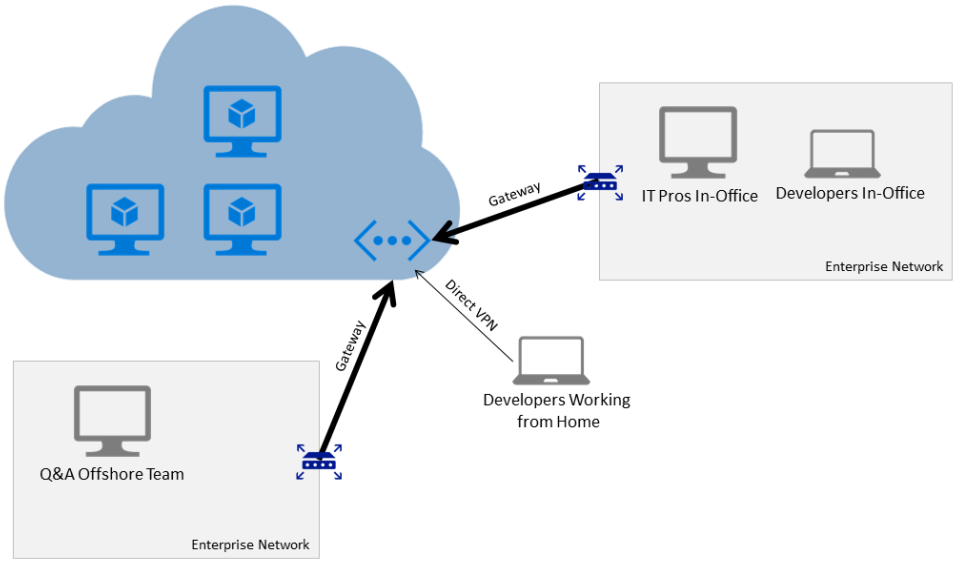
Site-to-Site and Point-to-Site connections can be combined for a variety of reasons. In the diagram below, the enterprise has decided to use a Site-to-Site connection to connect the in-office networks to Azure. Developers who are working remotely can connect to the Virtual Networks directly using a Point-to-Site connection.



Combining ExpressRoute and Site-to-Site Connectivity

You can connect ExpressRoute and a Site-to-Site VPN on the same virtual network. There are many reasons you may wish to do this.

- You may have multiple branch offices and it would be cost prohibitive to purchase peering for every location. You can use Site-to-Site VPN for the locations that don't require the fastest or most reliable connections.
- You may have multiple networks within your enterprise and may wish to connect one to Azure using ExpressRoute and one to Azure using Site-to-Site VPN so there are two active connections. The ExpressRoute connection could be used for higher-risk traffic.
- You can use the Site-to-Site VPN as a failover link if the ExpressRoute connection fails.



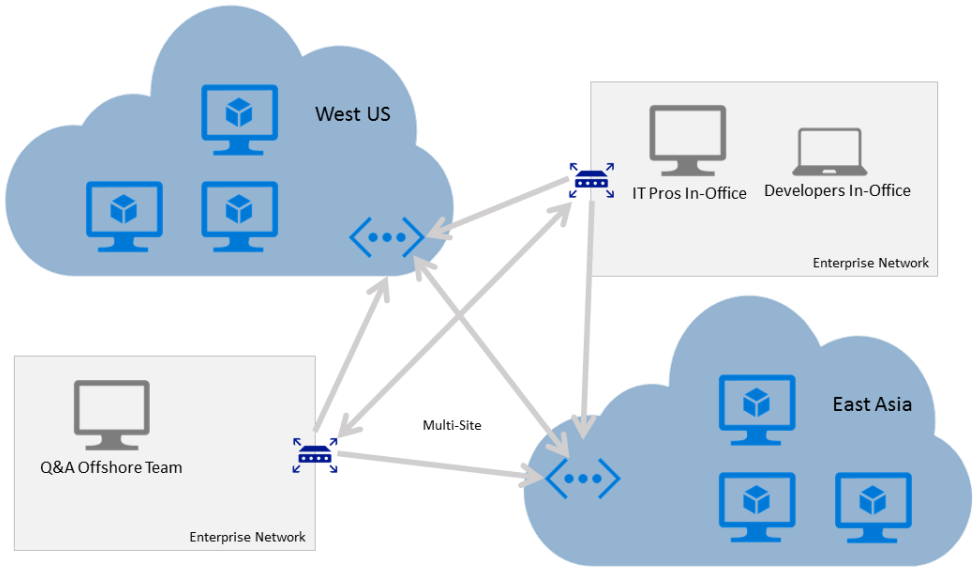
For more information , you can see:
Azure: https://aka.ms/edx-dev205bx-az34

Virtual Network to Virtual Network

Bookmark this page

Virtual Network to Virtual Network Connectivity

VNet-to-VNet connectivity utilizes the Azure VPN gateways to connect two or more virtual networks together securely with IPsec/IKE S2S VPN tunnels. Together with the Multi-Site VPNs, you can connect your virtual networks and on-premises sites together in a topology that suits your business need. The diagram below shows a simple example of a fully connected topology between virtual networks and on premises sites.

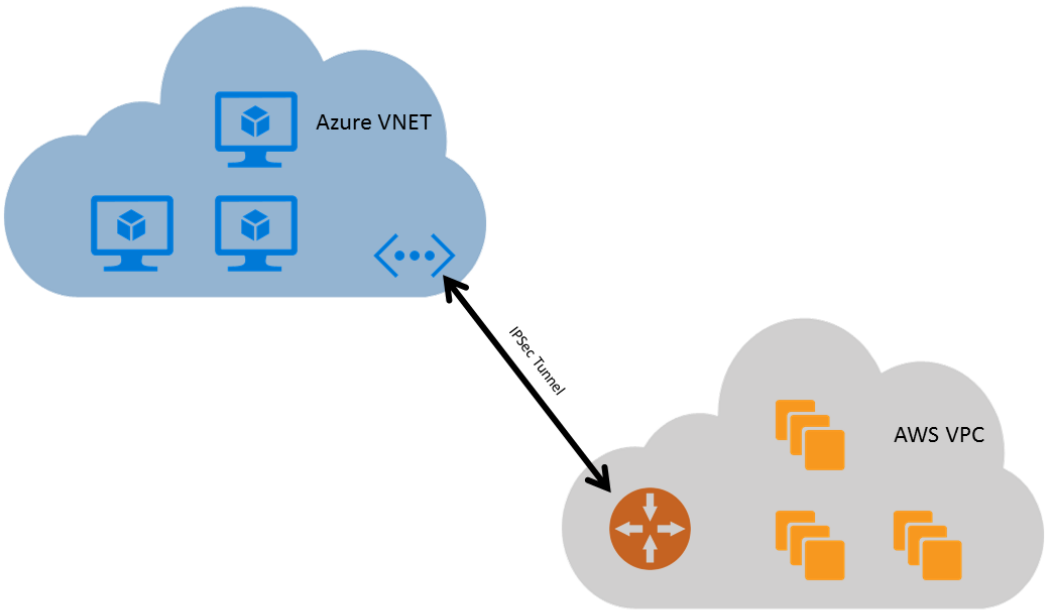


Connecting Across Cloud Providers

Since a Site-to-Site connection is simply an IPsec tunnel, you can connect to networks across cloud providers. This scenario could be used for failover, backup or even migration between providers.

Amazon Web Services (AWS)

In AWS, you can create a Virtual Private Cloud (VPC) which provides network capabilities similar to a Virtual Network in Azure. An EC2 instance with OpenSwan (VPN software) can then be created for VPN functionality. After those instances are running, you simply create a gateway on the Azure VNET side using static routing. The Gateway IP Address from Azure is then used to configure OpenSwan for a tunnel connection between the two virtual networks.



Affinity Groups

Bookmark this page

Affinity Groups

You can use an affinity group to ensure that resources created within the same affinity group are physically hosts by servers that are close together, enabling these resources to communicate quicker. In the past, affinity groups were a requirement for creating Virtual Networks (VNets). At the time, the network manager service that managed VNets could only work within a set of physical servers or scale unit. Recent architectural improvements have increased the scope of network management to a region.

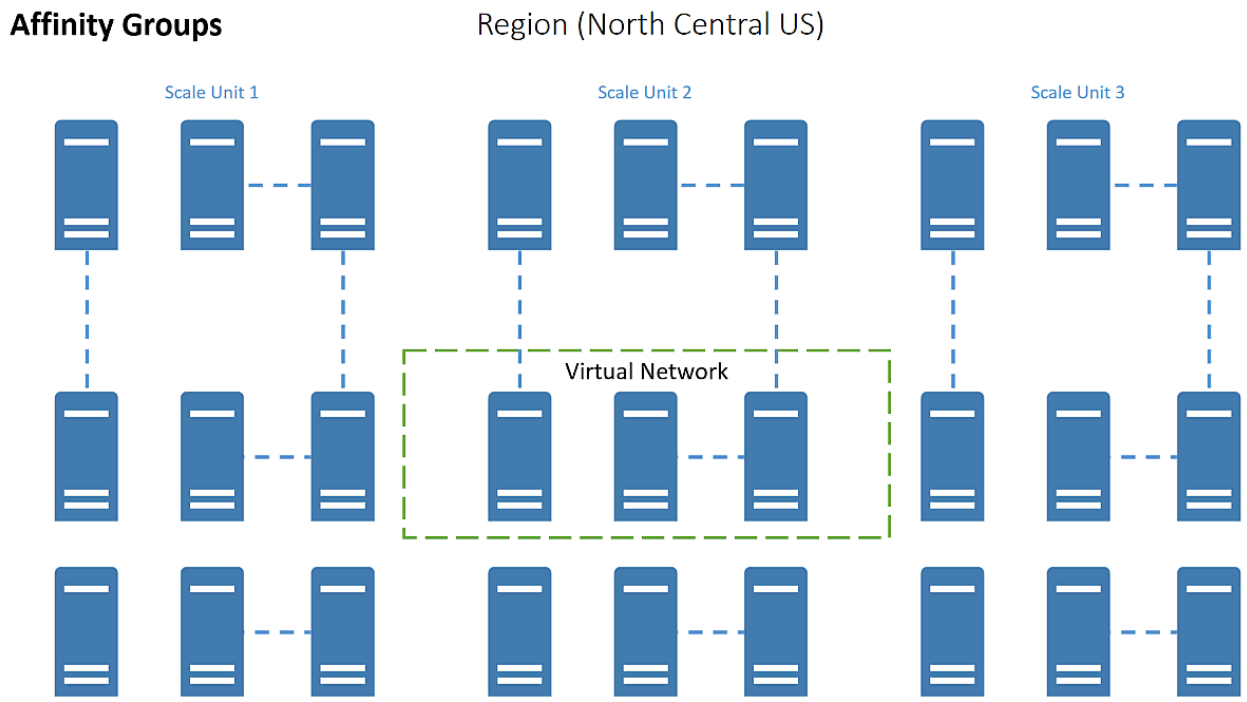
As a result of these architectural improvements, affinity groups are no longer recommended, or required for virtual networks. The use of affinity groups for VNets is being replaced by regions. VNets that are associated with regions are called regional VNets

Affinity groups are primarily still around for compatibility with existing virtual networks. Although it is still technically possible to create a virtual network that is associated with an affinity group, there is no compelling reason to do so. Many new features, such as Network Security Groups, are only available when using a regional VNet and are not available for virtual networks that are associated with affinity groups.

Regional Virtual Networks

Within Affinity Groups, VM deployments are confined to a single scale unit. With Regional Virtual Networks, VMs can be deployed across multiple scale units within the entire region. This allows VMs with new features such as Reserved IP Address, Internal Load Balancing, Instance Level Public IPs and A8/A9 sizes to coexist with existing

VMs. From a management perspective, the only difference is that you must specify a region when new Virtual Networks are created instead of an affinity group.



Cloud Services

Bookmark this page

Azure Compute Options

Azure provides different hosting models for running applications. Each one provides a different set of services, so which one you choose depends on your application workload.

COMPUTE OPTION	SCENARIOS
App Service	Scalable Web Apps, Mobile Apps, API Apps, and Logic Apps for any device
Cloud Services	Highly available, scalable n-tier cloud apps with more control of the OS

Virtual Machines	Customized Windows and Linux VMs with complete control of the OS

Cloud Services

Cloud Services is an example of Platform-as-a-Service (PaaS). The technology is designed to support applications that are scalable and resilient. Cloud Services are hosted on VMs that you can control through startup scripts or a remote connection. Cloud Services provides two slightly different VM options: instances of web roles run a variant of Windows Server with IIS, while instances of worker roles run the same Windows Server variant without IIS. A Cloud Services application relies on some combination of these two options.

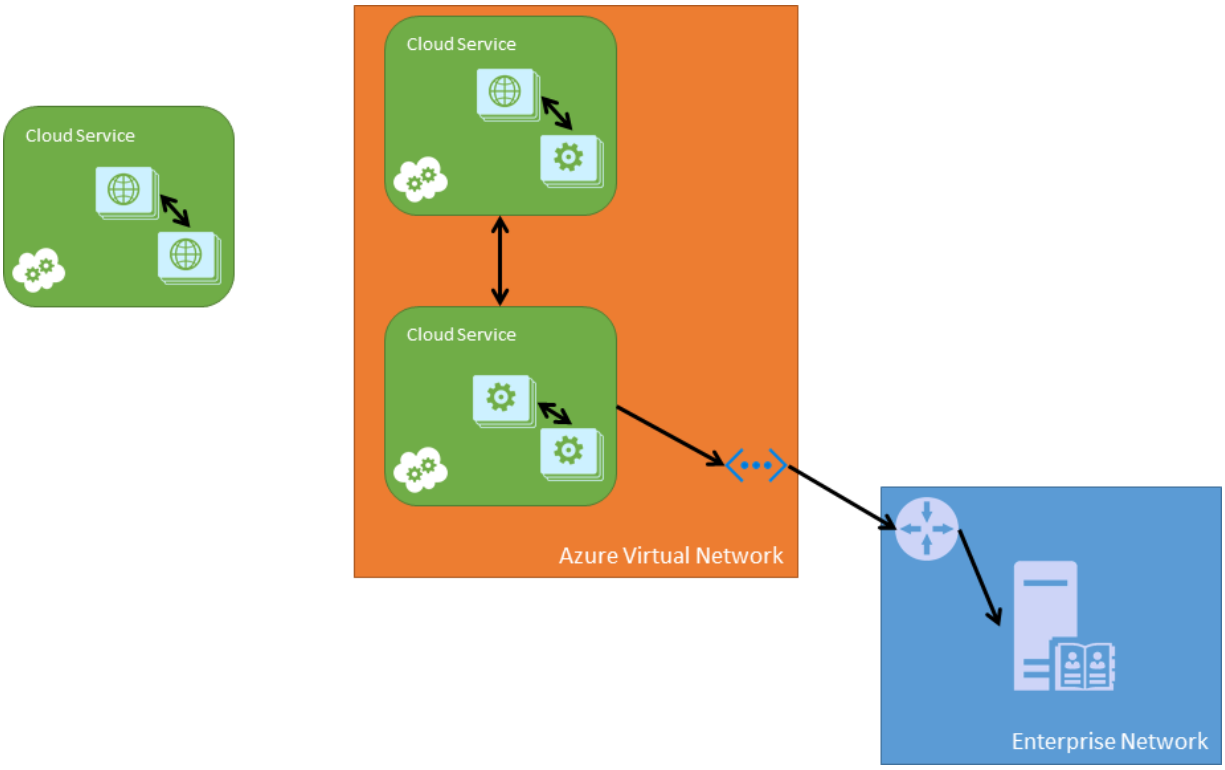
Cloud Services uses your configuration files, startup scripts and binaries to automate scaling and deployments tasks. This is what distinguishes Cloud Services as a PaaS service instead of an IaaS service. You still choose what size those backing VMs should be, but you don't explicitly create them yourself. If your application needs to handle a greater load, you can use configuration to requrest more VMs, and Azure will create those instances. If the load decreases, you can use configuration or automation to scale down your VM instance count.

More control also means less ease of use; unless you need the additional control options, it's typically quicker and easier to get a web application up and running in Websites compared to Cloud Services. It is also not recommended to use a remote desktop connection to install software on a Cloud Service role instance. Startup scripts should be used for these tasks as the steps are repeated in automated deployments.

Cloud Services and Virtual Networks

When creating a new instance of a Cloud Service, you can elect to add the Cloud Service to a Virtual Network. Cloud Services that are not connected to a Virtual Network are isolated and can only communicate with other instances/roles within the Cloud Service or with publically available internet endpoints. Once created, you must create a new Cloud Service if you wish to add your service instance to a Virtual Network.

Cloud Services that are in a Virtual Network can communicate with other service instances within the network using Private IP Addresses. This is ideal for scenarios where you would like a Cloud Service worker role to connect to a specialized database on an Azure Virtual Machine. This can also be used to allow communication between roles in separate Cloud Services or to allow a Cloud Service role to communicate with resources in an on-premise network.



For more information , you can see:
Cloud Services: https://aka.ms/edx-dev205bx-az10
Azure: https://aka.ms/edx-dev205bx-az34
windows sever: https://aka.ms/edx-dev205bx-ws
virtual network: https://aka.ms/edx-dev205bx-az12
Azure Virtual Machine: https://aka.ms/edx-dev205bx-az11

Virtual Machines

Bookmark this page

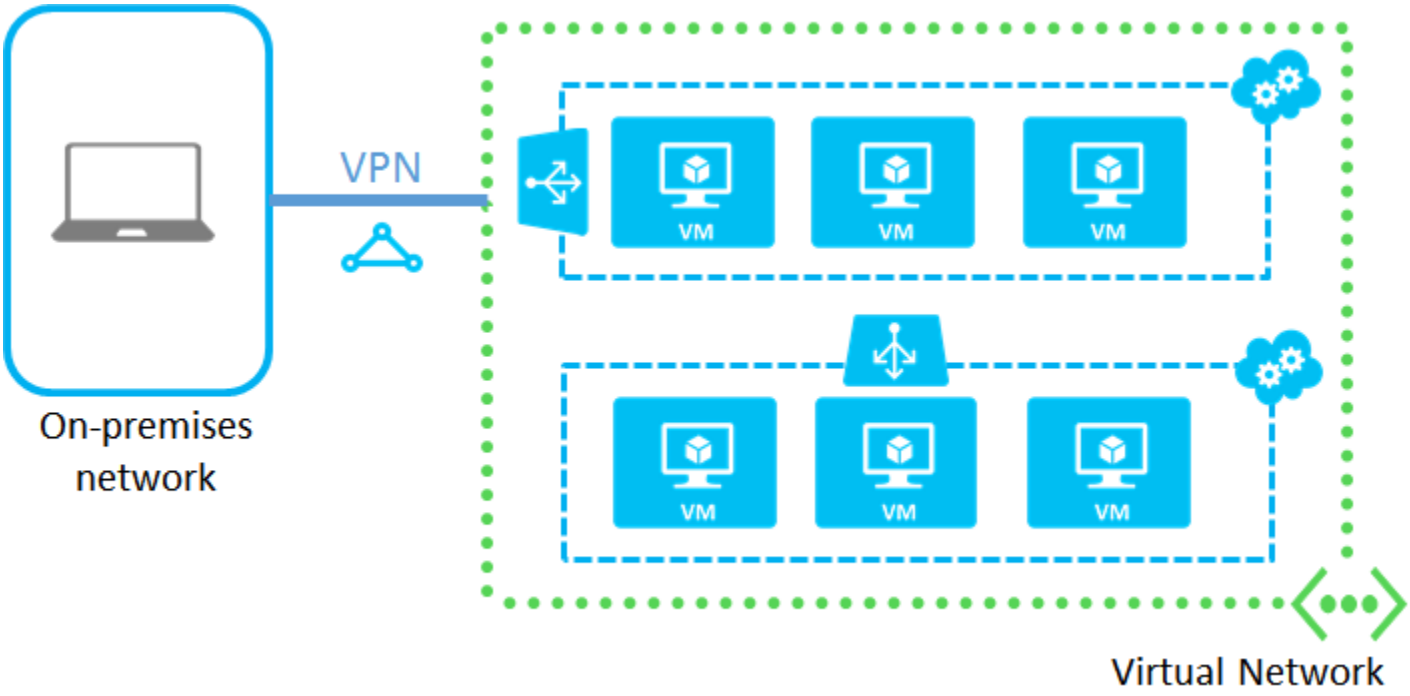
Virtual Machine Connectivity

Virtual machines are always placed in a cloud service, which acts as a container and provides a unique public DNS name, a public IP address, and a set of endpoints to access the virtual machine over the Internet. The cloud service can optionally be in a virtual network.

If a cloud service isn't in a virtual network, it's called a standalone cloud service. The virtual machines in that cloud service can only communicate with other virtual machines through the use of the other virtual machines' public DNS names, and that traffic would travel over the Internet. If a cloud service is in a virtual network, the virtual machines in that cloud service can communicate with all other virtual machines in the virtual network without sending any traffic over the Internet.

Generation 2 Virtual Machines no longer require you to place them in a Cloud Service as a logical container.

If you place your virtual machines in the same standalone cloud service, you can take still use load balancing and availability sets. However, you can't organize the virtual machines on subnets or connect a standalone cloud service to your on-premises network. If you place your virtual machines in a virtual network, you can decide how many cloud services you want to use for load balancing and availability sets. Additionally, you can organize the virtual machines on subnets in the same way as your on-premises network and connect the virtual network to your on-premises network.



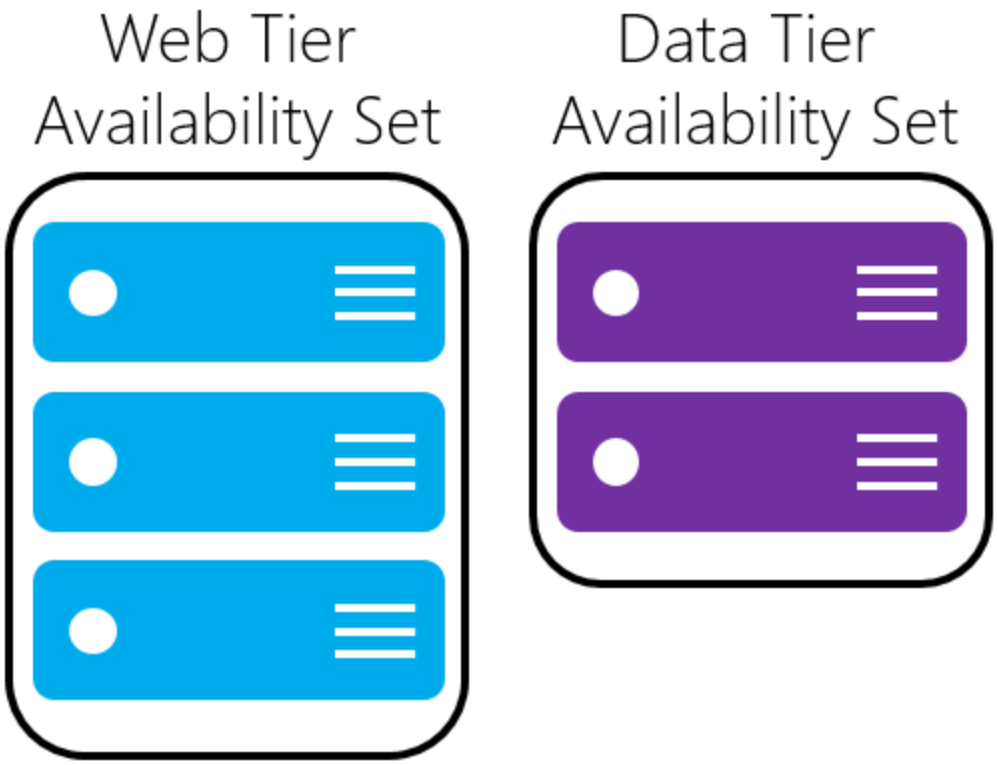
Availability Sets

An availability set helps keep your virtual machines available during downtime, such as during maintenance. Placing two or more similarly configured virtual machines in an availability set creates the redundancy needed to maintain availability of the applications or services that your virtual machine runs.

It's a best practice to use both availability sets and load-balancing endpoints to help ensure that your application is always available and running efficiently.

To provide redundancy to your application, we recommend that you group two or more virtual machines in an Availability Set. This configuration ensures that during either a planned or unplanned maintenance event, at least one virtual machine will be available and meet the 99.95% Azure SLA.

If the virtual machines in your availability set are all nearly identical and serve the same purpose for your application, it is recommend that you configure an Availability Set for each tier of your application. If you place two different tiers in the same Availability Set, all virtual machines in the same application tier could be rebooted at once. By configuring at least two virtual machines in an Availability Set for each tier, you guarantee that at least one virtual machine in each tier will be available.



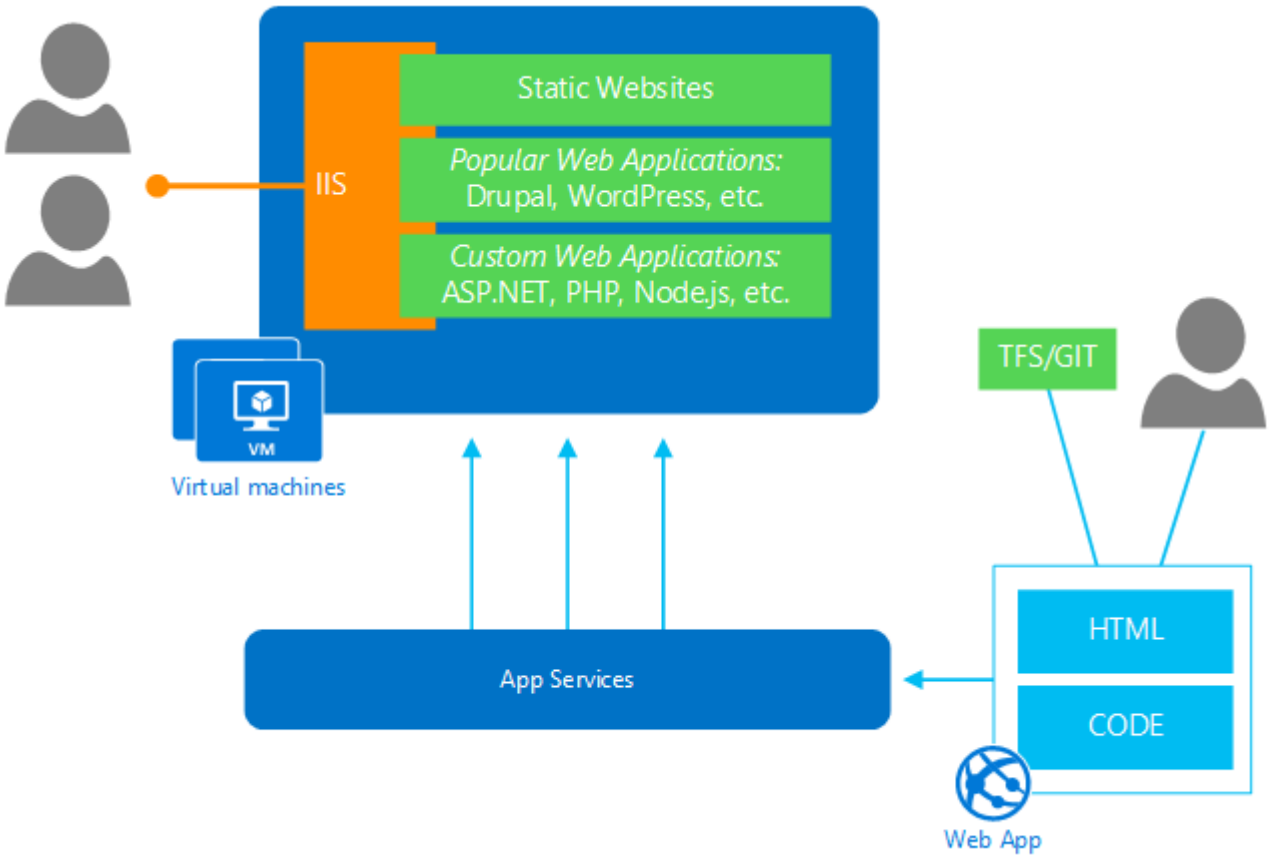
App Service

Bookmark this page

App Service

Azure Virtual Machines can handle a wide range of cloud hosting tasks. But creating and managing a VM infrastructure requires specialized skills and substantial effort. If you don't need complete control over the VMs that run your web apps, mobile app backends, API apps, etc., then you can use a Platform as a Service (PaaS) offering. App Service is a fully managed PaaS offering that allows you to manage your code and rely on Azure's infrastructure for management and scaling operations.

Azure App Service gives you the option of running on shared VMs that contain multiple apps created by multiple users, or on VMs that are used only by you. VMs are a part of a pool of resources managed by Azure App Service and thus allow for high reliability and fault tolerance.

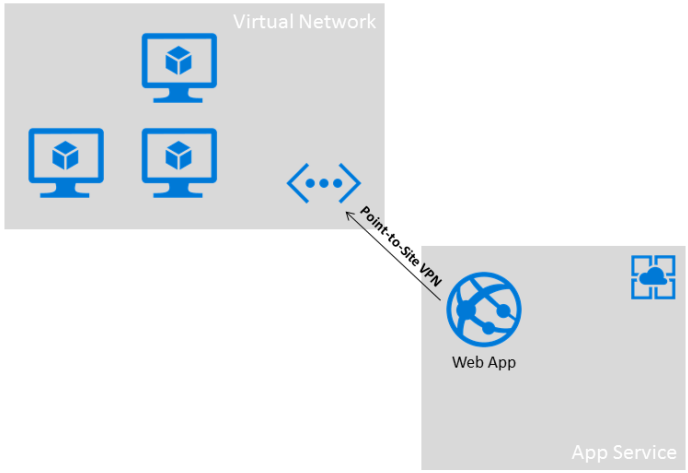


App Service Connectivity

Web Apps, Mobile Apps and API Apps can be connected to resources in a virtual network or on-premise using two different technologies.

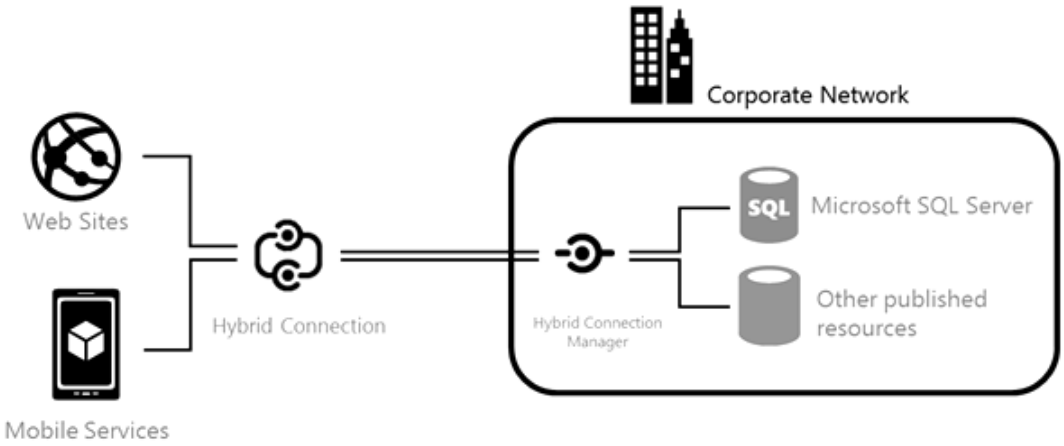
Virtual Networks

App Service apps can connect to resources with a Virtual Network and resources that are connected to an Azure Virtual Network using a Site-to-Site connection or ExpressRoute. To connect an App Service app to a VNET, the VNET must have Point-to-Site enabled and a dynamic routing gateway configured. This is because the Web App is not actually "inside" the VNET but is instead using a Point-to-Site connection to connect to the VNET for secure access.



Hybrid Connections

Hybrid Connections are powered by Azure BizTalk Services and allows you to connect your App Service apps to on-premise resources behind a firewall. Hybrid Connections connects to on-premise resources that expose a static TCP port. Hybrid Connections can be shared between multiple apps and are restricted to only the specific resource that is published through your Hybrid Connection.



For more information , you can see:
Azure App Service: https://aka.ms/edx-dev205bx-az07

Network Load Balancing (Internal, External, etc.)

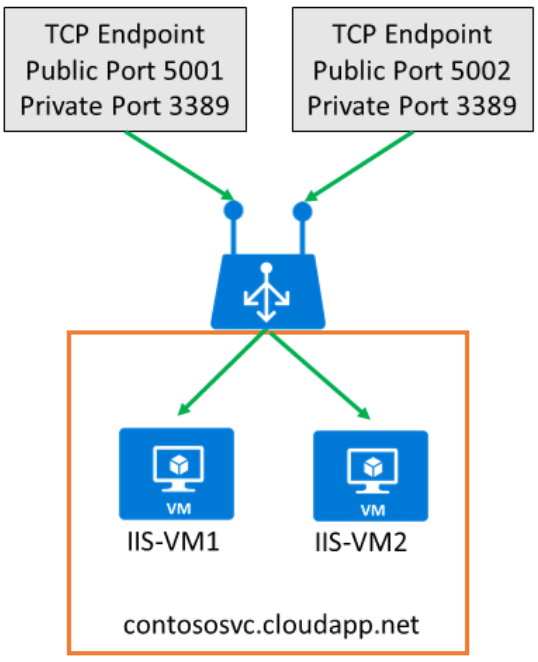
Bookmark this page

Network (External) Load Balancer

Azure load balancer delivers high availability and network performance to your applications. It is a Layer-4 (TCP, UDP) type load balancer that distributes incoming traffic among healthy service instances in cloud services or virtual machines defined in a load balancer set.

Azure Load Balancer provides you control over how inbound communication, such as traffic initiated from Internet hosts or virtual machines in other cloud services or virtual networks is managed. This control is represented by an endpoint (also referred as Input Endpoint). An endpoint listens on a public port and forwards traffic to an internal port. You can map the same ports for an internal or external endpoint or use a different port for them. For example: you can have a web server configured listen to port 81 while the public endpoint mapping is port 80. The creation of a public endpoint triggers the creation of an Azure Load Balancer.

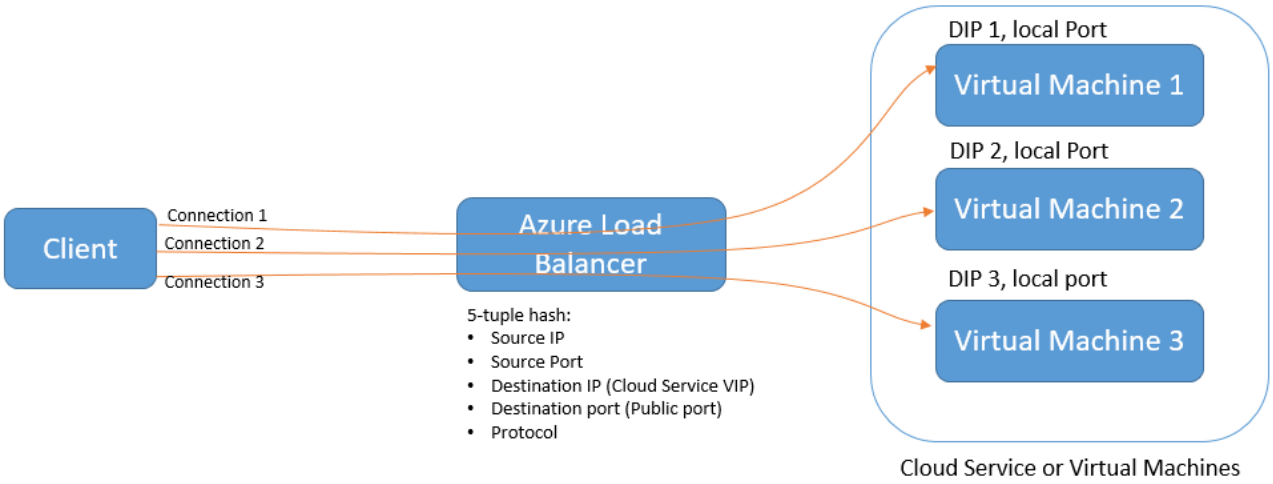
The default use and configuration of endpoints on a virtual machine that you create with the Azure Management Portal are for the Remote Desktop Protocol (RDP) and remote Windows PowerShell session traffic. These endpoints allow you to remotely administer the virtual machine over the Internet.



In the above example, a request to the public DNS entry **mycloudsvc.cloudapp.net** at port **5001** would resolve to the virtual machine **IIS-VM1** at port **3389**. Similarly, a request to **mycloudsvc.cloudapp.net:5002** would resolve to **IIS-VM2** at port **3389.**

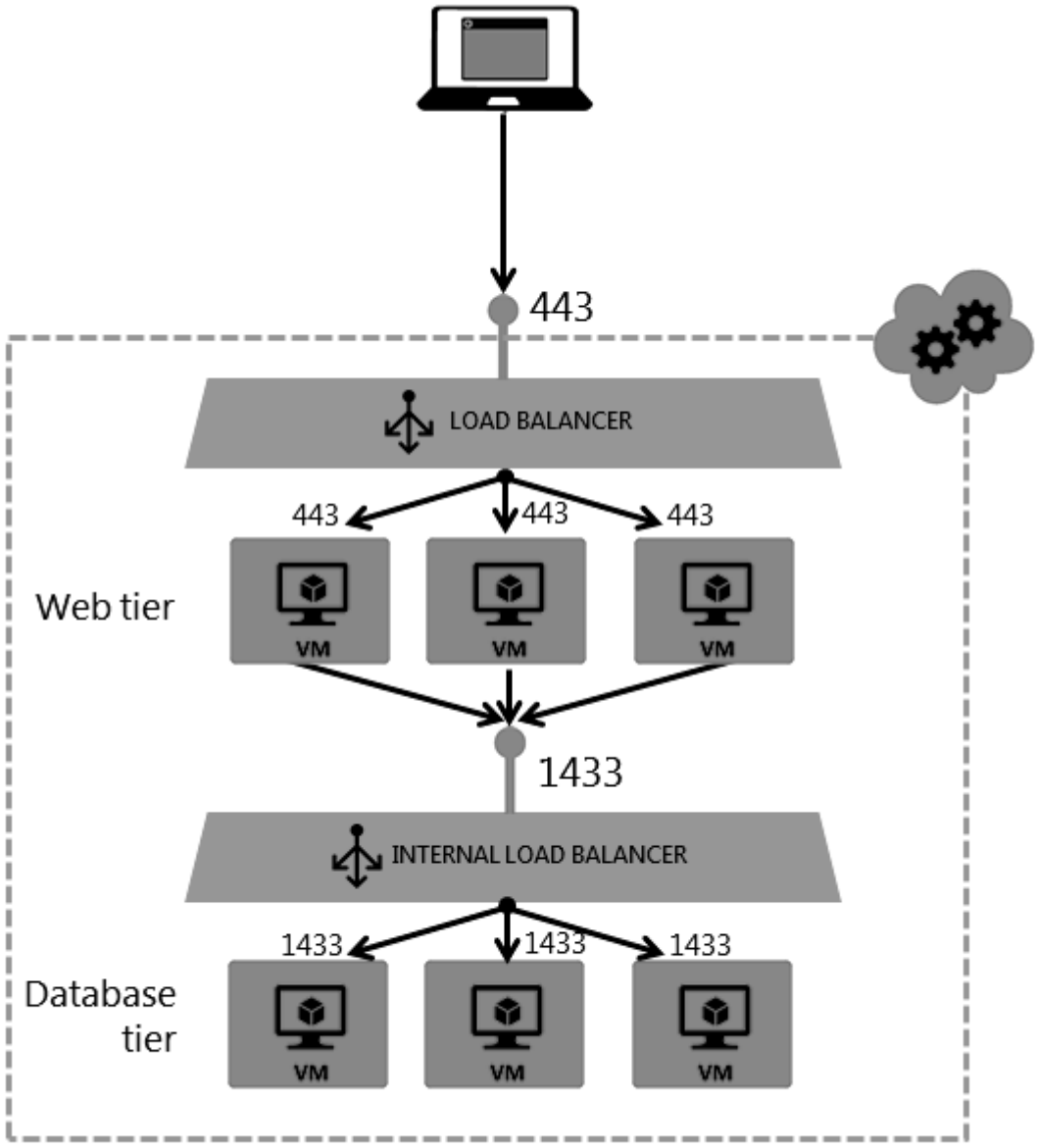
Layer-4 Load Balancer, Hash based distribution

Azure Load Balancer uses a hash based distribution algorithm. By default it uses is a 5 tuple (source IP, source port, destination IP, destination port, protocol type) hash to map traffic to available servers. It provides stickiness only within a transport session. Packets in the same TCP or UDP session will be directed to the same datacenter IP (DIP) instance behind the load balanced endpoint. When the client closes and re-opens the connection or starts a new session from the same source IP, the source port changes. This may cause the traffic to go to a different DIP endpoint.



Internal Load Balancer

Internal Load Balancer (ILB) is a security enhancement over the current Internet facing load balancer that is offered in Azure. ILB allows access only to resources inside the same cloud service or resources using a VPN connection to access the Azure infrastructure. This allows you to have the same benefits of a load balancer without requiring your resources to expose a public IP address. The infrastructure restricts the accessibility and creates a trust boundary between the load balanced virtual IP addresses to a Cloud Service or a Virtual Network and will never be exposed to a Internet endpoint directly. This enables internal Line of Business applications to run in Azure and be accessed within the cloud or from on-premises.



For more information , you can see:
Azure load balancer: https://aka.ms/edx-dev205bx-az23
Cloud Services: https://aka.ms/edx-dev205bx-az10
Windows PowerShell: https://aka.ms/edx-dev205bx-ps

Designing a "Lights-Out" Deployment Using Azure Resource Manager

Bookmark this page

"Lights-Out", Immutable Deployment

Applications are typically made up of many components – maybe a web app, database, database server, storage, and 3rd party services. You do not see these components as separate entities, instead you see them as related and interdependent parts of a single

entity. As software-defined data centers have grown, many DevOps professionals have tried to find ways to improve their deployments by following one of these two principles:

"Lights Out" Deployment

This principle is simply to not allow any administrator access to a virtual network after it has been deployed. Any configuration that needs to be done should be done using script and without temporary endpoints or VPN connections. Manual configuration and changes in an environment tend to go undocumented and are difficult to repeat. If all changes are done using scripts or other automation tools, you can ensure that you are able to recreate most scenarios.

Immutable Deployment

This principle focuses on the ability to deploy your environment multiple times without a change in behavior. If the code has not changed, you should be able to run the same script and deploy your environment many times without any side effects, errors or manual intervention. With an immutable script, it is easier to destroy and recreate environments instead of trying to figure out "what went wrong" with an errant virtual machine. Disks and physical hardware will fail, an immutable deployment will make it a lot easier to recreate your environment quickly in the event of a physical failure. An immutable deployment is also useful for replicating your production environments in other subscriptions, resource groups or for other purposes (Development, Test, Q&A).

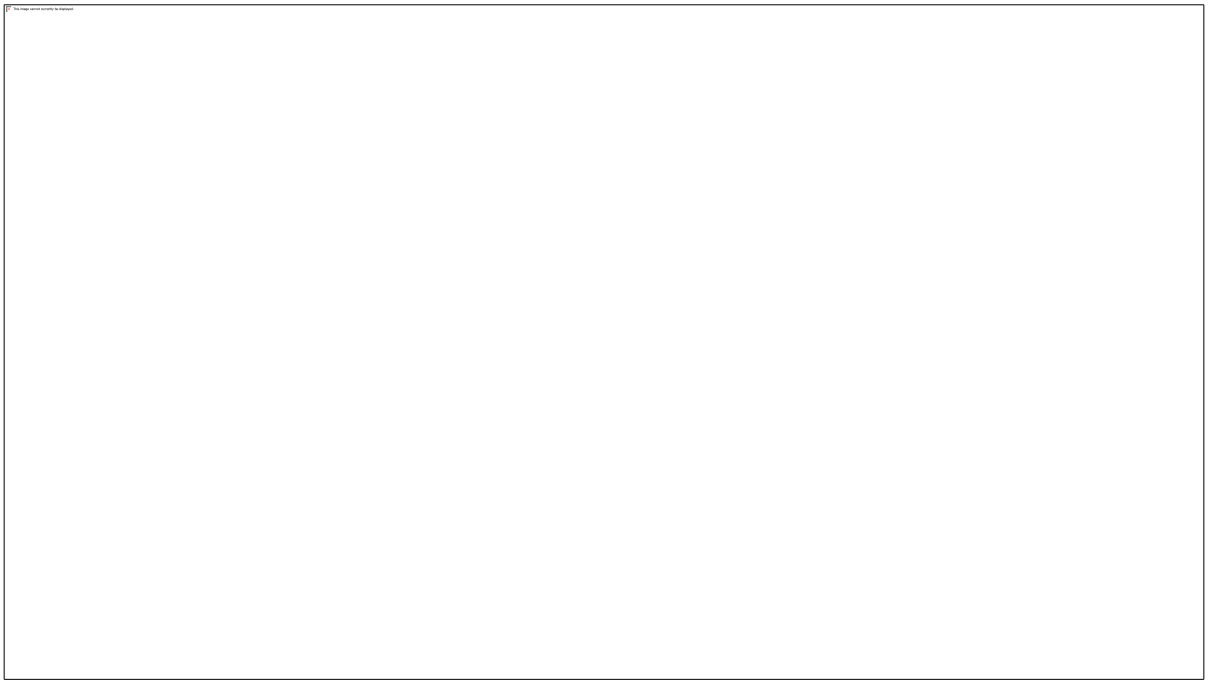
Designing a “Lights-Out” Deployment Using Azure Resource Manager

Azure Resource Manager (ARM) enables you to work with the resources in your application as a group. You can deploy, update or delete all of the resources for your application in a single, coordinated operation. You use a template for deployment and that template can work for different environments such as testing, staging and production. You can clarify billing for your organization by viewing the rolled-up costs for the entire group.

The below example shows how you can use ARM to create an immutable, "lights-out" deployment pipeline.

1. Your application developers release a new version of your source code and the ARM templates. The ARM templates detail the infrastructure necessary to host your application.
2. The continuous integration utility starts working by building the source code and creating code packages. These code packages are then uploaded to a storage account and temporary access tokens are generated for the code packages.
3. The continuous integration utility pulls the latest version of the ARM template and starts a new ARM deployment. The template has parameters to specify source code packages and the URIs of the previously uploaded packages are specified at this point.

4. ARM creates a new resource group with a unique name and unique instances of each service that your application needs to function. Once this resource group is validated, your CI tool then updates the external load balancer, pointing it to your latest resource group.
5. Whenever a new version of your code or ARM template is ready, the CI tool uses the latest bits and creates a new resource group, again updating the load balancer to point to the latest resource group.
6. Whenever an infrastructure error occurs with your code, your CI tool can easily create a new "fresh" deployment. New deployments can also be created in other regions if there is a widespread outage or created in other environments (Dev, Test, QA) to mimic production.



Your workload is **immutable** as long as the resulting resource group created by your CI tool has no change in behavior from a previous resource group created with the same code and ARM template. This is useful because you can easily destroy and recreate your environment if there are ever any edge-case issues that are related to a specific errant machine.

Your workload is **"lights-out"** because all deployment and management is done without directly accessing your resource group. The group is generated from a template, but you never have to use a VPN connection or create endpoints to configure your environment. Any manual access by an administrator can cause side effects or leave undocumented changes, so a "lights-out" deployment is preferable.

For more information , you can see:
Azure Resource Manager: https://aka.ms/edx-dev205bx-az24

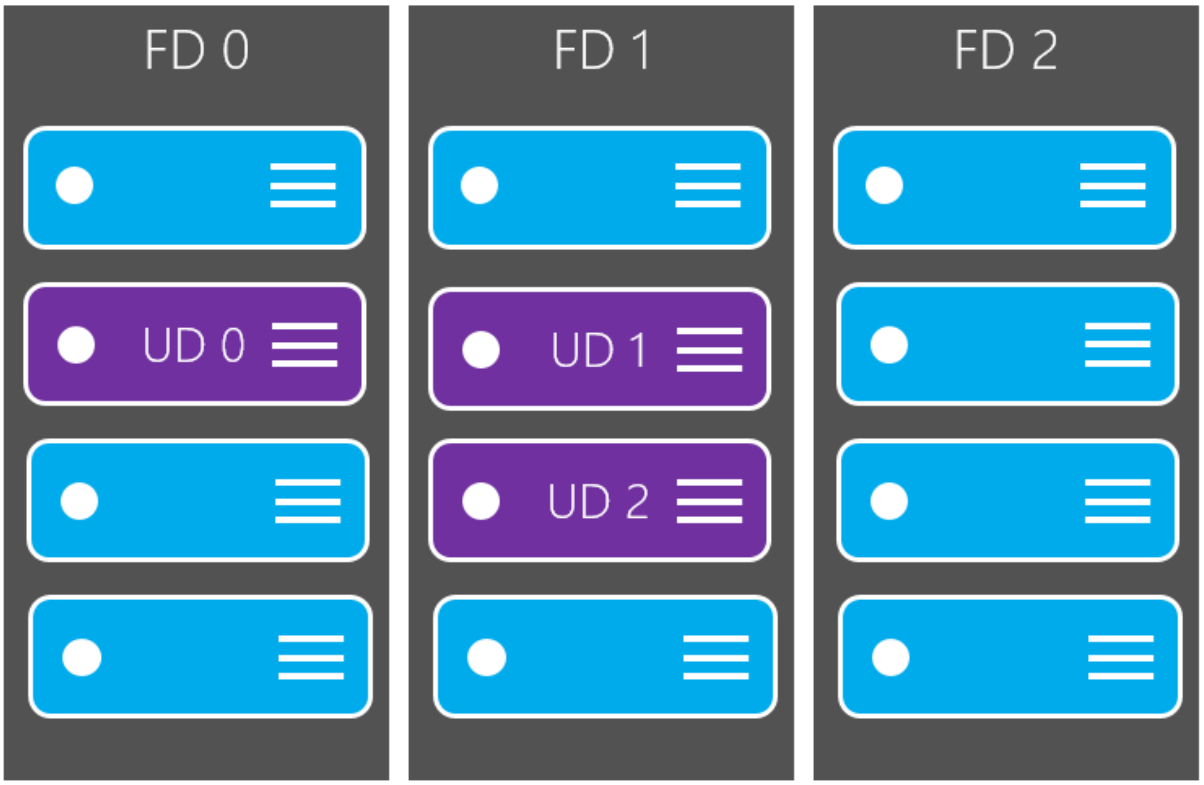
Fault/Update Domains

Bookmark this page

Fault/Update Domains

Each virtual machine in your Availability Set is assigned an Update Domain (UD) and a Fault Domain (FD) by the underlying Azure platform. For a given Availability Set, five non-user-configurable UDs are assigned to indicate groups of virtual machines and underlying physical hardware that can be rebooted at the same time. When more than five virtual machines are configured within a single Availability Set, the sixth virtual machine will be placed into the same UD as the first virtual machine, the seventh in the same UD as the second virtual machine, and so on. The order of UDs being rebooted may not proceed sequentially during planned maintenance, but only one UD will be rebooted at a time.

FDs define the group of virtual machines that share a common power source and network switch. By default, the virtual machines configured within your Availability Set are separated across two FDs. While placing your virtual machines into an Availability Set does not protect your application from operating system nor application-specific failures, it does limit the impact of potential physical hardware failures, network outages, or power interruptions.



For more information , you can see:
Virtual Machine: https://aka.ms/edx-dev205bx-az11

Domains and System Updates/Failover

Bookmark this page

Fault/Update Domains and System Updates/Failover

There are two types of Azure platform events that can affect the availability of your virtual machines: planned maintenance and unplanned maintenance.

- **Planned maintenance events** are periodic updates made by Microsoft to the underlying Azure platform to improve overall reliability, performance, and security of the platform infrastructure that your virtual machines run on. The majority of these updates are performed without any impact upon your virtual machines or cloud services. However, there are instances where these updates require a reboot of your virtual machine to apply the required updates to the platform infrastructure.
- **Unplanned maintenance events** occur when the hardware or physical infrastructure underlying your virtual machine has faulted in some way. This may include local network failures, local disk failures, or other rack level failures. When such a failure is detected, the Azure platform will automatically migrate your virtual machine from the unhealthy physical machine hosting your virtual machine to a healthy physical machine. Such events are rare, but may also cause your virtual machine to reboot.

Avoid leaving a single instance virtual machine in an Availability Set by itself. Virtual machines in this configuration do not qualify for SLA guarantee and will face downtime during Azure planned maintenance events. Furthermore, if you deploy a single VM instance within an availability set, you will receive no advanced warning or notification of platform maintenance. In this configuration, your single virtual machine instance can and will be rebooted with no advanced warning when platform maintenance occurs.

Virtual Machine Sizes and Relation to Available Features

Bookmark this page

Extended Virtual Machine Features By Size

Specific Virtual Machine sizes offer various features that are not typically available for the standard VM sizes. Below is a list of some of the extended features exclusive to particular sizes. Sizes A0-A7 represent the **Standard** tier of VMs within the Azure services.

A8/A9

While the latter designation is the same as the standard tier, A8/A9 VMs also add extra features not available to VMs in the A0-A7 tiers. These VMs have a newer generation CPU than other A-series VMs. These VMs also have access to a 40Gbit/s InfiniBand network connection with support for remote direct memory access (RDMA) features. These VMs are ideal for scenarios where you need to manage clustered

messaging, modeling or simulations. Due to the RDMA support, these VMs are also typically used in HPC scenarios.

A10/A11

These machines are identical to the A8/A9 VMs but they do not include the InfiniBand network or RDMA technology.

D-Series

D-series virtual machines feature SSD drives for the temp disk. The processors for D-service VMs are 60% faster than the standard tier. This tier is designed for applications that need better I/O performance on disk and a faster CPU.

G-Series

G-series virtual machines feature a newer generation CPU that is faster than the ones included in the standard tier or D-series. G-series VMs also tend to have exponentially more memory and SSD storage than the equivalent sizes in any other tier. This tier is designed for the most demanding application workloads.

Customer Scenario

Who is the customer?

Fabrikam Residences (http://fabrikamresidences.com) is a national real estate services group whose rapid growth was being slowed by an expensive and unresponsive datacenter infrastructure.

Fabrikam has two data centers in the United States, but it really doesn't want to be in the datacenter business. "We are a national real estate firm," says Craig Jones, Chief Information Officer for Fabrikam. "We want to make investments that support our core business, and buying and managing servers is not our core business. In fact, we have what we call a DOS strategy—'don't own stuff.' We were not an asset-intensive organization in any area but IT, where we had many underutilized assets."

What does the customer already have?

Fabrikam has about 250 servers in its datacenter in California, and another 110 in its Virginia datacenter, and hundreds of servers scattered across several branch offices throughout the United States. Fabrikam ended up overprovisioning servers each time it deployed an application to ensure that capacity would be there at peak times. This meant that millions of dollars' worth of hardware and software was sitting idle much of the time.

In addition to the primary data centers, Fabrikam also has several branch offices scattered across the United States that have connectivity to the primary data center through an MPLS based wide area network. Their partner is a Microsoft Azure ExpressRoute partner. To reduce costs, Fabrikam has made the decision to move its West coast datacenter to a colocation site in Silicon Valley and to virtualize the remainder of the servers in its branch offices and Virginia data center into the cloud. Fabrikam's current virtualization and management solution is based on System Center so a solution that integrates well with these known tools is ideal.

Customer Goal

What is the customer's goal?

Fabrikam Residences would like to eventually migrate the majority of their workloads to Azure. There are several workloads that will be migrated, but the most critical for Fabrikam is their CRM application.

The CRM application is a custom web application that runs on IIS 8 and SQL Server 2012 that stores sensitive documents for all of their customer's transactions. This application needs to perform well at peak time while mitigating the problem of overprovisioned capacity. The centralized nature of the application means that any downtime will block the activity of a significant portion of the company so the solution must be highly available. Due to the sensitive nature of this application security is key so access to the application is restricted to only authorized users from the corporate network including branch offices.

For more information , you can see:
Microsoft Azure: https://aka.ms/edx-dev205bx-az34
SQL Server 2012: https://aka.ms/edx-dev205bx-sql03

Solution Considerations

What does the customer need?

- Reduce the number of existing on-premises servers through public cloud consolidation to reduce the costs of their current overprovisioned deployments. Servers running in the Virginia data center and remote branch offices will be moved to the closest Azure region. Due the sheer amount of servers being virtualized latency and performance of the network is a big concern.
- Because of the sensitive nature of the data that Fabrikam Real Estate works with ensuring the security and privacy of their infrastructure connects through is critical.
- As part of the migration efforts the CRM application must be deployed in a way that mitigates their current problem of overprovisioning capacity when not needed but able to scale to meet

peak demand. The CRM application must be highly available and only accessible from the corporate intranet.

What things worry the customer?

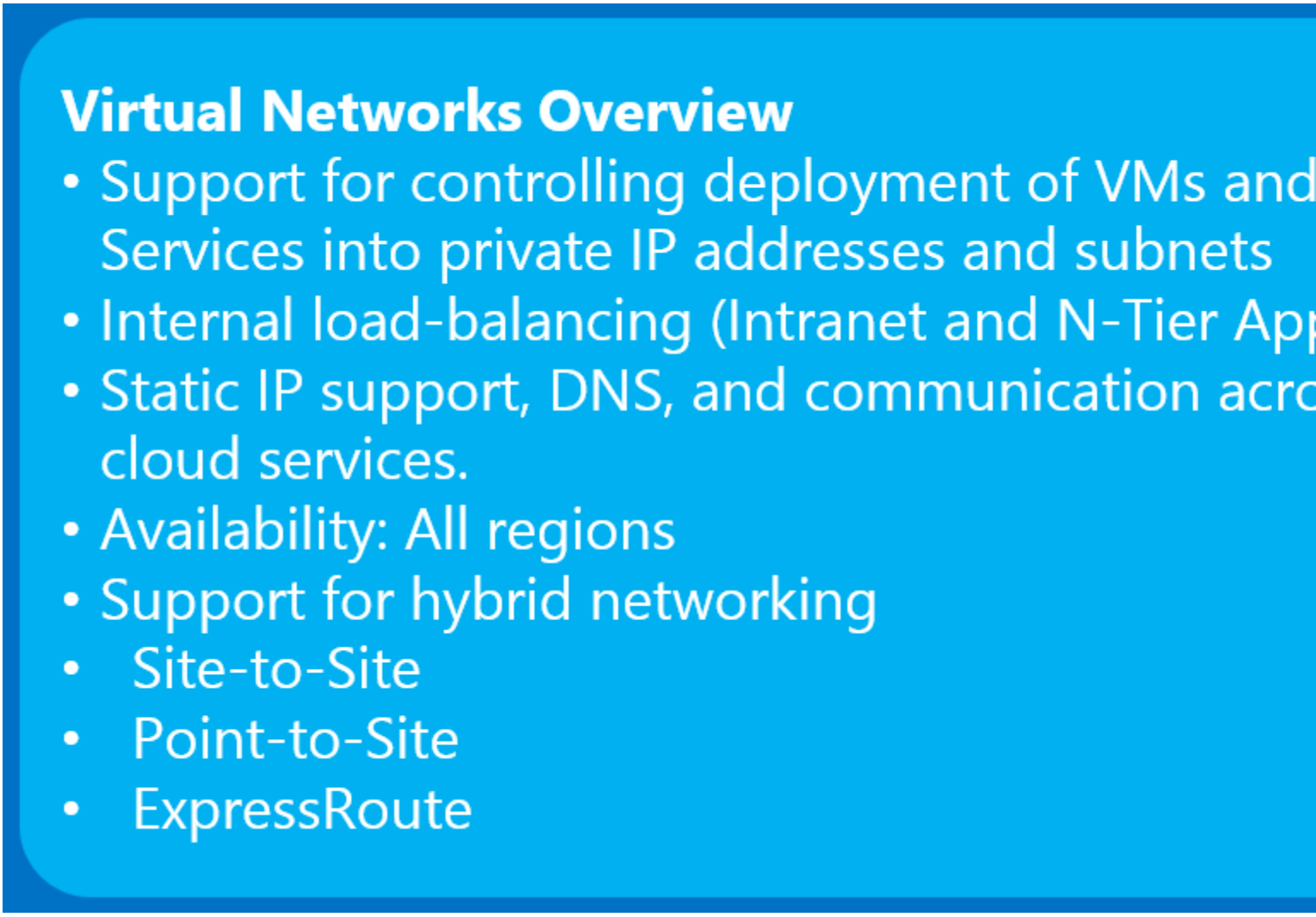
- We have a national business and we need connectivity that can accommodate connectivity from coast-to-coast.
- Our workloads are very seasonal. I do not want to pay for more resources than I need.
- The data that crosses our network is very confidential. Is Azure secure?
- I need to deploy an intranet-based solutions and I have heard that Azure requires an on-premises load-balancer for internal facing workloads.
- I have heard that the public IP address of an Azure deployment can change and break my application.
- My workloads require static IP addresses. I have heard Azure does not support this scenario.
- I have some workloads that require multiple network interfaces on my virtual machines.
- Some deployments require the segmenting of network traffic. Does Azure support this?

Common Implementations

What are some of the available Azure services?

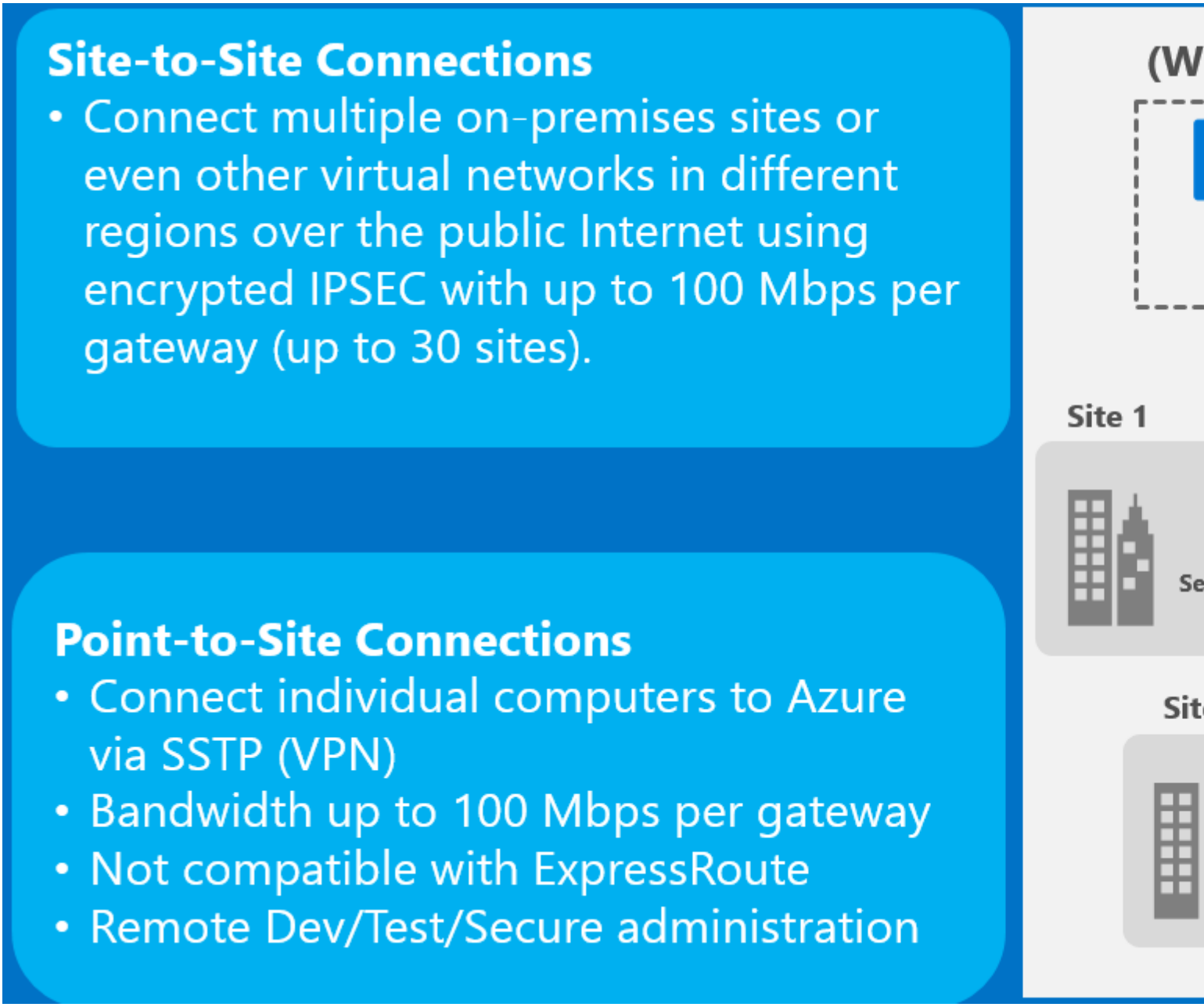
Azure Virtual Networks

Virtual network provides an isolated and secure environment to run your VMs and applications. You can bring your private IP addresses, define subnets, access control policies and much more.



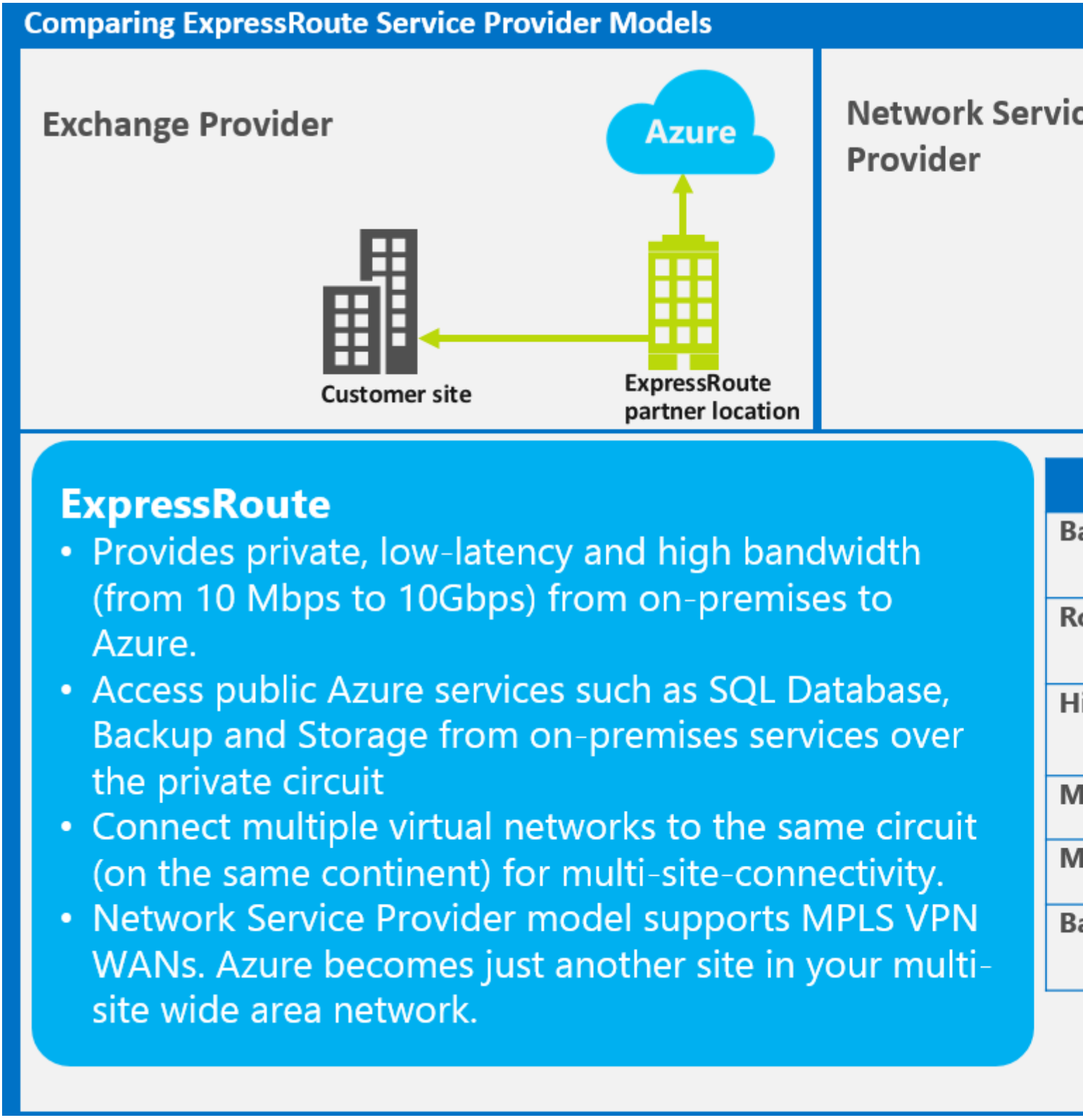
Site-to-Site and Point-to-Site Hybrid Connectivity

A site-to-site VPN allows you to create a secure connection between your on-premises site and your virtual network. To create a site-to-site connection, a VPN device that is located on your on-premises network is configured to create a secure connection with the Azure Virtual Network Gateway. Once the connection is created, resources on your local network and resources located in your virtual network can communicate directly and securely. Site-to-site connections do not require you to establish a separate connection for each client computer on your local network to access resources in the virtual network.



Azure ExpressRoute Hybrid Connectivity

Microsoft Azure ExpressRoute lets you create private connections between Microsoft datacenters and the infrastructure that’s on your premises or in a co-location environment. With ExpressRoute, you can establish connections to Microsoft cloud services, such as Microsoft Azure and Office 365, at an ExpressRoute partner co-location facility. Or, you can directly connect from your existing WAN network by using, for example, an MPLS VPN provided by a network service provider.



Additional References

Description	Links

ExpressRoute: Connecting Private and Public Clouds through WAN ProvidersExpressRoute: Connecting Private and Public Clouds through WAN Providers	http://channel9.msdn.com/Events/TechEd/NorthAmerica/2014/DCIM-B42
Extending Your Premises to Microsoft Azure with Virtual Networks and ExpressRoute	http://channel9.msdn.com/events/TechEd/NorthAmerica/2014/DCIM-B38
ExpressRoute: Connecting Private and Public Clouds through Exchange Providers	http://channel9.msdn.com/Events/TechEd/NorthAmerica/2014/DCIM-B42
Virtual Network to Virtual Network: Connecting Virtual Networks in Azure across Different Regions	http://azure.microsoft.com/blog/2014/06/17/vnet-to-vnet-connecting-virtu networks-in-azure-across-different-regions/

For more information , you can see:
Azure Virtual Networks: https://aka.ms/edx-dev205bx-az12
Microsoft Azure ExpressRoute: https://aka.ms/edx-dev205bx-az22
Microsoft cloud services: https://aka.ms/edx-dev205bx-az10
Office 365: https://aka.ms/edx-dev205bx-o365

Call to Action

Bookmark this page

Design the Solution

You will now design a potential solution for **Fabrikam Residences**. Prior to completing this exercise, please ensure that you have read all of the previous units in this case study. Particularly, make sure that you have read and you understand the **Customer Scenario** and **Solution Considerations**.

In this exercise, you will tackle a few primary tasks.

1. You will need to determine who you should present this solution to. Who is the target customer audience (stakeholder)? Who are the decision makers? You will answer these questions below in the problem sections.

2. You will need to determine what you intend to address with your solution. What customer business needs do you need to address with your solution? How will you address each business need? You will provide a brief explanation below in the problem sections.
3. You will need to create a diagram for your proposed solution. This can be done with Visio or with any image editor of your choice. The ideal output is either a high-resolution PNG file or a PDF document. You will upload this diagram in the final section below.

Once you have completed these three tasks, create a new post in the solution forum, attach an image file to the post for your diagram and your response to prompts #1 & #2.

Networking Case Study

Topic: Case Studies / Module 5: Networking Case Study

Show Discussion

Sample Solution

You are highly encouraged to try and come up with your own solution for this case study and post that solution in the forums. Once you have completed that, you can review the sample solution linked below and compare it against your own solution.

Sample Solution

Introducing High-Performance Computing (HPC)

Bookmark this page

Introducing High-Performance Computing (HPC)

Traditionally, complex processing was something saved for universities and major research firms. A combination of cost, complexity and accessibility served to keep many from pursuing potential gains for their organization from processing large and complex simulations or models. Cloud platforms have democratized hardware so much that massive computing tasks are within the reach of hobbyist developers and small to medium-sized enterprises.

High Performance Computing (HPC) typically describes the aggregation of complex processes across many different machines thereby maximizing the computing power of all of the machines. Through HPC in the cloud, one could create enough compute instances to create a model or perform a calculation and then destroy the instances immediately afterward. Advancements in the HPC field have led to improvements in the way that machines can share memory or communicate with each other in a low latency manner.

Remote Direct Memory Access

Remote Direct Memory Access, or RDMA, is a technology that provides a low-latency network connection between processing running on two servers, or virtual machines in Azure. This technology is essential for engineering simulations and other compute applications that are too large to fit in the memory of a single machine. From a developer perspective, RDMA is implemented in a way to make it seem that the machines are "sharing memory". RDMA is efficient because it copies data from the network adapter directly to memory and avoids wasting CPU cycles.

The A8 and A9 VM sizes in Azure use the InfiniBand network to provide RDMA virtualized through Hyper-V with near “bare metal” performance of less than 3 microsecond latency and greater than 3.5 Gbps bandwidth.

For more information , you can see:
Azure: https://aka.ms/edx-dev205bx-az34

HPC using Azure Virtual Machines

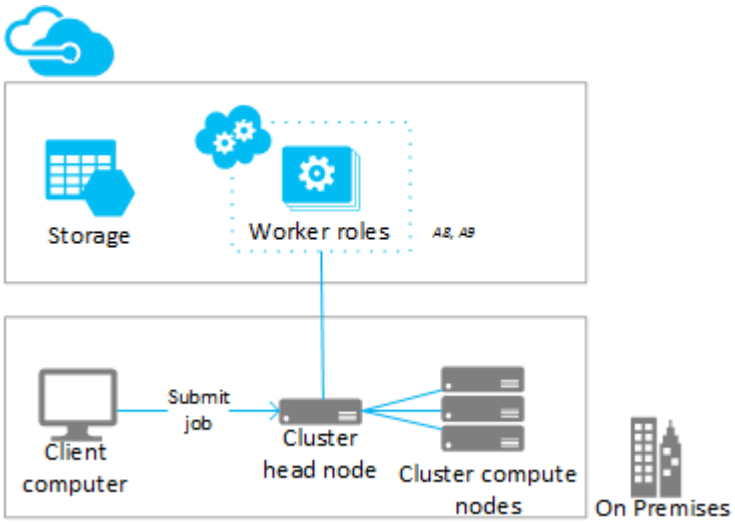
Bookmark this page

HPC Pack using Azure Virtual Machines

HPC Pack is Microsoft’s HPC cluster and job management solution for Windows. The HPC Pack can be installed on a server that functions as the "head node" and the server can be used to manage compute nodes in a HPC cluster.

HPC Pack is not required for you to use the A8, A9, A10, and A11 instances with Windows Server, but it is a recommended tool to create Windows HPC clusters in Azure. In the case of A8 and A9 instances, HPC Pack is the most efficient way to run Windows MPI applications that access the RDMA network in Azure. HPC Pack includes a runtime environment for the Microsoft implementation of the Message Passing Interface for Windows.

HPC Pack can also be used in hybrid scenarios where you want to "burst to Azure" with A8 or A9 instances to obtain more processing power.



RDMA on Linux Virtual Machines in Azure

Starting with HPC Pack 2012 R2 Update 2, HPC Pack supports several Linux distributions to run on compute nodes deployed in Azure VMs, managed by a Windows Server head node. With the latest release of HPC Pack you can deploy a Linux-based cluster that can run MPI applications that access the RDMA network in Azure. Using HPC Pack, you can create a cluster of virtual machines using either Windows or Linux that use the Intel Message Passing Library (MPI) to spread the workload of simulations and computations among compute nodes of many virtual machines.

For more information , you can see:
Windows Server: https://aka.ms/edx-dev205bx-ws
Microsoft MPI: https://aka.ms/edx-dev205bx-mpi
Azure Batch: https://aka.ms/edx-dev205bx-az25

Azure Services for HPC (Batch)

Bookmark this page

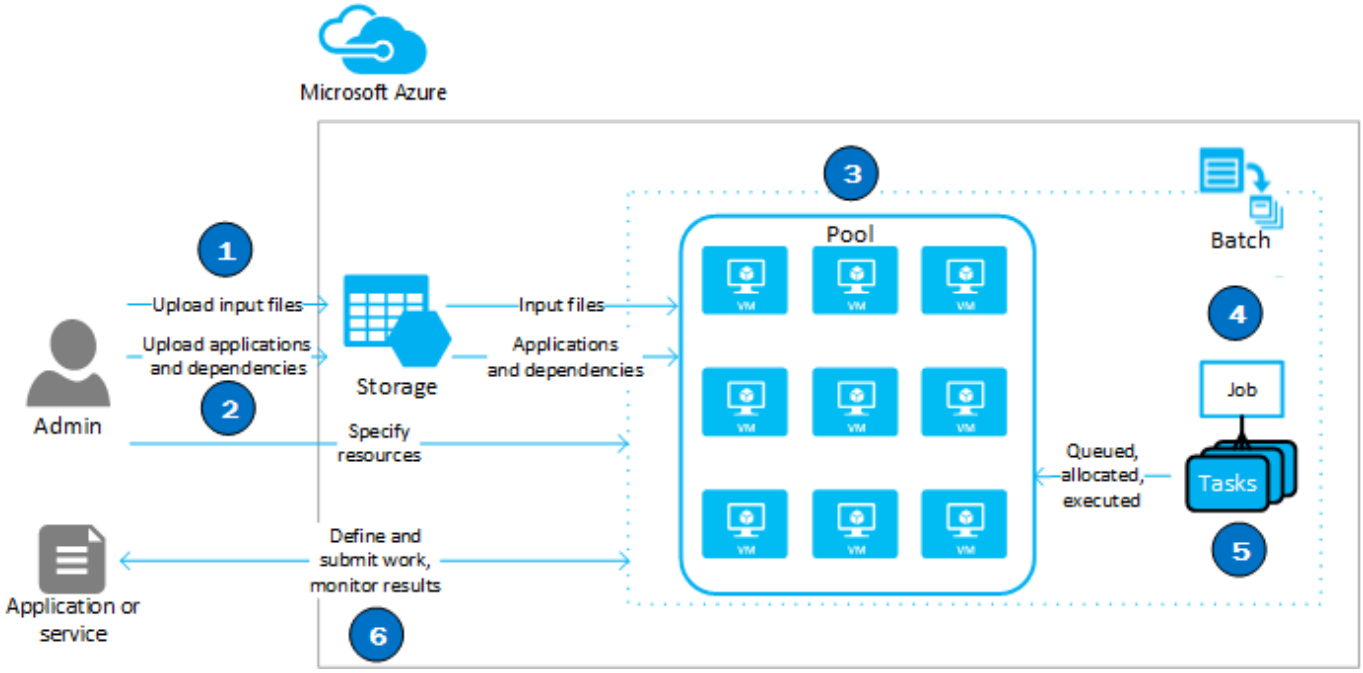
Azure Batch

Azure Batch is a service that manages Virtual Machines for large-scale parallel and high performance computing (HPC) applications. Batch is a Platform as a Service (PaaS) offering that manages the VMs necessary for your compute jobs for you instead of forcing you to manage an HPC cluster or job schedule. Batch provides autoscaling functionality and job scheduling functionality in addition to managing compute nodes.

Batch computing is a common pattern for organizations that process, transform, and analyze large amounts of data, either on a schedule or on-demand. It includes end-of-cycle processing such as a bank’s daily risk reporting or a payroll that must be done on schedule. It also includes large-scale business, science, and engineering applications that typically need the tools and resources of a compute cluster or grid. Applications include traditional HPC applications such as fluid dynamics simulations as well as specialized workloads in fields ranging from digital content creation to financial services to life sciences research. Batch works well with intrinsically parallel (sometimes called "embarrassingly parallel") applications or workloads, which lend themselves to running as parallel tasks on multiple computers.

Scaling out Parallel Workloads

The Batch API can handle scale out of an intrinsically parallel workload such as image rendering on a pool of up to thousands of compute cores. Instead of having to set up a compute cluster or write code to queue and schedule your jobs and move the necessary input and output data, you automate the scheduling of large compute jobs and scale a pool of compute VMs up and down to run them. You can write client apps or front-ends to run jobs and tasks on demand, on a schedule, or as part of a larger workflow managed by tools such as Azure Data Factory.



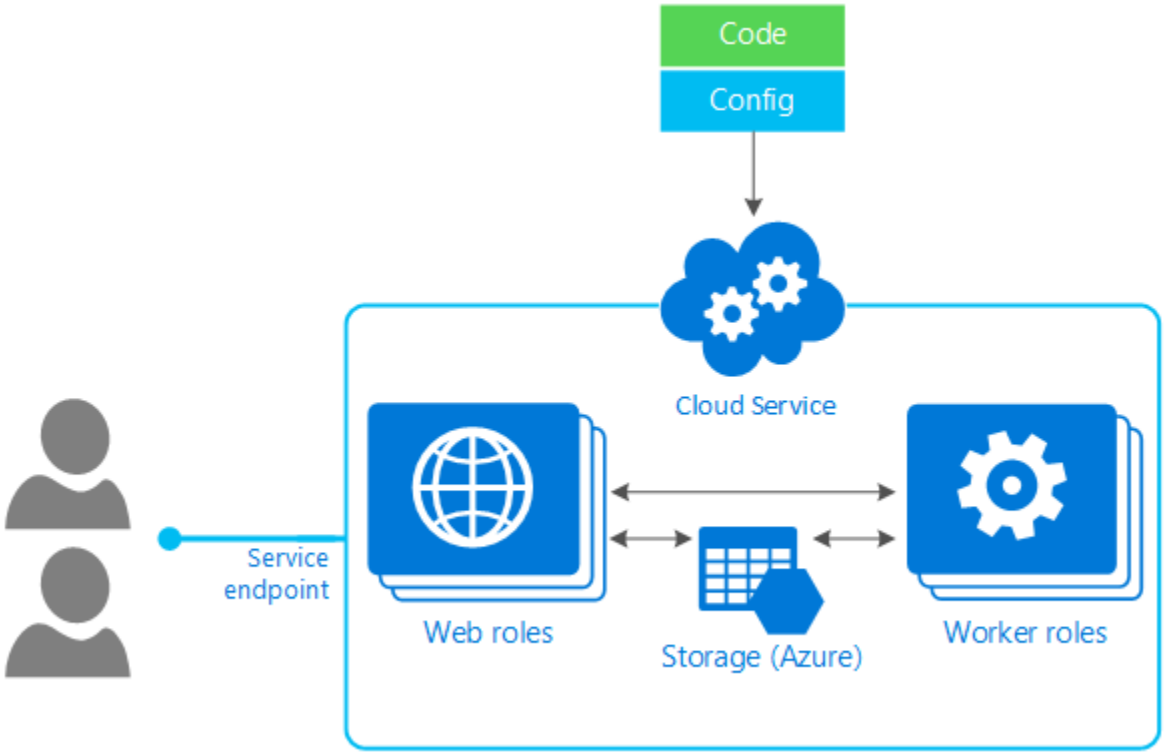
You can also use the Batch Apps API to wrap an existing application so it runs as a service on a pool of compute nodes that Batch manages in the background. The application might be one that runs today on client workstations or a compute cluster. You can develop the service to let users offload peak work to the cloud, or run their work entirely in the cloud. The Batch Apps framework handles the movement of input and output files, the splitting of jobs into tasks, job and task processing, and data persistence.

Cloud Services Worker Roles

Bookmark this page

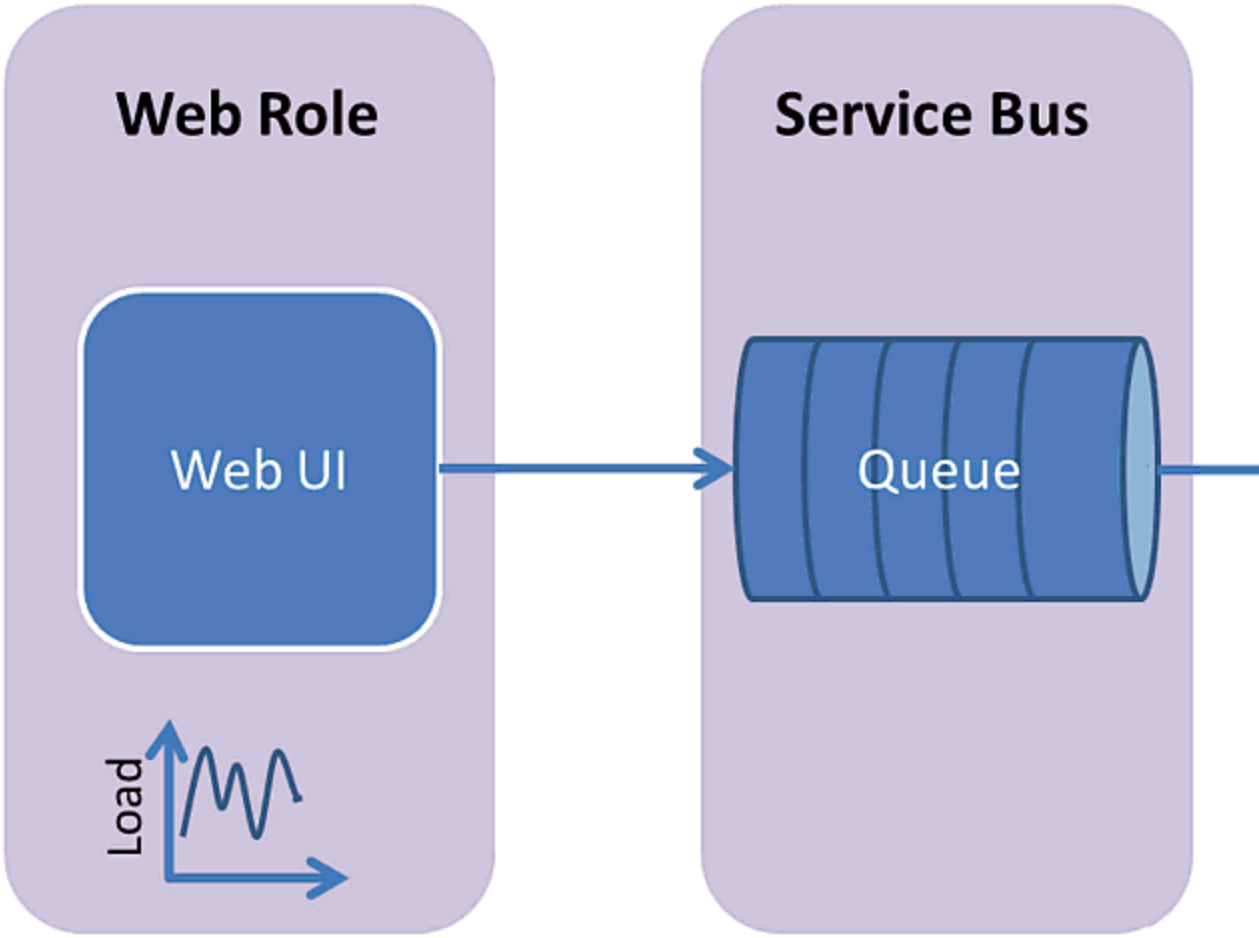
Cloud Services Worker Roles

Cloud Service Worker Roles are Windows Server Virtual Machines running a continuous process. This process could scan data, poll queues or listen for asynchronous events to assist with offloading the processing of data from an application's web-front end or another task such as an ETL workflow. Worker Roles are infinitely flexible and can be used in a variety of scenarios.



Worker Roles can be developed in .NET, PHP, Python, Java or Node.js. You can also run a variety of other languages (such as Ruby) if you can script the framework install as part of the cloud service package.

Many Worker Role implementations in Azure use an external queueing mechanism for inter-role communication. The front-end UI component may place requests in a queue for the worker role to manage later. The advantage to this technique is that the front-end can scale to accurately match the number of web requests while offloading costly, time-consuming work for a worker role to do later. The web front-end is always under a time crunch to respond to requests as quickly as possible while the back-end (worker role) can have an entirely different SLA. This logical separation gives you the flexibility to scale each component in isolation to meet your organization's SLA.



Advantages to Asynchronous Messaging/Processing

This communication mechanism has several advantages over direct messaging, namely:

- **Temporal decoupling.** With the asynchronous messaging pattern, producers and consumers need not be online at the same time. Service Bus reliably stores messages until the consuming party is ready to receive them. This allows the components of the distributed application to be disconnected, either voluntarily, for example, for maintenance, or due to a component crash, without impacting the system as a whole.

Furthermore, the consuming application may only need to come online during certain times of the day.

- **Load leveling.** In many applications, system load varies over time, while the processing time required for each unit of work is typically constant. Intermediating message producers and consumers with a queue means that the consuming application (the worker) only needs to be provisioned to accommodate average load rather than peak load. The depth of the queue will grow and contract as the incoming load varies. This directly saves money in terms of the amount of infrastructure required to service the application load.
- **Load balancing.** As load increases, more worker processes can be added to read from the queue. Each message is processed by only one of the worker processes. Furthermore, this pull-based load balancing allows for optimum utilization of the worker machines even if the worker machines differ in terms of processing power as they will pull messages at their own maximum rate. This pattern is often termed the competing consumer pattern.

For more information , you can see:
.NET: https://aka.ms/edx-dev205bx-ne
Azure: https://aka.ms/edx-dev205bx-az34

Web App WebJobs

Bookmark this page

Web App WebJobs

In Azure App Service Web Apps, you can upload and run an executable file such as:

- cmd
- bat
- exe
- .NET
- ps1
- sh
- php
- py
- js
- jar

These executable programs are referred to as WebJobs. WebJobs can be triggered manually, run on a schedule (cron) or run continuously. In .NET, a WebJobs SDK makes it easy to integrate a continuously running WebJob with a Storage Account or Service Bus instance. The SDK allows you to write methods that handle changes in the status of either of those services.

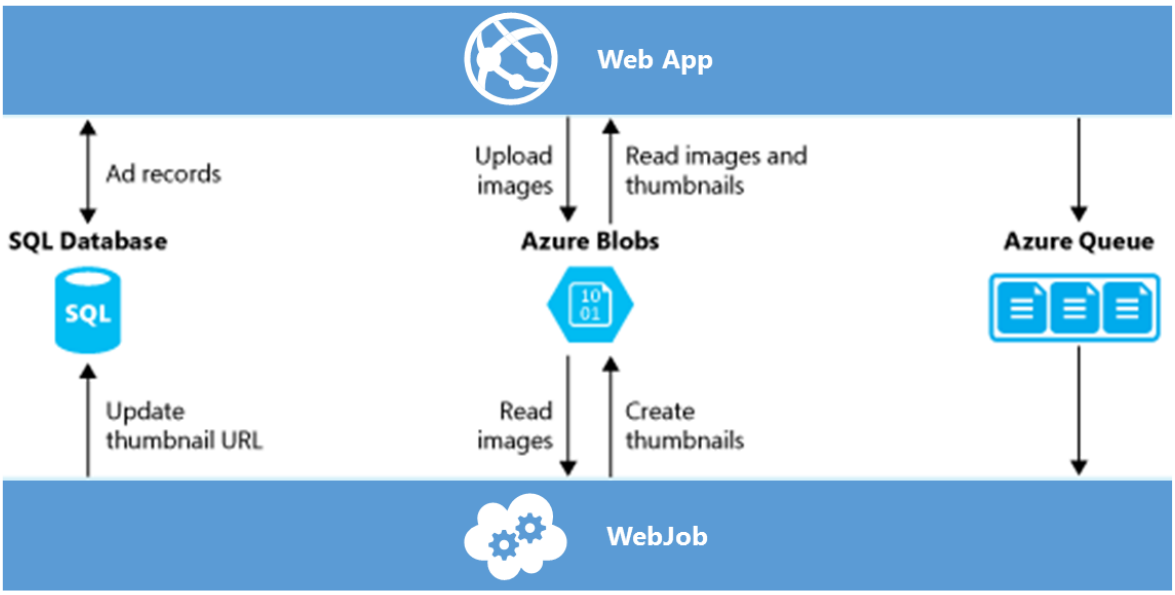
A WebJob dashboard (KUDU) is also available that allows you to view verbose log data, initiate manual runs and configure your WebJob.

Asynchronous Messaging/Processing with WebJobs

WebJobs can be used for Asynchronous messaging scenarios in the same way you would use a Cloud Service Worker Role.

WebJobs have the same restrictions as Web Apps. If you need GDI access or to install dependencies, you will need to use a worker role.

The below diagram illustrates an asynchronous messaging solution using a Web App instance (front-end) and a WebJob instance (back-end). This illustration also serves as an example of the **Competing Consumers** pattern.



For more information , you can see:
Web Apps: https://aka.ms/edx-dev205bx-az08

Implementing a Basic WebJob

Bookmark this page

Implementing a Basic WebJob

Channel 9: https://channel9.msdn.com/Shows/Azure-Friday/Azure-WebJobs-101-Basic-WebJobs-with-Jamie-Espinosa

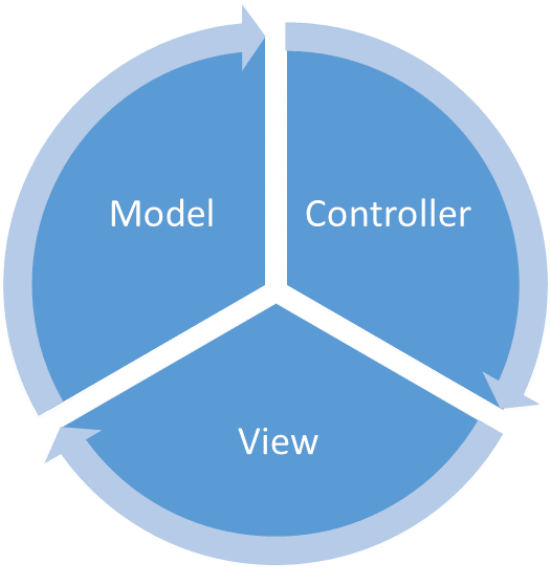
ASP.NET MVC

Bookmark this page

Model-View-Controller (MVC) Pattern

The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller. MVC is a standard design pattern that many developers are familiar with across various application platforms and frameworks. The MVC pattern includes the following primary components:

- **Models:** Model objects are the parts of the application that implement the logic for the application's data domain. Often, model objects retrieve and store model state in a database. For example, a Product object might retrieve information from a database, operate on it, and then write updated information back to a Products table in a SQL Server database. In small applications, the model is often a conceptual separation instead of a physical one. For example, if the application only reads a dataset and sends it to the view, the application does not have a physical model layer and associated classes. In that case, the dataset takes on the role of a model object.
- **Views:** Views are the components that display the application's user interface (UI). Typically, this UI is created from the model data. An example would be an edit view of a Products table that displays text boxes, drop-down lists, and check boxes based on the current state of a Product object.
- **Controllers:** Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render that displays UI. In an MVC application, the view only displays information; the controller handles and responds to user input and interaction. For example, the controller handles query-string values, and passes these values to the model, which in turn might use these values to query the database.



The MVC pattern helps you create applications that separate the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements. The pattern specifies where each kind of logic should be located in the application. The UI logic belongs in the view. Input logic belongs in the controller. Business logic belongs in the model. This separation helps you manage complexity when you build an application, because it enables you to focus on one aspect of the implementation at a time. For example, you can focus on the view without depending on the business logic. The loose coupling between the three main components of an MVC application also promotes parallel development. For example, one developer can work on the view, a second developer can work on the controller logic, and a third developer can focus on the business logic in the model.

ASP.NET MVC

ASP.NET MVC framework is a lightweight, highly testable presentation framework that (as with Web Forms-based applications) is integrated with existing ASP.NET features, such as master pages and membership-based authentication. The MVC pattern makes it easier to test applications than it is to test a Web Forms-based ASP.NET Web application. For example, in a Web Forms-based ASP.NET Web application, a single class is used both to display output and to respond to user input. Writing automated tests for Web Forms-based ASP.NET applications can be complex, because to test an individual page, you must instantiate the page class, all its child controls, and additional dependent classes in the application. Because so many classes are instantiated to run the page, it can be hard to write tests that focus exclusively on individual parts of the application. Tests for Web Forms-based ASP.NET applications can therefore be more difficult to implement than tests in an MVC application. Moreover, tests in a Web Forms-based ASP.NET application require a Web server. The MVC framework decouples the

components and makes heavy use of interfaces, which makes it possible to test individual components in isolation from the rest of the framework.

Features

The ASP.NET MVC framework provides the following features:

- Separation of application tasks (input logic, business logic, and UI logic), testability, and test-driven development (TDD). All core contracts in the MVC framework are interface-based and can be tested by using mock objects, which are simulated objects that imitate the behavior of actual objects in the application. You can unit-test the application without having to run the controllers in an ASP.NET process, which makes unit testing fast and flexible. You can use any unit-testing framework that is compatible with the .NET Framework.
- An extensible and pluggable framework. The components of the ASP.NET MVC framework are designed so that they can be easily replaced or customized. You can plug in your own view engine, URL routing policy, action-method parameter serialization, and other components. The ASP.NET MVC framework also supports the use of Dependency Injection (DI) and Inversion of Control (IOC) container models. DI enables you to inject objects into a class, instead of relying on the class to create the object itself. IOC specifies that if an object requires another object, the first objects should get the second object from an outside source such as a configuration file. This makes testing easier.
- Extensive support for ASP.NET routing, which is a powerful URL-mapping component that lets you build applications that have comprehensible and searchable URLs. URLs do not have to include file-name extensions, and are designed to support URL naming patterns that work well for search engine optimization (SEO) and representational state transfer (REST) addressing.
- Support for using the markup in existing ASP.NET page (.aspx files), user control (.ascx files), and master page (.master files) markup files as view templates. You can use existing ASP.NET features with the ASP.NET MVC framework, such as nested master pages, in-line expressions (<%= %>), declarative server controls, templates, data-binding, localization, and so on.
- Support for existing ASP.NET features. ASP.NET MVC lets you use features such as forms authentication and Windows authentication, URL authorization, membership and roles, output and data caching, session and profile state management, health monitoring, the configuration system, and the provider architecture.

ASP.NET Web API

ASP.NET Web API is an extension of ASP.NET MVC directly geared towards making RESTful web services. In MVC, a controller can handle HTTP methods such as GET, PUT, POST or DELETE. Web API removes the "View" aspect of MVC and provides a simple and extensible framework for consuming models from requests and serving models as a

response to requests. Web API controllers are based on the same controller classes as MVC making it easy to serve both HTML data (MVC) and JSON/XML data (Web API) from the same web application.

Web API also introduces advanced routing features and deeper support for customizing request/response serialization.

For more information , you can see:
ASP.NET: https://aka.ms/edx-dev205bx-asp
.NET Framework: https://aka.ms/edx-dev205bx-net
IIS: https://aka.ms/edx-dev205bx-iis

ASP.NET Ecosystem

Bookmark this page

ASP.NET Ecosystem

Moving forward, ASP.NET Web API and MVC will begin to share more of a code base. With this change there is now a de-emphasis on development with Web Forms. The ASP.NET ecosystem is changing rapidly and it is recommended to follow the posts on http://www.asp.net to keep up with where ASP.NET is going in the future.

ASP.NET Identity

Over the years, web developers have learned that users are less likely to create a new user name and password for every web application they visit. The web has become more social and users would like to leverage their existing social identities. ASP.NET Identity is a modern membership system that enables redirection-based login-ins to authentication providers such as Facebook, Twitter, Google, Microsoft and others.

ASP.NET Identity enables a user to register an account directly on your site by creating a username and password or by signing in with an existing social provider. ASP.NET Identity is based on OWIN and is portable enough to be used with MVC, Web Forms on IIS or with self-hosting for Web API or SignalR. ASP.NET Identity uses Entity Framework to manage the membership database. The membership context classes are implemented using Entity Framework Code First making it easy for you to modify the base implementation for your application's specific needs.

Other Azure Services

Bookmark this page

Azure Services

Azure is a collection of a wide variety of services for many different types of applications. In the typical enterprise application, you may not even approach 10-20% of the surface area of Azure as a platform. For exam preparation, it is typical to focus on the narrow slice of Azure that is covered in the exam objectives. In the following units, we will explore some of the other services in Azure that may interest you.

This is a shallow dive and we will not go to deep into any of these other services. If you want to learn more, links to Microsoft Virtual Academy videos are provided at the top of the page for each service.

Azure Services

For more information , you can see:
Azure: https://aka.ms/edx-dev205bx-az34

Machine Learning

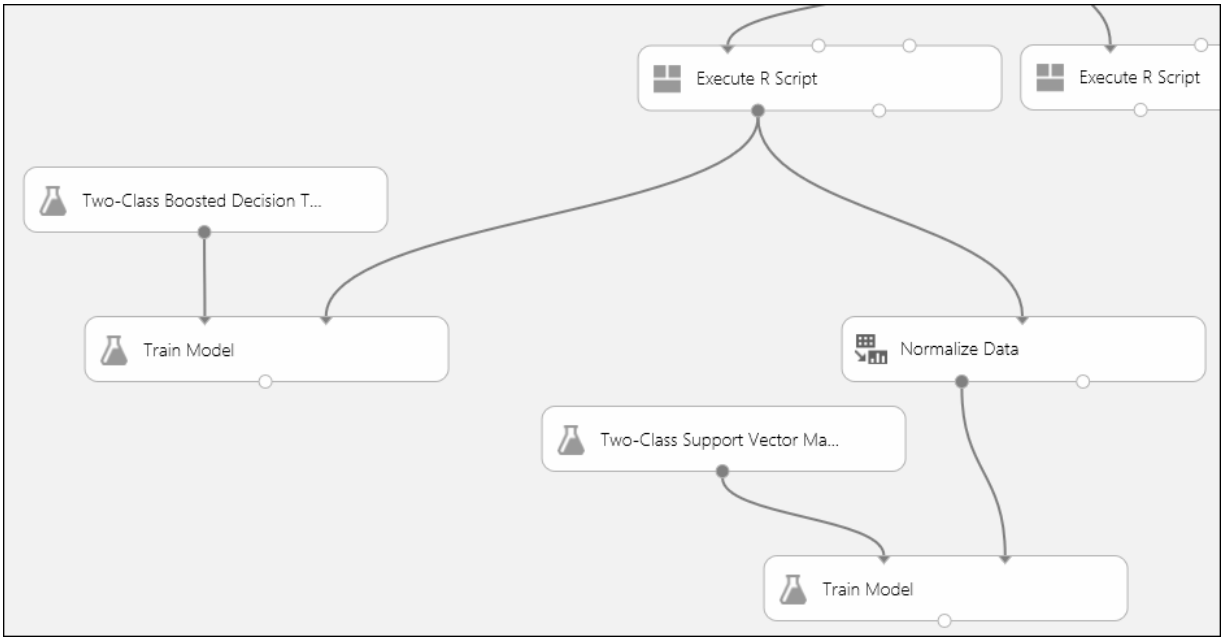
Bookmark this page

Machine Learning

https://mva.microsoft.com/en-US/training-courses/getting-started-with-microsoft-azure-machine-learning-8425

Machine learning uses computers to run predictive models that learn from existing data in order to forecast future behaviors, outcomes, and trends. These forecasts or predictions from machine learning can make apps and devices smarter. When you shop online, machine learning helps recommend other products you might like based on what you've purchased. When your credit card is swiped, machine learning compares the transaction to a database of transactions and helps the bank do fraud detection.

Azure Machine Learning is a powerful cloud-based predictive analytics service that makes it possible to quickly create and deploy predictive models as analytics solutions. Azure Machine Learning not only provides tools to model predictive analytics, but also provides a fully-managed service you can use to publish your predictive models as ready-to-consume web services. Azure Machine Learning provides tools for creating complete predictive analytics solutions in the cloud: Quickly create, test, operationalize, and manage predictive models. You do not need to buy any hardware nor manually manage virtual machines.



Predictive analytics uses various statistical techniques - in this case, machine learning - to analyze collected or current data for patterns or trends in order to forecast future events. Azure Machine Learning is a particularly powerful way to do predictive analytics: You can work from a ready-to-use library of algorithms, create models on an internet-connected PC without purchasing additional equipment or infrastructure, and deploy your predictive solution quickly. You can also find ready-to-use examples and solutions in the **Microsoft Azure Marketplace** or **Machine Learning Gallery**.

For more information , you can see:
Azure Machine Learning: https://aka.ms/edx-dev205bx-az26

HDInsight

Bookmark this page

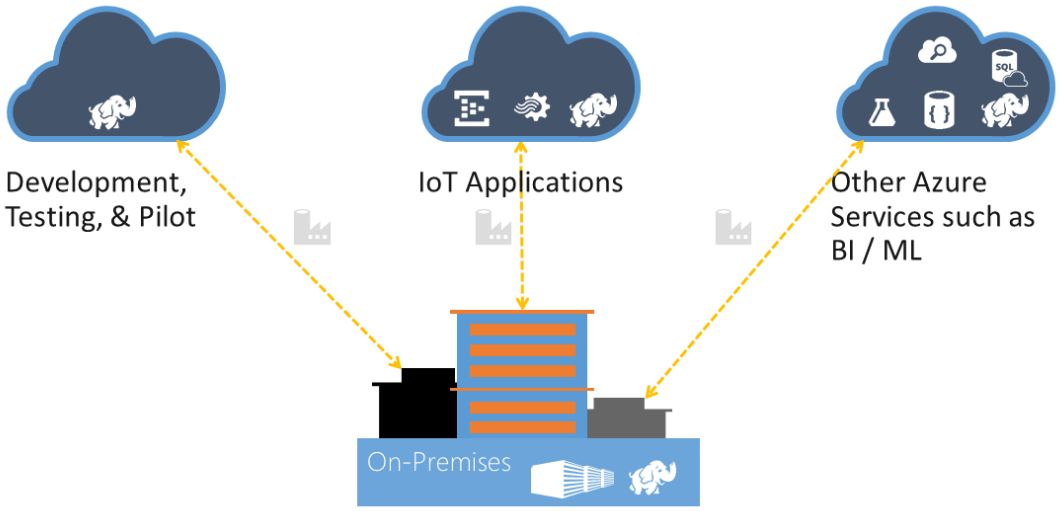
HDInsight

https://mva.microsoft.com/en-US/training-courses/big-data-analytics-with-hdinsight-hadoop-on-azure-10551

Big data refers to data being collected in ever-escalating volumes, at increasingly high velocities, and for a widening variety of unstructured formats and variable semantic contexts. Big data describes any large body of digital information, from the text in a Twitter feed, to the sensor information from industrial equipment, to information about customer browsing and purchases on an online catalog. Big data can be historical (meaning stored data) or real-time (meaning streamed directly from the source). For big data to provide actionable intelligence or insight, not only must the right questions be asked and data be relevant to the issues be collected, the data must be accessible, cleaned, analyzed, and then presented in a useful way.

Azure HDInsight deploys and provisions Apache Hadoop clusters in the cloud, providing a software framework designed to manage, analyze, and report on big data with high reliability and availability. HDInsight uses the Hortonworks Data Platform (HDP) Hadoop distribution. Hadoop often refers to the entire Hadoop ecosystem of components, which includes Storm and HBase clusters, as well as other technologies under the Hadoop umbrella. Azure HDInsight supports deploying Hadoop clusters on Windows Server or Ubuntu. Apache Spark is also currently supported in HDInsight.

The **HDInsight on Linux** functionality is currently in preview. Use this if you are familiar with Linux or Unix, are migrating from an existing Linux-based Hadoop solution, or want easy integration with Hadoop ecosystem components built for Linux.



For more information , you can see:
Azure HDInsight: https://aka.ms/edx-dev205bx-az18

Search

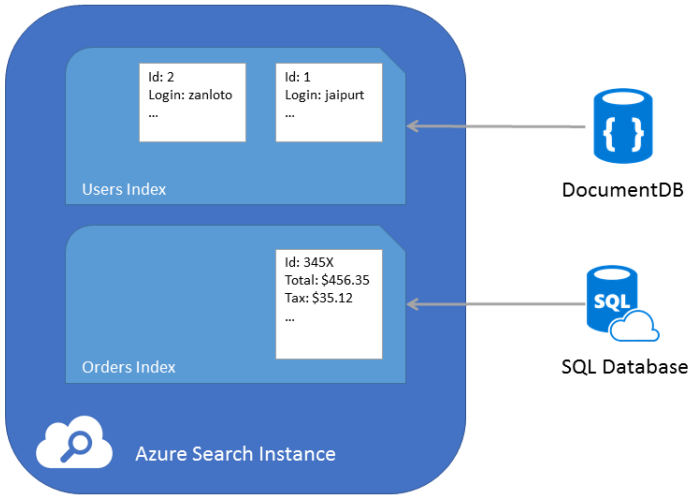
Bookmark this page

Search

https://mva.microsoft.com/en-US/training-courses/adding-microsoft-azure-search-to-your-websites-and-apps-10540

Azure Search Service is a fully managed cloud service that allows developers to build rich search applications using a .NET SDK or REST APIs. It includes full-text search scoped over your content, plus advanced search behaviors similar to those found in commercial web search engines, such as type-ahead query suggestions based on a partial term input, hit-highlighting, and faceted navigation. Natural language support is built-in, using the linguistic rules that are appropriate to the specified language.

You can scale your service based on the need for increased search or storage capacity. For example, retailers can increase capacity to meet the extra volumes associated with holiday shopping or promotional events. Azure Search is also an API-based service for developers and system integrators who know how to work with web services and HTTP. You can use existing platforms and frameworks since search only requires HTTP requests.



Azure Search is a PaaS service that delegates server and infrastructure management to Microsoft, leaving you with a ready-to-use service that you populate with search data, and then access from your application. Depending on how you configure the service, you'll use either the free service that is shared with other Azure Search subscribers, or the Standard pricing tier that offers dedicated resources used only by your service.

Standard search is scalable, with options to meet increased demands for storage or query loads. Azure Search stores your data in an index that can be searched through full text queries. The schema of these indexes can either be created in the Azure Portal, or programmatically using the client library or REST APIs. The schema can also be auto-generated from an existing data source such as SQL Database or Document DB. Once the schema is defined, you can then upload your data to the Azure Search service where it is subsequently indexed.

For more information , you can see:
cloud services: https://aka.ms/edx-dev205bx-az10
Azure Document DB: https://aka.ms/edx-dev205bx-az06

DocumentDB

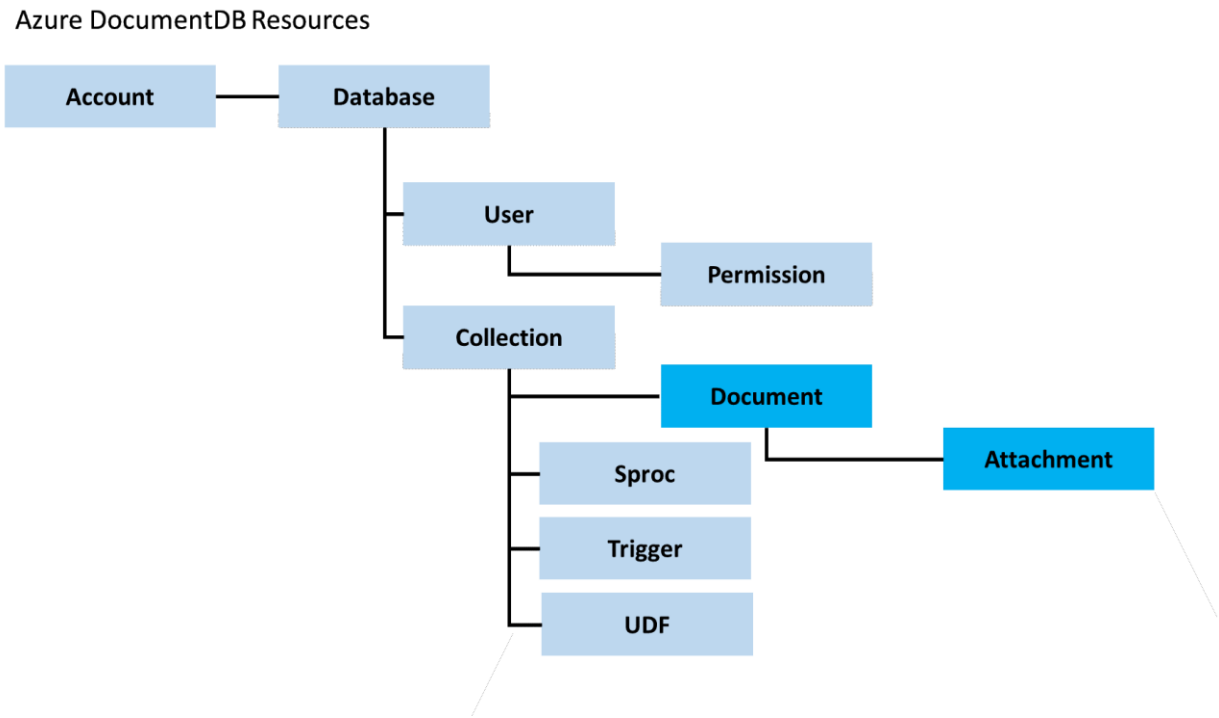
Bookmark this page

DocumentDB

https://mva.microsoft.com/en-US/training-courses/developing-solutions-with-azure-documentdb-10554

DocumentDB is a NoSQL document database service designed to deliver consistently fast reads and writes, schema flexibility, and the ability to easily scale a database up and down on demand. DocumentDB enables complex ad hoc queries using a SQL language, supports well defined consistency levels, and offers JavaScript language integrated, multi-document transaction processing using the familiar programming model of stored procedures, triggers, and UDFs. DocumentDB natively supports JSON documents enabling easy iteration of application schema. It embraces the ubiquity of JSON and JavaScript, eliminating mismatch between application defined objects and database schema. Deep integration of JavaScript also allows developers to execute application logic efficiently and directly - within the database engine in a database transaction.

DocumentDB manages data through well-defined database resources. These resources are replicated for high availability and are uniquely addressable by their logical URI. DocumentDB offers a simple HTTP based RESTful programming model for all resources. All resources within DocumentDB are modeled and stored as JSON documents. Resources are managed as items, which are JSON documents containing metadata, and as feeds which are collections of items. Sets of items are contained within their respective feeds.



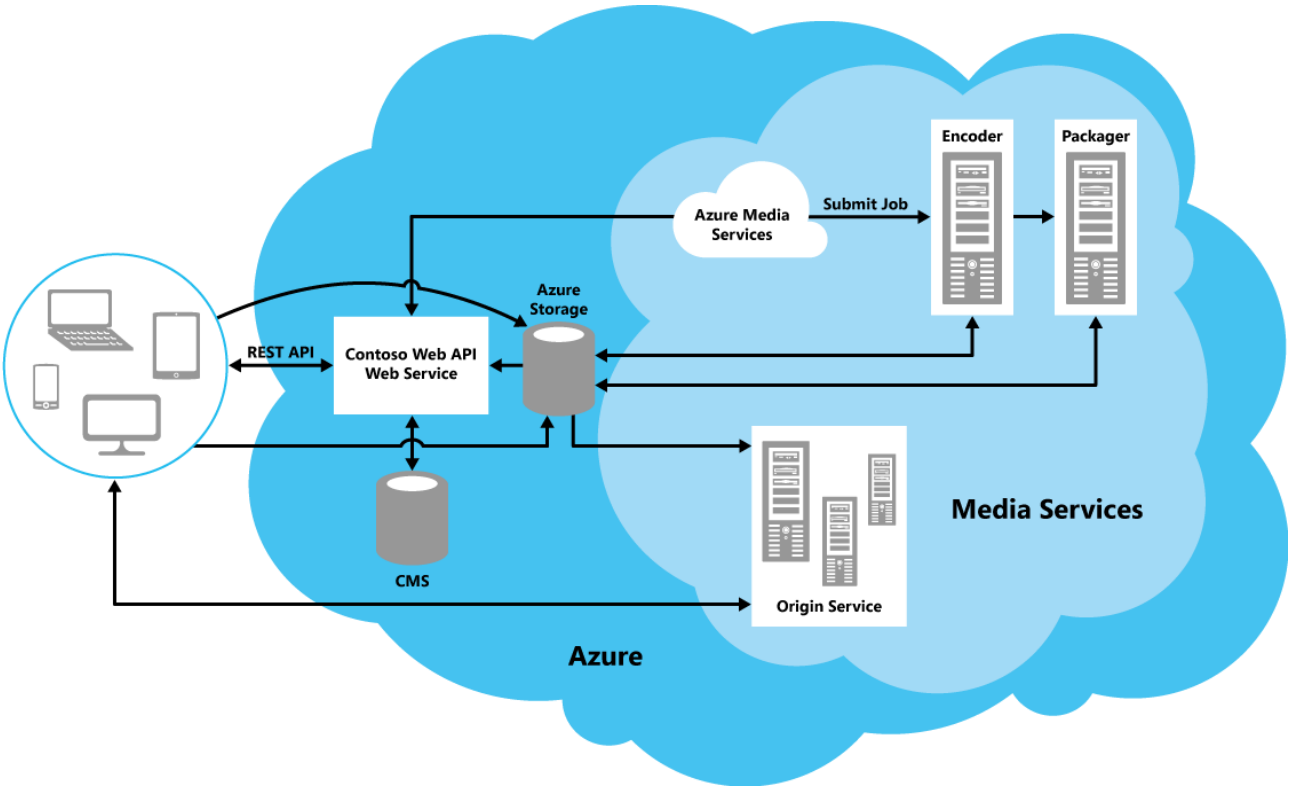
Media Services

Bookmark this page

Media Services

Microsoft Azure Media Services is an extensible cloud-based platform that enables developers to build scalable media management and delivery applications. Media Services is based on REST APIs that enable you to securely upload, store, encode and package video or audio content for both on-demand and live streaming delivery to various clients (for example, TV, PC, and mobile devices). You can build end-to-end workflows using entirely Media Services. You can also choose to use third-party components for some parts of your workflow. For example, encode using a third-party encoder. Then, upload, protect, package, deliver using Media Services.

Building and hosting a video-on-demand service is a major undertaking and can require a significant investment in hardware, management, and other infrastructure resources. Connectivity and security are also major concerns because users require timely and responsive access, and at the same time the system must maintain the integrity of the data and the privacy of users' information. To support a potentially large number of concurrent users against an ever-expanding collection of videos, you can implement video-on-demand service by using Azure Media Services. Media Services allows you to build scalable, cost effective, end-to-end media distribution solutions that can upload, encode, package, and stream media to a variety of devices and platforms. In addition, the Azure environment provides the necessary scalability, reliability, security, and performance necessary for supporting a large number of concurrent, distributed users. The below diagram illustrates an example of this:



For more information , you can see:
Microsoft Azure Media Services: https://aka.ms/edx-dev205bx-az27

Azure Active Directory

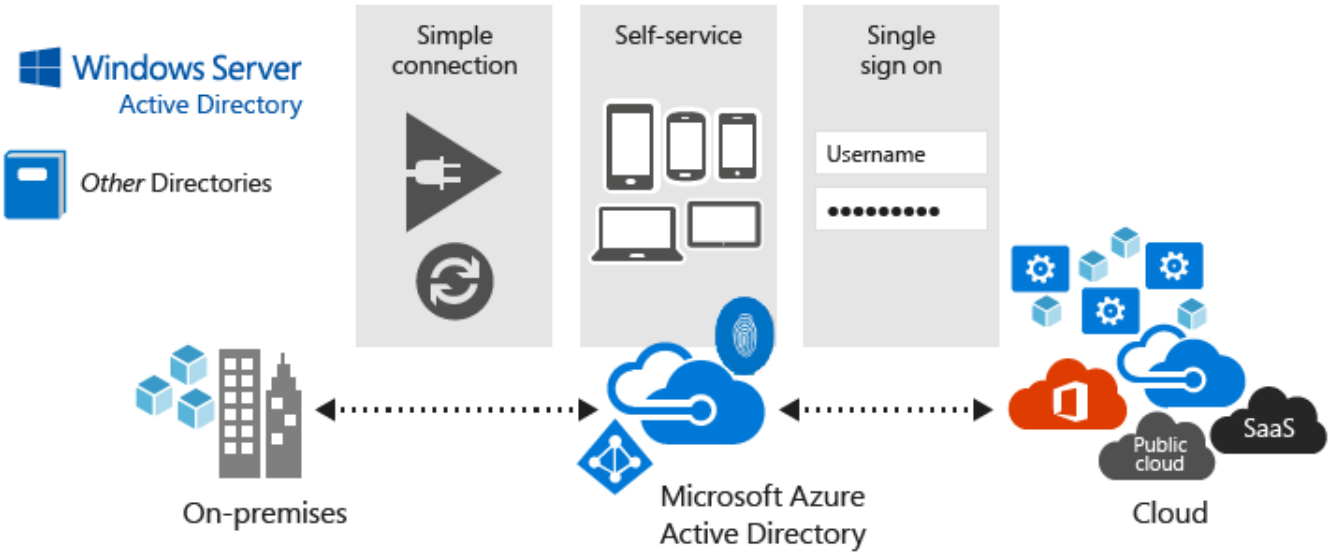
Bookmark this page

Azure Active Directory

Azure Active Directory (Azure AD) is a multi-tenant cloud based directory and identity management service. Azure AD includes a suite of identity management capabilities including multi-factor authentication, device registration, self-service password management, self-service group management, privileged account management, role based access control, application usage monitoring, auditing and security monitoring/alerting.

For IT professionals, Azure AD allows seamless integration with SaaS applications such as DropBox or Salesforce.

For Developers, Azure AD exposes itself as an identity provider using common identity standards such as OAuth 2.0.



For more information , you can see:
Azure Active Directory: https://aka.ms/edx-dev205bx-azad

Microsoft Monitoring Solutions

Bookmark this page

Monitoring Application Solutions

Many architecture diagrams, bootcamps and conference sessions/videos focus on going from concept to a fully-featured application solution or workload. While this is very important, a new set of concerns becomes your focal point once an application is deployed into your production systems. How do you know what hardware you need for your application long-term? How do you handle edge case bugs that are reported by users? How do you make decisions about chainging hardware levels to accomodate more or less users? You must measure and monitor your applications to answer these questions and this unit introduces some of the monitoring solutions available to you.

Benefits of Monitoring

Application monitoring offers a couple of primary benefits:

- Giving you more information to shape your resource utilization to your real-world application performance baseline
- Saving OPEX when diagnosing edge case issues reported by users
- Determing application-wide behavior and performance quirks

Flexibility

Monitoring your application helps you learn about your real-world resource utilization. For example, your expense tracking system's business logic may be distributed across three instances of an API App with each instance reporting an average CPU utilization of 50-65%. You may decide to decrease the amount of instances to

two. Over the next period of time, your monitoring solution may report that CPU utilization for these two instances are between 80-90%. The application's performance is still acceptable and you are able to shape your resource utilization to real-world numbers and eventually save money versus traditional over-provisioning of hardware.

Bug Tracking

What if a user reports a bug that you cannot reproduce? Users are incredible at finding the "perfect storm" of conditions to cause an edge-case bug to occur. With an application analytics/monitoring solution, you can capture the real-time debug and symbol information for the application instance where the error occurred. This dump is tremendously useful for developers as they can use the real-world environmental conditions to quickly determine root cause for the issue and solve it.

Behavior/Performance

Over time you may determine that your application "feels" like it's running slow. Monitoring solutions will help you track behavior of your application over time. For example, you may have a digital receipt image processing solution that uses the local disk. At design time you may not have considered some of the artifacts left on the disk by the processing solution. Over time, the disk will "fill up" and cause an exceptional condition. With a monitoring solution, you can view many metrics in a dashboard and determine quickly that your disk utilization does not match your planned numbers. In this scenario, a monitoring solution helped you learn about the behavior of your application. This information is also incredibly useful when you inherit an existing application and you want to learn enough about it to plan for a refactoring of the application for scalability.

Existing Infrastructure Monitoring Solutions

System Center

System Center is a management and monitoring tools for IT professionals to monitor virtualized and physical environments across datacenters, cloud compute and physical devices. System Center Operations Manager is a component of System Center that enables you to monitor services, devices, and operations for many computers in a single console. The Operations Manager provides in-depth insight into the state of the IT environment and the IT services running across different systems and workloads by using numerous views that show state, health, and performance information, as well as alerts generated for availability, performance, configuration and security situations.

Existing Application Monitoring SOlutions

Microsoft Monitoring Agent

Microsoft Monitoring Agent is a .NET Application Performance Monitoring (APM) agent in System Center with the full functionality of IntelliTrace Collector in Visual Studio for gathering full application-profiling traces. Microsoft Monitoring Agent can collect traces on demand or can be left running to monitor applications and collect traces. You can use

Microsoft Monitoring Agent together with System Center Operations Manager or as a stand-alone tool for monitoring web applications that were written on the Microsoft .NET Framework. In both cases, you can direct the agent to save application traces in an IntelliTrace log format that can be opened in Microsoft Visual Studio Ultimate. The log contains detailed information about application failures and performance issues.

Third-Party

There are many different application monitoring solutions provided by third-parties. Many of these third parties already provide tutorials for using their application monitoring solution with Azure applications:

- New Relic http://newrelic.com/
- Stackify http://stackify.com/
- App Dynamics http://www.appdynamics.com/

For more information , you can see:
API App: https://aka.ms/edx-dev205bx-az05
System Center 2016 Operations Manager: https://aka.ms/edx-dev205bx-scom
Microsoft Monitoring Agent: https://aka.ms/edx-dev205bx-mmag
.NET: https://aka.ms/edx-dev205bx-ne

Application Monitoring using Application Insights

Bookmark this page

Application Monitoring using Application Insights

Application Insights is an Application Performance Monitoring (APM) solution for applications hosted anywhere (not just Azure). Application Insights gives you the metadata that can help detect issues, solve long-term performance or behavior problems and start a workflow towards the continuous improvement of your applications.

Note: Application Insights began as an extension of Visual Studio Online (VSO). A newer version of Application Insights has been released that is no longer tightly associated with VSO. Newer Application Insight instances are deployed as isolated resources in Azure.

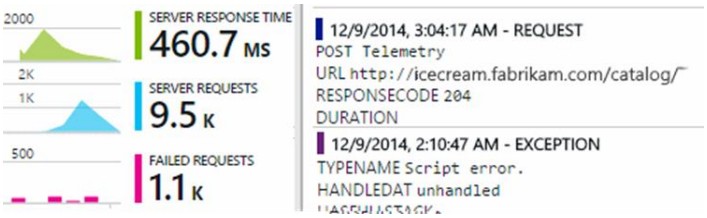
Application Insights runtime monitoring is supported for iOS, Android and Windows mobile applications. Runtime monitoring is also available for .NET web applications (ex. ASP.NET, WCF) and J2EE web applications. Application Insights can also capture analytics about your application's usage using JavaScript.

Application Insights Features

Application Insights is available in a wide variety of environments and can be used for many different purposes:

Capture Performance and Usage Analytics for an ASP.NET Application

Add Application Insights SDK to your web project. ASP.NET Telemetry is immediately available on the dashboard.



Get Additional IIS Performance Telemetry or
Diagnose Issues in a Live Website on IIS Without Redeploying

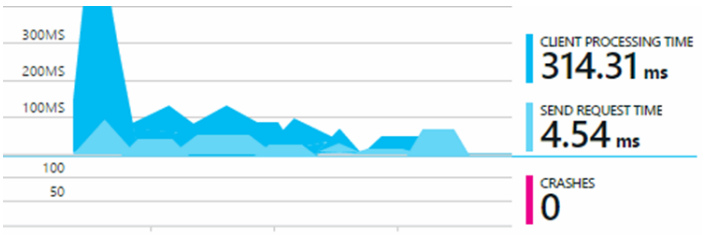
Install Status Monitor on your IIS server and view IIS telemetry on the dashboard.

Get Additional Azure Web App or VM Telemetry

Enable Insights in your Azure Web App or VM.

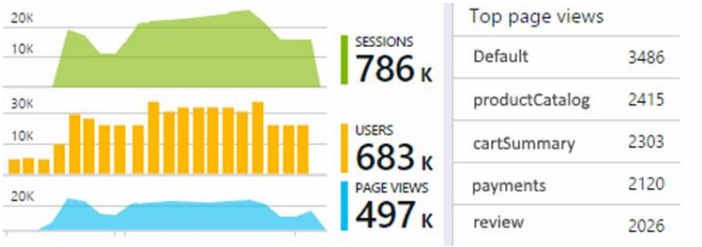
Obtain Performance and Usage Analytics for a Java Website

Add the SDK to your Java project.



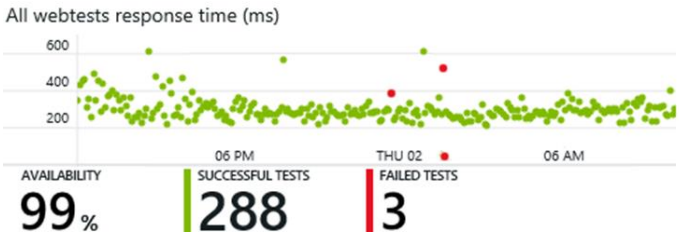
View Usage Analytics for Web Pages From Any Server (Cloud or On Premise)

Insert the Application Insights script into your web pages.

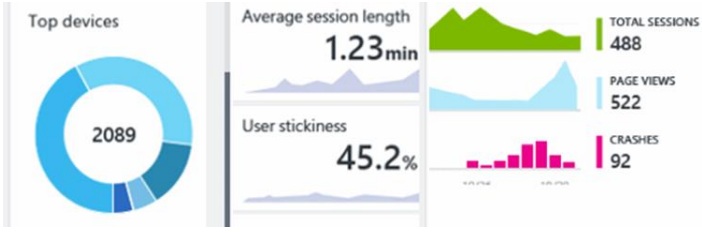


Monitor Availability of Your Website

Create web tests.



Add Application Insights to your device app project.



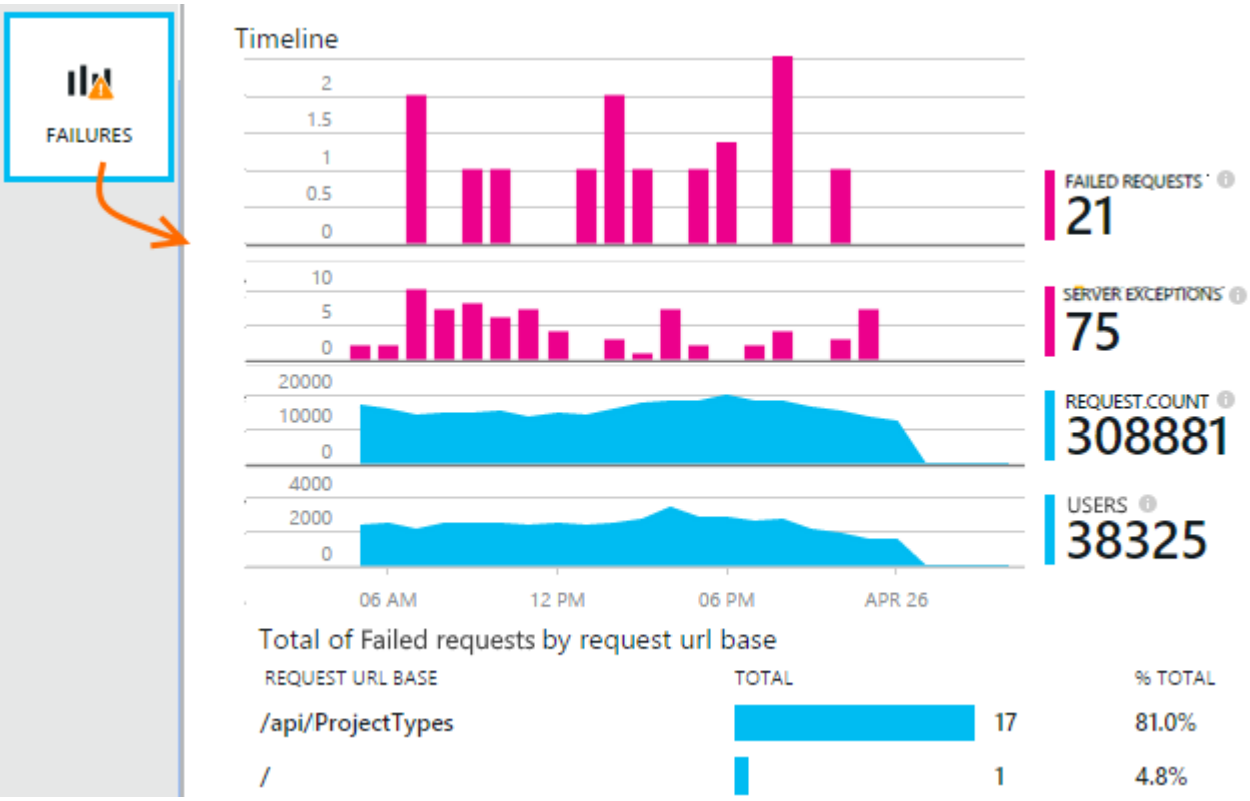
Monitoring Exceptional Scenarios using Application Insights

By monitoring your application with Application Insights, you can correlate failed requests with exceptions and other events at both the client and server, so that you can

quickly diagnose the causes. You can also insert code directly in your application to capture additional telemetry data.

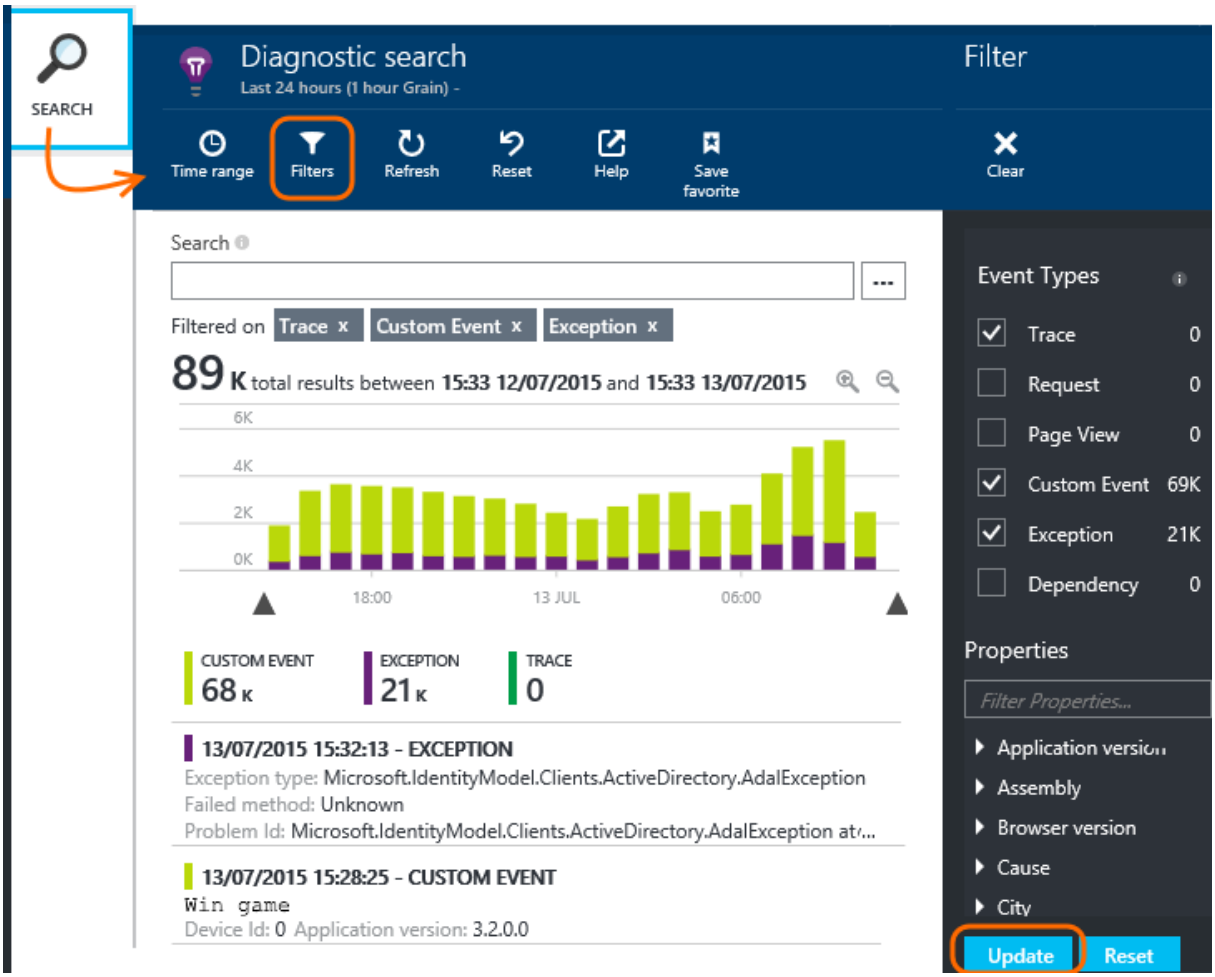
Client/Server Events

In the Application Insights blade in the portal, you can view Request failures using the Failures tile. These charts show HTTP requests and even capture additional data such as the most frequent URLs resulting in a failed request. You can drill down further into each request and see additional metadata about the individual failed requests such as request headers.



Telemetry Data

You can use the SDK-provided trace mechanisms to log custom events that you can see using the Application Insights blade in the portal. If you already have your own logging solution, adapters are available via NuGet to route the logs to Application Insights. These logs are captured under the "Custom Event" or "Trace" event types when searching events using the portal.



For more information , you can see:
Azure: https://aka.ms/edx-dev205bx-az34
Visual Studio Online (VSO): https://aka.ms/edx-dev205bx-vs02

Using the Application Insights Dashboard

Bookmark this page

Using the Application Insights Dashboard

Channel 9: https://channel9.msdn.com/Series/Application-Insights-for-Visual-Studio-Online/Customer-Usage-Analytics-with-Application-Insights

Infrastructure Monitoring using Operational Insights

Bookmark this page

Operational Insights

Operational Insights is an analysis service that enables IT administrators to gain deep insight across on-premises and cloud environments. It enables you to interact with real-time and historical machine data to rapidly develop custom insights, and provides Microsoft and community-developed patterns for analyzing data.

With Operational Insights, you can transform machine data into operational intelligence with the following features:

Icon	Feature	Description
	Capacity Planning	You can use the Capacity Planning solution in Microsoft Azure Operational Ins help you understand the capacity of your server infrastructure.
	System Update Assessment	You can use the System Updates solution in Microsoft Azure Operational Insig help you apply missing updates to servers in your infrastructure.
	Log Management	You use the Log Management solution to gather event and IIS logs for log sear throughout Operational Insights.
	Malware Assessment	You can use the Antimalware solution in Microsoft Azure Operational Insights you protect the servers in your infrastructure from malware.

	Security and Audit	You can use the Security and Audit solution to get a comprehensive view into y organization's IT security posture with built-in search queries for notable issues require your attention.
	Active Directory and SQL Assessment	You can use Assessment solutions to assess the risk and health of your server environments on a regular interval.
	Alert Management	You can use the Alert Management solution to manage alerts from servers mon Operations Manager.

Extended Features

Connectivity to System Center

You can connect Operational Insights to an existing System Center Operations Manager environment. This will allow you to use existing Operations Manager agents for data collection. If you use Microsoft Azure Operational Insights with Operations Manager, then your configuration relies on a distribution of Operations Manager agents and management groups to collect and send data to the Operational Insights service for analysis. However, if you use agents that connect directly to the web service, then you do not need Operations Manager.

Centralized Machine Data/Log Management

Operational Insights uses data from servers in your on-premises or cloud infrastructure. You can collect machine data from Azure storage when generated by Azure diagnostics. Using the data you collect from Azure storage, you can quickly search event and IIS logs for cloud services and virtual machines by enabling Azure diagnostics. You can also get additional insights from your virtual machines by installing the Microsoft Monitoring Agent. The Update Assessment, Change Tracking, and SQL Assessment solutions all

work with the Microsoft Monitoring Agent to provide deeper insights on your virtual machines.

Solutions

Solutions are a collection of logic, visualization and data acquisition rules that address key challenges for your applications. Solutions are driven by Operational Insights log search to bring you metrics pivoted around a particular problem area. They allow deeper insights to help investigate and resolve operational issues faster, collect and correlate various types of machine data and help you be proactive with activities such as capacity planning, patch status reporting and security auditing. Some example solutions include:

- **System Update Assessment**: This solution determines missing updates for monitored servers and facilitates applying those updates to each server.
- **Capacity Planning**: This solution uses performance counters on monitored servers and custom logic to determine usage patterns over time. These patterns are then analyzed to project capacity based on current consumption trends.
- **SQL Assessment**: This solution assesses the health of your SQL Server environments by scanning your systems and consolidating the information into a monthly rollup.

For more information , you can see:
Azure storage: https://aka.ms/edx-dev205bx-az01
SQL Server: https://aka.ms/edx-dev205bx-sql01

Using the Operational Insights Dashboard

Bookmark this page

Using the Operational Insights Dashboard

Channel 9: https://channel9.msdn.com/blogs/satyavel/Azure-Operational-Insights-Overview

Availability Groups and Update Domains for Patching

Bookmark this page

Azure Host OS Patching

Azure updates the operating system in the root partition, sometimes referred to as the host OS, at least every quarter to keep the environment secure for all customer applications running on the platform. The virtualization architecture used by Azure is similar to that of Windows Hyper-V. It includes a root partition that hosts the Azure root partition OS and Azure agent that’s responsible for creating child partitions to execute Azure services on the Azure guest OS.

Updating the Azure root partition OS and Azure Hypervisor requires that the virtual machines on the server being updated are shut down and subsequently restarted. **To implement the Azure SLA, Azure must not simultaneously shut down virtual machines hosting different update domains of the same Azure service role. To make sure that it does not do so, Azure determines the optimal order to update servers while honoring update domain constraints.**

Cloud Service Patching Workflow

Once Azure initiates the update of a server, it proceeds according to the following steps:

1. Virtual machines running on the server that have an Input Endpoint in their role's service model are removed from the load balancer rotation so that no new requests will come to the virtual machine and instead new requests are sent to other instances of that role as per the Azure load-balancing policies.
2. Each virtual machine hosting a Web or Worker Role receives a Stopping event, whereas VM Roles receive a standard Windows shutdown event.
3. Worker, Web, and Virtual machine roles are allowed five minutes to respond to the stopping and shutdown event before they are forcibly stopped.
4. After all guest virtual machines are stopped, the root partition OS shuts down and the server reboots.
5. The updated root partition OS starts.
6. The virtual machines hosted on the server boot and start their application code.
7. Virtual machines hosting service roles with Input Endpoints reconnect to the load balancer, enabling them to receive client requests.

Azure waits until the role code in each virtual machine on an updated server enters the Ready state before updating different servers that host the same roles, but that reside in different update domains. However, to guarantee a timely update for a secure platform, each role instance has only 15 minutes to reach the healthy Ready state before Azure stops waiting and starts updating other servers, potentially hosting roles of different update domains. The 15 minute timer starts when a role instance begins executing **Startup** tasks.

SLA Considerations

Regardless of whether you are using Virtual Machines, App Service or Cloud Services, avoid leaving a single instance of your compute instance. Compute instances in this configuration do not qualify for a SLA guarantee and will face downtime during Azure planned maintenance events.

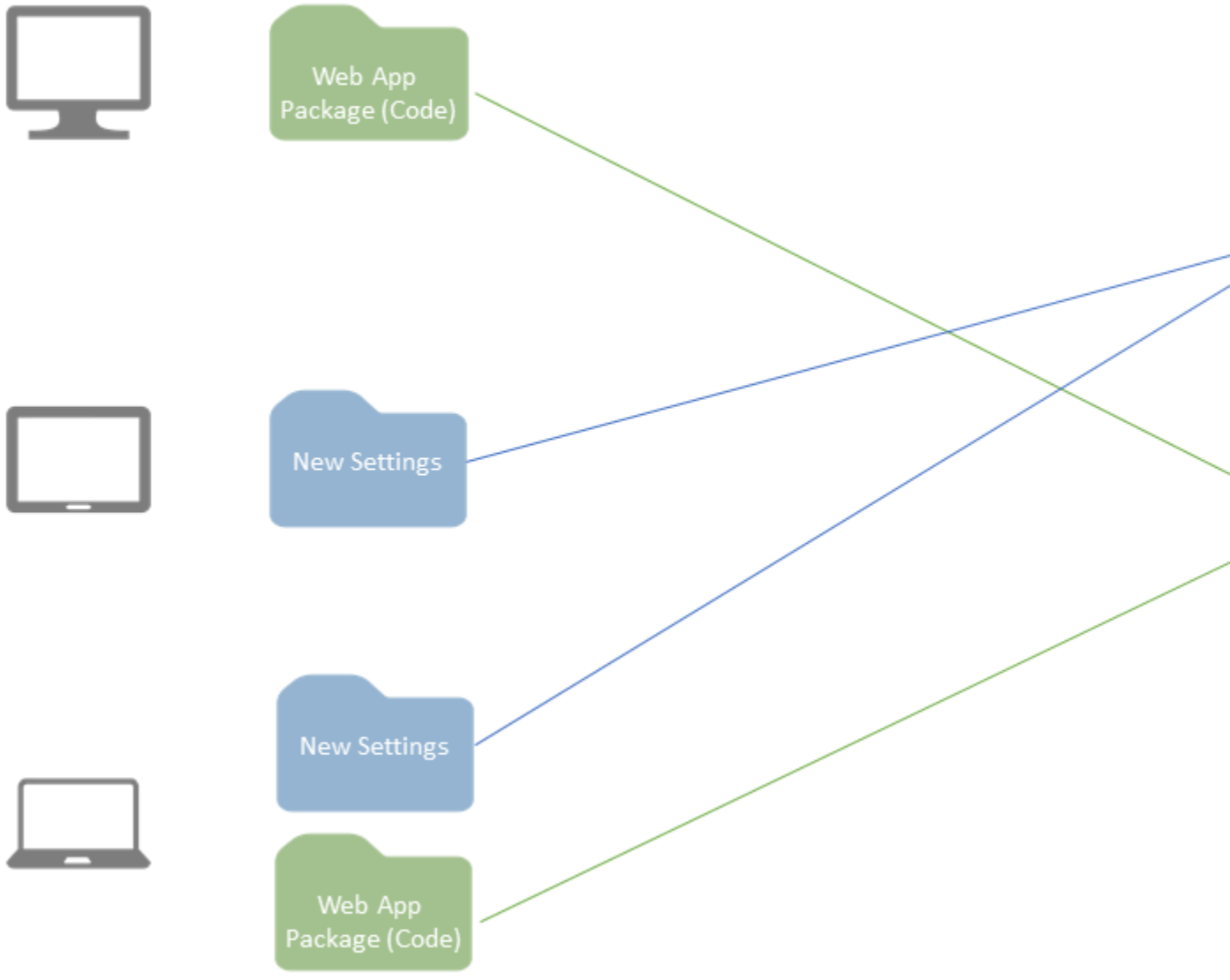
For more information , you can see:
Windows Hyper-V 2012 R2: https://aka.ms/edx-dev205bx-whv

Virtual machines: https://aka.ms/edx-dev205bx-az11
Azure: https://aka.ms/edx-dev205bx-az34
App Service: https://aka.ms/edx-dev205bx-az07
Cloud Services: https://aka.ms/edx-dev205bx-az10

Updating Compute Instances

Bookmark this page

Updating PaaS Instances



Why Separate Stores for Settings and Code?

Cloud Services in Azure have traditionally used a separate code and settings store for your applications. This separation allows you to deploy your code as many times as possible without having to know the production secrets and keys and without overwriting any existing production configuration settings. This separation also allows operation professionals in your organization to change or modify configuration settings without requiring a new code deployment.

For example, if you have an ASP.NET application, your configuration settings are stored in a Web.config file. This file may contain a connection string titled "DatabaseConnStr". The developers (even the build managers or architects) in your organization ideally would not have access to the production environment and the connection string value for the production database. The application should be developed without this information stored in a plain-text file (Web.config). When it is time to deploy, your operations team will obtain a code package from the developer team and deploy that code package. The operations team will then modify the app settings directly in the compute instance with the appropriate connection string value for the production database. If the production database changes, they can update this setting without needing to update the application code. If the application is updated, this setting is not lost. If Azure scales the compute instance, the application code package is installed in the new instance and all instances still share the same application settings. If changes were made to the Web.config directly, they could be potentially lost when instances are created or destroyed.

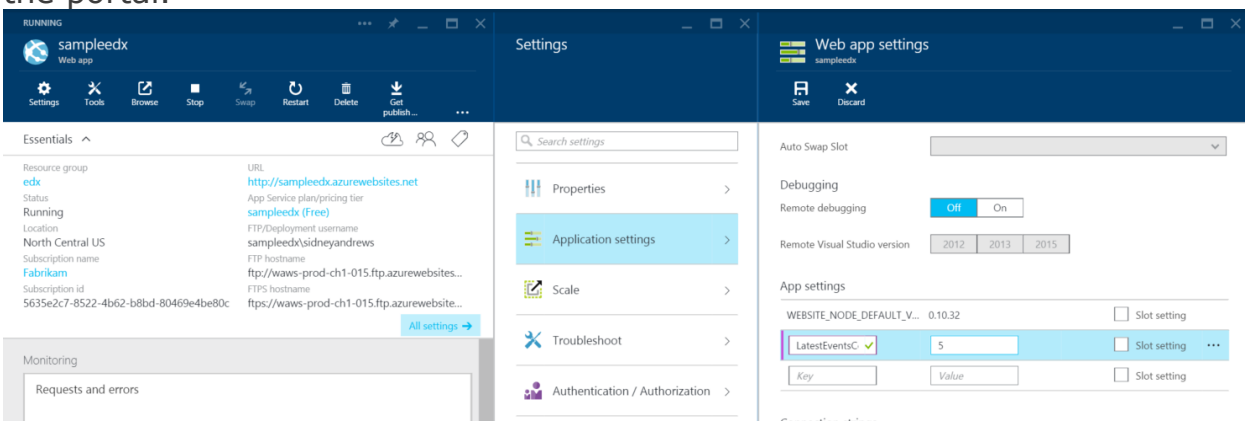
How do separate stores work?

How do seperate stores work?

Azure Web Apps uses the following pattern for managing the configuration settings for your applications:

1. Your application code is deployed to your compute instance[s]. When the code runs, it will request a configuration value. In this example, we will use a "LatestEventsCount" value that determines how many events to show on the home page. This value is set to 3 in your Web.config file.
2. The stakeholders for the web application have determined that they would like 5 events to show on the home page. Traditionally, this would require re-deployment of your web application with an updated Web.config file. You can do this by adding a new configuration setting to your application store with they key "LatestEventsCount" and the value "5". In Azure Web Apps, this is done using the "Application Settings" blade in

the portal.



3. When the web application request the "LatestEventsCount" configuration value, the request is intercepted and the value "5" that you added to the web app is provided in lieu of the value in the Web.config file.

4. Your developers can deploy a new version of your application which will overwrite the Web.config file. This new version however will still use "5" for the "LatestEventsCount" value due to interception.

5. At any time, the operations team can update the "LatestEventsCount" value for the Web App and this change will occur in all instances of your distributed web application without requiring the code to be redeployed or the Web.config file to be modified in any particular instance.

   Mobile, API App instances use a similar pattern to the one used for Web Apps. The pattern used for Cloud Services is also largely similar.

   For more information , you can see:
   ASP.NET: https://aka.ms/edx-dev205bx-asp
   Azure Web Apps: https://aka.ms/edx-dev205bx-az08
   API App: https://aka.ms/edx-dev205bx-az05

Azure's Architecure: Discussing Host Updates

Bookmark this page

Azure's Architecure: Discussing Host Updates

Channel 9: https://channel9.msdn.com/Shows/Azure-Friday/FAQ-with-Mark-Russinovich-Does-Windows-Azure-run-Windows-

PowerShell

Bookmark this page

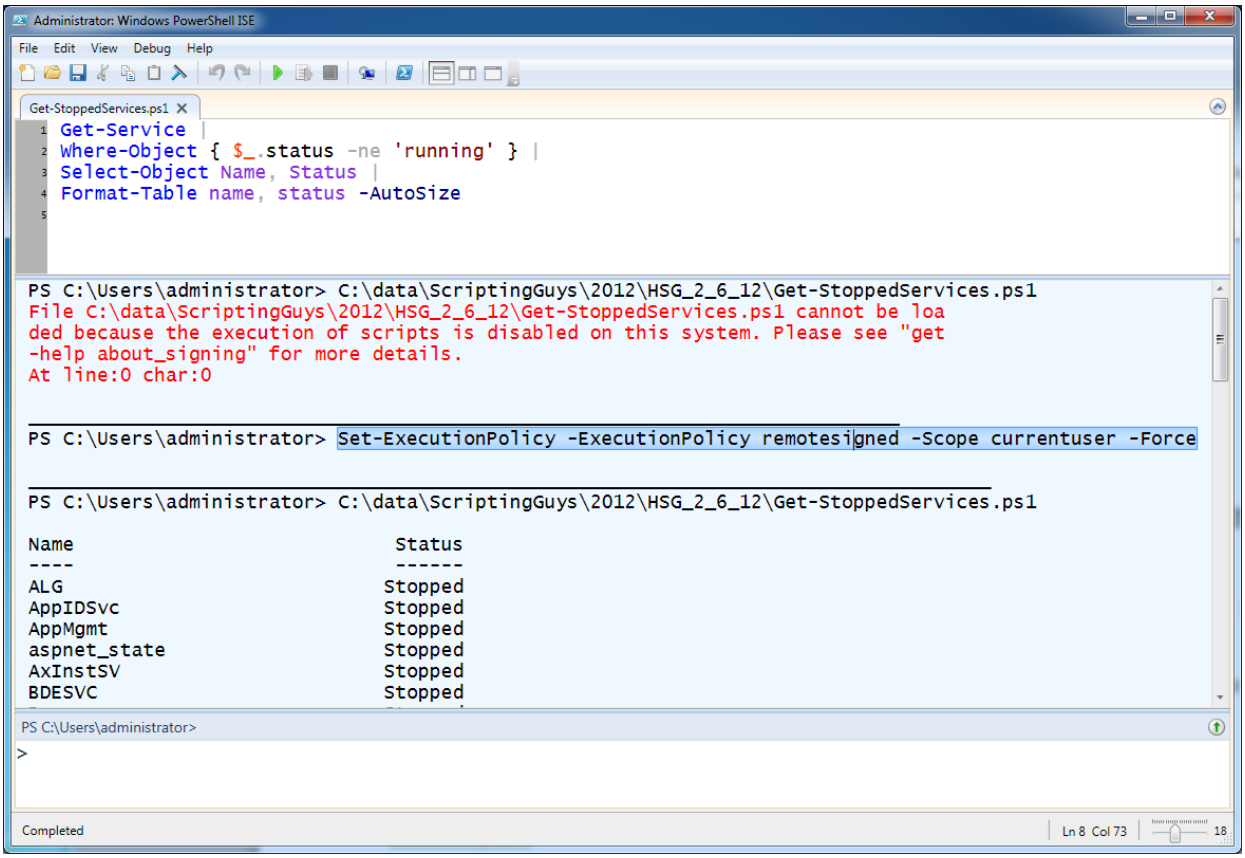
Windows PowerShell

Windows PowerShell is an automation platform and scripting language for Windows and Windows Server that allows you to simplify the management of your systems. Unlike other text-based shells, PowerShell is build on the .NET Framework, providing rich objects and a massive set of built-in functionality for taking control of your Windows environments. Through this relationship, PowerShell provides full access to managed objects in the .NET Framework along with full access to unmanaged components in COM and WMI.

PowerShell ISE

The PowerShell Integrated Scripting Environment (ISE) is a Windows application that supports enhanced usage of PowerShell. The ISE is a host application for PowerShell that provides menu items and shortcuts to perform many of the most common tasks that are normally performed line-by-line in the PowerShell console. The ISE also provides a rich explorer to drill-down into the various cmdLets available from loaded PowerShell modules. The ISE's many features include:

- A built-in editor for writing, testing, and debugging scripts
- Full IntelliSense tab completion, syntax highlighting, and context-sensitive help
- Keyboard shortcuts
- Extension through add-ons provided by both Microsoft and the community.



For more information , you can see:
Windows PowerShell: https://aka.ms/edx-dev205bx-ps
Windows Server: https://aka.ms/edx-dev205bx-ws
.NET Framework: https://aka.ms/edx-dev205bx-net

Azure PowerShell Module

Bookmark this page

PowerShell Modules

A module is a set of related Windows PowerShell functionalities that can be dynamic or persist on disk. Modules that persist on disk are referenced, loaded, and persisted as script modules, binary modules, or manifest modules. A module can include cmdlets, providers, functions, variables, aliases, and much more. The most common type of module is a script module.

Script Modules

A script module is a file (.psm1) that contains any valid Windows PowerShell code. Script developers and administrators can use this type of module to create modules whose members include functions, variables, and more. A script module can also optionally contain a Module Manifest file (.psd1) that describes the contents and attributes of the module, determines the processing order for components and define the prerequisites for the module.

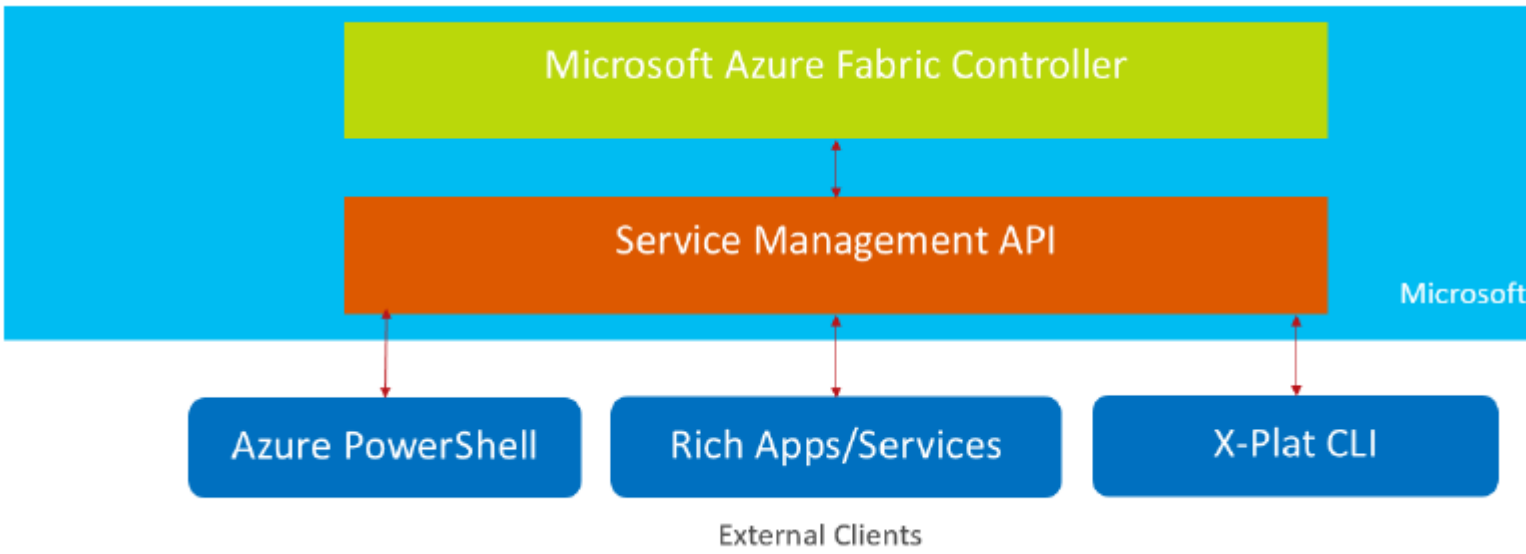
Azure PowerShell Module

The Azure PowerShell Module is a set of PowerShell cmdLets for developers and administrators to manage their Azure instances or resources. The module can be used with general Azure datacenters, the Azure China datacenter environment or the Azure Pack environment installed in your existing datacenter.

CmdLets are provided to manage your Azure subscriptions, accounts and environments. Using this module, you can specify which subscription or account is used for the context of any executed calls.

The module has two separate modes for managing your services in Azure:

- **Service Management**: These cmdLets allow you to directly manage service instances in Azure.
- **Resource Manager**: These cmdLets allow you to manage resources in Azure using the Azure Resource Manager paradigms (resources, resource groups, resource group templates)



One of the major advantages of using the Azure PowerShell modules for service management is the availability of new features. Typically new features are first introduced to the Service Management API and then are added to the Azure PowerShell modules. Finally, these features are available in the portal.

Using the Azure PowerShell Module

Creating a Redis Cache Instance using the Service Management mode

You can import a publish settings file that contains your account settings. This is useful if you intend to run your scripts in an unattended manner.

```
Import-AzurePublishSettingsFile C:\Scripts\youraccount.publishsettings
```

You then create a new cache instance

```
New-AzureManagedCache -Name FabrikamCache -Location "West Europe"
```

Once created, you can view a list of cache instances in your subscription

```
Get-AzureManagedCache
```

You can even get your individual cache instance with additional metadata in the returned object

```
Get-AzureManagedCache -Name FabrikamCache
```

You may wish to create a secondary cache access key for your web application

```
New-AzureManagedCacheAccessKey -Name FabrikamCache -KeyType Secondary
```

Finally, you can remove your cache instance

```
Remove-AzureManagedCache -Name FabrikamCache
```

Creating a Web App and SQL Database Resource Group using the Resource Manager mode

You can login using the interactive login UI with this cmdLet. You also can optionally store your credentials for an unattended script execution. If you are using Azure Automation, you must use Add-AzureAccount to set your script's context.

```
Add-AzureAccount
```

You can get a resource group template's description from the gallery. The template is a JSON file describing the resources to be created within the resource group.

```
Get-AzureResourceGroupGalleryTemplate -Identity Microsoft.WebSiteSQLDatabase.0.2.0-
preview
```

You can then create the resource group using the previously specified template.

```
New-AzureResourceGroup –Name FabrikamGroup –Location "East" Us” -
GalleryTemplateIdentityMicrosoft.WebSiteSQLDatabase.0.2.0-preview
```

Once created, you can view your resource group.

```
Get-AzureResourceGroup –ResourceGroupName FabrikamGroup
```

Finally, you can delete the resource group and all of the child resources within the group.

```
Remove-AzureResourceGroup –Name FabrikamGroup
```

For more information , you can see:
Azure PowerShell: https://aka.ms/edx-dev205bx-az28
Azure Automation: https://aka.ms/edx-dev205bx-az29
Web Apps: https://aka.ms/edx-dev205bx-az08

PowerShell Desired State Configuration (DSC)

Bookmark this page

PowerShell Desired State Configuration (DSC)

Desired State Configuration (DSC) is a management platform in PowerShell that enables deploying and managing configuration data for software services and managing the environment in which these services run. DSC provides a set of PowerShell language extensions, new PowerShell cmdLets, and resources that you can use to declaratively specify how you want your software environment to be configured. It also provides a means to maintain and manage existing configurations.

In DSC, you define a PowerShell script with a configuration element. This element is then compiled into a configuration folder containing Managed Object Format (MOF) files. Once compiled, this configuration can then be applied to a specific machine instance or set of machines. You can optionally use parameters with your template to maximize the amount of code-reuse in your templates. To use parameters, you simply specify the parameter values when compiling the configuration element.

Using PowerShell DSC

DSC introduces a new keyword called Configuration. To use DSC to configure your environment, first define a Windows PowerShell script block by using the Configuration keyword, follow it with an identifier, and then with braces ({}) to delimit the block.

```
Configuration WebServerConfig
{
   Node "WebServer"
   {
      WindowsFeature ServerRoleExample
      {
          Ensure = "Present"
          Name = "Web-Server"
      }
   }
}
```

Once the example configuration is defined, you can create the configuration block by using the configuration's name.

```
PS C:\Scripts> WebServerConfig
```

Invoking the configuration's name creates Managed Object Format (MOF) files and places them in a new directory with the same name as the configuration block. The new MOF files contain the configuration information for the target nodes.

To apply the saved configuration, run the following command.

```
PS C:\Scripts> Start-DscConfiguration -Wait -Verbose -Path .\WebServerConfig
```

Configuration Management using PowerShell

Bookmark this page

Configuration Management using PowerShell

PowerShell DSC is beyond the scope of this course. If you are interested in learning more, you can view these Microsoft Virtual Academy sessions with Jeffrey Snover, Microsoft Distinguished Engineer and inventor of PowerShell, and Jason Helmick, Windows PowerShell MVP:

Getting Started with PowerShell Desired State Configuration (DSC)

[Advanced PowerShell Desired State Configuration (DSC) and Custom Resources]

PowerShell DSC and Linux

Bookmark this page

PowerShell DSC and Linux

The PowerShell DSC platform can be used to manage the configuration of both Windows and Linux workloads. In order to use PowerShell DSC with Linux, your machine must have the Open Management Infrastructure (OMI) installed. Once OMI is installed, you can install the DSC resources directly on your machine. The configuration elements are similar to the ones used in DSC for Windows, but there are individual modules purpose-built for Linux machines.

The following Linux operating system versions are supported for DSC for Linux. Installation packages are available for various distros, OpenSSL versions and processor architectures:

- CentOS 5, 6, and 7
- Debian GNU/Linux 6 and 7
- Oracle Linux 5, 6 and 7
- Red Hat Enterprise Linux Server 5, 6 and 7
- SUSE Linux Enterprise Server 10, 11 and 12
- Ubuntu Server 12.04 LTS and 14.04 LTS

The resources used to implement PowerShell DSC on Linux are currently available on GitHub:

https://github.com/MSFTOSSMgmt/WPSDSCLinux

Using PowerShell DSC in Linux

To use DSC to configure your environment, first define a Windows PowerShell script block by using the Configuration keyword, follow it with an identifier, and then with braces ({}) to delimit the block.

```
Configuration HelloWorldFileConfig
{
  Import-DSCResource -Module nx

  Node  "linuxhost.fabrikam.net"
  {
    nxFile ExampleFile {

      DestinationPath = "/tmp/example"
      Contents = "hello world `n"
      Ensure = "Present"
      Type = "File"
    }
  }
}
```

Once the example configuration is defined, you can create the configuration block by using the configuration's name.

```
HelloWorldFileConfig -OutputPath:"C:\dscpkg"
```

To apply the saved configuration, run the following command.

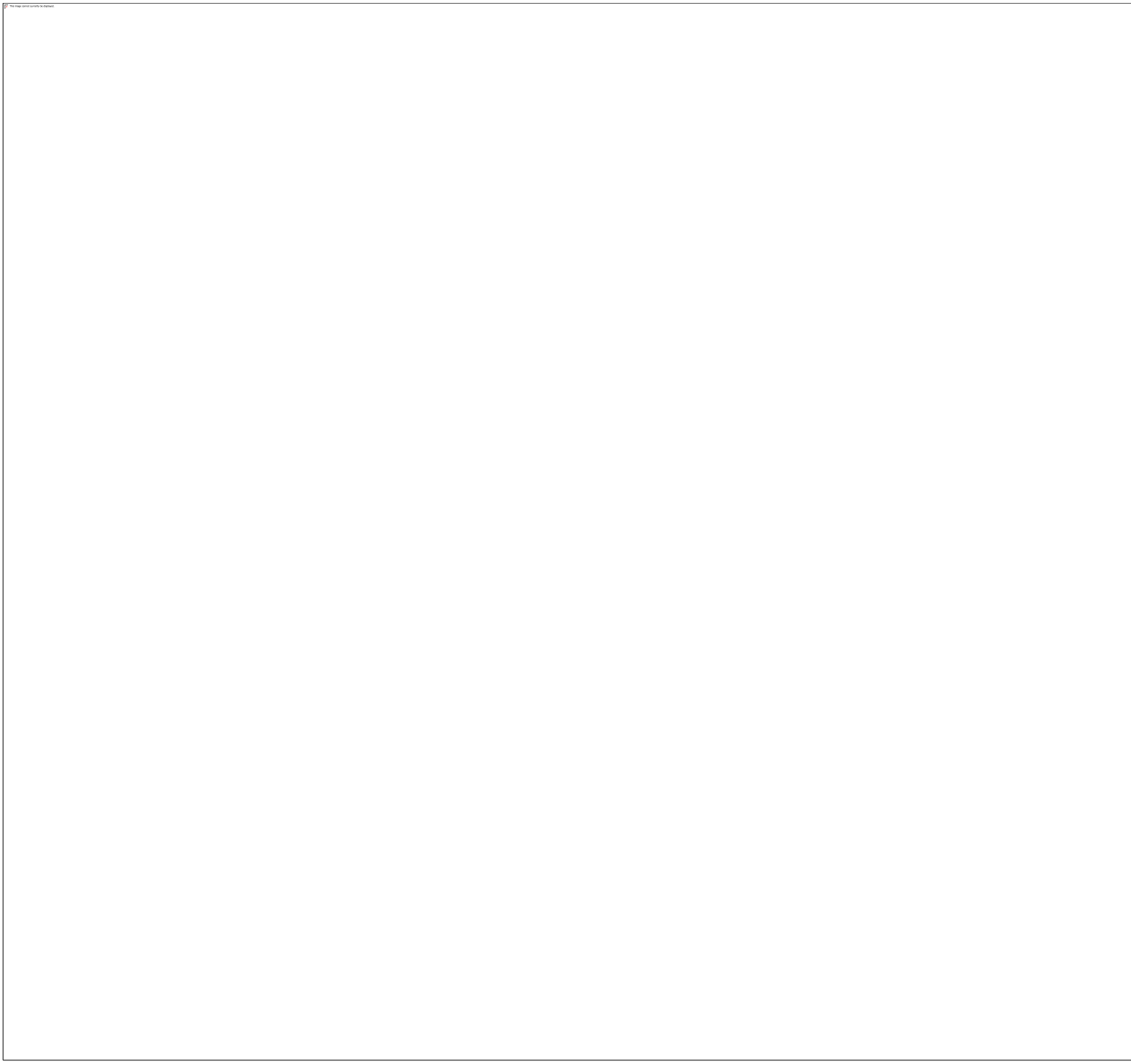
```
Start-DSCConfiguration -Path:"C:\dscpkg" -wait -verbose
```

You can also alternatively deploy a configuration to a remote Linux machine using the `New-CimSession` cmdLet.

PowerShell DSC and Azure Resource Manager

Bookmark this page

PowerShell DSC and Azure Resource Manager



Azure Resource Manager DSC Extension

One of the resources that can be created using Azure Resource Manager is the DSC extension. This extensions applies the DSC configuration to the VM created in the ARM template. The workflow the the extension is below:

1. The ARM template (JSON) is deployed to the Azure Resource Manager.
2. The manager creates a VHD as specified in the template. Other necessary resources are also created.

3. The manager creates the VM using the VHD created earlier in the template.
4. The manager encounters the DSC Script extension and then downloads a package (compressed folder) containing the DSC Script.
5. The DSC Script is then ran on the VM created earlier in the template. The DSC Script installs IIS, .NET 4.5 and the Remote Web Deploy IIS extension.

DSC and ARM templates with Jeffrey Snover

Channel 9: https://channel9.msdn.com/Shows/Tuesdays-With-Corey/Tuesdays-with-Corey-DSC-and-ARM-templates-with-Jeffrey-Snover

For more information , you can see:
Azure Resource Manager: https://aka.ms/edx-dev205bx-az24
IIS: https://aka.ms/edx-dev205bx-iis
.NET Framework 4.5: https://aka.ms/edx-dev205bx-net
Web Deploy: https://aka.ms/edx-dev205bx-wd

Configuration Management Tooling

Bookmark this page

Configuration Management Utilities

Configuration management is the task of implementing, tracking, and controlling changes to software across a large variety of machines.

Many of the common practices from source control management such as revision control are also seen in configuration management (CM) software. Configuration management utilities offers a lot of features such as:

- **Baselines.** Establishing a base configuration that is used as the default for new virtual or physical machine
- **Revision History.** Enables your team to determine who or what changes specific configuration settings
- **Replication.** Enables your configuration changes to be replicated across multiple machines.
- **Agents.** Software specifically installed on machines to receive configuration change requests and apply them to the local machine

Chef and Puppet are two of the most common examples of configuration management software used throughout the industry. Both Chef and Puppet are written in Ruby and are licensed under the Apache license. Template images are available for both Chef and Puppet in Azure. Both configuration management utilities discussed in this topic support Windows and Linux operating systems.

Puppet

http://puppetlabs.com/

Puppet is an open-source configuration management utility that is produced by Puppet Labs. Puppet has a unique declarative language that can be used to describe system configuration. These configuration changes can be applied directly to a virtual machine or distributed to multiple virtual machines by using a catalog. The Puppet agent periodically polls the machine for its current configuration and then syncs that configuration data to the Puppet master, a machine that manages all of the other machines with agents installed. The Puppet master ensures that the machines with the agent installed are in compliance with the latest configuration defined in the catalog.

Chef

http://www.chef.io

Chef is another popular open-source configuration management utility. Chef is unique because configuration changes are composed into recipes. Recipes are composed of individual configuration changes that are called resources, which can include:

- A file to store
- A template configuration change
- A software package to install

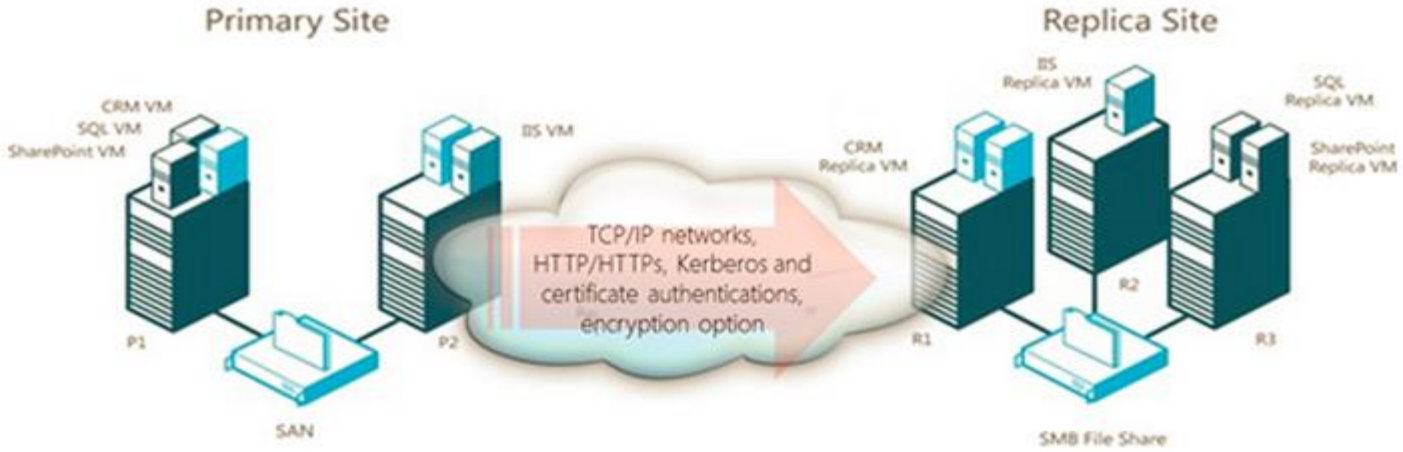
Recipes can be combined and used to automate the most common infrastructure tasks along with software configuration changes. Using an agent called a node, the Chef server is polled for changes that are made to the installed recipes and ensures that individual machines (physical or virtual) are in compliance with the latest version of the recipe.

Hyper-V Replica

Bookmark this page

Hyper-V Replica

Hyper-V Replica is a built-in feature of Windows Server 2012 that manages the replication of your virtual machines. Using Hyper-V Replica, you can asynchronously replicate a virtual machine in a primary site to a replica virtual machine in a secondary site.



In this example, we have a primary site with a Hyper-V host server running Windows Server 2012 or later and a secondary site with another host server. Virtual machines on the primary site replicate to machines on the secondary site, enabling workload continuity and recovery when outages occur.

Scenarios

Hyper-V replica is straight-forward to implement and can be used in a variety of scenarios. Some examples include:

Head office and branch office

In this scenario, there are two sites: a main head office and one or more branch offices in different physical locations. Taking advantages of virtualized workloads, Hyper-V Replica can be used to provide disaster recovery support for the branch offices.
For this situation, day-to-day operations would run on the virtual machines running on primary servers at the various branch offices. Each branch office would have a Replica server standing by at the head office to take over the workload in the event that the primary server must go offline for any reason.

Cloud service provider

In this scenario, the hosting provider sets up a Replica server at their datacenter which receives replication data from a number of primary servers running virtualized workloads on the premises of their various customers. The hosting provider's Replica server thereby provides disaster recovery capability for the customers who subscribe to it. The Trusted Group feature of Replica allows the hosting provider to segregate the replicated data from each customer, using separate storage locations and tagging to prevent data from various customers from being mixed.

For more information , you can see:
Windows Server 2012: https://aka.ms/edx-dev205bx-ws01
Virtual machines: https://aka.ms/edx-dev205bx-az11

Windows Server Backup

Bookmark this page

Windows Server Backup

Windows Server Backup is an essential backup feature built-in to the Windows Server OS. Windows Server Backup is intended for use by everyone who needs a basic backup solution—from small business to large enterprises—but is even suited for smaller organizations or individuals who are not IT professionals. You can use Windows Server Backup to create and manage backups for the local computer or a remote computer. And, you can schedule backups to run automatically.

Backup and Restore using Windows Server Backup

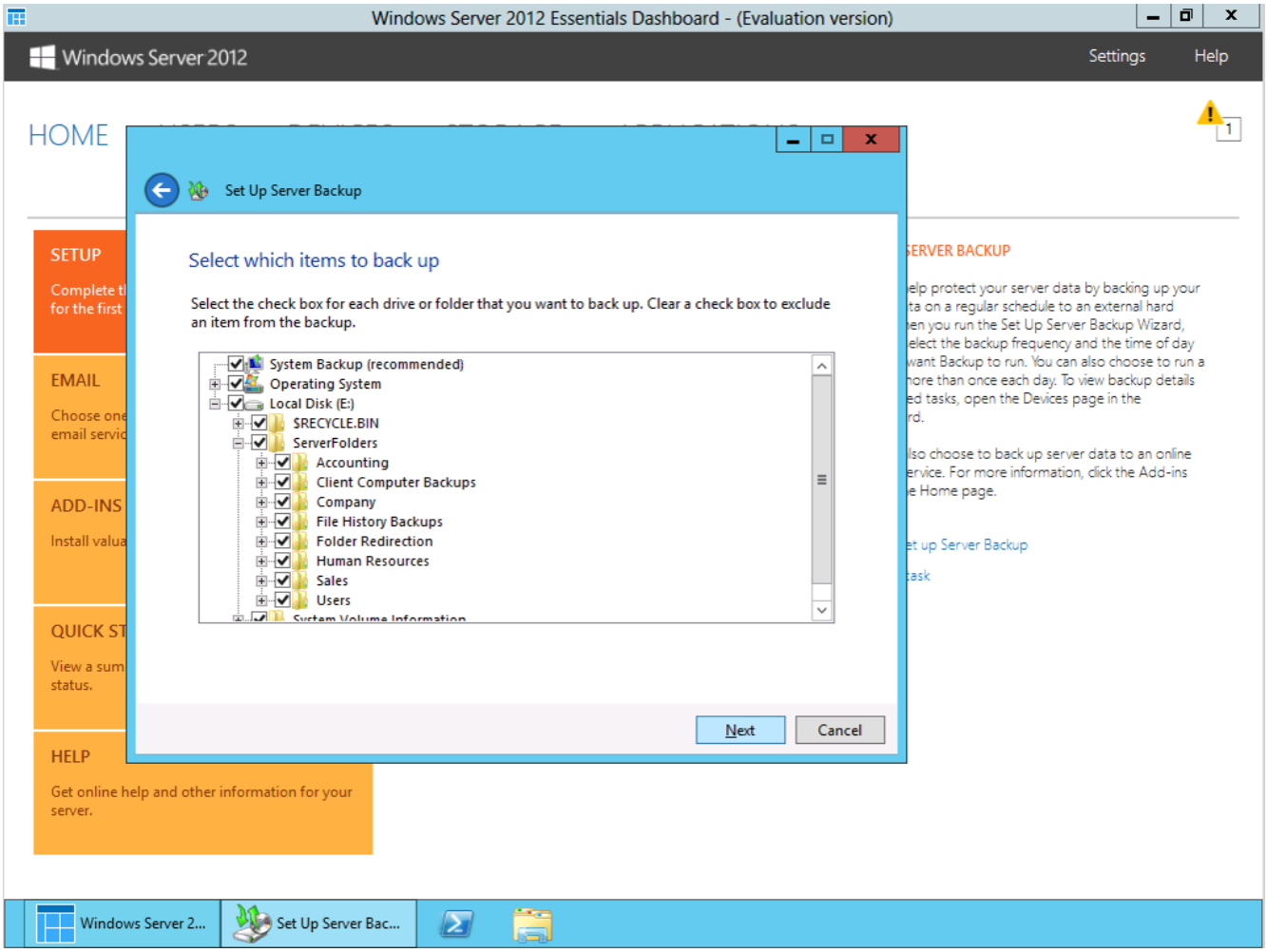
Bookmark this page

Usage

The Windows Server Backup feature in Windows Server 2008 consists of a Microsoft Management Console (MMC) snap-in and command-line tools that provides the coverage a complete solution. You can use four wizards to guide you through running backups and recoveries. You can use Windows Server Backup to back up a full server (all volumes), selected volumes, or the system state. You can recover volumes, folders, files, certain

applications, and the system state. And, in case of disasters like hard disk failures, you can perform a system recovery, which will restore your complete system onto the new hard disk, by using a full server backup and the Windows Recovery Environment.

Backup and Restore using Windows Server Backup



Windows Server Backup is designed to support the restore of files and folders, as well as a bare metal recovery of the server. The server backup feature is also designed to support the restore of the client backup database and factory reset of the server. This feature is built into the OS and can be configured directly on the host machine.

For more information , you can see:
Windows Server 2008: https://aka.ms/edx-dev205bx-ws02

System Center Data Protection Manager

Bookmark this page

Data deduplication

Data deduplication (dedup) was introduced in Windows Server 2012 as a next-generation replacement for the Single-Instance Storage (SIS) feature in Windows Storage Server 2008. It uses an advanced, variable block-size chunking algorithm to

provide maximum deduplication savings per volume. A post-processing approach is used to preserve all file system semantics and to ensure negligible impact on the primary data path performance.
Data deduplication is designed to be installed on primary data volumes without adding additional dedicated hardware so that it doesn't impact the primary workload on the server. The default settings are nonintrusive because they allow data to age for five days before processing a particular file, and has a default minimum file size of 32 KB. The implementation is designed for low memory and CPU usage.

Using deduplication with DPM can result in large savings. The amount of space saved by deduplication when optimizing DPM backup data varies depending on the type of data being backed up. For example, a backup of an encrypted database server may result in minimal savings since any duplicate data is hidden by the encryption process. However backup of a large Virtual Desktop Infrastructure (VDI) deployment can result in very large savings in the range of 70-90+% range, since there is typically a large amount of data duplication between the virtual desktop environments.

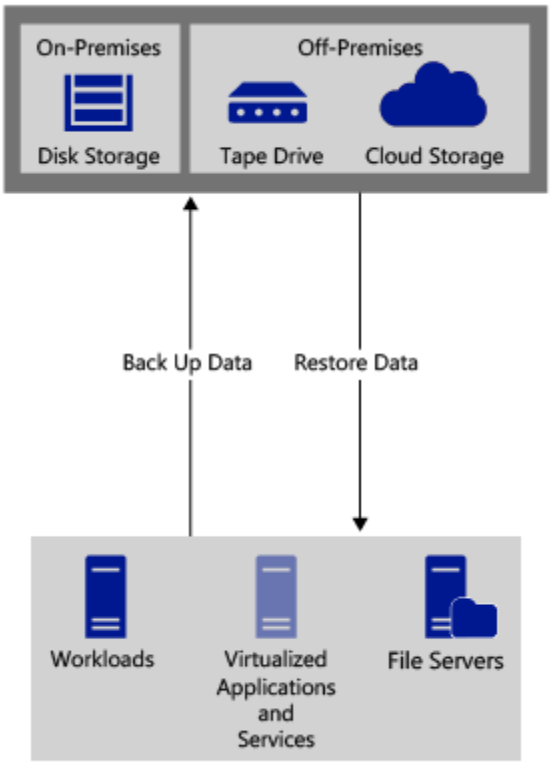
System Center Data Protection Manager

System Center Data Protection Manager (DPM) is an enterprise backup system. Using DPM you can backup (copy) data from a source location to a target secondary location. If original data is unavailable because of planned or unexpected issues, you can restore data from the secondary location. Using DPM you can back up application data from Microsoft servers and workloads, and file data from servers and client computers. You can create full backups, incremental backups, differential backups, and bare-metal backups to completely restore a system.

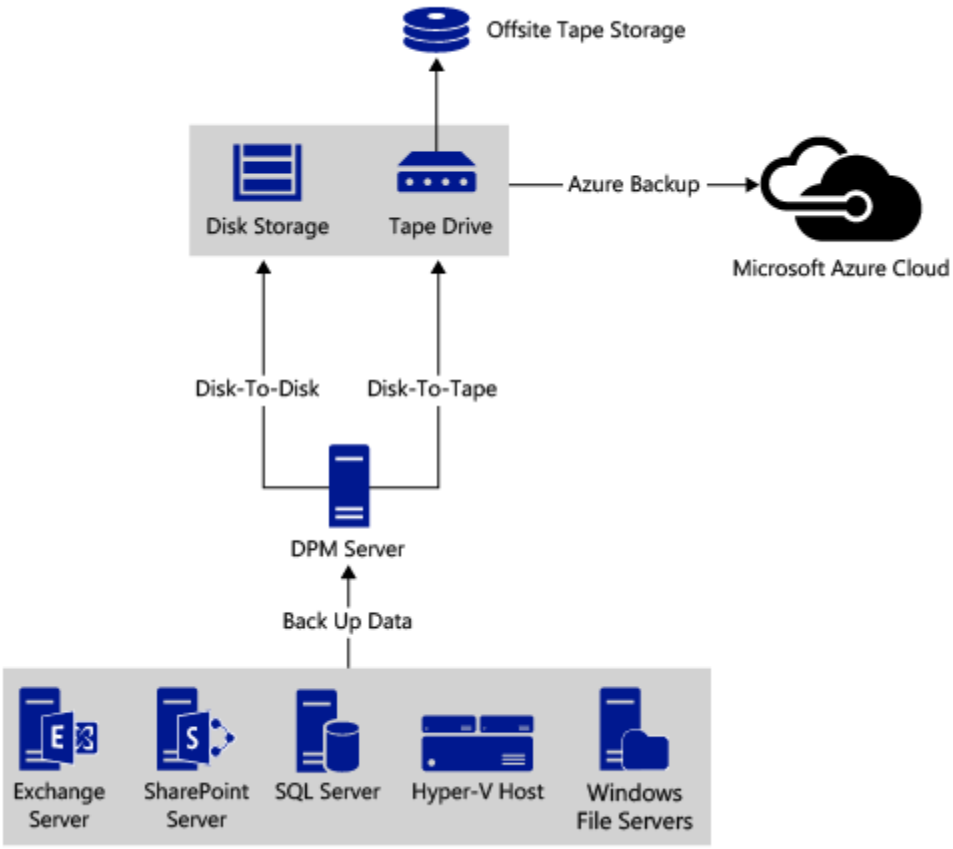
DPM also supports data deduplication. In the most common scenario, DPM runs in a Hyper-V virtual machine and stores backup data to VHDs in shared folders on a Windows File Server with data deduplication enabled.

Backup Using Data Protection Manager

In a traditional backup scenario, each machine is responsible for backing up to your long-term stores and recovering from the stores in the case of an event.



DPM acts as a conduit for your traditional backups. All of your application and workload servers can backup to DPM and DPM will manage the backup to one or many backup tiers. Using DPM, you disassociate your final backup solution from the workloads that are attempting to backup.



With DPM you can implement tiers of backup. For example, you can use Disk-to-Disk backup for non-critical data with a low Recovery Point Objective (RPO). You can use Disk-to-Tape backup for compliance data where you cannot tolerate data loss and require a higher RPO. You can even store the tape offsite or in the cloud using Azure Backup.

For more information , you can see:
Windows Storage Server 2008 r2: https://aka.ms/edx-dev205bx-wsstor
System Center Data Protection Manager: https://aka.ms/edx-dev205bx-sdpm
Azure Backup: https://aka.ms/edx-dev205bx-az30

Azure Site Recovery

Bookmark this page

Azure Site Recovery

The Azure Site Recovery service is part of a robust business continuity and disaster recovery (BCDR) solution that protects your on-premises physical servers and virtual machines by orchestrating and automating replication and failover to Azure, or to a secondary on-premises datacenter.

For more information , you can see:
Azure Site Recovery service: https://aka.ms/edx-dev205bx-az31
Azure: https://aka.ms/edx-dev205bx-az34

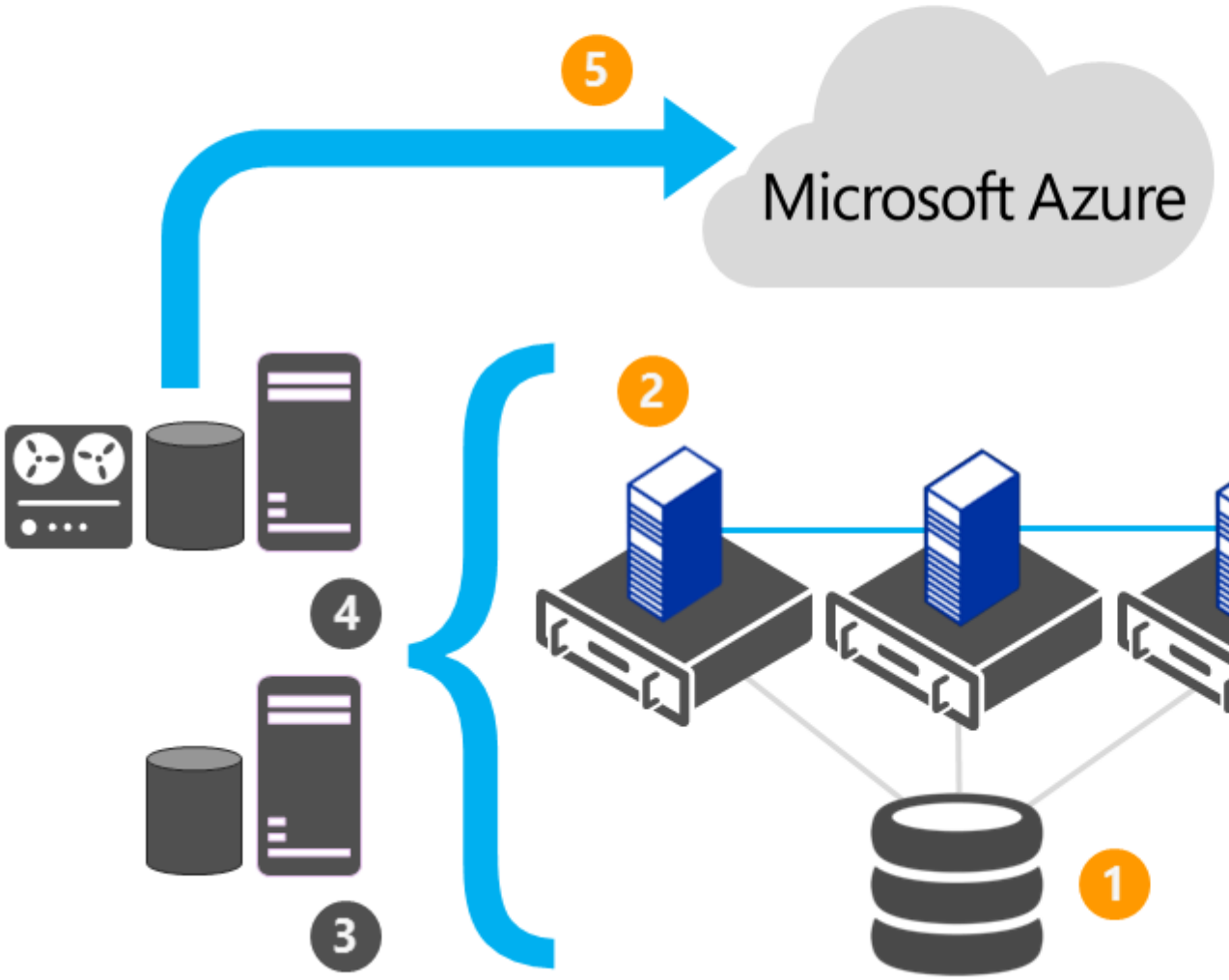
Orchestration using Azure Site Recovery

Bookmark this page

Business Continuity

Azure can be combined with existing Microsoft solutions to provide breadth & depth for a business continuity solution

1. Hyper-V Failover , Clustering for VM Resilience
2. Hyper-V Guest Clustering for app- level HA, i.e. SQL Server AlwaysOn FCI
3. Simplified protection with Windows Server Backup
4. Centralized backup with Data Protection Manager
5. Integration of WSB/DPM with Microsoft Azure Backup
6. Orchestrated Physical, Hyper-V & VMware VM Replication & Recovery using Azure Site Recovery, between on-premises locations, or between on-premises & Microsoft Azure



Recovery Plans

Recovery plans consist of one or more ordered groups that contain protected virtual machines or replication groups (for SAN replication). Machines fail over according to group they are in. Virtual machines in a particular group fail over in parallel.

The way in which you create a recovery plan depends on your Site Recovery deployment.

- **Hyper-V replication (VMM)**: If you're replicating from a VMM site to a secondary on-premises site or to Azure using Hyper-V replication you add protected Hyper-V virtual machines from a VMM cloud to a recovery plan.
- **Hyper-V replication (Hyper-V site)**: If you're replicating from a Hyper-V site (without a VMM server) to Azure you add protected Hyper-V virtual machines from a protection group to a recovery plan.
- **SAN replication**: If you're replicating to a secondary on-premises site using SAN replication you add a replication group that contains virtual machines to the recovery plan. You select a replication group rather than specific virtual machines because all virtual machines in a replication group must fail over together (failover occurs at the storage layer first.
- **VMware replication**: If you're replicating VMware virtual machines to Azure you add replication groups that contain virtual machines to a recovery plan.

For more information , you can see:
Hyper-V: https://aka.ms/edx-dev205bx-whv
Microsoft Azure Backup: https://aka.ms/edx-dev205bx-az30
Virtual machines: https://aka.ms/edx-dev205bx-az11

Executing Recovery Plans

Bookmark this page

Executing Recovery Plans

This recovery plan orchestrates VMs from the secondary site (Azure) to the primary site (on-premise) using a Site Recovery recovery plan.

Azure Backup

Bookmark this page

Azure Backup Overview

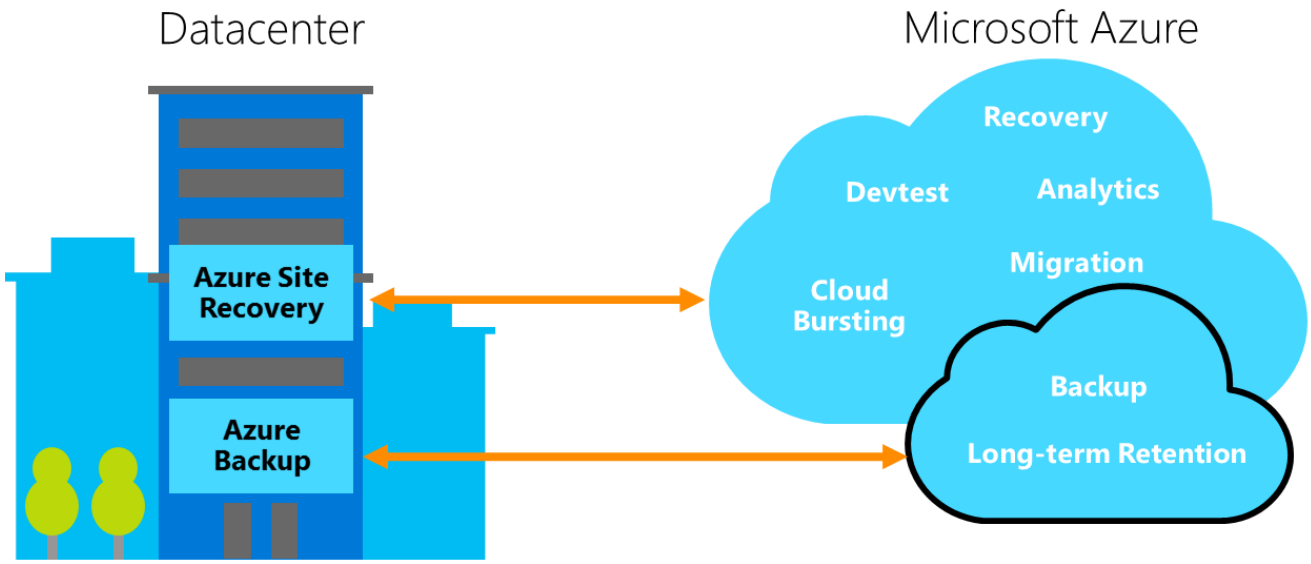
Azure Backup is a multi-tenanted Azure service which enables you to back up your data present anywhere: on-premises or in Azure. It replaces your existing on-premises or offsite backup solution with a cloud based offering. It also gives the flexibility of protecting assets running in the cloud. Using this solution, you can backup data and applications from their System Center Data Protection Manager (SCDPM) servers, Windows servers, Windows client machines or Azure IaaS virtual machines. Azure Backup and SCDPM are the fundamental technologies which make up Microsoft's cloud-integrated backup solution.

Azure Backup

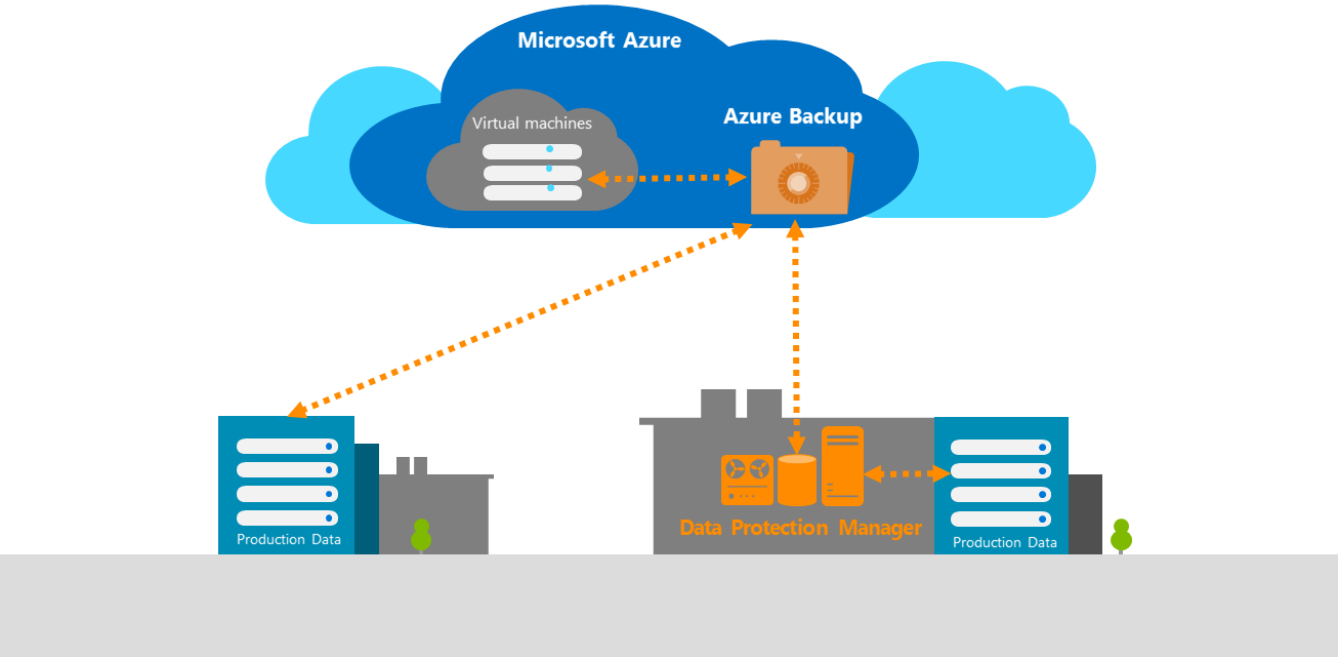
Channel 9: https://channel9.msdn.com/Blogs/Windows-Azure/What-is-Azure-Backup

Azure Backup Scenarios

Azure Backup can be used as part of an wholesale backup and continunity strategy.



Azure Backup is ideal for long-term retention and hard backup of point-in-time state or data. In a hybrid solution, Azure Backup can be used as the backup destination and recovery source for both your application data and your infrastructure.&nbs



Virtual Machines can backup and recover using Azure Backup along with application data. Azure Backup can also be used in scenarios where you would like to backup client devices or remote (branch) offices without investing in a long-term solution for each office.

For more information , you can see:
Windows servers: https://aka.ms/edx-dev205bx-ws

StorSimple

Bookmark this page

StorSimple

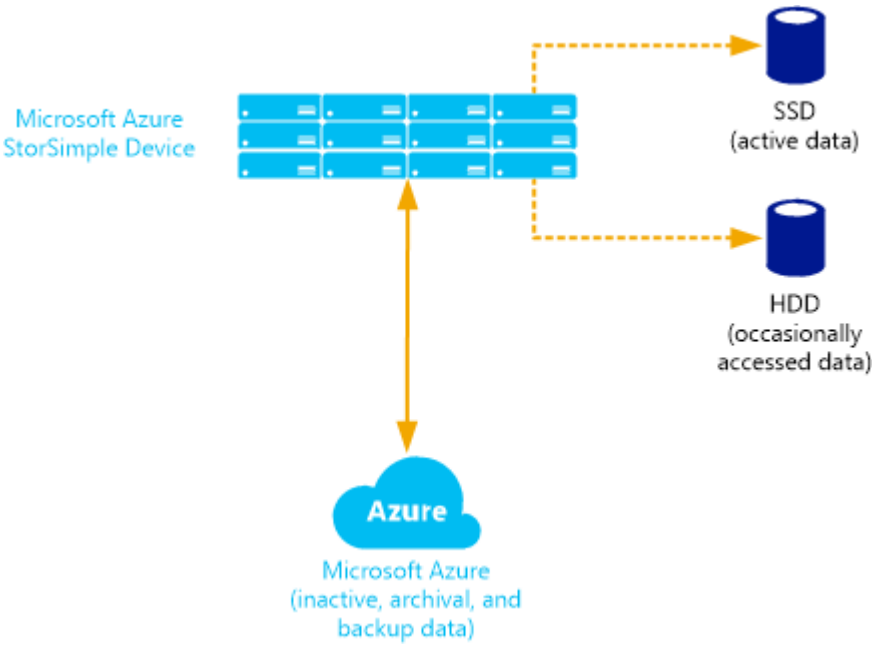
Microsoft Azure StorSimple is an efficient, cost-effective, and manageable solution that eliminates many of the issues and expense associated with enterprise storage and data protection. It uses a proprietary device (the Microsoft Azure StorSimple device) and integrated management tools to provide a seamless view of all enterprise storage, including cloud storage.

StorSimple: A Hybrid Cloud Storage Solution

Channel 9: https://channel9.msdn.com/Blogs/Windows-Azure/StorSimple-Hybrid-Cloud-Storage-Solution

StorSimple Storage Management

Microsoft Azure StorSimple automatically arranges data in logical tiers based on current usage, age, and relationship to other data. Data that is most active is stored locally, while less active and inactive data is automatically migrated to the cloud.



To enable quick access, StorSimple stores very active data (hot data) on SSDs in the StorSimple device. It stores data that is used occasionally (warm data) on HDDs in the device or on servers at the datacenter. It moves inactive data, backup data, and data retained for archival or compliance purposes to the cloud. StorSimple adjusts and rearranges data and storage assignments as usage patterns change. For example, some information might become less active over time. As it becomes progressively less active, it is migrated from SSD to HDD and then to the cloud. If that same data becomes active again, it is migrated back to the storage device.

Thin provisioning

Thin provisioning is a virtualization technology in which available storage appears to exceed physical resources. Instead of reserving sufficient storage in advance, StorSimple uses thin provisioning to allocate just enough space to meet current requirements. The elastic nature of cloud storage facilitates this approach because StorSimple can increase or decrease cloud storage to meet changing demands.

Deduplication and compression

Microsoft Azure StorSimple uses deduplication and data compression to further reduce storage requirements. Deduplication reduces the overall amount of data stored by eliminating redundancy in the stored data set. As information changes, StorSimple ignores the unchanged data and captures only the changes. In addition, StorSimple reduces the amount of stored data by identifying and removing unnecessary information.

Customer Scenario

Who is the customer?

Fabrikam Publishing is a media and publishing company in Seattle, Washington, with approximately 5,000 employees.

What does the customer already have?

Fabrikam has a single data center that primarily runs Microsoft server software, including Active Directory Domain Services AD DS) and a number of AD-integrated services, including Exchange 2013, as well multi-tier, internal, AD-integrated IIS-based web applications with SQL Server 2014 as the database platform. The services are consumed by client systems hosted in three buildings located in adjacent areas of the city. Buildings are connected to the data center by using site-to-site VPN with the throughput of 100Mbps. Each building has also an independent connection to the Internet.
Server backups are performed by using tape libraries with autoloaders. Tapes are periodically shipped for permanent storage to an offsite location.

If something catastrophic were to happen to the data center, the IT team would have to deploy replacement physical servers by reinstalling the operating system and restoring data from backups in the equipment rooms in small server rooms located in each building.

Fabrikam's IT staff likes to stay current with the latest offerings from Microsoft so that the department functions as cost-effectively as possible. In order to reduce costs, the IT staff has recently started planning the initiative to virtualize majority of its physical servers using the Hyper-V platform and to deploy System Center 2012 R2 Virtual Machine Manager (SCVMM) for managing the resulting virtualized environment.

For more information , you can see:
Active Directory Domain Services: https://aka.ms/edx-dev205bx-adds
Exchange 2013: https://aka.ms/edx-dev205bx-mex
SQL Server 2014: https://aka.ms/edx-dev205bx-sql04
System Center 2012 R2 Virtual Machine Manager: https://aka.ms/edx-dev205bx-scvmm

Customer Goal

Bookmark this page

What is the customer's goal?

"We needed greatly improved disaster, server, and application recovery processes," says Anthony Ciske, IT Director for Fabrikam. "We've had some near-disasters in the past that were a real pain to recover from. We needed a real disaster recovery solution for our critical workloads that was compatible with our budget—and our staffing bandwidth." The team had explored building a secondary data center and employing commercial disaster recovery solutions in the past, but both turned out to be too expensive for serious consideration.

Solution Considerations

Bookmark this page

What does the customer need?

- The ability to perform data-center level recovery for critical workloads that can be executed in the event of a data center failure, with an automated and orderly recovery process so different tiers of the application start in the correct order and remain in a consistent state without manual intervention.
- The ability to perform failback following restoring on-premises data center functionality that can be executed in the automated and orderly manner.
- The ability to perform multi-tier application and individual server-level recovery of critical workloads.
- Support for server-level and application-level high availability whenever possible.

- Quick testing and validation of recovery processes with minimal interruption to the production environment.
- Minimized capital and operational expenses.
- Optimized authentication for AD-integrated services and applications.
- Centralized management of backups and reduced or eliminated dependency on offsite tape storage.
- The level of security and privacy commensurate with highly sensitive and competitive nature of the business.

What things worry the customer?

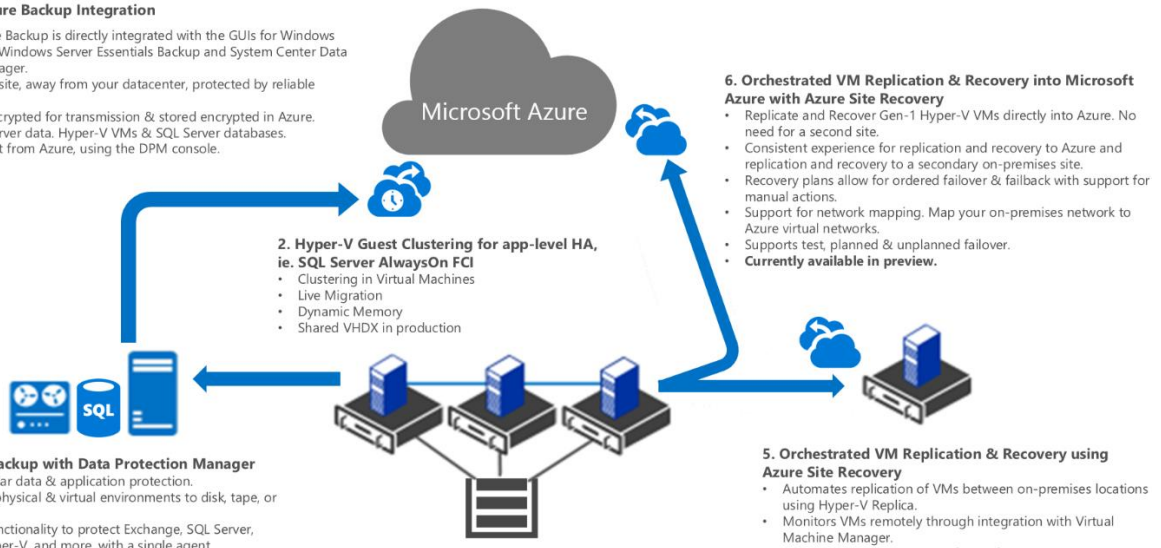
- Solution must significantly improve their current recovery point/time objectives (which today is a manual process).
- Overall cost of the solution.
- Protecting a diverse environment such as physical servers or other hypervisors.
- The management tools for the solution must be available in the event one of the data centers is unavailable.
- Protect data that is not hosted within a virtual hard disk (VHD/X).
- The protected data must be secure.
- Unsure about which workloads are supported on azure.

Common Implementations

Backup

4. Microsoft Azure Backup Integration

- Microsoft Azure Backup is directly integrated with the GUIs for Windows Server Backup. Windows Server Essentials Backup and System Center Data Protection Manager.
- Backups are offsite, away from your datacenter, protected by reliable Azure storage.
- Backups are encrypted for transmission & stored encrypted in Azure.
- Supports file server data. Hyper-V VMs & SQL Server databases.
- Recover straight from Azure, using the DPM console.



6. Orchestrated VM Replication & Recovery into Microsoft Azure with Azure Site Recovery

- Replicate and Recover Gen-1 Hyper-V VMs directly into Azure. No need for a second site.
- Consistent experience for replication and recovery to Azure and replication and recovery to a secondary on-premises site.
- Recovery plans allow for ordered failover & failback with support for manual actions.
- Support for network mapping. Map your on-premises network to Azure virtual networks.
- Supports test, planned & unplanned failover.
- **Currently available in preview.**

2. Hyper-V Guest Clustering for app-level HA, ie. SQL Server AlwaysOn FCI

- Clustering in Virtual Machines
- Live Migration
- Dynamic Memory
- Shared VHDX in production

3. Centralized backup with Data Protection Manager

- Scalable, granular data & application protection.
- Protect across physical & virtual environments to disk, tape, or Azure.
- App-specific functionality to protect Exchange, SQL Server, SharePoint, Hyper-V, and more, with a single agent.
- Supports protection of up to 800 VMs per DPM Server.
- Protects both Windows & Linux VMs.
- Provides low-cost DR through DPM chaining.
- Centralized monitoring of multiple DPM servers through Operations Manager.

1. Hyper-V Failover Clustering for VM Resilience

- Automatic failover of clustered VMs build into Windows Server
- Huge scale (up to 64 physical hosts & 8,000 VMs per cluster).
- Built-in hardware and software validation and diagnostics reduce misconfiguration and support compliance.

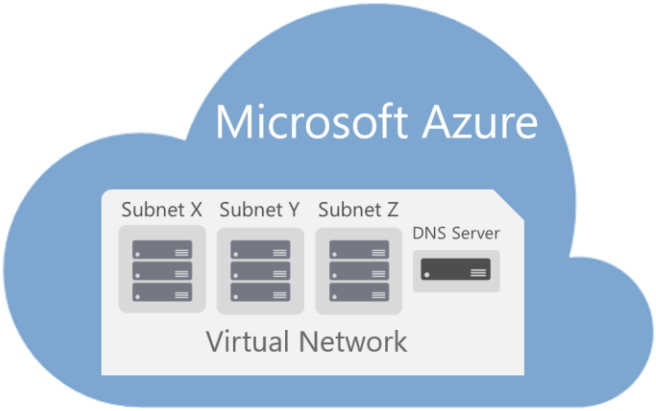
5. Orchestrated VM Replication & Recovery using Azure Site Recovery

- Automates replication of VMs between on-premises locations using Hyper-V Replica.
- Monitors VMs remotely through integration with Virtual Machine Manager.
- Secure replication using certificates for encryption.
- Recovery plans allow for orderly failover for multi-tier applications with support for manual or scripted actions.
- Support for network mapping.
- Supports non-disruptive testing on your schedule.

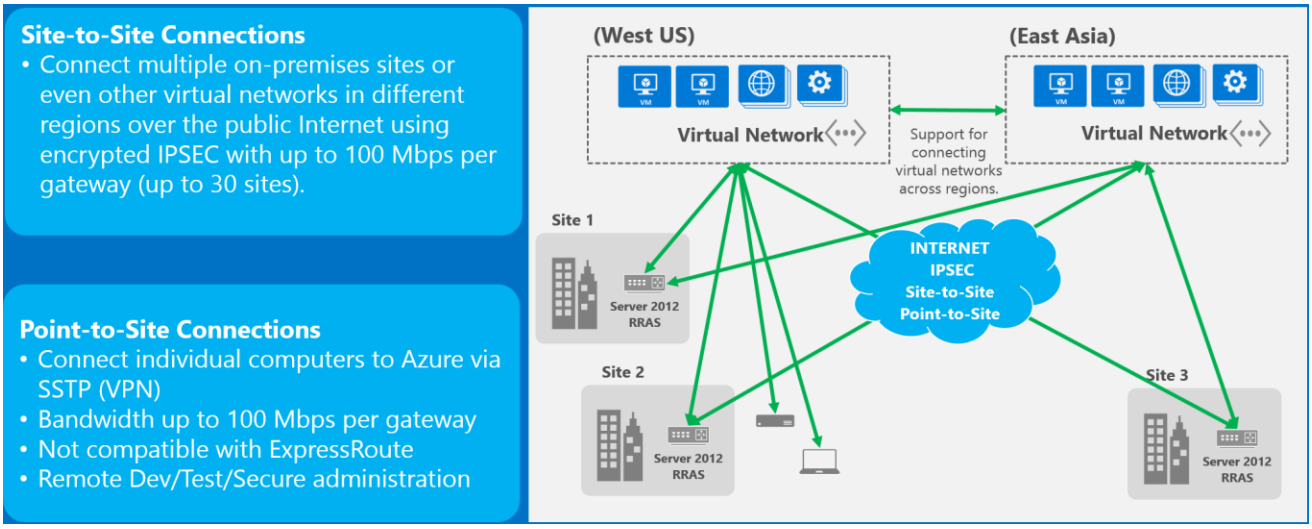
Virtual Networks

Virtual Networks Overview

- Support for controlling deployment of Virtual Machines and Cloud Services into private IP addresses and subnets
- Internal load-balancing (Intranet and N-Tier Apps)
- Static IP support, DNS, and communication across cloud services.
- Access resources such as SQL, CRM, and AD from VMs and Cloud Services in Azure.
- Secure virtual machines with Network Security Groups
- Availability: All regions
- Support for hybrid networking:
  - **Site to Site**
  - **Point to Site**
  - **ExpressRoute**



VNET Connectivity



Call to Action

Bookmark this page

Design the Solution

You will now design a potential solution for **Fabrikam Publishing**. Prior to completing this exercise, please ensure that you have read all of the previous units in this case study. Particularly, make sure that you have read and you understand the **Customer Scenario** and **Solution Considerations**.

In this exercise, you will tackle a few primary tasks.

1. You will need to determine who you should present this solution to. Who is the target customer audience (stakeholder)? Who are the decision makers? You will answer these questions below in the problem sections.
2. You will need to determine what you intend to address with your solution. What customer business needs do you need to address with your solution? How will you address each business need? You will provide a brief explanation below in the problem sections.
3. You will need to create a diagram for your proposed solution. This can be done with Visio or with any image editor of your choice. The ideal output is either a high-resolution PNG file or a PDF document. You will upload this diagram in the final section below.

Once you have completed these three tasks, create a new post in the solution forum, attach an image file to the post for your diagram and your response to prompts #1 & #2.

Business Continuity Case Study

Topic: Case Studies / Module 8: Business Continuity Case Study

Show Discussion

Sample Solution

You are highly encouraged to try and come up with your own solution for this case study and post that solution in the forums. Once you have completed that, you can review the sample solution linked below and compare it against your own solution.

Sample Solution