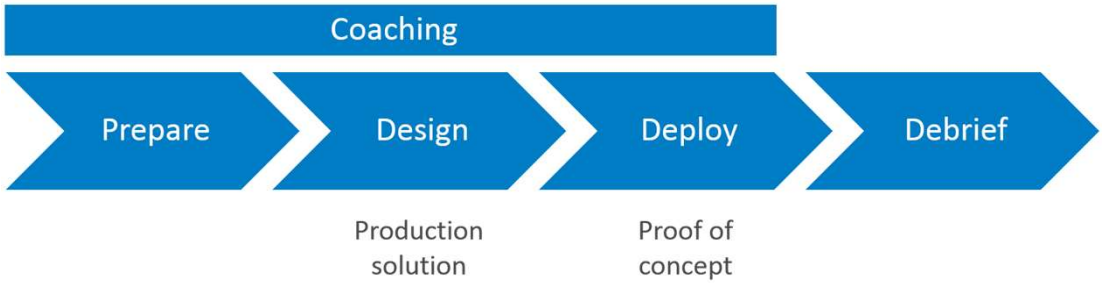

Global Knowledge
Architecting Microsoft Azure Solutions Master Class
LG-6342-002

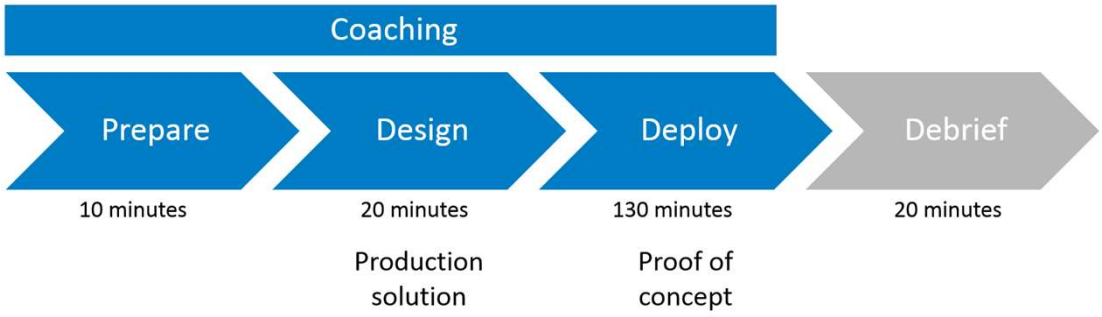
Challenge process



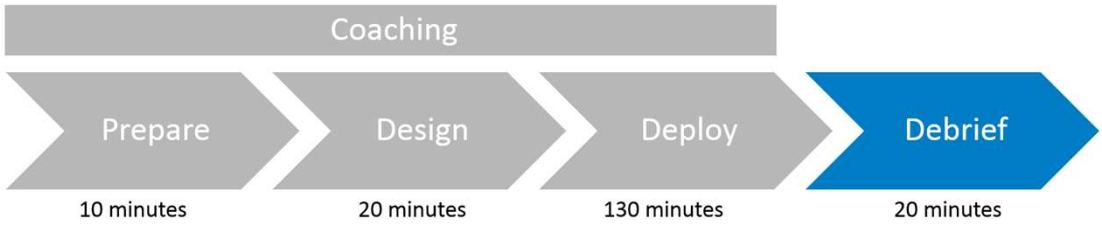
Briefly explain process students will through for each Challenge
•Prepare – Review the Challenge
•Design – Design the production solution
•Deploy – Implement a proof of concept solution
•Debrief and coaching – Coach will monitor work and go over final products and share preferred solution


Global Knowledge
Challenge: Migrate a multi-tenant application to Azure

Challenge: Migrate a multi-tenant application to Azure

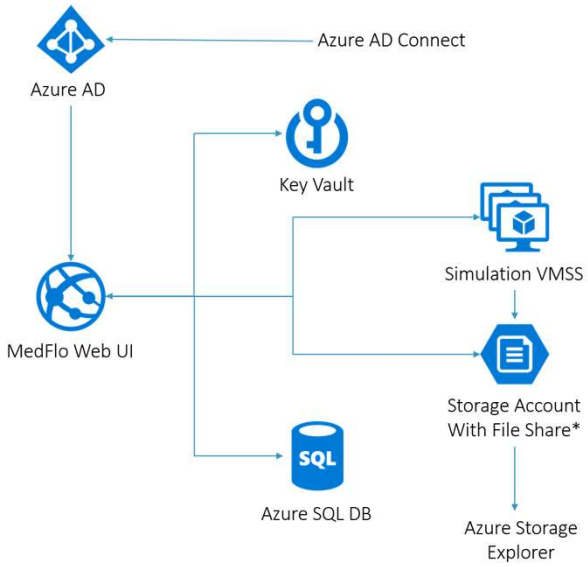


Challenge: Migrate a multi-tenant application to Azure



MedFlo Azure components

- Azure AD
- Web App
- Key Vault
- Azure SQL DB
- Simulation Processor
- Storage



This is an overview of what the components are and how they work together. Details on provisioning (including options) and security settings are included in follow-up slides

Components

Azure AD. This is used for authentication for all user access, including via the web app and direct.

MedFlo Web UI. This is a migration from the on-premises UI. The app will need to be modified to work with Azure AD authentication and tie into the Azure resources.

Key Vault. Key vault is used to hold sensitive information such as storage keys and VMSS admin account information

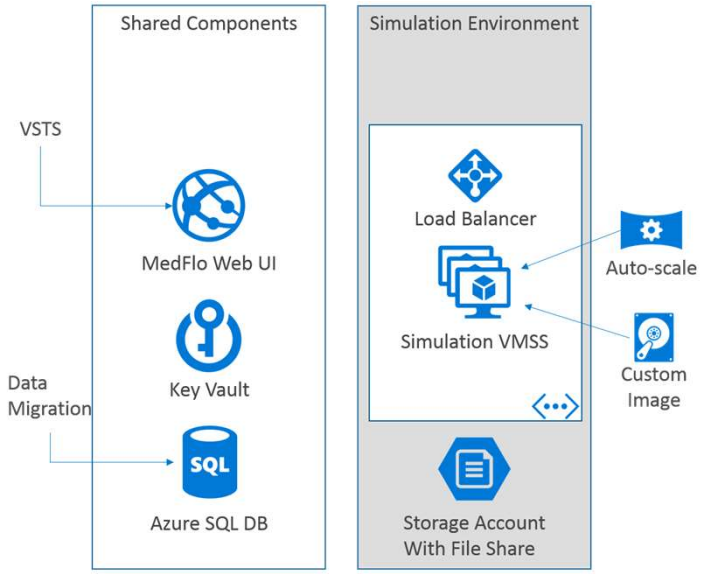
Simulation VMSS. The VMSS will be a Linux-based scale set. It will be implemented so as to minimize any recoding of the models.

***Storage account with File Share.** The file share is selected because the existing code (UI and simulation) works with an SMB network share. However, it will likely be better to use BLOB storage which gives you better authorization options.

Azure Storage Explorer. This is the obvious choice for direct file access (via either file or blob storage). Point out that it is free, cross platform, and has a familiar interface

MedFlo provisioning

- Resource templates
- Shared components
- Simulation environment



Deployment
The deployment is broken into two groups – shared components and simulation environment.
Shared Components
•Deploy via ARM template
•**App Service Plan** – S1 (standard, one instance). Add autoscale to scale to two as necessary. Only one is necessary for availability SLA; use app identity option
•**Web App** – Associate SSL/certificate – sensitive info; configure staging slot; configure CI/CD between VSTS and staging slot; plan for swap
•**Key vault** – Grant web app identity access; store DB admin account/pwd, VMSS account/pwd per deployment, storage keys/deployment
•**Azure SQL DB** – transparent database encryption; Azure AD access – grant access to web app identity; no direct user access; admins use AD account via SSMS; force encrypted connections; migrate data from on prem – standard data use choice of tools such as SSMS, SSIS, etc..
Simulation Environment
Deploy via ARM template and script. Script is necessary to add sensitive data to key vault and to set policy on premier tier.
May need to deploy multiple simulation environments for shared tier – 500TB limit for storage

Each project on basic and premier has a dedicated simulation environment.

•**Parameters** – Cost Center, customer name, project name; VMSS size (ie D2_V3). All resources are tagged with cost center.

•Create ARM template based on Linux VMSS

•**Virtual network** – IP address range 10.x.x.0/24, subnets default 10.x.x.0/25, GatewaySubnet 10.x.x.248/27, rest as spare, GatewaySubnet for premier tier. This gives 123 IP addresses for simulation process.

•**Storage account** – standard GRS. Add name, keys into key vault

•**Load balancer** – balance port 443 into VMSS; set NAT rules for high port (50000) map to port 22 for direct SSH support

•**VMSS** – managed disks, based on custom image with software installed – quicker load, add autoscale to meet requirements – scale out at > 70%, scale in < 10%, max 10 – shared, 5 – basic; 25 premier

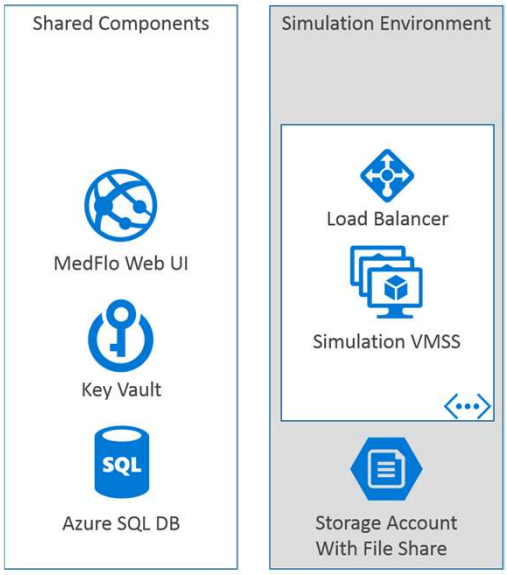
Create policy for premier tier that adds cost center to any resource that they add.

MedFlo security

- User authentication and authorization
  - Authentication is through Azure AD
  - All Azure AD directories are synchronized via Azure AD Connect.
  - Requires Azure AD Premium for password write-back.
  - Wunhillpharm.com is default Azure AD directory.
  - Each customer has a tenant Azure AD directory with three Global Admins.
  - A security group is added for each project.
  - Global Admins manage group membership through Azure Portal.
- Web App
  - Configure to use Azure AD authentication
- SQL Server
  - Only allow access from Azure Services
  - Implement Transparent Data Encryption
- Storage
  - All storage accounts configured for encryption at rest
  - VMSS VHDs use encryption or disk encryption with managed disks
  - Direct access (shared and premier tier) using Azure Storage Explorer
  - Geo-redundant storage

Demonstration

- Resource templates
- Shared components
- Simulation environment



Prior to demo:

1.If you do not have an Azure AD user in your default Azure AD directory create one (this is easiest via the portal

2.Retrieve the object ID of a user in your default Azure AD directory

    1.Open PowerShell
    2.Log in to Azure RM
    3.Run the following: Get-AzureRmAdUser
    4.Copy the id from one of the users. If there are no users create one then repeat

3.Download the demonstration files from https://github.com/GKLabContent/6432-AzureArchitecture

4.Open the challenge1\MultitenantAppSolution.sln solution.

5.Right click the solution and select Deploy and the New

6.Set the following:

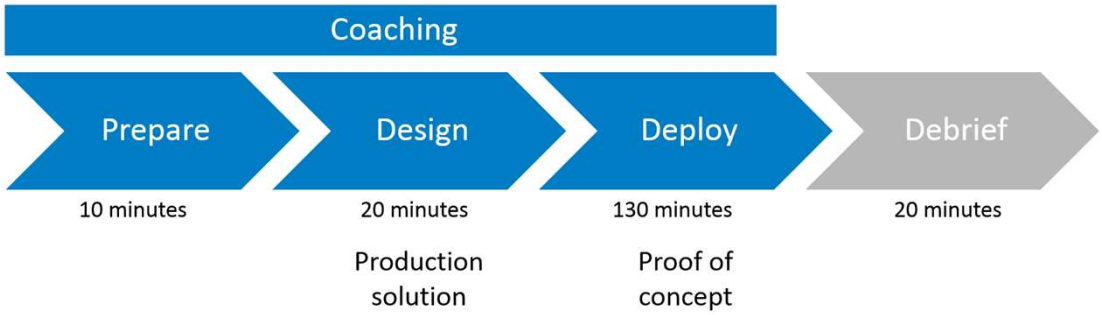
    1.Subscription: <your subscription>
    2.Resource Group: <new> 6342C1-Shared (EastUS)
    3.Deployment template: sharedComponents.json
    4.Template parameters file: sharedcomponents.parameters.json
    5.Click Edit Parameters and set parameters accordingly. Set the ObjectID to the id you copied for the Azure AD user

7.Click Deploy

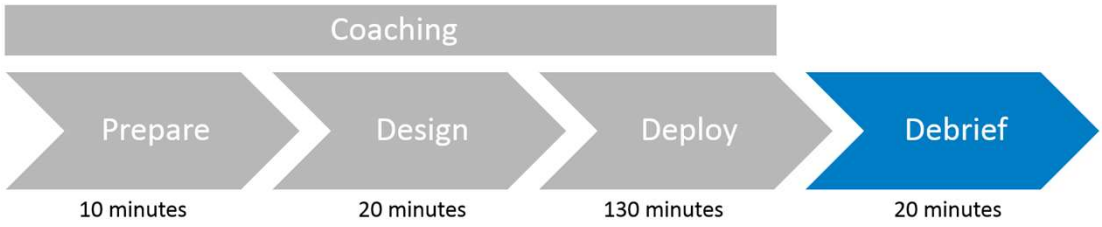
8. Wait for the deployment to complete. I twill likely take about 2-3 minutes
9.Right click the solution and select Deploy and the New
10.Set the following:
    1.Subscription: <your subscription>
    2.Resource Group: <new> 6342C1-Shared (EastUS)
    3.Deployment template: sharedComponents.json
    4.Template parameters file: sharedcomponents.parameters.json
    5.Click Edit Parameters and set parameters accordingly.
11.Click Deploy
12.Once the deployment is complete you are ready for the demonstration. Deployment should take somewhere in the 4-5 minute range.
Demonstration:
1.Review the deployment templates
2.Configure the web app to use a managed service identity. This is easiest via the portal.
3.Create a SQLAdmin group in the default Azure AD directory via the portal
4.Add your account to the SQLAdmin group
5.Optional: Add the web app service account to the SQLAdmin group. This requires the AzureAD PowerShell commandlets and a Global Admin user account in your default Azure AD directory. To do this, run a variation of the following from PowerShell:
    1.Login-AzureRmAccount
    2.Connect-AzureAd
    3.$spn = (Get-AzureRmADServicePrincipal | ? {$_.DisplayName -like "*Managed*"})[0] ##This worked for me but probably needs some tweaking
    4.$group = Get-AzureRmAdGroup --SearchString SQLAdmins
    5. Add-AzureADGroupMember -ObjectId $group.Id -RefObjectId $spn.Id
6.Set the SQL Server instance to use the SQLAdmin group as the Active Directory Admin.
7.Open the key vault.
8.Add yourself as an administrator
9.Create an access policy for the Managed Service Identity.
10.Create an access policy for your account
11.Review the load balancer and VMSS
12.Copy the storage account key
13.Add the storage account name and storage account key to the key vault. Let students know that this would use well defined names in production and that the web app would access these settings.
14.Create a fileshare in the storage account
15.Open Azure Storage Explorer (download if necessary) and navigate to the file share.
16.That's really it for the demo
17.You might want to point out that most of the demo steps would be scripted in a production environment. Several of the steps cannot be accomplished via templates


Global Knowledge
Challenge: Design a massive-scale data processing environment

Challenge: Design a massive-scale data processing environment

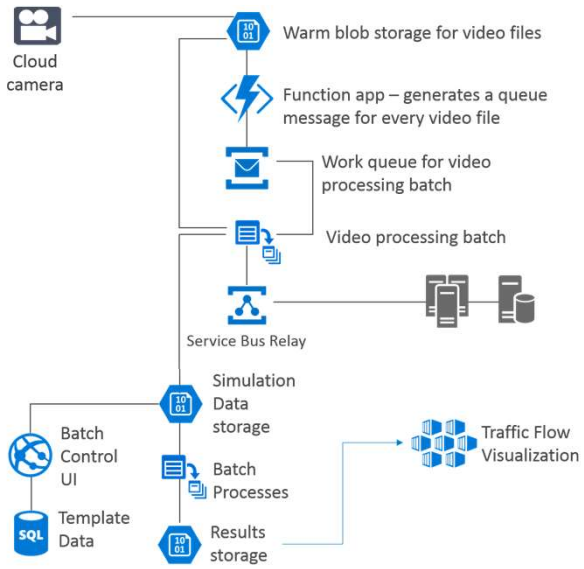


Challenge: Design a massive-scale data processing environment



RoadwayImpact In Azure

- Storage
- Cameras
- Queues
- Function App
- Batch Services
- Service Bus Relay
- Traffic Flow Visualization



Storage:
•Raw Video – Warm Blob storage – write once, read at least once. Create a storage container for each project
•Traffic data files/ control files – Standard blob storage. Data files will be written once and read many times. Control files may be written many times but are relatively small. Create a storage container for each project
•Simulation results – File Share. Create a file share for the results with a new folder for each project. Map a drive to the file share from analysts' workstations
Cameras
•Cameras upload videos via REST interface for storage account
•Create a new container for each project
•Use storage policy as SAS for secure access. Cameras cannot have storage account keys.
Queues
•Video processing queue – Service bus queue. These messages are small so there isn't much difference between SB queues and storage queues. Files may be written twice which might generate multiple queue messages. SB queues have built in duplicate detection.
•Lock time – set the lock time to 30 minutes. This is three times the average processing time for a video
Function app
•Generate a queue message any time a video is uploaded from the camera. This eliminates

any complex searching through blobs, which has pretty limited query capabilities.

Batch services

- Batch account
  - User Subscription – custom images
  - Separate batch jobs for video processing and simulation
- Pools
  - Control – used for relatively low impact batch control tasks
    - Relatively low-end VMs – A2, D2 v2, etc.
    - Running Windows Server 2012 or higher.
  - Video processing
    - NV-12 VMs (2 Nvidia Tesla M60 GPUs)
    - Custom Windows 2012 R2 image
    - Scale to meet demand
  - Simulation
    - E4 v3 VM size
    - Custom Ubuntu 16-04-LTS image
    - Scale to demand
- Jobs
  - Video processing
    - Schedule to run once daily
    - Batch control task – Set up logging and I/O. Generate and run video processing tasks. Pull processing messages from queue and push video files to processing tasks. Write results to blob storage. Written in C#. Run in control pool.
    - Video processing task –. Process videos. Use SB relay to add vehicle data. Read and write data to local storage. Run on video processing pool.
  - Simulation
    - Run on demand
    - Batch control task – Set up logging and I/O. Generate and run simulation and analysis tasks. Pull traffic and control files form blob storage and push to simulation instances. Write analysis results to Azure file share. Written in C#. Run in control pool.
    - Simulation task – Run simulations based on local data and control files. Write results to local file system. Run in Simulation pool.
    - Analysis task – Use custom R based code to create results files from simulation results. Read and write data locally. Run in Simulation pool.

Service bus relay

- Add service bus namespace (use the same as the queue)
- Configure WCF service to use SB relay
- Modify video processing batch task to call SB relay endpoint

Traffic Flow Visualization

- Deploy AKS (Azure Kubernetes Container Service)
- Pay only for worker nodes
- Public IP address

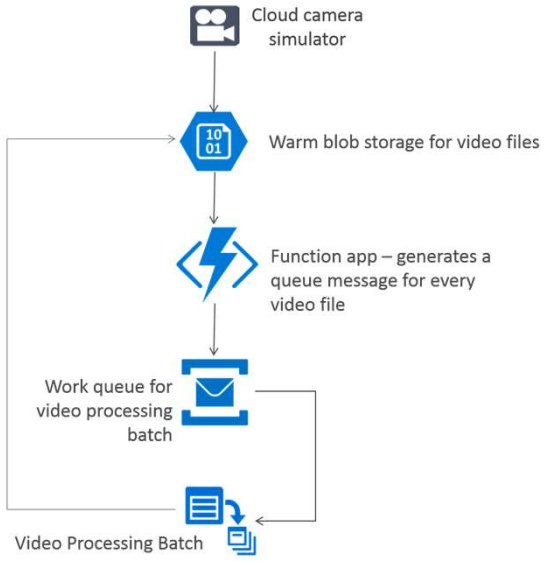
•50GB storage
•This will handle load balancing
•VM size are not really specified, so choose something small but reasonable D2_V3

Web app and database

•Create an Azure web app with an Azure SQL database
•Requirements are otherwise a bit vague because, really, if a student gets everything else, just give them this one.

Demonstration

Storage and function app integrations



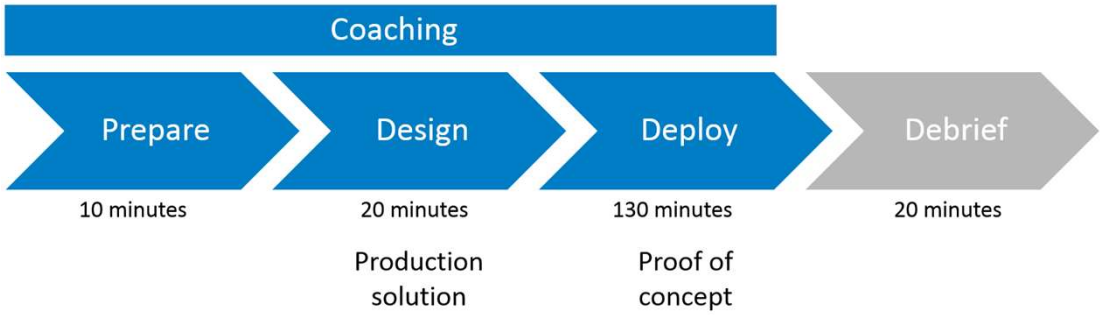
Preparation
This will take somewhere between 30 minutes and 1 hour to process.
Run the setup script
•Open PowerShell ISE
•Open the script <labfiles>\challenge2\demo\challenge2-deploy.json
•Edit the variable values in the #Set these variables section
•Save the file
•Run the script
NOTE - Be sure to run the script, as opposed to selecting sections of script and running it. Two file path variables (batchAppFile and environmentSettingsFile) use the built-in $PSScriptRoot variable which only has context when running a script. If you want to run the script piecemeal, modify the definition of the variables.
•Log in to Azure
•Select your subscription
•Wait for the process to complete.
•Verify that the storage accounts, Service Bus, and batch account were created
Create the function app
•Log in to the Azure portal
•Navigate to the challenge2 resource group (use the name you specified if you changed this in the script)

•Click the + New item
•Type function app in the Everything blade search box and click the Function App item in the search box
•Click Function App in the Everything blade
•Click Create
•Specify a unique App name
•Set the location to the same location as the rest of the challenge
•Leave everything else as the default
•Click Create
•Wait for the function app to create
•Navigate to the blade for the function app
•Click the + next to Functions in the navigation for the function app blade
•Click the custom function link on the create panel
•Select Blob Trigger – C#
•Name the function videoFunction
•Set the Path to video
•Click the new link next to the Storage account connection box
•Select the data storage account (it will be the name you used for your storage account + data)
•Click Create
•Once the function is created click Integrate under the function in the navigation for the function app blade
•Click + New Output
•Click Azure Service Bus
•Click Select
•Set Message type to Service Bus Queue
•Set Queue name to videoprocess
•Click new next to the Service Bus connection box
•Ensure that the Namespace and Policy have values. The Namespace should match the manespace that you specified in the setup script
•Click Select
•Click Save
•Click videoFunction in the navigation
•Open the file <labfiles>\challenge2\demo\function.csx
•Replace the contents of videoFunction with the contents of function.csx
•Click Save
•Click the logs link to open the Log panel on the function app blade. The link is towards the bottom of the blade
•Verify that the function saved successfully
•Leave the function app blade open.
Run the camera simulator
•Open the file <labfiles>\challenge2\executables\camera\CameraSim.exe.config
•Open the file <labfiles>\challenge2\demo\environment.json
•In the config file, set the values of the VIDEO_STORAGE_ACCOUNT and VIDEO_SAS to the

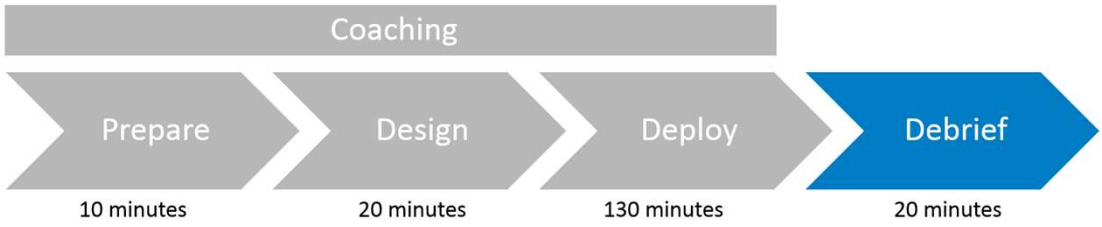
value from the environment.json file.
•Open a command window or use powershell
•Navigate to <labfiles>\challenge2\executables\camera
•Execute CameraSim.exe
•You should receive a message that three files were created
•Return to the portal
•The function log should indicate that it has been fired three times
•Verify the messages are in the queue
•Navigate back to the resource group (challenge2 unless you used something different)
•Click the Service Bus item in the resource group blade
•Click the videoprocess queue
•Verify that there are messages in the queue
•Create the video process task
•Navigate back to the resource group (challenge2 unless you used something different)
•Click your batch service
•Click Pools and verify that you have a pool with dedicated nodes at either 0 or 1
•Click Applications and verify that you have an application named ProcessVideo
•Click Jobs
•Click the Demo job
•Click Tasks
•Click + Add
•Complete the Add task blade with the following information
•Task ID: testTask
•Command line: cmd /c %AZ_BATCH_APP_PACKAGE_PROCESSVIDEO%\VideoProcessor.exe
•User identity: Task autouser, Admin
•Application packages: select ProcessVideo v. 1.0
•Click OK
•Wait for the task to complete. Navigate to the task and refresh the Overview panel until it is complete
•Clicked Saved logs on the task blade navigation
•There should be two files – stderr.log and stdout.log. stderr.log should be empty and there should be data in the stdout log.
Demonstration
•Walk students through the resources provisioned for this challenge. Don't go into details.
•Open the function log panel.
•Walk students through the Challenge2-deploy.ps1 script
•Show students the CameraSim.exe.config file. Point out the storage settings
•Run the CameraSim console app to generate three files
•Return to the function log panel. It should show three activations
•Navigate to the Service Bus queue. There should be three messages in the queue.
•Navigate to the batch account.
•Navigate to the job
•Show students the environment variables
•Repeat the steps for the "Create the video process task" task above. Verify that it processed


Global Knowledge
Challenge: Design a global compute strategy

Challenge: Design a global compute strategy

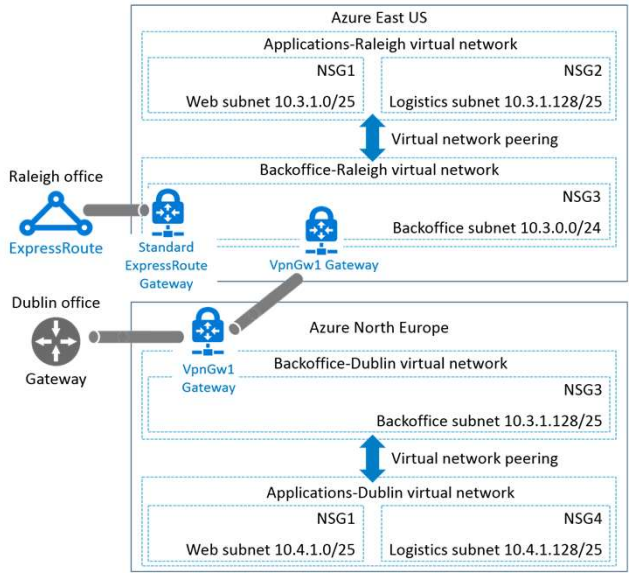


Challenge: Design a global compute strategy



Networking

- Networks
- Gateway connections and peering
- Network security groups



Networks
•East US
    •Back office
        •Backoffice – 10.3.0.0/24
        •GatewaySubnet – 10.255.255.0/27
    •Applications
        •Web – 10.3.1.0/25
        •Logistics – 10.3.1.128/25
        •GatewaySubnet – 10.255.255.32/27
•North Europe
    •Back office
        •Backoffice – 10.4.0.0/24
        •GatewaySubnet – 10.255.255.64/27
    •Applications
        •Web – 10.4.1.0/25
        •Logistics – 10.4.1.128/25
        •GatewaySubnet – 10.255.255.96/27
•Virtual networks for back office and applications because they are managed by different groups
•Split the applications network to easily apply network security groups

•Gateway subnets are basically pushed to the end of the line

Gateways

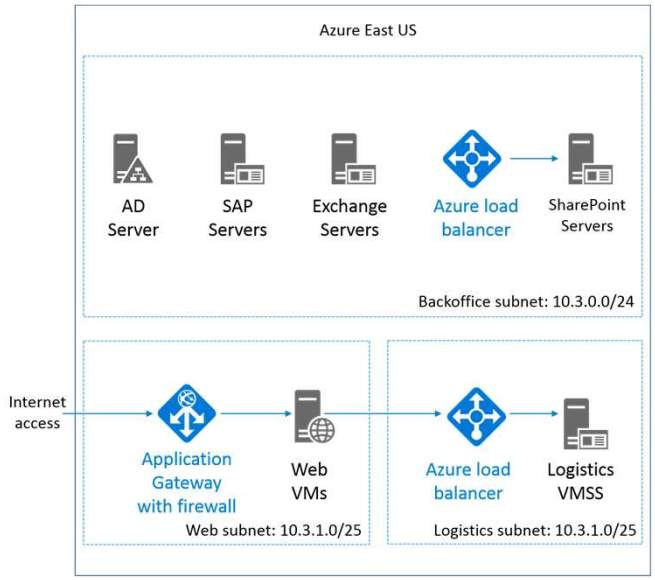
•Backoffice - Raleigh
    - •ExpressRoute Standard Gateway. Uses Euinix ExpressRoute provider in Atlanta or Washington DC to connect to Raleigh office
    - •VpnGw1 size VPN gateway. Connect to backoffice-dublin virtual network
    - •Configure peering with applications-raleigh virtual network.
- •Applications – Raleigh
    - •Establish peering with backoffice-Raleigh
- •Backoffice-Dublin
    - •VpnGw1 size VPN gateway. Connect to backoffice-Raleigh. Connect to existing gateway at Dublin office
- •Use complex keys for all connections

Network Security Groups

- •NSG1
    - •Allow 80 and 443 from internet
    - •Allow 3389 (RDP) from 10.0.0.0/16
- •NSG2
    - •Allow 1234 from 10.3.1.0/25
    - •Allow 22 (SSH) from 10.0.0.0/16
- •NSG3
    - •Allow * from 10.0.0.0/16
    - •Allow 4346 (SAP) from 10.0.0.0/8
- •NSG4
    - •Allow 1234 from 10.4.1.0/25
    - •Allow 22 (SSH) from 10.0.0.0/16
- •NSG3 could be modified so that only the web subnets can use port 4346 for requests. However, other regions may be added and this makes it easier to maintain.
- •SAP is only running in the East US region, so the backoffice-Dublin backoffice subnet does not really need port 4346, but it makes it easier to use backoffice-Dublin as a DR site in the future.

Compute

Access and load balancing



Load balancers

•Internal load balancer for SharePoint web front ends

•Internal load balancer for Logistics VMSS. This will load balance requests over port 1234 and set up a NAT rule mapping requests from ports 50000-50020 to ports 22 on the instances of the VMSS

Application gateway

•Implement an application gateway with web application firewall for the web and API server.

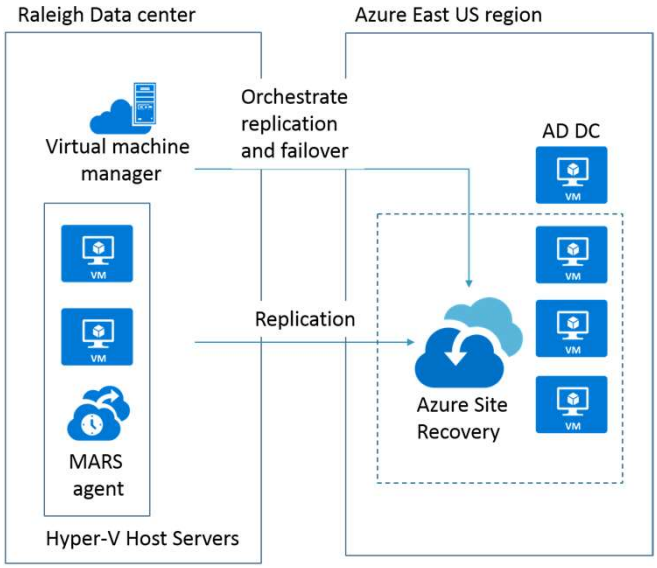
Compute

- VM sizes
- VM configuration

Role	#	Configuration
Back office		
Active Directory	1	Standard A2 v2
Exchange	2	Standard D4 v3 2x5 TB data disks in a storage pool
SharePoint	2	Standard E4 v3 1 x 5 TB data disk
SAP	2	Standard E2 v3 1 x 5 TB data disk
Applications		
Web server	2	Standard D2 v3
Logistics server	2-5	Virtual machine scale set Standard D4 v3 Autoscale extension Custom script extension

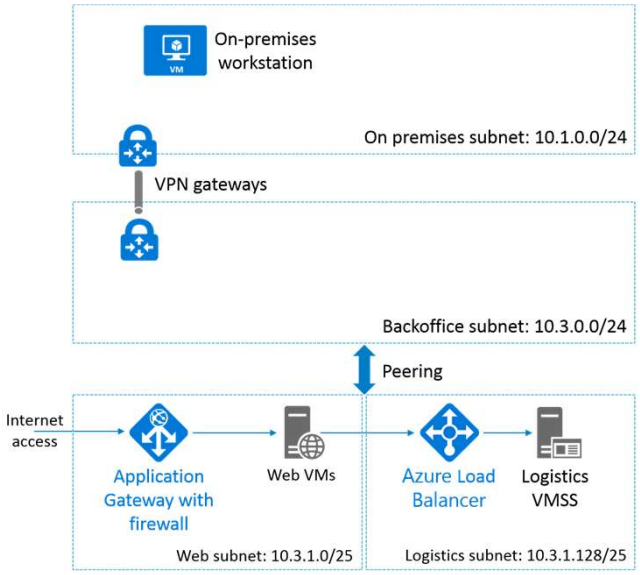
Transition

- Active Directory
- Azure Recovery Services
- Failover
- DNS



Active directory
•Provision AD domain controller in Azure
•Configure Azure subnets in AD sites
Recovery services
•Create recovery vault
•Install Azure recovery provider in SAVMM
•Install MARS agent on managed Hyper-V machines
•Configure replication for Echange, SharePoint and SAP servers
•Initiate replication
Fail over 1 VM at a time
Update DNS
Test

Demonstration



Demonstration setup:
There are five deployment templates in the Challenge1.sln solution:
Challenge1-main.json – Control deployment template. This will deploy the other four templates
Challenge1-network.json – This sets up the basic network components, including three networks, several subnets, NSGs and two VMs
Challenge1-connections – This sets up the VPN gateways for the on-premises and back office networks and establishes a connection between them. It also establishes peering between the backoffice and applications networks.
LinuxVirtualMachineScaleSet.json – This deploys the Logisics VMSS.
Deployment:
1.Open the challenge1.sln file in Visual Studio 2017
2.Deploy the challenge1-main.json template.
3.Complete the parameters in the corresponding parameter file.
4.Deploy the solution. It will take somewhere between 30 and 60 minutes to deploy. When I tested it took 35 minutes.
Alternatively you can deploy the four templates individually.
Testing:
1.Connect to the workstation VM via RDP
2.From the workstation VM establish an RDP session to the web server (10.3.1.4)

3. Install the web server role on the web server
    1. Open Server Manager (it should open automatically)
    2. Select Manage in the top right menu
    3. Select Add Roles and Features
    4. Click Next
    5. Click Next
    6. Click Next
    7. Select Web Server IIS from the list
    8. Click Add Features in the pop up
    9. Click Next
    10. Click Next
    11. Click Next
    12. Click Next
    13. Click Install
    14. Wait for IIS to install

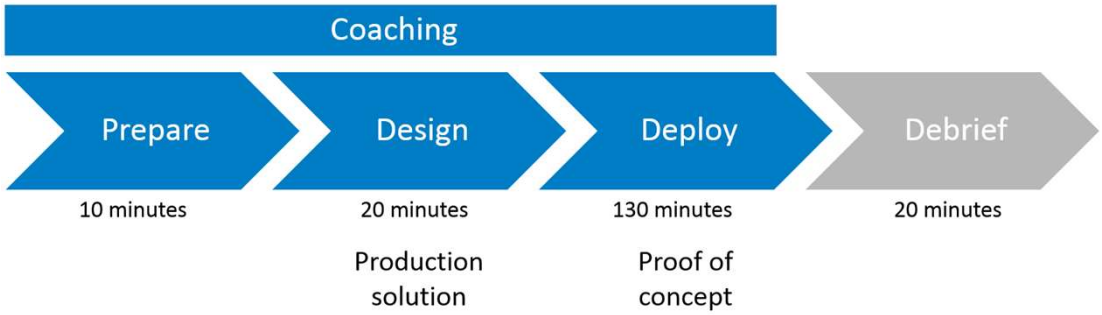
4. Test the web server locally by opening Internet Explorer and navigating to http://localhost. You should get the IIS welcome page
5. Close the RDP session to the web browser
6. Return the the RDP session for the workstation (you should be there upon closing the RDP session with the web server
7. Disable Enhanced IE security (Server Manager -> Local Server -> IE Enhanced Security Configuration -> Off)
8. Open Internet Explorer and go to https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
9. Install the Putty client.
10. Open putty
11. Enter 10.3.1.254 as the destination IP address and 50001 as the port
12. Confirm that you can establish an SSH session with the logistics server via the internal load balancer. You will be prompted to accept the certificate. Accept the certificate
13. Close the putty session
14. Close the RDP session with the workstation
15. In your own browser, open the Azure portal and navigate to the application load balancer that you installed. Note the public IP address.
16. Open a new tab in your browser and navigate to the public IP address of the app load balancer
17. Confirm that the IIS welcome page loads.
18. This confirms that the challenge 1 hands on components have successfully installed

Demonstration

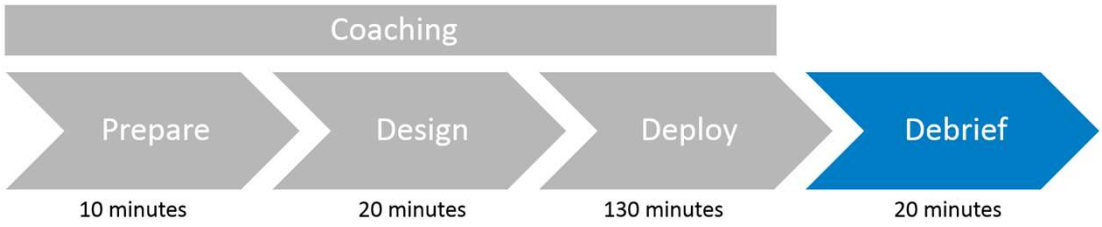
To demonstrate the solution walk the students through the components that have been installed and how they are related. You can use either the deployment templates or the portal depending on your preferences and the level of your students. Verify that you can reach the web server via the application gateway. If time permits, connect to the workstation via RDP and from there connect to the web server via RDP and the logistics server via SSH (putty)


Global Knowledge
Challenge: Design a data analytics environment

Challenge: Design a data analytics environment

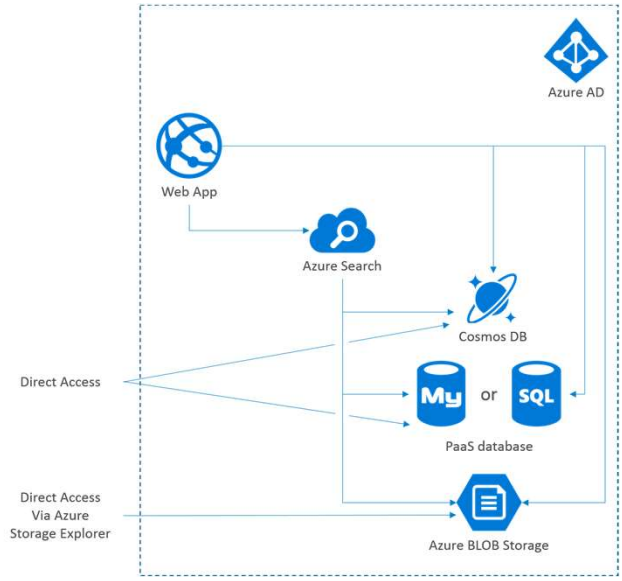


Challenge: Design a data analytics environment



Data access

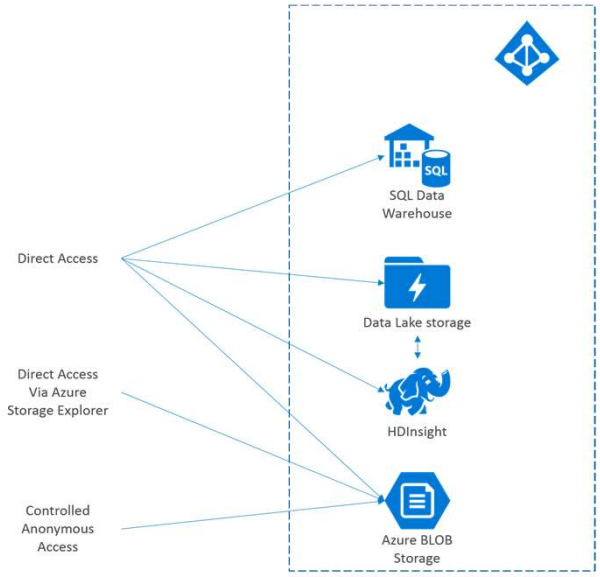
- **Targeted active research projects**
- Massive scale data survey



Deployment for TARP:
•**App Service Plan**. Deploy as a Linux-based App Service Plan. No performance specs are given so deploy as a Standard tier S1 size with one instance.
•**Web App**. Deploy as a Linux-based web app. There are no details other than that it uses Node.js version 6.1.
•**PaaS database**. This can be either Azure SQL or Azure MySQL. Both provide encryption at rest and can force encryption in transit.
•**CosmosDB**. Deploy with the MongoDB API.
•**Blob Storage**. Standard LRS storage. There is no specified requirement for geo-redundancy.
•**Search**. Basic search account. Configured with all three data storage mediums as index sources. Configure indexing schedules as appropriate
All resources protected by Azure AD.
Deploy Azure Storage Explorer to users' desktops.
Deploy by a combination of templates and scripts.

Data access

- Targeted active research projects
- **Massive scale data survey**



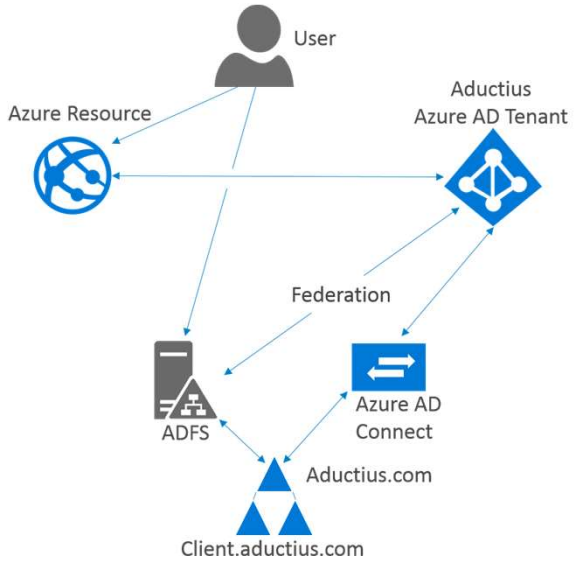
Deploy via templates
Based on the flexibility of requirements, the best option would be to use conditional templates, or modularize templates, or both. To use modularized templates create templates for each type of resource and then create "master" templates for various options.
Deployment resources:
•**SQL Data Warehouse**. Large-scale storage with simple table structures and large scale data (up to approx. 1PB when fully compressed. Polybase – variant of SQL, semi-relational
•**Data lake Storage**. PB level storage for MapReduce. Based on HDFS (Hadoop file system). Always encrypted, auditing RBAC; accessibly via WebHDFS for completely custom development
•**HDInsight.** Hadoop cluster as a platform service. Supports Hadoop and R, as well as Spark, Hive, LLAP, Kafka, and Hbase.. Data Lake as good store for this
•**Blob Storage**. Outcomes storage. Control access via containers.
•**Security**. Deploy to resource group. Set RBAC at resource group level based on Azure AD. Team members are owners.

Security requirements

- Azure AD Premium Level II
  - Azure AD Connect with ADFS
  - Advanced reports
  - Privileged Identity Management
- Log Analytics
  - Log all data access activity
- Encryption
  - SQL DB – TDE and enforce encrypted connections
  - MySQL – Enforce SSL connections
  - CosmosDB – all data is encrypted at rest and in transit
  - Connection keys used by web app stored in Key Vault
- RBAC
  - Resource group per project
  - Principals are owners of resource group
- Study Outcome Access
  - Use BLOB storage for study outcomes*
  - Set container security for public access
  - Use Azure storage explorer for access

Authentication flow

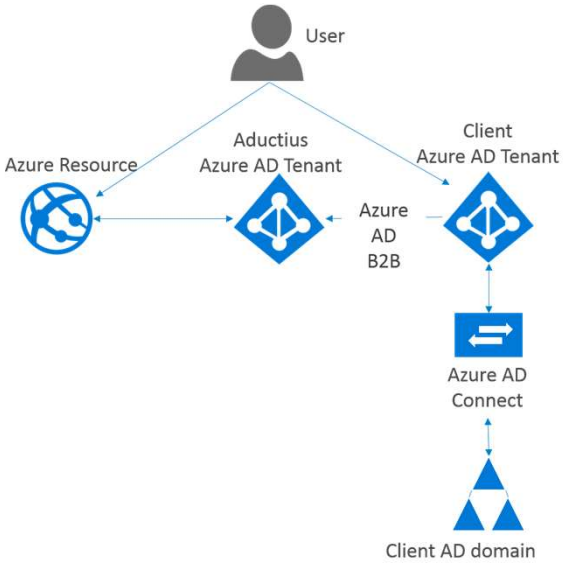
- **Clients with Aductius managed AD domains**
- Clients with their own Azure AD tenants
- Clients using ADFS without Azure AD tenants



Configure an Azure AD directory for each client
Configure Azure AD connect with ADFS. This will provide consistent SSO for web and on-prem access.

Authentication flow

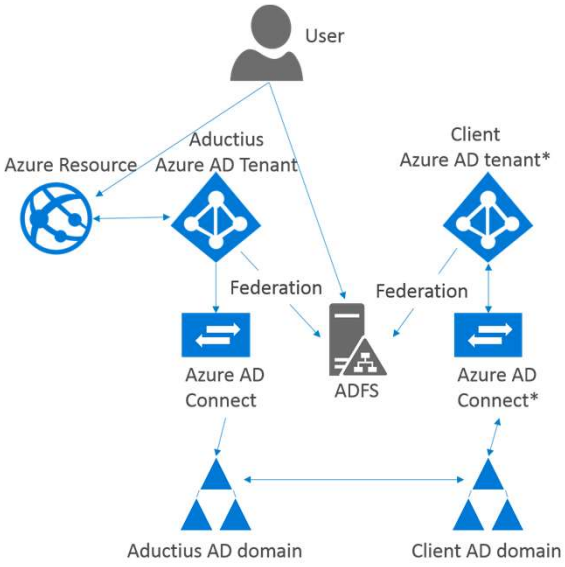
- Clients with Aductius managed AD domains
- **Clients with their own Azure AD tenants**
- Clients using ADFS without Azure AD tenants



Customers with their own Azure AD directories can be given direct access to resources. Simply add user accounts as owners of the appropriate resource group

Authentication flow

- Clients with Aductius managed AD domains
- Clients with their own Azure AD tenants
- **Clients using ADFS without Azure AD tenants**



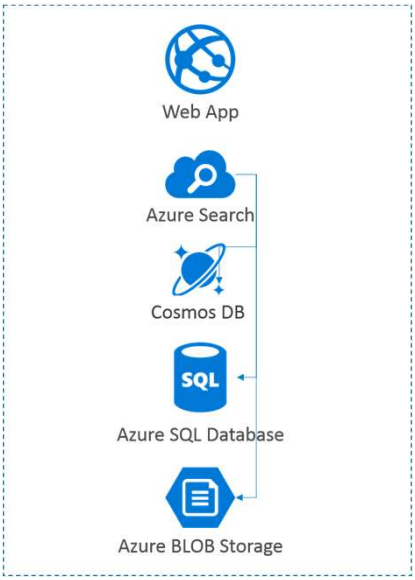
User's with their own AD domain but no presence in Azure AD pose a more complex problem. You will need to set up a directory for them and establish Azure AD connect to pull the basic info of their accounts into the directory. This will require coordinating the installation of Azure AD connect on a server with access to one of their domain controllers. You should also set up ADFS for SSO. This is very complex and you do not need to go into details

Supported topologies:
https://docs.microsoft.com/en-us/azure/active-directory/connect/active-directory-aadconnect-topologies

ADFS with multiple on-prem domains:
https://docs.microsoft.com/en-us/azure/active-directory/connect/active-directory-aadconnectfed-single-adfs-multitenant-federation

Demonstration

Targeted active research projects



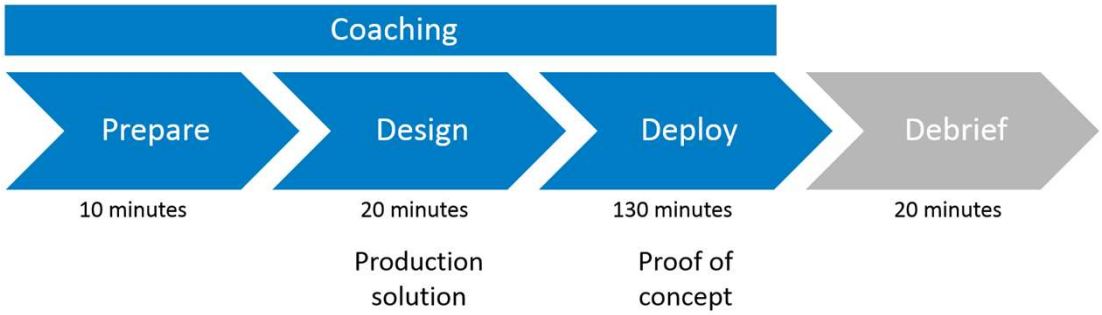
Demonstration
In this demonstration you will deploy the 5 components for a TARP project via a template prior to the demonstration. You will then load data via a script and an app. Then you will demonstrate search
Demonstration preparation
1.Download the demonstration files from https://github.com/GKLabContent/6432-AzureArchitecture
2.Open the challenge2\DataAnalytics.sln solution
3.Right click the DataAnalytics project and select Deploy -> New
4.Deploy with the following settings:
5.Wait for the deployment to finish. Deployment should take 3-4 minutes
Data load
Note: This can be done as demo prep or at the beginning of the demo. If you do this at the beginning of the demo, you should show the deployment template first.
1.Open the challenge2\DataAnalytics.sln solution if necessary
2.Right click the CreateData.ps1 file and select open with PowerShell ISE
3.Review the script with students
4.Replace any settings at the top of the file that may be different than what is in the script
5.Run the script
6.Record the Server Name and CosmosDB Key values

7. Return to Visual Studio
8.Open cosmosDBLoad\Program.cs
9.Briefly review the code with students.
10.Replace the serverName value (between the double quotes) and the cosmosDBKey value with the values recorded in step 5
11.Right click the cosmosDBLoad project and select Debug -> Start new instance
12.Click Enter to close the window.
13.Data is now loaded
Demonstration
1.Review the azuredeploy.json file in the DataAnalytics project
2.If you are demonstrating data load, run the Data Load process defined above
3.If you preloaded the data:
    1.Review the CreateData.ps1 script
    2.Optionally Review the cosmosdbLoad\Program.cs file
4.Log in to the Azure Portal
5.Navigate to the resource group for this demo
6.Point out the resources that have been created
7.Navigate to the CosmosDB account
8.Open the Data Explorer
9.Navigate down to the documents and move through them
10.Click Add Azure Search
11.Select the search service that is created for the demo.
12.Select the database and collection that show up in the list (there should only be one of each)
13.Accept the default index name and choose all options for the name column in the Index customization page
14.Give the indexer a name
15.From the Import data blade, click OK
16.Point out to the students that at the time of this writing, you can add a CosmosDB account to search from here but not from the search interface
17.Navigate to the search service blade
18.Click Import data
19.Click Data Source -> Azure SQL Database
20.Set the Name to sqldata (must be all lowercase)
21.Select the Demo database. It should be the first entry for your server. The name of the DB might not be isible so verify the name in the New data source blade after you sleect it
22.Enter the password that you used for the deployment. The default is Pa55w.rd1234
23.Click Test connection
24.Select the products table
25.Click OK
26.In the Index blade select RETRIEVABLE for the id field, and select all options for the name field.
27.Set the Key to the id field
28.Click OK

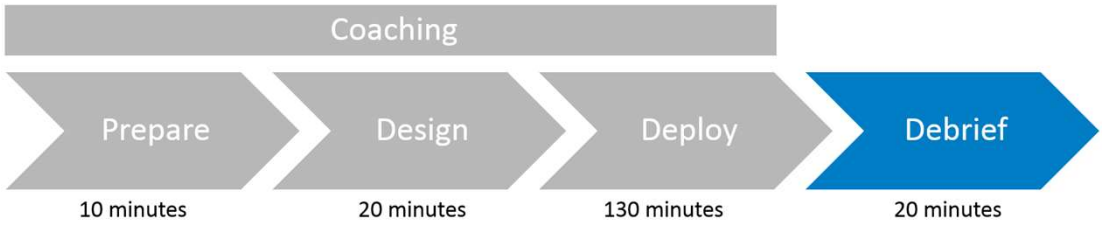
29. Set the Name to sqlsched
30. Click OK
31. Click OK to create the SQL indexer
32. Click Import data
33. Set the Data Source to Azure Blob Storage
34. Set the following properties in the New data source blade:
    1. Name: blobdata
    2. Storage container: Select your storage account and the unstructured container
35. Click OK
36. Accept the defaults on the Index blade and click OK
37. Set the name of the indexer to blobsindexer and click OK
38. Click OK
39. From the Overview Lens click Search explorer
40. Enter eggs and click Search.
41. Repeat with each index (click Change index to switch)
42. That's probably enough at this point


Global Knowledge
Challenge: Design an IoT solution architecture

Challenge: Design an IoT solution architecture

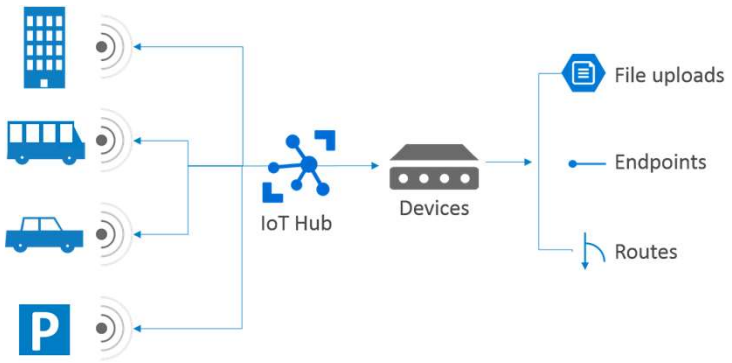


Client request: Design a data analytics environment



Setting up devices connection to IoT Hub

1. S1 IoT Hub Plan with 5 Units
2. Configure IP filter in IoT Hub with device address range
3. Use job framework to register new devices and assign keys.
4. Configure endpoints and set up consumer groups

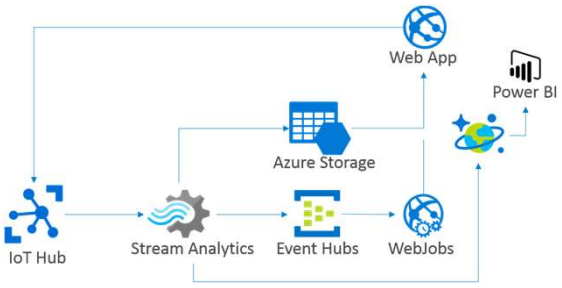


IoT Hub Configuration
IP filter to ensure only devices from IP range are allowed
The ImportDevicesAsync method in the RegistryManager class enables you to perform bulk import and synchronization operations in an IoT hub identity registry. Like the ExportDevicesAsync method, the ImportDevicesAsync method uses the Job framework. This uses two parameters, one that contains a string URI to blob storage with a SAS token that contains read access, and another URI to blob storage container with read,write and delete SAS token. They can be the same container.
Device message routing will provide endpoints such as Event Hubs (There is a default endpoint for all device messages), file uploads to Azure Blob storage automatically generates SAS tokens with write access for a TTL, and offers notifications, and Routes can deliver messages that meet a condition directly where they need to go without having to go through any other services. Needs to be in the same region to avoid cross-region charges
IoT Hub Plan
•S1 with 5 Units $250 (As of August 2017)
•Support of up to 2,000,000 messages a day
•400,000 Messages/Day/Unit
•Billing is evaluated each month on number of units consumed
IoT Hub Configuration

•IP filter to ensure only devices from IP range are allowed.
•The ImportDevicesAsync method in the RegistryManager class enables you to perform bulk import and synchronization operations in an IoT hub identity registry. Like the ExportDevicesAsync method, the ImportDevicesAsync method uses the Job framework. This uses two parameters, one that contains a string URI to blob storage with a SAS token that contains read access, and another URI to blob storage container with read,write and delete SAS token. They can be the same container.
•Device message routing will provide endpoints such as Event Hubs (There is a default endpoint for all device messages), file uploads to Azure Blob storage automatically generates SAS tokens with write access for a TTL, and offers notifications, and Routes can deliver messages that meet a condition directly where they need to go without having to go through any other services. Needs to be in the same region to avoid cross-region charges.

Configuring Stream Analytics for apps and analysis

- Assign an “Input” associated with messages from devices.
- Assign an “Output” to send data to storage and event hubs.
- Create a “Query” to select the message details from “Input” to send to “Output” based on alert and trigger metrics.
- Cosmos DB is used to connect to data for analysis.
- Applications use data generated by stream analytics to view alerts and access data.
- IoT Service Client SDK to manage Cloud-To-Device through Web Apps.
- Use Bing Map API to view geolocation data based on alerts.
- Cosmos DB Geo-Spatial Polygon Search.

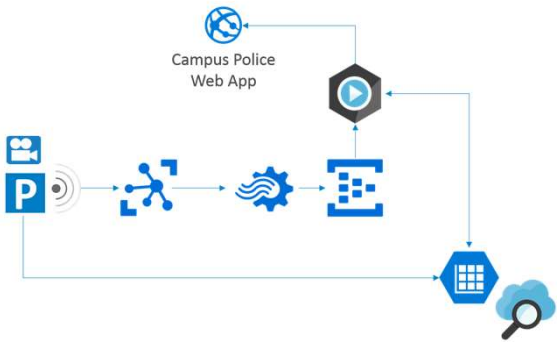


Stream Analytics
Perform queries on input as messages arrive to send data that matches the criteria to a destination. Send only the specific details needed by Cosmos DB Geo-Spatial search capabilities.
Can have up to two free Bing Map API connections for web apps used by building management and campus police. Alerts are generated through event hubs and table storage used by the associated web applications for both departments.

Devices can be managed directly in IoT hub or these features can be programmatically added to an application to send messages to the IoT devices.

Configuring Media Services

- IoT Hub Receives a Message that Emergency is triggered and is processed by Stream Analytics.
- Campus Police Web App gets notified and has direct access to camera feed through the live channel in Media Services.
- REST API to access media services and upload new videos for live viewing and archival.
- Azure Search can be used to access video content stored in Azure Blobs.



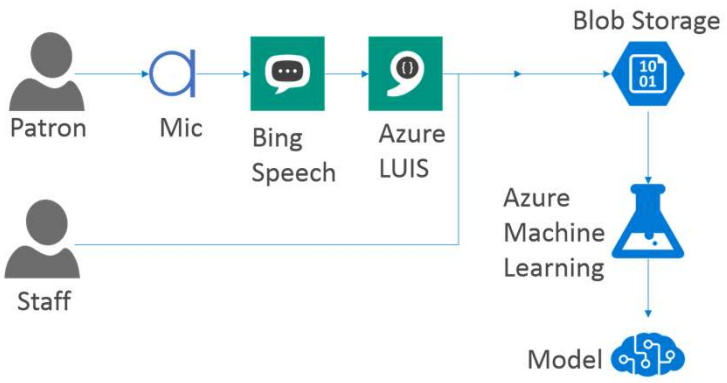
Media Services .Net SDK and Rest API can be used to allow Campus Police web app to access live channels and allow incoming videos generated from the emergency station to be uploaded and trigger an online channel for viewing live.

The system running the camera in the Emergency Station makes a REST API call to Azure Media Services to begin live streaming to the channel. https://docs.microsoft.com/en-us/rest/api/media/operations/azure-media-services-rest-api-reference

In addition to this solution, it is also possible to incorporate Azure Functions and Logic Apps for use with Azure Media Services, helping to provide custom triggers and workflows in the implementation.

Alcohol Safety Assurance Program - training

- Voice capture
- Voice processing
- Data storage
- Model training



Develop questions designed by socialogists to suggest truthfulness in patrons attempting to enter a bar

Use Bing API and Azure LUIS to capture each patron's responses to questions

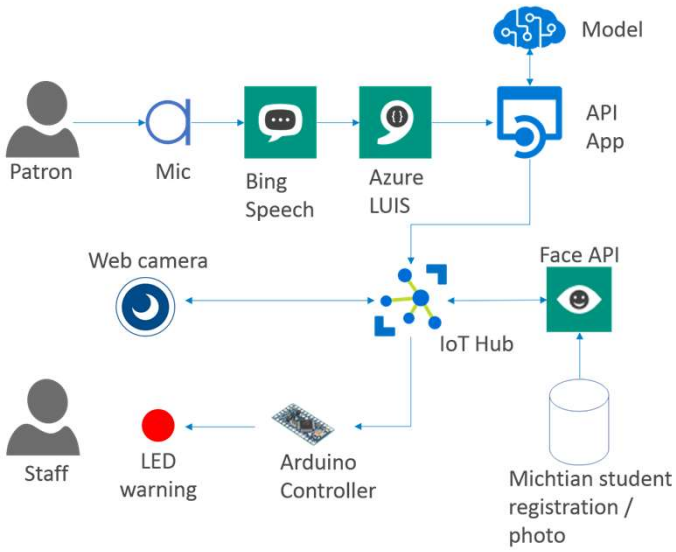
For each patron, have the expert bar personnel determine whether or not they are trying to enter the establishment on false ID

Save this data to blob storage

Process data through machine learning to develop and test a predictive model to determine the likelihood that specific answers indicate usage of a false id

Alcohol Safety Assurance Program - production

- Audio processing
- Model test
- Camera capture
- Facial recognition
- Staff indicator



In production:

Capture patron responses to questions through Bing speech and LUIS. Call an API app that processes the responses against the model

If the model shows a probability greater than xx then

Capture an image of the patron with the web camera. Send the command to the camera and receive the result via IoT hub.

Process the image against the Mightain registration database using the Face API

If the ID matches a student who is of age do nothing more

If there is a match with a student id that is under age send a message to light the red LED light via the IoT hub

If there is no match via the face API send a message to light the red LED light via the IoT hub

Demonstration

- Configure an Azure Cognitive Services account for Bing Speech
- Configure an Azure Cognitive Services account for Intent Detection/Language Understanding (LUIS)
- Download and run the sample Speech app from Azure

Bing Speech – Start here - https://azure.microsoft.com/en-us/services/cognitive-services/speech/
LUIS – Start here - https://azure.microsoft.com/en-us/services/cognitive-services/language-understanding-intelligent-service/
Demonstration app - https://github.com/Azure-Samples/Cognitive-Speech-STT-Windows