FullStack.Cafe - Kill Your Tech Interview

Q1: What is WebJob in Azure? ☆☆

Topics: Azure

Answer:

WebJobs is a *feature* of **Azure App Service** that enables you to run a program or script in the same instance as a web app, API app, or mobile app. There is no additional cost to use WebJobs. You can use the Azure WebJobs SDK with WebJobs to simplify many programming tasks. Azure Functions provides another way to run programs and scripts.

Q2: Is there a way to view deployed files in Azure? ☆☆

Topics: Azure

Answer:

If you're just trying to look around, and see the various directories and files in your deployment, you can enter the site's "Kudu" dashboard (called Advanced Tools), using the url format `http://<yoursitename>.scm.azurewebsites.net`

Kudu is the engine behind git deployments in Azure Web Sites. This will give you a web-based dashboard, including a debug console (web-based) where you can explore your various directories (and the directories will show up visually as well).

Q3: What is the difference between an *Azure Tenant* and *Azure Subscription*? ☆☆

Topics: Azure

Answer:

- a **Tenant** is associated with a single identity (person, company, or organization) and can own one or several subscriptions
- a **Subscription** is linked to a payment setup and each subscription will result in a separate bill
- in every subscription, you can add virtual resources (VM, storage, network, ...)

- Every tenant is linked to a single Azure AD instance, which is shared with all tenant's subscriptions
- Resources from one subscription are isolated from resources in other subscriptions
- An owner of a tenant can decide to have multiple subscriptions:
  - when Subscriptions limits are reached
  - to use different payment methods
  - to isolate resources between different departments, projects, regional offices, and so on.

Q4: What would you use: Azure CLI or PowerShell? Explain. ☆☆

Topics: Azure

Answer:

Both, Azure CLI and the PowerShell package use the REST API of Azure.

- Azure CLI is a PowerShell-like-tool available for all platforms. You can use the same commands no matter what platform you use: Windows, Linux or Mac and it's open-source
- The real biggest benefit of Powershell is that it operates on objects, not on strings. Therefore it's really easy to work with the output of commands
- You might end up replacing all my ARM Templates with Azure CLI scripts as it is less verbose and easy to read

PowerShell has some significant advantages over CLI:

- PowerShell is a language of Azure Functions. So you can easily write a module or code and push it to a secured Azure Function. CLI is not a supported language.
- Same goes for Azure Automation Runbooks, whereas CLI is not a supported language.
- If you want to orchestrate a hybrid workload covering on premise and cloud, i.e. deploy a Service to a Windows Server and then deploy an endpoint in Azure for it to interact with, PowerShell can but CLI can't (and I would surprised if it did).
- If you want to invest in one language that can call legacy Modules for management, PowerShell can but CLI can't.
- If you want to create your own DLL in C# or any other language and incorporate that into your scripts. I've had to reverse engineer DLLs and wrap them in PowerShell to incorporate them into orchestrations.
- I've had significantly more success extending DevOps pipelines with PowerShell than CLI.
- PowerShell has very good multithreaded workload support for scenarios such as Azure unit and smoke testing and large Azure parallel deployments (beyond DevOps parallel tasks which get clunky). TBH, I am not aware of native multithreading in Azure CLI.
- I can develop a module, use it in all the above scenarios and then give it to an end client; TBH, I don't believe I can do that for CLI (create a module, digitally sign it and then nuget feed distro).

Q5: What is the difference between Azure AD App *Application Permissions* vs *Delegated Permissions*? ☆☆

Topics: Azure

Answer:

You typically use **delegated permissions** when you want to call the Web API as the logged-on user. Say for example that the Web API needs to filter the data it returns based on who the user is, or execute some action as the logged-in user. Or even just to log which user was initiating the call.

Application permissions are used when the application calls the API itself. For example to get the weather forecast for a certain zipcode (it does not matter which user is logged on). The client can even call the API when there's no user present (some background service calling the API to update some status).

Q6: What is the difference between_Azure API Apps_, *Logic Apps*, *Web Apps* and *Azure Functions*? ☆☆

Topics: Azure

Answer:

Logic Apps:

Logic Apps provide a way to simplify and implement scalable integrations and workflows in the cloud. It provides a visual designer to model and automates your process as a series of steps known as a workflow. There are many connectors across the cloud and on-premises to quickly integrate across services and protocols. A logic app begins with a trigger (like 'When an account is added to Dynamics CRM') and after firing can begin many combinations of actions, conversions, and condition logic.

API Apps:

API apps in Azure App Service offer features that make it easier to develop, host, and consume APIs in the cloud and on-premises. With API apps you get enterprise-grade security, simple access control, hybrid connectivity, automatic SDK generation, and seamless integration with Logic Apps.

Web Apps:

App Service Web Apps is a fully managed to compute platform that is optimized for hosting websites and web applications. This platform-as-a-service (PaaS) offering of Microsoft Azure lets you focus on your business logic while Azure takes care of the infrastructure to run and scale your apps.

Functions:

Azure Functions is a solution for easily running small pieces of code, or "functions," in the cloud. You can write just the code you need for the problem at hand, without worrying about a whole application or the infrastructure to run it. Functions can make development even more productive, and you can use your development languages of choice, such as C#, F#, Node.js, Python or PHP. Pay only for the time your code runs and trust Azure to scale as needed. Azure Functions lets you develop serverless applications on Microsoft Azure.

Q7: What is Azure CDN (Content Delivery Network) and why to use it? ☆☆

Topics: Azure CDN

Answer:

Almost each and every site has HTML, JS, CSS, image, video and font files that do not change much from user to user. To minimize the application load time we should use CDN.

Azure Content Delivery Network (CDN) is CDN service provided by Azure Cloud Platform that enables in storing and accessing data on different content servers and locations – used by online or cloud services. It provides:

- Better performance and user experience for end-users who are far from the destination content source or where there are many intermediary nodes in between to reach the content.
- Ability to scale to handle instantaneous high load and demand
- Caching content from publicly available Azure storage blobs
- Caching web content, images, scripts, and other website content from CDN
- Enables in caching objects to the CDN that are provided by an Azure cloud service
- Cache content based on specific query strings
- Accessing cached content from a custom domain name by mapping the CDN HTTP Endpoint.

Q8: What are ARM Templates in Azure? ☆☆

Topics: Azure

Answer:

ARM templates are a form of infrastructure as code, a concept where you define the infrastructure you need to be deployed. The template defines the resources, and the Azure ARM management layer is responsible for creating the infrastructure. Templates use a JavaScript Object Notation (JSON) syntax that also includes advanced capabilities. Templates use declarative syntax. In this way, you can specify what to deploy without writing a series of commands that specify how to deploy it.

The template includes the following parts:

- **Parameters** - values that allow you to use the same template in different environments during deployment.
- **Variables** - values to be reused in different templates. Variables can use values from parameters.
- **User-Defined Functions** - lets you define customized elements to simplify templates.
- **Resources** - specifies Azure resources to be deployed.
- **Outputs** - the return value of the deployed resource.

Q9: What are *On Demand WebJobs* aka. *Scheduled WebJobs* aka. *Triggered WebJobs* in Azure? ☆☆☆

Topics: Azure

Answer:

Triggered WebJobs are WebJobs which are run once when a URL is called or when the schedule property is present in `schedule.job`. **Scheduled WebJobs** are just WebJobs which have had an Azure Scheduler Job created to call URL on a schedule.

Summary:

- `+` Executable/Script on demand
- `+` Scheduled executions
- `-` Have to trigger via `.scm` endpoint
- `-` Scaling is manual
- `-` VM is always required

Q10: Compare Azure WebJobs vs Azure Function. When to use one? ☆☆☆

Topics: Azure

Answer:

Azure Functions offers more developer productivity than Azure App Service WebJobs does. It also offers more options for programming languages, development environments, Azure service integration, and pricing. For most scenarios, it's the best choice.

Here are two scenarios for which **WebJobs** may be the best choice:

- You need more control over the code that listens for events, the `JobHost` object. Functions offers a limited number of ways to customize `JobHost` behavior in the host.json file. Sometimes you need to do things that can't be specified by a string in a JSON file. For example, only the WebJobs SDK lets you configure a custom retry policy for Azure Storage.
- You have an App Service app for which you want to run code snippets, and you want to manage them together in the same Azure DevOps environment.

Azure Functions is built on the WebJobs SDK, so it shares many of the same event triggers and connections to other Azure services. Here are some factors to consider when you're choosing between Azure Functions and WebJobs with the WebJobs SDK:

	Functions	WebJobs with WebJobs SDK
Serverless app model with automatic scaling	✔	
Develop and test in browser	✔	
Pay-per-use pricing	✔	
Integration with Logic Apps	✔	
Trigger events	Timer Azure Storage queues and blobs Azure Service Bus queues and topics Azure Cosmos DB Azure Event Hubs HTTP/WebHook (GitHub, Slack) Azure Event Grid	Timer Azure Storage queues and blobs Azure Service Bus queues and topics Azure Cosmos DB Azure Event Hubs File system
Supported languages	C# F# JavaScript Java Python PowerShell	C#[1]
Package managers	NPM and NuGet	NuGet[2]

- WebJobs (without the WebJobs SDK) supports C#, Java, JavaScript, Bash, .cmd, .bat, PowerShell, PHP, TypeScript, Python, and more. This is not a comprehensive list. A WebJob can run any program or script that can run in the App Service sandbox.
- WebJobs (without the WebJobs SDK) supports NPM and NuGet.

Q11: What is the difference between an *Azure Web App* and an *Azure Web Role*? ☆☆☆

Topics: Azure

Answer:

Azure Web Role is like a virtual private host. You get a VM that acts as your web server, and you own that VM instance.

Azure Web Apps are like an elastic shared hosting service. You deploy your app to a web server that is not controlled by you and which also servers other users' sites. You can scale your site up and down (at some extra charge) to make it more elastic as your resource needs shift.

Web Roles give you several features beyond Web Apps (formerly Web Sites):

- Ability to run elevated startup scripts to install apps, modify registry settings, install performance counters, fine-tune IIS, etc.
- Ability to split an app up into tiers (maybe Web Role for front end, Worker Role for backend processing) and scale independently
- Ability to RDP into your VM for debugging purposes

- Network isolation
- Dedicated virtual IP address, which allows web role instances in a cloud service to access IP-restricted Virtual Machines
- ACL-restricted endpoints (added in Azure SDK 2.3, April 2014)
- Support for any TCP/UDP ports (Web Sites are restricted to TCP 80/443)

Web Apps have advantages over Web Roles though:

- Near-instant deployment with deployment history / rollbacks
- Visual Studio Online, github, local git, ftp, CodePlex, DropBox, BitBucket deployment support
- Ability to roll out one of numerous CMS's and frameworks, (like WordPress, Joomla, Django, MediaWiki, etc.)
- Use of SQL Database or MySQL
- Simple and fast to scale from free tier to shared tier to dedicated tier
- Web Jobs
- Backups of Web Site content
- Built-in web-based debugging tools (simple cmd/powershell debug console, process explorer, diagnostic tools like log streaming, etc.)

Q12: How would you choose between *Azure Blob Storage* vs. *Azure File Service*? ☆☆☆

Topics: Azure

Answer:

1. You can't mount Azure Blob Storage as a native share on a virtual machine.
2. Pricing: Blob storage is much cheaper than file storage.
3. Azure Blob Storage isn't hierarchical beyond containers. You can add files that have / or \ characters in them that are interpreted as folders by many apps that read blob storage.
4. Azure File Service provides a SMB protocol interface to Azure Blob Storage which solves the problem with (1).
5. Portability:
    - With blob storage, if you decide to migrate to a diff platform in future you may have to change your app code but
    - with File storage you can migrate your app to any other platform that supports SMB (assuming you are using native file system APIs in your app)

Use this rule of thumb:

- If you are developing a new application then leverage the native Azure API directly into Blob Storage.
- If you are porting an existing application that needs to share files then use Azure File Service.

Q13: Using Azure Functions, can I reference and use *NuGet* packages in my C# function? ☆☆☆

Topics: Azure

Answer:

You can use Nuget packages in your Azure Functions. The easiest way will be to use Visual Studio 2017 15.4 where there is a template for Azure Functions. Alternatively, the runtime supports NuGet references and will make sure they are correctly used when compiling and executing your functions.

In order to define your dependencies, you need to create a `Project.json` file with the required NuGet package references. Here is an example that adds a reference to `Microsoft.ProjectOxford.Face` version 1.1.0:

```
{
  "frameworks": {
    "net46":{
      "dependencies": {
        "Microsoft.ProjectOxford.Face": "1.1.0"
      }
    }
   }
}
```

Q14: What are *Web Role*, *Worker Role* and *VM Role* in Azure? ☆☆☆

Topics: Azure

Answer:

All roles (web, worker) are essentially Windows Server. Web and Worker roles are nearly identical:

- Web roles are Windows Server VMs with IIS enabled
- Worker roles are Windows Server VMs with IIS disabled (and you could manually enable it)
- VM roles are Windows Server 2008 images you construct locally via Hyper-V and upload to Azure **(and are now discontinued and no longer available as of May 31, 2013**
- Virtual Machines are Windows or Linux images created in Azure, stored as a vhd in your own storage, and have several enhancements over VM role. For example: since the vhd is in your own storage account, you can easily create an image template from your vhd, copy it to a new vhd, or even upload it to VM Depot (Linux only).

- With Web and Worker roles, the OS and related patches are taken care of for you; you build your app's components without having to manage a VM.
- With Virtual Machines, you simply pick an OS image from a gallery, which gets created for you and stored as a vhd in blob storage. You then RDP/ssh and set it up how you like.

Q15: What is the difference between an API App and a Web App? ☆☆☆

Topics: Azure

Answer:

App Services now replaces all Mobile, Api and Web Apps flavors as a single app framework with all the functionality rolled over to make things more accessible across application types. **Currently all of Web, Mobile and Api Apps are collectively called App Services.**

Function apps are now the exception. Creating a function app changes the user interface in the portal. The underlying web app, however, is no different. Setting an app setting named `FUNCTIONS_EXTENSION_VERSION` = `~1` turns any web app into a function app (minus the user interface in the portal).

Q16: What are the differences between *New* and *Classic* Storage Accounts in Azure? ☆☆☆

Topics: Azure

Answer:

- Classic storage accounts are created using existing Service Management API's (the REST API stack that's been available for the past several years).
- The newer storage accounts are created with the new Azure Resource Manager (ARM) API's (which are also wrapped in PowerShell and CLI now). One advantage of the new over the classic storage accounts is Storage Service Encryption (SSE) (to automatically encrypt your data when it is persisted to the cloud).

Ultimately they provide the same resources to your apps, but they're created/managed differently, and there are a few nuanced differences (such as the ability to tag resources that are created via ARM scripts).

Q17: What to use: many small Azure Storage Blob containers vs one really large container with tons of blobs? ☆☆☆

Topics: Azure

Answer:

You must consider:

- From a scalability/parallelization perspective, it doesn't matter because partitioning in Win Azure blobs storage is done at the blob level, not the container, which means you could (and often will) have two different blobs in a same container being served out by different servers.
- The extra containers can be nice as additional security boundaries and access control (for public anonymous access or different SAS signatures for instance),
- If you need to list blobs in a container, you will likely see better performance with the many-container model,
- Extra containers can also make housekeeping a bit easier when pruning (deleting a single container versus targeting each blob),
- In extreme cause you can pay a lot more if you create and delete many containers

Q18: When to choose *Worker Role* vs *Web Jobs* on Azure? ☆☆☆

Topics: Azure

Answer:

- **WebJobs** are good for lightweight work items that don't need any customization of the environment they run in and don't consume a lot of resources. They are also really good for tasks that only need to be run periodically, scheduled, or triggered. They are cheap and easy to set up/run. They run in the context of your Website which means you get the same environment that your Website runs in, and any resources they use are resources that your Website can't use.
- **Worker Roles** are good for more resource-intensive workloads or if you need to modify the environment where they are running (ie. a particular .NET framework version or something installed into the OS). Worker Roles are more expensive and slightly more difficult to set up and run, but they offer significantly more power.
- A **Worker Role** is self-hosted on a dedicated VM, and a Web Job is hosted in a Web App container.
- A **Worker Role** will scale independently, a Web Job will scale along with the Web App container.

In general, I would start with WebJobs and then move to Worker Roles if you find that your workload requires more than WebJobs can offer.

Q19: When shall we use Azure Table Storage over Azure SQL? ☆☆☆

Topics: Azure MongoDB

Answer:

- **SQL Azure** is great when you want to work with structured data using relations, indexes, constraints, etc.
- **Azure Storage Table** is great when you need to work with centralized structured data without relations and usually with large volumes.

I would always use azure tables as a much cheaper solution if:

- I perform table selects ONLY by PK (select on the property is slow due to the of the whole deserialization),
- I can live with a limited Linq set (Query Operators (Table Service Support),
- I don't need to *join* tables and perform complex queries *on the server*,
- I need horizontal partitioning "sharding" of my data (SQL Azure Federations is a step in that direction by Tables have PartionKey from day 0),
- Azure Tables are only cheaper than SQL Azure if the data access pattern is relatively light since tables have a per-transaction fee and SQL Azure doesn't.

Q20: What is difference between *Keys* and *Secrets* in Azure Key Vault? ☆☆☆

Topics: Azure

Answer:

The Azure Key Vault (KV) can store 3 types of items: (1) secrets, (2) keys, & (3) certificates (certs).

1. **Secrets** - provides secure storage of secrets, such as DB connection strings, account keys, or passwords for PFX (private key files). An auth app can retrieve a secret for use in its operation.
2. **(Cryptographic) Keys** - keys represented as JWK (JSON Web Key). Supports multiple key types and algorithms, and enables the use of Hardware Security Modules (HSM) for high value keys.
3. **Cert** - is a managed X.509 certificate, which are built on top of keys and secrets and add an automated renewal feature/auto-rollover.

FullStack.Cafe - Kill Your Tech Interview

Q1: How to define an *Environment Variable* on Azure using Azure CLI? ☆☆☆

Topics: Azure

Answer:

To do it using Azure CLI:

```
az webapp config appsettings set -n $webappname -g $resourceGroupName --settings
ConnectionStrings__Default=$connectionString
```

To set an environment variable `ConnectionStrings__Default` to the variable `$connectionString`

Q2: What is equivalent of a Windows Service on Azure? ☆☆☆

Topics: Azure

Answer:

There is no specific way to run your code in Azure. You have lots of choices, and which you choose is really up to you (and a matter of opinion). But, objectively speaking:

- Install your service as you always have, in a Windows Server VM
- Run your code, without the Windows Service wrapper, in a **VM** (either Windows or Linux, depending on language)
- Pull your core code out of the service, and run it within a web/**worker role** (cloud service).
- Run your code in a **WebJob**.
- Run your code in a Web App (you'd need to add some way to get to it, like a REST API sitting in front of it)
- **Azure Function** can be triggered by a timer and so like a windows service can be scheduled at a certain time of the day for example.

Q3: How is *Azure App Service* different from *Azure Functions*? ☆☆☆

Topics: Azure

Answer:

- An Azure function is triggered by an external event or a timer. It then executes the code of the function. When hosted on a consumption plan this execution is allowed to run for 5 or 10 minutes max. When you need a longer execution time you need to run it on an App Service Plan.
- An App Service can host any app you've created. Like a website (that runs continuously and doesn't need to be triggered before it starts doing something) or an API for example.
- Azure Functions consumption plan you pay per execution
- Azure Functions in an App Service (dedicated plan) you pay for the allocated hardware per minute.

So, for the integration scenario azure functions are a very natural fit. For websites an App Service might be the best solution.

Q4: What's the difference between_ Azure SQL Database_ and *Azure SQL Managed Instance*? ☆☆☆

Topics: Azure SQL T-SQL

Answer:

- With **Azure SQL Managed Instance**, you essentially get a full-fledged SQL Server that you can control any way you want, just like you would control a locally configured SQL Server. All the power and access and customization you want. With SQL Managed Instance, auditing happens at the server level, because, you are getting the full database server.

- With, **Azure SQL DB PaaS**, you are essentially getting a database service, so, you give up a lot of control. Resources dedicated to individual DBs like a container. They are grouped under an Azure SQL Server but that SQL Server is shared.

Q5: What Is *Azure Resource Manager*? ☆☆☆

Topics: Azure

Answer:

Azure Resource Manager (ARM) is an Azure service you can use to manage and deploy resources using an infrastructure as code paradigm. It enables you to provision, modify, and delete resources using a variety of features including access controls, tags, and locks.

When using Azure Resource Manager, there is some specific terminology you should be aware of. The most common terms include:

- **resource**—an asset that can be managed. Assets include virtual machines (VMs), virtual networks, databases, web apps, and storage accounts. Resources may also refer to tags, subscriptions, resource groups, or management groups.
- **resource group**—a container that groups together related resources. An Azure resource group enables you to manage multiple resources as a whole.
- **resource provider**—an individual service in Azure that you can create resources in. For example, Microsoft Storage or Microsoft Compute.
- **Resource Manager template**—a file that defines how resources should be deployed to groups, subscriptions or tenants. This file is defined in JavaScript Object Notation (JSON).
- **declarative syntax**—the syntax used for Resource Manager templates. It enables you to define how resources should be handled without having to know programming commands.

Q6: How can I test my ARM template before deploying It? ☆☆☆

Topics: Azure

Answer:

You can run the ARM test toolkit and “what-if” operation on your templates before deploying them. The toolkit tries to validate if your templates use best practices. If the test toolkit finds elements that require improvements, it provides warnings.

The what-if operation helps you see how the template will change your environment. It can help you discover unintended changes before deployment.

Q7: What is the difference between *Enterprise Application* and *App Registration* in Azure? ☆☆☆☆

Topics: Azure

Answer:

An **App Registration** is a way of reserving your app and URL with Azure AD, allowing it to communicate with Azure AD, hooking up your reply urls, and enabling AAD services on it. When you have an application that you are developing and want to integrate with Azure, you need to register your application in App Registrations, where you will configure your reply URL, logout URL, and API access if needed. When you register your application, Azure AD assigns a unique Application ID to it and allows you to add certain capabilities such as credentials, permissions, and sign-ons. The default settings allow only users from the tenant under which your app is registered to sign into your application.

The Enterprise Applications blade might be confused with App Registrations because the Enterprise Application blade contains the list of your service principals. However, the term **Enterprise App** generally refers to applications published by other companies in the AAD gallery that can be used within your organization. For example, if you want to integrate Facebook and manage SSO within your organization, you can integrate it from the Enterprise Applications dropdown in the applications blade. **Your own applications** will *also* be represented in the Enterprise Applications blade as Service Principals, which are instantiations of your applications in the tenant.

App Registration are basically the apps local to the tenant/organization in which they have been registered to generate unique application id. Enterprise apps blade shows global apps (belonging to other tenants) which can be configured and used within your tenant/organization.

The workflow is you create the App Registration (Application) in your tenant, which also creates the Enterprise Application (Service principal) in your tenant. Then when another tenant user wants to login to your app, they grant your app the permissions it requires and the Enterprise Application (Service Principal) is created in their tenant. This effectively mirrors your application in their tenant.

Q8: When should I use *Azure SQL* vs *Azure Table Storage*? ☆☆☆☆

Topics: Azure

Answer:

- SQL Database (SQL Azure) is relatively expensive for gigabyte of storage, does not scale super well and is limited to 500 gigs/database. To go beyond that, you'd need to partition your data. However, and this is very important, there are no transaction fees against SQL Azure and your developers already know how to code against it.
- SQL Azure supports transactions; table storage doesn't,
- If you have an existing application that uses SQL Server, SQL Azure provides a good migration path,
- ATS is capable of mega-scalability, it is dirt cheap to store but gets expensive to frequently access,
- ATS does offer transactions at the partition level,
- ATS is not a relational database (limited SQL is available, don't expect joins of any type, besides what you implement in Linq)
- Frequently accessed data that does not need huge scalability and is not super large in size should be destined for SQL Azure, otherwise Azure Table Services.

Azure Table's Good points:

- Strong point is its ability to store lots of little data; Azure table is based on Azure Blob, but is geared towards smaller data
- Much cheaper than Azure SQL Server
- Anytime you know both the partition key and the row key, the data access is very fast.
- `Entity transactions` are possible if you place two different "schemas" in the same partition key.
- Where the total row size is LESS THAN 980K (SQL Row)
- Where each property is LESS THAN 64K (SQL Column)
- It can act as a poor man's SQL.

Azure table's bad points:

- SQL is the weak point. Don't expect to use this on any large SQL table or you will suffer performance issues.
- Limited SQL is available, don't expect joins of any type, besides what you implement in Linq
- Azure Table SQL doesn't scale as well as its ability to store an infinite amount of data
- Anytime you don't specify both a partition key and row key in a WHERE clause, expect a slow table scan to occur
- Expect table scan performance to degrade as you add more rows
- Don't expect Azue Table queries to be fast as you add more rows
- Bottom line, if you're using Azure Table to act like SQL don't add lots of data. If you have lots of data (gigabytes), just don't plan on getting high-performance SQL queries against it. You will be saving money if you don't need those regular SQL features.

Q9: Explain what is the difference between *Block Blobs*, *Append Blobs* and *Page Blobs* in Azure? ☆☆☆☆

Topics: Azure

Answer:

Block blobs Block blobs are used to hold text or binary files up to ~5 TB (50,000 blocks of 100 MB) in size. The primary use case for block blobs is the storage of files that are read from beginning to end, such as media files or image files for websites. They are named `block blobs` because files larger than 100 MB must be uploaded as small blocks, which are then consolidated (or committed) into the final blob.

Page blobs
Page blobs are used to hold random-access files up to 8 TB in size. Page blobs are used primarily as the backing storage for the VHDs used to provide durable disks for Azure Virtual Machines (Azure VMs). They are named `page blobs` because they provide random read/write access to 512-byte pages.

Append blobs
Append blobs are made up of blocks like block blobs, but they are optimized for append operations. These are frequently used for logging information from one or more sources into the same blob. For example, you might write all of your trace logging to the same append blob for an application running on multiple VMs. A single append blob can be up to 195 GB

Q10: How would you choose between *Azure Table Storage* vs *MongoDB*? ☆☆☆☆

Topics: Azure MongoDB

Answer:

Azure Table Storage is a core Windows Azure storage feature, designed to be scalable (100TB 200TB 500TB per account), durable (triple-replicated in the data center, optionally geo-replicated to another data center), and schemaless (each row may contain any properties you want). A row is located by partition key + row key, providing a very fast lookup. All Table Storage access is via a well-defined REST API usable through any language (with SDKs, built on top of the REST APIs, already in place for .NET, PHP, Java, Python & Ruby).

MongoDB is a document-oriented database. To run it in Azure, you need to install MongoDB onto a web/worker roles or Virtual Machine, point it to a cloud drive (thereby providing a drive letter) or attached disk (for Windows/Linux Virtual Machines), and optionally turn on journaling (which I'd recommend), and optionally define an external endpoint for your use (or access it via virtual network). The Cloud Drive / attached disk, by the way, is actually stored in an Azure Blob, giving you the same durability and geo-replication as Azure Tables.

- if your queries are really simple (key->value) go with ATS.
- If you need more elaborate queries and don't want to go to SQL Azure, Mongo is likely your best bet.

Q11: *Cosmos DB* vs *Azure Table Storage* vs *Azure SQL Database*: what to choose? ☆☆☆☆

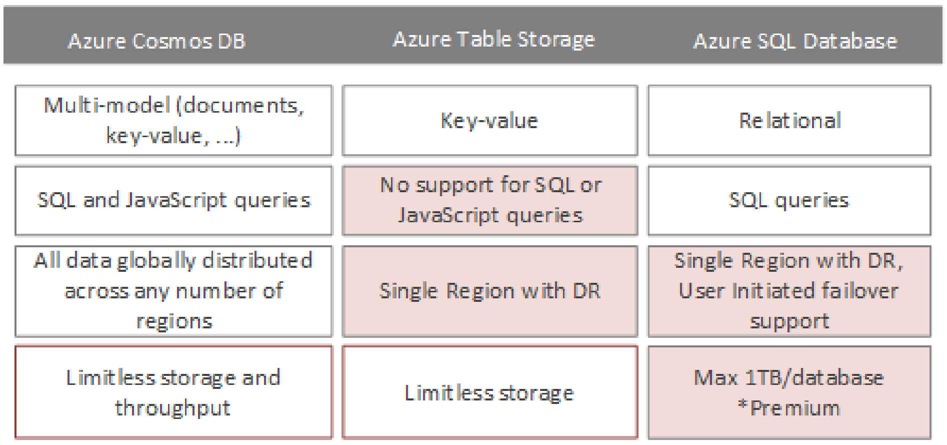
Topics: Azure

Answer:

Common attributes of Cosmos DB, Azure Table Storage, and Azure SQL Database:

- 99.99 availability SLA
- Fully-managed database services
- ISO 27001, HIPAA and EU Model Clauses Compliant

The following table shows the uncommon attributes of Azure Cosmos DB, Azure Table Storage:



Azure Cosmos DB	Azure Table Storage	Azure SQL Database
Multi-model (documents, key-value, ...)	Key-value	Relational
SQL and JavaScript queries	No support for SQL or JavaScript queries	SQL queries
All data globally distributed across any number of regions	Single Region with DR	Single Region with DR, User Initiated failover support
Limitless storage and throughput	Limitless storage	Max 1TB/database *Premium

Q12: Name some advantages and disadvantages of Azure CDN ☆☆☆☆

Topics: Azure CDN

Answer:

Advantage of Azure CDN

- Better performance and improved user experience.
- Decrease the load on Original server because files are delivered from the edge server (CDN).
- Only configuration required to enable feature.
- Robust and less maintenance.

- Easy to configure and from resource only.

Disadvantage of Azure CDN

- You need to spend around $0.10 per GB. It doesn’t seem much but comparing it with your regular hosting plan bandwidth, It is too costly.
- Storing some sensitive files on the CDN network may open potential security vulnerabilities. Because it does not matter what you store at CDN, it will be copied to all distributed servers.
- Using CDN, you are creating an additional “point of failure”. If the CDN network goes down you may lose website visibility.

Q13: What is the difference between *Azure Active Directory* and *Azure Active Directory Domain Services*? ☆☆☆☆

Topics: Azure

Answer:

- **Azure Active Directory** is meant to be a secure authentication store, which contains users and groups. AAD is a cloud-based identity management store for modern applications. AAD allows you to create users, groups, and applications that work with modern authentication mechanisms like SAML and OAuth.

- **Azure AD Domain Services** meant for If you need more than just user management, then it is possible to extend Azure AD to offer more AD based services using Azure AD Domain Services. Azure AD Domain Services enables you to use managed domain services—such as Windows Domain Join, group policy, LDAP, and Kerberos authentication—without having to deploy, manage, or patch domain controllers. An Azure AD DS managed domain lets you run legacy applications in the cloud that can't use modern authentication methods.

Q14: Compare ARM Templates vs Azure CLI for Azure deployments ☆☆☆☆

Topics: Azure

Answer:

Compare:

- **ARM templates** /*Azure CLI*
- **Declarative** / *Imperative*
- JSON** (JavaScript object notation) for defining ideal state — easier to read/understand objects and used to define what** / *Batch/shell for defining implementation — easier to read/understand order of events/logic and define HOW*
- **Can download template after manual creation of resources** / easier *understand and implement logic developers are familiar with for code efficiency, such as iterative code (for each loop)*
- **Potentially lots of lines of code and files** / *Potentially long lines of code*

Q15: Why are *Azure Resource Groups* associated with a specific *region*? ☆☆☆☆☆

Topics: Azure

Answer:

For compliance reasons, you may need to ensure that your data is stored in a particular region.

The main reason for specifying the location of a resource group is to specify a location for data/metadata for the deployment to be stored in. It also makes the API consistent (think of the paths in REST API calls) but the primary reason is storage during deployment.

The location of the resources in the group is independent/not related to the location of the group itself.