Azure Expertise and Questionnaire

What do you feel like your level of expertise is in Azure? Junior, Senior, Architect? Why?

My level of expertise in Azure is Architect. There are many reasons for this, but it's based on my career arc. My first 13 years in IT prepared me for Azure as I was in a variety of IT roles. I was focused on Data Center Operations which lead me to an Automation and Scheduling team. There I was lucky enough to work with senior mainframe engineers and to truly understand how processes and development worked. One of the engineers that I worked with, Rod Smith (who also happens to be my lead reference), was a seasoned developer who taught me how to learn. As a team, we worked together to automate a variety of processes at First Niagara. I learned best practices on how to develop and gained insight into what to do and what not to do in an IT role. The most important thing I learned was to build processes and programs thinking of the next person in the role. This leads to design patterns and practices that focus more on creating repeatable building blocks that are easy to understand and build upon rather than complex designs that may work but are not able to be maintained or well-understood.

My formal Azure experience began when I started my role in 2018. As a DevOps Engineer, I already had programming, scripting, and ETL experience. I previously worked in areas where I had to understand the end-to-end processes. I had the technical expertise to be able to both zoom in to specific technical problems as well as being able to zoom out and to see the whole picture.

My first role at my current company was to assist in the migration of 50 applications from on-prem to the cloud. This wasn't a simple lift and shift. Instead of keeping the existing architecture, in most instances we were modifying the architecture to utilize PaaS services in Azure including App Services and Azure SQL. I was brought on to be the sole DevOps Engineer originally, but that quickly expanded to not just DevOps, but also architecture. In order to understand what I needed to automate, I also needed to be included on the discussions of what we were doing from an architecture perspective. My first couple months were focused heavily on the creation of PowerShell scripts (which I had never used previously, but quickly picked up), build/release pipelines, and heavy work within Azure DevOps (VSTS). My predecessor was focused on getting pipelines up and running which lead to a lot of pipelines that could not be reused. My focus was to parameterize the pipelines so that they could be reused for a variety of jobs. My goal with any code that I write is for it to be parameter driven so that it can be reused and built upon.

As my role quickly expanded, I learned more about the various aspects and complexities of Azure. I felt like everything I was learning required me to learn something new. In my first year, it went something like this…I need somewhere to put the resources in Azure (Resource Groups). Well, where do the applications live? Those are the app services. How can I stand up those app services automatically? Oh, it's pretty easy to do with ARM templates and PowerShell. Once those services are created, how do they connect to on-prem? Oh, that's where Hybrid Connections come into place. Where can we store all the passwords for everything so it's not in code…well we have two spots, one at the app service level and another within key vault. Essentially each problem led to a different aspect of Azure that I needed to learn and in my first year it was mostly about App Services, Key Vaults, Azure SQL, and App Insights. To add to that, I helped design and maintain a PowerApps Application and became more familiar with Serverless designs including utilizing logic apps. Finally, to monitor everything, we had to register the applications with Application Insights which allowed our developers to see what was happening and where failures and issues were occurring.

As the migrations moved to a close, we moved more to a maintain and improve stage with the applications. This led to close evaluations of how the applications were performing, bug fixes, and improvements in architecture. I assisted the development team in backend improvements including query/LINQ optimizations and redesigns of the application databases. As our knowledge of the architecture and design patterns grew, so did my experience in Azure. In the next two years, I was exposed to storage accounts, azure automation, azure active directory, and function apps. As I previously stated, our focus was mostly on the PaaS offering, but towards the second year in my role we had an on-going effort to implement SAP in Azure which required my assistance. In this role, I was working with two other architects designing not just the environments but accounting for Disaster Recovery scenarios as well. I was tasked to write PowerShell scripts that would execute an automated DR plan in Azure Site Recovery. This project lasted six months and made me very familiar with networking architecture, VMs, and in general IaaS setup within Azure.

In early 2020, the team that worked on this project which was core to the company's success was recognized and I was promoted to IT Section Head. My day-to-day responsibilities transitioned to managing the Development Team (mix of on-prem and off-shore developers) and helping to design future-state architecture for corporate applications. This included following Agile and DevOps practices.

After a corporate reorg, this expanded to working with not just my own development team, but also other development teams around the organization. The goal was to provide enablement and to teach the application teams the best patterns and practices in order to build/modify/move their applications into Azure. This included weekly meetings where we'd review blockers to the processes moving forward and my team would assist in removing those blockers in order to ensure success.

After nearly trying for a year to push creation of a SRE team, I was told by the director that the organization was not ready for this, and we didn't have the funding. We did, however, have a great need to improve our Data Services space. That is when I began to bring my understanding of DevOps to Data.

That's where in July 2021, I began my role in the Data Services space. There were a variety of projects that were occurring across the organization that required some updates in the Cloud space. Two come to mind. The first was heavily involved in the IoT space. This introduced me to offerings I previously hadn't worked on including Azure Synapse Analytics, Stream Analytics, IoT Hub, Event Hubs, and TSI (Time Series Insights). My primary responsibility was to design an automated process to capture raw json telemetry dynamically (with no defined schema) and to store it for analysis. The solution required me to gain an understanding of Python and Spark and greatly increased my understanding of these offerings.

The second was the Enterprise Data Warehouse. The warehouse in its current state is more of an OLTP Warehouse than a true data warehouse. This led to investigations into more modern Data Warehouse designs where I explored Azure Synapse and Databricks. This is part of the current on-going effort to build up a true Enterprise Data Warehouse in the Cloud utilizing Azure Synapse which includes the Dedicated SQL Pool, Serverless Components, and Spark Engines.

Hopefully this insight into the arc gives you and idea of the various things I have worked on and gained expertise in.

Describe what your day-to-day Azure support looks like

I think the prior question does a pretty good job of explaining my current day-to-day. At this point, my day-to-day focuses on a mix of Powershell scripting, Azure Data Factory and Azure Synapse Analytics updates, and configuration and updates required for the data and reporting world. Right now, we are trying to establish best patterns and practices regarding both data and reporting. I am working with various business groups to establish what is needed for reporting and designing a solution utilizing Azure Synapse and Delta Lake.

In the past, I had additional responsibilities day-to-day including monitoring application performance, architecture design of various projects, and database monitoring and support. Please see above question to get a sense of the various things I have done in the cloud.

Describe a project you worked on that you are proud of, and why?

One of the application migrations to the cloud I had was not going so well. The application was a .NET application that was recoded to work in an Azure App Service and had an on-prem SQL backend. The customers of the application were upset because one of the key functions of the application would stop working randomly. This was an application that needed to be available 24/7/365 as it was critical to a business function.

We had the developers take a look at the application and they could not determine the root cause of the problem. We did of course have the application configured to log to Application Insights and while it did provide some useful information about the problem, it didn't paint the whole picture.

What was happening and what we didn't really know or understand at the time was that we were suffering from the hybrid model in which the application was architected. The developers could not find the root cause to the problem. In analyzing the logs, I discovered the issue.

The issue was simple. The application existed in the cloud and had to connect to not just one but several on-prem databases. When the application would perform this function (in this case a search), it was a linked server query. In this particular case, the linked server query was executed against a SQL DB that had high cpu/memory load. This caused the query to take an extremely long time to process.

There was really only one course of action. I had to migrate the data into an Azure SQL instance. This was a delicate process as I had to investigate the various queries that touched databases other than the application database. Once determined, I had my sources of data that I needed to migrate. The next step was to build an automated process that could perform this on a regular basis. The frequency was something that had to be determined from the end users. I worked with the Product Owner (on my team) and we had a face-to-face to understand the requirements. The end users didn't understand the difference between the frequency of the data refresh. In this particular case, the application needed to reference data from our warehouse (which was already at a delayed frequency) and then I had to migrate it. In the end, we determined that a daily process was sufficient to update the relevant data to the application database.

At first, I tried all the solutions I knew about to transfer data. From my previous roles as an ETL Analyst, I had a strong understanding of what was needed to do this and in particular I wanted my solution to focus on changes only. My first choice was to utilize tools for bulk data transfer (BCP utility) which could be automated on an on-prem VM and orchestrated with Azure DevOps. That seemed to work ok until it failed after 3 minutes. Not knowing entirely what the problem was, my next step was to build a custom application in .NET. That also was a success but had the same issue.

It seemed like an ideal solution would be to utilize something like Azure SQL Sync but I didn't have data ownership of that server and was quite confident that any attempt to install an additional service on the box would be denied.

This led to my finalized solution which utilized Azure Data Factory. What I discovered in a project parallel to this one that any query which didn't have a where clause was capped on the warehouse after 3 minutes. While I could have gone with any of the other solutions I had created, at that point I had grown used to and was quite proficient at a parameter driven approach utilizing ADF.

The end solution was great. The data transfer of approximately 5 million rows took about 40 minutes to process (not because of the data process, but more bandwidth constraints). I finalized the updates to the application, migrated the data myself, pointed to the new database in the cloud and created all the necessary indexes for performance. This was entirely automated in the pipeline that I had created (in ADF).

The result was immediately noticeable. Not only did the application respond instantaneously to searches, it never timed out. That process has been updated only once in the two years since the solution was created and the customers remain happy to this day.

Describe a project you worked on that went extremely wrong, and why?

One project that immediately comes to mind is an application that we were migrating to the cloud. A large portion of the details of this application had already been discussed prior to me joining the company, so I joined the project mid-implementation. In my opinion, it was a classic example of everything that could go wrong with a migration.

Let's start with the details. From a risk/importance factor, this application was crucial to a business unit within our company. It was one of if not the most important applications that existed in that business unit. It had a high number of users that performed specialized functions in the application. If you could for a second, imagine an application that was built over time, jerry-rigged with other processes and automation until the picture of how the end-to-end processes work was lost. The application was also written in Java utilizing Struts and was being modernized by the development team to account for the various cloud services (for example, rewriting the authentication library).

At some point in time, parallel development began with the application to bring it to a more modern framework (spring). Developers were doing development actively both with legacy code and the modernized code. In addition to that, some of the various pieces that were jerry-rigged in the past with services/bat files had to be rewritten to utilize some of the benefits of the cloud. We decided to write some of the services as Logic Apps.

Suffice it to say the transition of this application was a journey and not delivered on time. There were multiple issues with this project that we took as takeaways for the future.

1. There was no defined Business Owner.
2. This in turn led to no groups of qualified QA testers.
3. There was no code freeze on the legacy version.
4. The overall architecture and design was hard to test and implement in one step.

Our development team had several takeaways from this and we learned several key lessons in the transition to the cloud. The biggest one was probably to take more of an incremental approach and attempt to test in parallel to ensure that you are matching the requirements of the legacy system. Another key takeaway was to identify the business users/testers that were required prior to engaging on a future migration.

Are you available for on-call rotations?

My answer to this question generally is it depends. I'll be honest and say that I've been burned by this in the past. My first seven years in IT were nights. Then my next three years were days that had on-call rotation in which it was a team of myself. If transparent about the ask, I may be open to it, but I'd like to know ahead of time what I'm signing up for.

My general preference is to leverage my expertise in DevOps following SRE principals so that on-call rotations are not necessary in the first place. I'd like the solutions the team builds to be self-sustained and require very little outside intervention.

Walk through how to do cloud migration

This is a loaded question. There is no straightforward answer to this question and the real answer is…it depends. Let me try to break it down into the various scenarios to give you an understanding of what I mean.

First, let's focus on the goal. Is the goal to migrate servers to the cloud? Perhaps it is to migrate applications? How much time to do you have to achieve this goal? Do they need to be highly available? Is DR (disaster recovery) a concern? Are these internal functions to the business or external? What is the classification of the data that you are working with? What is the size of the team that you are working with? Is this being done in parallel with existing processes or has there been an agreement to pause any changes to an environment until this is done? These are not all of the questions that are out there, but I will also focus on one more…how much do you want to spend?

In general, there are three types of strategy you can focus on…IaaS (infrastructure as a service), PaaS (platform as a service), and SaaS (software as a service) depending on exactly what you are trying to do. When many people first run into the cloud, they are primarily thinking of IaaS. Let's do some data center consolidation, not have to pay for hardware and data center costs and move it to one of the big 3 (AWS, Google Cloud, and my personal preference Azure).

I'm going to focus on one particular instance for IaaS which is the migration of a server to the cloud. These are the questions of relevance for this migration…

What OS is the server? Linux/Windows?

What networking needs to be in place around the server? Virtual Networks, Security Protocols, Firewalls, Isolation, Clustering

What is the necessary availability of the server? What uptime is desired?

Is DR required?

What are the specs of the server? CPU/RAM/Disks?

What software is required on the server?

After this list of questions is answered, the next question is do you have any type or desire any type of automation around the creation and deployment of this VM. It doesn't make much sense to always have to click through a portal or have a manual process each time a new VM is needed…This goes back to one of the questions listed above…how much time do you have, what team are you working with, and how much can you spend?

Perhaps you want to build all your servers and cloud environment with IaC (Infrastructure as code) and use tools like Terraform/Ansible, Powershell DSC or just plain old ARM templates. Perhaps you are using a tool like Azure DevOps to handle your deployments. I'd say making sure you have tools like this available are crucial for future success as you don't want every build to be a one-off. You want to be able to create building blocks that the team and the organization can use in the future.

Assuming you do have the team, funding, etc… You would then have to script out the environment in Terraform including all the various resources (VM, VNET, ASGs (Azure Security Groups), Disks)…It would be a matter of inputting those variables for the build (as parameters…), having a Service

Principal that had access to perform the necessary actions, and kick off the job. After the infrastructure is built, the next step would be to register to the domain (either in the cloud or on-prem depending on the desired configuration), install the company software (antivirus, monitoring, etc.…), and then last but not least migrating the data. There are several built in tools to perform this action.

Let’s run through the lesser-known cloud migration which is more of a “lift and adjust”. Azure in particular has a growing list of services that are offered as platforms. The benefit of utilizing these services is ease of use, high-availability, and more often than not, lower cost (both in flat charges and savings in maintenance). Let’s take the simplest of these examples…an Azure SQL instance.

Let’s say your domain has several MS SQL databases that need to be migrated to the cloud. It’s actually been quite challenging over the years because your DBAs and System Engineers have had to maintain a large number of servers/databases keeping them up to date, applying patches, software updates, etc… You can’t count the number of times you’ve moved from SQL Server 2008 -> 2008R2 -> 2012 -> 2014 -> 2016 -> 2017 -> 2019. Maybe you’ve skipped a couple iterations because it was too much work to get your engineers/DBAs to update all of the systems. You know at some point they will go to end of life, but the cost to keep them up to date and the time….

By moving to Azure SQL, you don’t need to worry about that. Microsoft handles the servers, infrastructure, patching, and updates to the databases. There is no longer a version to worry about in the future so it will always stay up to date without any additional intervention. The best part, it supports about 99% of the features that a standard version does so more than likely it can be used. There are some ancillary services that can be set up of course including monitoring and security packages that can all be purchased as part of the deployment. You can use the same tool you used before…Terraform, Ansible, Powershell…and the deployment takes perhaps a couple minutes. The only necessary step at that point is to bring the application down, perform a backup of the database/s, and then use a tool to transfer the data (I’d probably use SQL Azure Data Sync or ADF (Azure Data Factory) depending on the scenario).

Hopefully these various scenarios have given you a preliminary understanding of how to migrate to the cloud.

Explain the journey and the challenges of a cloud migration

The road to a cloud migration is a long journey. There are several factors that influence how long the journey takes. One factor is cost and investment of the organization. What I mean by this is I've been in organizations that have different approaches. One is the top-down approach from Senior Management stating that an organization needs to migrate to the cloud. This typically comes with funding and hopefully also business buy-in which helps to speed up the process. This might include training and bringing in additional folks to help assist in the cloud journey.

The other approach I've seen is bottom-up and this is the teams themselves driving the need to be in the cloud. This doesn't seem to work as well, because a team itself can quickly transition, but it doesn't align other teams to the approach (obviously this is dependent on the size of the Company). There is also the inherent factor of cost which then becomes difficult to manage.

I'd say the quickest approach is "lift and shift" utilizing the cloud as IaaS (Infrastructure as a Service). While it's true that you need to have several teams involved in this endeavor (security, networking, and system engineering), this approach is probably the most straightforward because most folks are already used to VMs and it's not a hard transition to move into a bigger cloud. The downside to this approach is cost because this is probably the highest cost in the cloud.

The next approach "lift and adjust" has much more potential for cost savings and benefits. You can utilize the PaaS offerings in the cloud which provide higher availability and typically lower cost. This also reduces long-term maintenance costs in the organization as well. The difficulty here is typically time to implement. With adjustments, the existing processes are going to have to be well-understood and then adjusted and matched to the proper services in the cloud.

I would say the two biggest challenges involved in the cloud journey are keeping track of the overall costs and training employees. The cloud is really as expensive as you utilize it. Within the cloud, you can stand-up high-powered machines that can cost thousands of dollars per hour. Or you can reduce cost with low costing Azure SQL databases depending on your specific needs.

Training and learning are also crucial with the existing employees that you have. You need to make sure that folks are learning about the new technologies, how to interface with those technologies, and how they might change over time. As part of the journey, everyone needs to learn and as new members of the team come in, they need to be integrated with the best patterns and practices of the organization.