SAND2022-13755C

Sandia National Laboratories

Exceptional service in the national interest

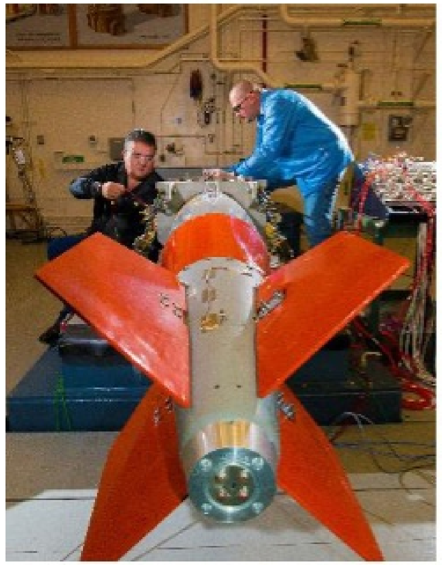
Building Capabilities in the Sky

NWM Cloud Project Lessons Learned

ERICA GRONG JOHN SEBASTIAN
RICHARD SCHETNAN WALTER WALKOW

SANDIA NATIONAL LABORATORIES
DATA SCIENCES DEPARTMENT

NLIT: October 18, 2022



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Agenda

- NWM Cloud Project Overview – 3 minutes
  - Erica Grong, IT Project Manager, Data Sciences
- NWM Cloud Environment – 15 minutes
  - John Sebastian, Solutions Architect, Data Sciences
- NWM Software Engineering – 15 minutes
  - Richard Schetnan, Software Systems Engineer, Data Sciences
- NWM Data, Databases, and Validation – 15 minutes
  - Walter Walkow, Solutions Architect, Data Sciences
- Q&A – 12 minutes

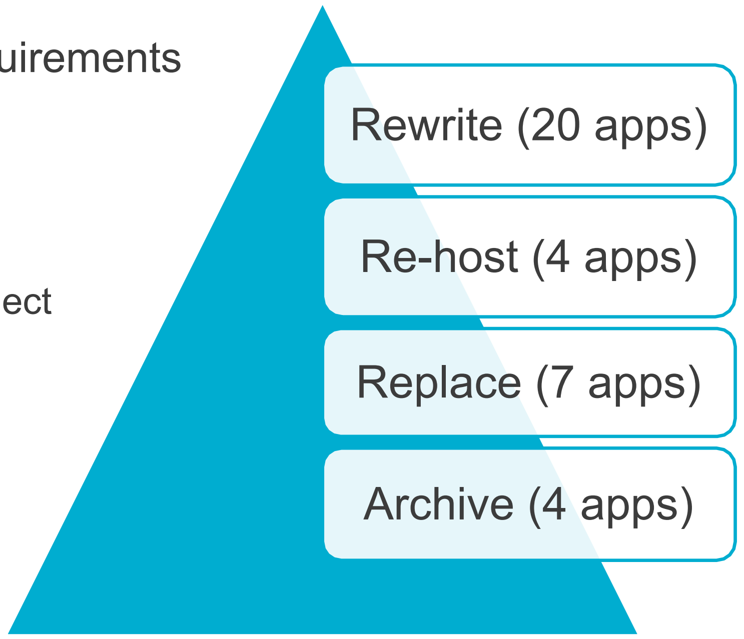
NWM Cloud Project Overview

Erica Grong

Nuclear Waste Management (NWM) Project Scope

Challenges

- Applications and content currently housed at DOE Office of Legacy Management on outdated hardware and software platforms.
- Complex or missing requirements
- Considerable effort to reverse engineer the applications to establish baseline requirements
    - 35 applications consisting of ~73TB of data files and content
    - Cycle Time = 4 years (from when baseline cloud services were set up to final go-live)
    - Division 8000 Subject Matter Experts (10-12)
    - IT Project Team size = 39 team members (approximately 14 FTE) – at height of the project
        - Cloud Administrators (authentication, account management, provisioning, DevSecOps, etc.)
        - Database Engineers
        - ISSM, ISSO, Cyber, 3rd party assessor, Incident Response
        - IT Project/Program Managers
        - Network Administrators
        - O365 Administrators (Exchange, SharePoint, O365, PowerApps, etc.)
        - Reporting/Data Analysts
        - Business Systems Analyst
        - Software QA/Testers
        - Software System Engineers
        - System Administrators



NWM Project Stakeholders & Partners

DOE Stakeholders

- DOE Office of Nuclear Energy (NE)
- DOE Office of General Council (OGC)
- DOE OCIO - Authorizing Official (AO)

Sandia Partners

- Division 8000 – Sandia Program Owner
- Division 9000 – IT Development Partner

Project Management Lessons Learned

Agile Methodologies

- Each application that we rewrote had its own Scrum development team
- Sprint lengths varied by team: anywhere from 1-3 weeks
- We had a COTS Team and an analysis/requirements team that was Kanban

Toolset

- Azure DevOps was used for everything: code repositories, wiki documentation, automated deployment, and Scrum /Kanban boards for task tracking. Also used Portfolio Plan add-on to do release planning

Team Skillset

- This project was largely learn-as-you-go for our project team
- Microsoft Workshops/Seminars that were available through Premier Support were incredibly useful but difficult to schedule just-in-time

Procurement

- Build vs. Buy vs. Rehost Analysis
- Probably best contract type to use would ideally be:
  - Time & Materials with Variable Scope and Cost Ceiling
  - Phased Development
- Try to coordinate the POs to all coincide with Sandia fiscal year

NWM Cloud Environment

John Sebastian

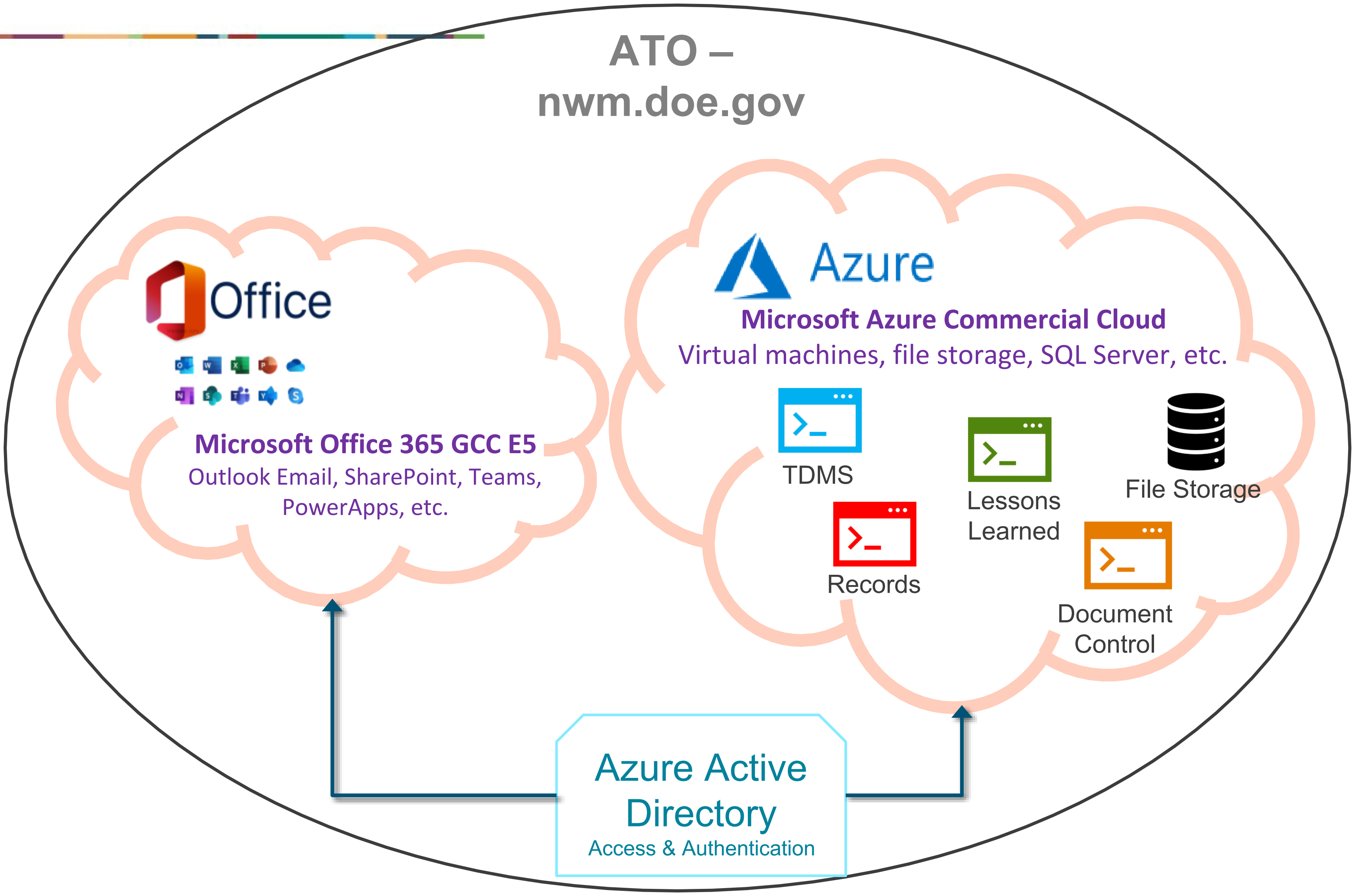
Commercial Cloud vs. Government Cloud

- In 2018, the NMM Project team invited the two largest cloud vendors, Microsoft and Amazon Web Services to make presentations to the team about their services and to show us how they could meet the needs of the NWM Project.

- It was clear from these presentations that the **Commercial Cloud from both vendors had many more available services and resources vs. the Government Cloud offerings**.
  - A key requirement for the NWM Project was that any Cloud Vendor we chose had to have the services be **FedRAMP certified at the moderate level**.
  - Not only did the Commercial Cloud offerings of both vendors have many more services and development resources devoted to adding and improving those services but also the number of FedRAMP approved services was growing at a faster pace

- **NWM chose Microsoft Azure Commercial Cloud**
  - Azure offered more FedRamp services at the time and we could run Office Apps as a service

NWM Cloud Environment


ATO –
nwm.doe.gov
Office
Microsoft Office 365 GCC E5
Outlook Email, SharePoint, Teams,
PowerApps, etc.
Azure
Microsoft Azure Commercial Cloud
Virtual machines, file storage, SQL Server, etc.
TDMS
Lessons
Learned
File Storage
Records
Document
Control
Azure Active
Directory
Access & Authentication

NWM Cloud Environment

Azure

Major Services that NWM Has Deployed

- **Compute**
  - Virtual Machines
  - Containers
  - Appliances
- **Data**
  - Databases (MS SQL, PostgreSQL)
  - Storage (in a myriad of shapes and sizes)
- **Security**
  - Vulnerability scans (VM, DB, Web Apps)
  - File Integrity Monitoring
  - Security Baseline scanning and alerting
- **Networking**
  - Virtual Private Cloud
  - Virtual LANs
  - Network Security Groups
  - Front Door/Web App Firewall
- **User Management/Identity**
  - Azure Active Directory
- **Version Control, Project**
  - Azure DevOps Boards
  - Azure DevOps Git
- **Monitoring**
  - Log Analysis
  - Microsoft Sentinel
  - Microsoft Defender for Cloud
  - Performance prediction and insights
  - Automation
- **Office 365**
  - Email
  - Microsoft Office
  - Power Platform
  - Sharepoint
- **Automation**
  - Patch Management
  - Deployments
  - Pipelines with Gates

IaaS / PaaS / SaaS

- **IaaS – Infrastructure as a Service**
  - Essentially Virtual Machines in the Cloud
  - Can be deployed/destroyed/resized through code or via a web portal
  - Pay as you go, pay for what you use, only pay for CPU while running
  - Customer manages the OS and anything deployed on the host
  - Cloud Vendor manages the hardware/infrastructure
- **PaaS – Platform as a Service**
  - Typically a hosted service that assists with development efforts
  - Examples include: databases, DevOps
  - Customer is responsible for how they use the service, what they put into it
  - Cloud Vendor is responsible for the infrastructure, backups, upgrades, patches, availability

IaaS / PaaS /SaaS (cont)

- **SaaS – Software as a Service**
  - Typically these are turn-key applications fully managed by the Cloud Vendor
  - Most often these applications are utilized via web browsers
  - Examples include: Webmail, O365, Web Conferencing, Cloud Storage drives (DropBox, OneDrive etc.)
  - The customer is responsible for what they put into the service and how they secure that data
  - The Cloud Vendor is responsible for the infrastructure on which the apps run, patching, upgrades, availability

How is NWM Using Infrastructure as a Service (IaaS)

- **Install Tools for DBAs:**
  - Our DBAs use specialty tools to manage databases in the environment.
  - We deployed a Windows virtual machine to install and run these tools from which they manage our databases.
- **Run old software:**
  - The NWM project had to deal with a variety of older software technologies (examples Lotus Notes, Cold Fusion)
  - We had a requirement to be able to read old email from the original Lotus Notes system and convert Lotus Notes application functionality to a modern development and execution platform (Power Apps)
- **Run SQL Server Reporting Services (SSRS):**
  - The original project had many reports written to run with SSRS. The team decided to continue to utilize SSRS to run these reports instead of converting them.
  - SSRS was installed onto an IaaS host.

How is NWM Using Infrastructure as a Service (IaaS)

- **Administration:**
  - The Cloud Administrators needed a host on which to run tools for managing the nwm.doe.gov domain.
  - We also use the administration server to apply upgrades with an Ansible Tower installation.
- **Appliances:**
  - The NWM team had the need to deploy a Proxy solution and an Ansible solution.
  - We deployed “appliances” from the Azure Marketplace to serve these needs
- **COTS:**
  - Numerous original applications were Commercial Off the Shelf products.
  - We needed virtual machines to run new versions of these COTS products (if they still exist) or run newly purchased products to replace the original functionality
- **Development Work:**
  - We used some virtual machines for software development efforts

How is NWM Using Infrastructure as a Service (IaaS)

- **Virtual Desktop:**
  - NWM is expecting to have a variety of users from different organizations using the applications.
  - For security purposes and control of the user’s desktop environment, NWM deployed the Azure Virtual Desktop service
  - This service provides a consistent Windows 10 desktop environment for end users to connect to. The hosts to which the AVD service connects users are IaaS virtual machines. Additional landing hosts can be deployed from a generalized image and added to the pool of landing hosts when the number of users grows.

How is NWM Using Platform as a Service (PaaS)

- **Azure DevOps:**
  - A fully managed Development environment with many tools for developers to use.
  - Microsoft manages all infrastructure, software, upgrades, patching and backups.
  - Provides a variety of development and operations type services to end users in an Agile style.
  - NWM uses many of the Azure DevOps features including: Git repositories, Wiki, Scrum and Kanban boards, as well as release planning.
- **Azure PostgreSQL database:**
  - A fully managed instance of PostgreSQL server. We use this for a specific application.
- **SQL Managed Instance:**
  - A fully managed SQL Server.
  - Microsoft manages all infrastructure, software, upgrades, patching and backups.
  - We have both development and production SQL Managed Instances deployed.
    - Note: Backups of SQL MI cannot be restored to on-prem SQL Server
- **Azure SQL Database**
  - Used in developing a specific application, not in production

How is NWM Using Software as a Service (SaaS)

Office Suite/Power Platform/Sharepoint

- One of the main objectives the NWM team decided on early on was to utilize as many vendor managed services as possible. With O365, there is no need for NWM team members to manage Exchange Servers or any of the O365 products. Microsoft does this for us.

- The NWM Project utilizes Microsoft's O365 Platform for Office apps (Outlook, Excel, Teams, PowerPoint, OneNote, OneDrive, Word, etc.) , SharePoint, and PowerApps.

Identity and Authentication

Azure Active Directory (Azure AD):

- NWM uses Azure AD Premium P2.
  - The default domain provided by the Azure AD tenant was nwmdoe.onmicrosoft.com.
  - Premium P2 allowed NWM to create a custom domain (nwm.doe.gov).
- Some of the Azure AD features we rely on are:
  - Multi Factor Authentication (MFA): All NWM users must pass an MFA challenge to gain access to the environment.
  - Conditional Access policy features. Provides a range of conditions to check and can allow or deny access. The policy can be tailored to specific groups and can exclude users or groups.
    - We use Conditional Access to
      - Ensure MFA is used
      - To ensure that that we are blocking non-US IP ranges from connecting
      - To block access from applications that perform legacy authentication

Lessons Learned

- If you require FedRAMP authorization, make sure that ALL of the services you require are FedRAMP certified
- Be sure that your cloud environments can communicate if you have that requirement. The NWM Environments were O365 Government Commercial Cloud (GCC) and Azure Commercial Cloud. Initially O365GCC could not reach databases in Azure Commercial.
- Be prepared for constant learning and trying to keep pace with changes. You will never know everything.
- Knowing a little about a lot is preferred to knowing a lot about a little.
- Be aware that you will pay for more services in the Cloud than you might when on-prem at Sandia.
- Be careful of resource drift.   It is quite easy to deploy things and forget them, especially when experimenting and researching.

Lessons Learned (cont'd)

- **Keep a careful eye on costs.** Consider using guard rails to prevent end users from deploying large servers or other costly resources without prior approval. Azure Policy is one example of a guard rail type service.
- **You are responsible for the security of everything you put into the cloud.** Having network engineering, database, web development, system administration and cyber skills are essential.
- **For resources that you expect to live for long periods of time, consider reservations to reduce cost.** Also consider turning off machines when they are not needed. You always pay for storage but consider tiering (cool/archive)
- **Make sure you are using a vulnerability assessment solution.** NWM utilizes a third party platform called Qualys and an Azure service Microsoft Defender for Cloud. NWM was not limited as to what services we could experiment with the exception of the service requiring FedRAMP certification

NWM Software Engineering

Richard Schetnan

Environments – Containerization

App Services

- Docker deployed onto a VM behind an IIS server
  - Sand boxed environment
  - Allowed using apps built for containerized environments with app services without additional work
  - Integrated OAuth/OpenID authentication flow at IIS server using managed identity in Azure AD
  - Our solutions:
    - Hasura
      - Originally on ACS, Hasura had an ARM template available
      - Moving to app service simplified integrating with the database, authentication, and other services without introducing the complexity of multiple containers, orchestration, sidecars, etc.
      - ACS was EOL and not recommended, AKS was considered but not chosen due to its complexity / tight timeline
    - Doclib
      - Rust server providing services for accessing and manipulating blob storage data
      - Rust runtime not provided in app service virtual machine environment, docker allows setting up and configuring any runtime on app services
      - FaaS was considered as an alternative but ruled out due to the need for long running operations
    - CI/CD
      - docker-in-docker strategy that allowed sand boxing build environments, easily cleaning up build artifacts / storage usage, restricting age system privileges

Environments – Orchestration

- Main options: Azure Container Service, Azure Service Fabric, Azure Kubernetes Service
  - ACS is EOL
  - ASF highly specific to Azure / Windows solutions, mostly eclipsed by the release of AKS
  - AKS generalized cloud agnostic orchestration solution
- Our solutions
  - Deploy application sets and integrate with azure resources using IaC
    - ARM, Terraform, Ansible
    - initiated manually or from CI/CD deployment pipelines
    - scale up/out using deployment configurations and Azure Slots (app services) or via terraform deployments creating new virtual machines
  - Lacked flexibility
    - service registration / discovery
    - application dependencies (ex. a needs b, both or deployed together, a may start before b is ready)
    - dynamic scale up/out
  - AKS was considered for some applications and would provide a more robust solution
    - chose not to utilize AKS due to development time constraints and increased complexity
    - increased flexibility / scalability was not a requirement for our customers

CI/CD – Azure DevOps

- Hosted source code repositories
- Hosted private npm artifacts
- Build configurations
  - triggered for changes on development branches, restricted merging into master branch
  - public build agents
    - provided by azure, specific runtime environment
    - outside the azure tenant, can run into network issues when targeting services, especially when a service is not externally accessible
  - private build agents
    - hosted on our vms, customized runtime environments
    - within our virtual network, simplifies nsg requirements when deploying artifacts to azure services
- Deployment / release configurations
  - trigger build and tag release
  - manual and automated, targeting azure services
  - integrated with azure service configuration, authentication, keystores
  - stores build artifacts
  - rollback support

CI/CD – Azure DevOps

Arm / terraform templates

- mostly run manually in our solutions
- could be fully integrated into the build chains (at the cost of development time) to provide more flexibility and ensure resource dependencies are resilient to redeployment of infrastructure
  - ex. If an app service were reconstructed resource ids would change, this would break the dependent builds/deployments until the CI/CD configuration is updated with the new resource ids. Setting up arm / terraform as a stage in the build would ensure the infrastructure exists and allow injecting the resource ids into the next step, thus removing the need to assume a resource already exists with a given id.

CI/CD – AAD/RBAC

Allowing DevOps to create and deploy services to Azure requires high level sensitive permissions

- Configure DevOps agent AAD service principle
  - Create the service principle in Azure then configure DevOps to use the identity
  - Requires CRUD permissions for desired services and scopes
  - Give DevOps agent permission to create additional service principles
    - Requires elevated permissions; creating credentials in AAD should be kept as restricted as possible
    - Can be scoped to a restricted set of permissions
  - If the DevOps agent does not have permission to write to AAD
    - An elevated user can create the required service principles, then CI/CD may use the existing service principle
      - Requires manual step when deploying new services reducing the resilience of the pipeline
      - Does not work with built in features such as the app services deployment center and managed identity providers

Our solution was to allow DevOps to use elevated permissions for specific resource group only, ie. a new identity could be create within specific-group-1 but not within global-scope-1. This enabled automated operations without granting global admin privileges to the agent.

Application Development – Networking

- Our environment
  - Public and private SaaS services, restricted access set by policies and NSGs
  - Private VNet hosting isolated virtual machines, restricted access set by policies and NSGs
  - Azure DevOps - not hosted within the azure tenant, ingress traffic to SaaS / VNet seen as coming from an external network
- Traffic between SaaS and VNet
  - Does not work by default in a restricted environment
  - Policies must be configured on the relevant NSGs to allow ingress / egress traffic for the specific network, machines, and services that will be talking
    - Ex. An app service dependent on managed SQL Server hosted with the VNet must configure the NSG to allow talking with SQL managed server; this will open ports only for the relevant service, on the specific machine, coming from / going to the app service resource group
- Traffic from DevOps to private SaaS resources / VNet
  - Public agents are blocked
    - Can work around this issue, however, its a tricky problem as the NSGs must allow the public ips of the agents (a domain name was not available) and those ips rotate on a weekly basis. A scheduled script was required to pull the ip addresses and update the relevant NSG policies. There was not an official solution available from Microsoft at the time.
  - Private agents
    - Deployments from the private build agent could be allowed through NSGs as the artifacts are hosted within the VNet. This is a fairly brittle solution as it requires restricting the machines that can build / store the build artifacts.

Application Development

Rehosting vs. rewriting advantages/disadvantages will be highly specific and very for each project.

Neither is a quick win, and depending on the underlying application, will require a significant amount of work, especially when working with legacy applications utilizing outdated languages, libraries, protocols, and security standards.

Application Development – Rehost

Application (CDIS / Doclib)

- Old java server/client web application
- Hand-rolled solutions (did not leverage J2EE or other libs)
- Outdated security practices, leveraged mix of http/https pages
- Depended on oracle database
- Depended on other applications and server file system resources

Solution

- Restricted network access to the application
- Deployed behind IIS server as reverse proxy
  - OAuth integration
  - Forced https
  - Tied into Azure SaaS monitoring, logging, and security services
- Migrated database from Oracle to Managed instance of SQL Server (licensing, consistency)
- Replaced file system resources with injected configuration and Azure Blob Storage
- Replaced system cron jobs with SQL Server scheduler
- Resolved various application issues
  - Deprecated html features no longer supported in modern browsers
  - URL / certificate updates --> hard-coded to injected from azure configuration / keystore
  - Switching to Https everywhere caused routing / application logic issues
  - Application dependencies --> other apps no longer existed or were moved to azure services
  - Updated authentication to recognize token passed by IIS reverse proxy

Rehost Application - Lessons Learned

- Private Vnet infrastructure can be used to isolate dated applications, provide legacy runtime environments, and reduce attack vectors
  - Leveraging / integrating with cloud native solutions will have increased complexity
- Need to integrate legacy authentication and authorization with AAD or opt out of using the same account
  - Integration is required to tie into AAA / RBAC, service-to-service OAuth flows, logging and monitoring, etc.
  - This can be accomplished with an Azure App Services reverse-proxy server
- Features will likely need to be rewritten / added due to changes in operating systems, browsers, protocols, etc. and deprecated features
  - Ex. http to https, flash, cors, etc.
  - Upgrading the language, frameworks, and libraries will likely take a significant amount of time. Using the existing dependencies and runtime will be more expedient, however, developers will likely need to learn the outdated approaches and hand-roll custom solutions to make required modifications.
- Determining Scope / level of effort can be challenging
  - Originally looked at the code base to determine the scope of the project. This inaccurately represented the level of effort as we later found there were dependencies on other applications, server resources, and scheduled cron jobs. The true complexity of rehosting an application may not become fully apparent until later, especially when documentation and/or original developers are not available. For this application, the additional dependencies were discovered during runtime testing.

Application Development – Rewrite

- Application (TDMS)
  - Old C client/server application web application
    - nonstandard version of c, poorly documented
  - Multiple applications served through main portal
  - Integrated with multiple other applications
  - Utilized scheduled jobs and system level resources
  - Ambiguous / missing requirements, fleshed out iteratively throughout development process
- Solution
  - React SPA (fabric-ui, graphql, msal), hosted on app services
  - Hasura server (generated graphql endpoints), hosted on app services
  - SaaS postgres database, migrated old data/schema, cleaned data during development
  - Implemented OAuth / OpenID flow using AAD
  - Integrated with various applications over http, OAuth for zero-trust permissions across services
  - Integrated with Blob Storage over http for storing/retrieving large files and binary data

Rewrite Application – Lessons Learned

- Compared to rehosting
  - Modern language, framework, libraries
  - Met modern security standards and allowed zero trust architecture using OAuth / OpenId with AAD
  - Allowed cloud first approach leveraging provided services
  - Significant development time, especially with unknown/changing requirements
  - Tying into cloud services was an advantage instead of an impediment.
- Ecosystem changes rapidly
  - Multiple Azure APIs changed - old approaches were deprecated and updates were required over the course of development
  - Downstream APIs were impacted by Azure changes (ex. msal)
- Lacking requirements and reverse engineering an old application made projecting scope especially challenging. Architecture and stack choices needed to allow for high degree of flexibility and re-engineering as requirements often changed.
- Generated GraphQL API provided a very high degree of flexibility to the data layer while also reducing backend development time. However, this approach required heavier application code to retrieve and structure entities, resulting in more frontend work curating and manipulating data.

Application Development – Skillsets

- Devops
  - CI/CD: repositories, builds, deployments, releases, controls
    - Azure specifics if using Azure DevOps, ie. azure-cli.yml structure
  - IaC: ARM, Terraform, Ansible, etc.
    - Alternatively the GUI can be used, however nothing will be reproducible, teardown/standup will be difficult
    - ARM templates can be generated from an existing configuration; this does not handle everything however, ex. configuration injecting a certificate / password from a keystore will be replaced with an empty property in the generated template
- Networking
  - Setup / troubleshoot NSGs, enable/disable traffic policies, trace routing problems
    - tcp/ip stack (layer 4-7), dhcp, subnets, firewalls, public vs. private addresses, routing, proxy/reverse-proxies, etc.
- Authentication flows
  - OAuth / OpenId
    - Specific flows for Azure, there are many, each with trade-offs
    - MSAL libraries
    - AAD / LDAP
  - RBAC
  - zero-trust architecture

Application Development – Skillsets

- Containerization / Orchestration
  - VMs vs. docker
  - hot vs. cold services
  - ASF, AKS
- SaaS, FaaS, PaaS, VMs
  - services available, trade-offs, when to use each
- Monitoring & Logging services
- Data Services: SaaS, managed, blob, full text, graph
- Distributed architecture: scale-up, scale-out, stateless vs stateful, microservices, function services

Application Development – Lessons Learned

- Had to document and proceduralism the pipeline and repo work
- Underestimated time estimates
  - Developing / configuring rewrite stack using cloud native approach
  - Onboarding / training team members with required skillsets
  - Hidden work, missing requirements, scope creep
- Iterative rework was required in multiple areas as team gained expertise and requirements were fleshed out
  - CI/CD pipelines, Azure services, UAT changes, etc.
- We had to upgrade our pipeline – so be wary of extra costs here
- Training opportunities that worked well
  - Microsoft Premier training workshops
  - Senior/Junior mentoring
  - Pair programming
- Code quality measures and training that did not meet expectations
  - Code reviews, quality gates, automated testing
  - Time and resource constraints, diverse experience levels, and shifting requirements resulted in these approaches causing more issues than they resolved

COTS lessons learned – Walter Walkow

Spend sufficient time on COTS IT requirements & considerations

- Evaluate Database requirements (industry supported DBMS, Use of Master DB or DB Catalog?, etc)
- Review system architecture (old/unsupported components, application software, security, neetc.)
- Insist on obtaining DB Design (unorthodox DB designs make migrations difficult)
- Review migration methodology early
- Determine if COTS package is supported in cloud environments
- Document responsibilities (who & what) for installation, migration as part of contract
- Avoid contracts where the vendor insists on performing installation, migrations
- Review IT documentation (Installation, migration, upgrades)
- Insist on delivering development and production environments are part of the contract
- Identify specialized skills required and availability of training
- Provide a final review to customer project manager (allow IT review to provide a basis for COTS rejection)
- Obtain profile and evaluations from Gartner if available (and/or previous customers).

COTS lessons learned – Walter Walkow

Ensure Azure Services necessary for COTS deployment are available

- Azure Virtual Machines are absolutely necessary to support software installation
- Network Virtual networks/subnet permissions to support access to required services (database, etc.)

Considerations for COTS in the Cloud:

§ For COTS tools especially, you will need to utilize containers and IaaS and PaaS

- The more COTS tools you have, the more flexibility you will need in IaaS and PaaS
- Make sure your COTS product is compatible to run in a public cloud/virtualized environment
- Make sure you select the correct license type (named user vs. floating)
- Software – what will run in the cloud, what does the tool install locally that you need to account for
- Thoroughly vet (run scans, understand architecture), of COTS tools to ensure no problems later

NWM Data, Databases, & Validation

Walter Walkow

Data Files and Content

Our Challenges

- Content originated in file servers located in a site with only an internal network (no outside connectivity)
- Files were both on Unix and Windows servers
- Approximately 165 million files and ~73 terabytes of data had to be copied
- Provided data and application code that was over 10 years old, some dating back to 1990s.

Data Files and Content

Our Solution

- ➢Data Box devices easily move data to Azure when busy networks aren’t an option
- ➢All data is AES-encrypted, and the devices are wiped clean after upload in accordance with NIST Special Publication 800-88r1

➢Data Box – Provides 80 TB of usable data. Supports standard NAS protocols (SMB/NFS). Weighs 40 pounds.

➢Data Box Disk – Provides 7 TB of usable data. Format is an SSD that supports USB/SATA II, III interface.

- ➢Complete data capture in Azure required 4 visits to the physical site. Data copy was limited by speed of internal network.

- ➢Other options that may be available are Express Route (private connection supports 100 gbs) or a larger network pipe

Data Files and Content

What NWM did with the Data

- Data Files in blob storage used to provide software, Data for Applications, O365 Files (Word, Excel, etc.), and data base backups
- Data files for Applications copied to File Servers provisioned using VMs
- Data files for standing up software copied to VMs that were used to host applications
- Data files for O365 copied to SharePoint document libraries
- Data files for database backups used to restore SQL Server, Oracle, and OpenText databases

Data Lessons Learned

- ➢ Uphold Chain of Custody[1]
  - ➢ Data file counts of data in Azure storage accounts were validated against file counts provided on source data files. Ensures no data lost as data changes custody.
- ➢ Ensure data copied was Forensically Sound[2]
  - ➢ MD5 Checksum comparisons were made by using MD5 checksums provided by source site IT and comparing to MD5 checksums generated by NWM Project Team of files in Azure Storage accounts. Ensures the data transferred is not corrupted during data transfer

(1) Chain of Custody, as it applies to Computer Forensics

A process that tracks the movement of evidence through its collection, safeguarding, and analysis lifecycle by documenting each person who handled the evidence, the date/time it was collected or transferred, and the purpose for any transfers (NIST SP 800-101 Rev.1)

(2) Forensically Sound

Forensically sound data collection refers to the process by which ESI (Electronically Stored Information) is collected for eDiscovery without any alteration or destruction of either the data or its metadata.

Databases Used in NWM

- **SQL Server – multiple PaaS services and IaaS (SQL VM)**
  - Tried to standardize on this DBMS. Used Azure SQL Server Managed Instance service
  - NOTE: Source Oracle Databases migrated to SQL Server
- **PostGreSQL** – **Paas DB Service**
  - Redeveloped applications used the Azure Postgres database service
  - NOTE: Source databases migrated to Postgres
- **OpenText Collection Server – Proprietary DBMS from Open Text**
  - Required by COTS package (Open Text Library Management)
  - Used Windows VM to host DBMS.
  - Used to restore Open Text database backup files.
  - Migrated Open Text database to SQL Server to support migration to COTS package

Azure SQL Database Services

SQL Server on Azure Virtual Machine – provides complete SQL Server with licensed products (SSRS, SSIS, SSAS) on a SQL VM

- Used to configure an SSRS report server required for NWM Report Center
- SQL Server used to generate on-premises compatible SQL Server backups.
- Customer response for upgrading, patching SQL products, scheduling backups. Microsoft updates OS

SQL Server Managed Instance – provides a SQL server instance that hosts multiple database

- Very close to providing an on-premises SQL Server Instance (Cross DBMS queries, Restore from On-Prem SQL backups, etc.)
- Ability to host multiple Databases required for NWM applications (Max is 100, 89 currently)
- Only supports native backups when Transparent Data Encryption (TDE) is turned off for the database.
- Needs to be deployed in a dedicated subnet in a virtual network.
- Managed Instance Database backups not compatible with on-premises SQL Server restore (requires migration tools)
- Supports vCore purchasing option with Gen5
- NWM project used SQL Server installed on laptops to restore older SQL Server version database backups then migrated to SQL Managed instance

Azure SQL Database – provides a single SQL Server database (creates a reusable virtual Sql server)

- Used to simplify providing database support (reduced network security requirements) and speed up application development (for one application - DIRS)
- No native backups, only automated to provide Point-In-Time-Recovery (PITR).
- No recovery from database backup files, need to use Microsoft Migration Tools (BACPAC, DACPAC files, etc.)
- Support both DTU (Database Transaction Unit) with multiple Service Tiers (Basic, Standard, Premium) as well as vCore purchasing options

Databases

Our Challenges

- Data needed to be available for each application that was re-hosted, re-developed, or replaced by a COTS package
- DBMS platforms were varied, old versions, and in some cases, no longer supported
- There were approximately 10-12 applications that required a database environment

Our Solution

- Database backups were copied to Azure storage accounts using same mechanism as data copy
- Azure’s SQL Managed Instances were provisioned for development and production
- DBs were restored to the SQL Managed Instance if the source platform was SQL Server
- Oracle DBs were converted to SQL Server through a multiple step manual process
- PostGreSQL database services were provisioned to support application development tools.

Database Lessons Learned

- **There are significant differences between Azure database services and on-premises database services**
  - Compatibility issues in various Azure database services (impact on COTS, etc.)
  - Network Security configuration requirements (Firewall and Vnet Requirements, etc.)
  - Backup and Recovery limitations (No native backups for SQL Database, etc.
  - DB Management and Migration tools (VM with DMS, SSDT Visual Studio, SSMS, etc.)
  - Choose purchasing options and service tiers (vCore, DTU, Service Tiers, etc.)
  - Developer and DBA skills (classes on managing SQL Database services)
  - Limit the number of different Database Management Systems (simplifies O&M, etc.)
  - Responsibilities – be prepared for what Customer Manages

Customer Manages	**Microsoft Manages**
Capacity Planning, purchasing options	Hardware, datacenter, virtualization
Migration	Operating System patching, upgrades
Monitoring (logs, etc.)	SQL Installation, configuration, patches
Configuring Network security	Network availability
Backup (additional native copies)	Backup and Restore (PITR, LTR)
Performance Tuning	High Availability and Disaster Recovery
Database level configuration	Security (SQL Defender, etc.)
Database Design	Scaling
Automation	Auditing
Cost Monitoring and Optimization	
Fixing outages (Refers to application outages due to deadlocks, etc.)	

Additional Resources

- **Book** - Professional Azure SQL Managed Database Administration by Ahmad Osama, Shashikant Shakya, 3rd edition – 2021
- **Network Connectivity** - https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/connect-application-instance?view=azuresql
- **Network Configuration** - https://docs.microsoft.com/azure/azure-sql/database/connectivity-architecture
- **Feature Comparisons** – https://docs.microsoft.com/azure/azure-sql/database/features-comparison
- **Services Tiers** - https://docs.microsoft.com/azure/azure-sql/database/service-tiers-vcore
- **Restore SQL MI DB to on-prem** - https://dba.stackexchange.com/questions/244167/backup-a-database-from-azure-sql-managed-instance-and-restore-to-on-premise-sql
- **Azure TDE Overview -** https://docs.microsoft.com/azure/azure-sql/database/transparent-data-encryption-byok-overview

Q&A