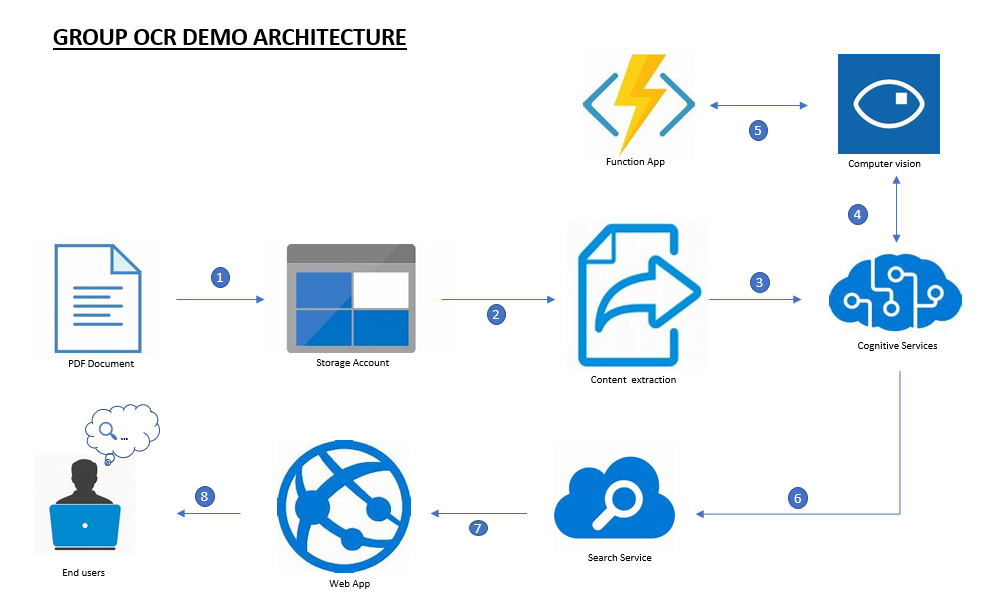
**Azure Group OCR Demo**

****

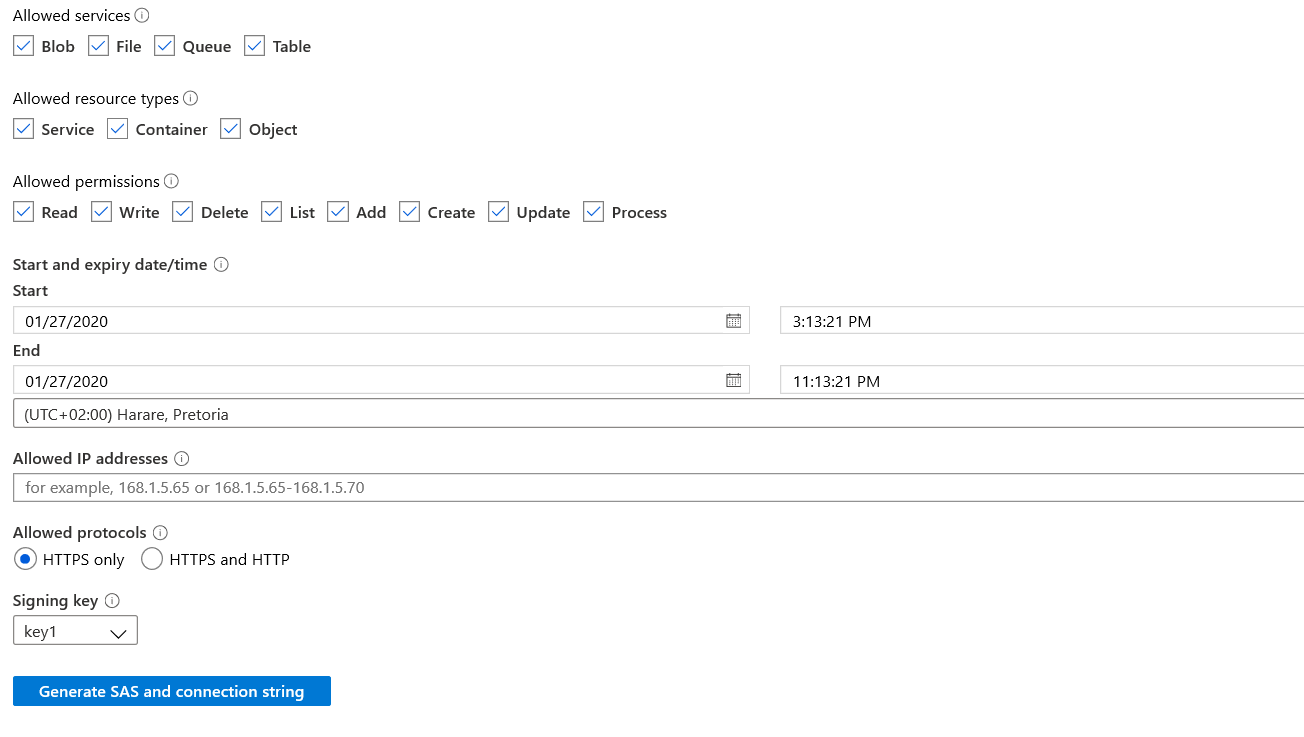
**Architecture**

1. Upload pdf documents from external source into the blob storage
2. Content of the documents is extracted and converted into images
3. We use Cognitive services to add Computer Vision (OCR) to extract text from the images
4. Use extensibility mechanism to add custom skills using Azure functions
5. Add to the Computer vision in order to send through the search service
6. Azure search service is used to index the content and power the UX experience
7. All the indexed content is sent to the website including the images from the storage account
8. Users can access the uploaded documents and search for them on a website

**Step-by-step guide on how to get the service up and running**

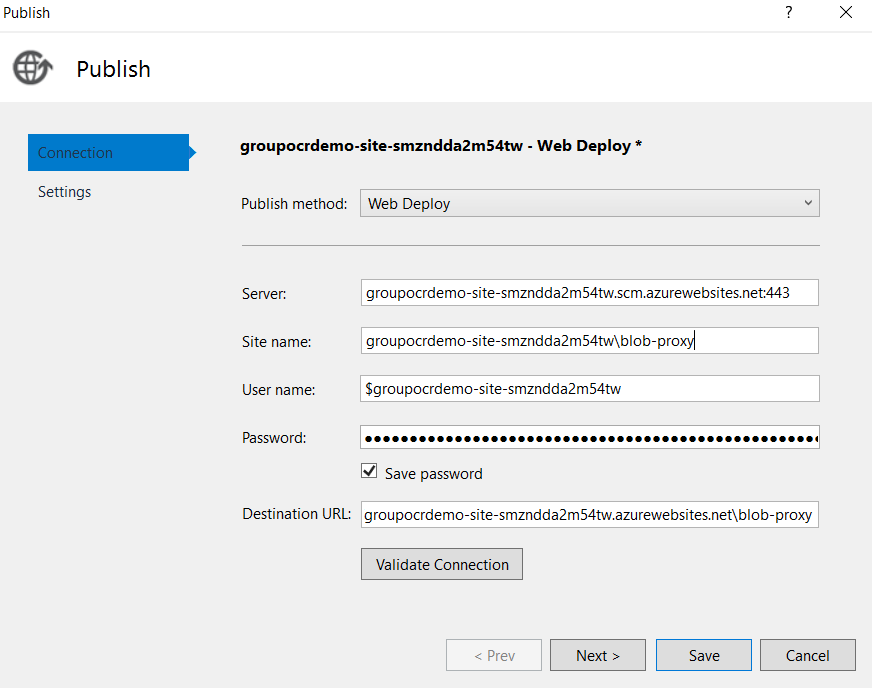
* To start this solution, we need to first install visual studio and node.js
* We will need to make use of the following services from Microsoft azure:
  + - Search service
    - Storage account
    - Computer vision
    - Function App
    - App service

1. **Deploy Azure resources:**
   1. Open the GitHub link <https://github.com/nabeelp/AzureSearch_JFK_Files>, and navigate to the bottom and click on “Deploy to Azure”.
   2. Navigate to the resource group where you deployed your resources
   3. Select deployments on the left blade
   4. Click on the recent deployment named “Microsoft.Template” and select “Outputs”
   5. Use the values from this output on the next step
2. **Prepare Visual Studio files**:
   1. Clone the repository from the above GitHub link
   2. Open the JfkWebApiSkills/JfkWebApiSkills.sln solution file in visual studio
   3. Open the app.config file in JfkInitializer project
   4. Take the settings from the deployment outputs list in step 1 above and, except for the SearchServiceQueryKey, copy the values into the appropriate settings in the app.config file
   5. For SearchServiceQueryKey, navigate to the Search service that was deployed previously in this resource group, then select “Keys” in the left blade. Copy the value in the key column in the “Manage query keys” section and paste this value into the SearchServiceQueryKey setting in the app.config file
   6. Change the JFKFilesBlobStorageAccountConnectionString and JFKFilesBlobContainerName settings
   7. The connection string can be obtained by going to the storage account that exists in the resource group, and creating a new Shared Access Signature, using settings as per the following:



* 1. Change the end date to your preference and then click on “Generate SAS and connection string”
  2. Copy the value of the “Connection string” text box and paste into the JFKFilesBlobStorageAccountConnectionString setting in the app.config file
  3. For the JFKFilesBlobContainerName setting, navigate to the “Blobs” section of the storage account and either use an existing container or create a new container here, and ensure the access level is set to private. Copy the name of this container into the JFKFikesBlobContainerName setting.
  4. Make sure the files you want to process are in this container. Save the app.config file

1. **Deploy customized functions**:
   1. The default functions are tied to the original GitHub repo, which we need to break and replace with some customizations we have done
   2. In your resource group, navigate to the Azure function resource
   3. Click on “Deployment options configured with ExternalGit” on the landing page of the function
   4. In the tab that opens, click “Disconnect”, and “Ok” to confirm
   5. While in the portal, navigate to the Overview tab and click “Configurations” in the links toward the bottom of the tab.
   6. Add a new application setting named “BlobProxyBaseUrl”, with a value of <https://[AzureWebSiteAppName].azurewebsites.net/blob-proxy>, where you replace the text [AzureWebSiteAppName] with the name of the web app in your resource group (also available in the list of values retrieved in step1). Click “OK” then “save”
   7. Return to Visual Studio, right-click on the “JFKWebApiSkills” project and select publish
   8. In the publish wizard, choose “Select existing” option and click publish.
   9. Select the appropriate subscription, resource group, and the function app that is in your resource group, e.g.:
   10. Click Ok to start the publishing process
   11. If you are prompted to upgrade the version of functions, accept and agree to this
   12. Ensure that the publishing process is completed, as indicated in the output window in Visual Studio
2. **Deploy blob-proxy service:**
   1. Navigate to your resource group in Azure, and select the web app that was deployed
   2. Select the “Console” option on the left blade (just over half-way down)
   3. In the console window that comes up type “mkdir blob-proxy” to create a new directory
   4. Navigate to configuration in the left blade, and to Path mappings in the tabs for the configuration settings and click the “New virtual application or directory” link
   5. For virtual path, enter /blob-proxy
   6. For physical path, enter site\wwwroot\blob-proxy and uncheck the “Directory” checkbox then click Ok and save (at the top of the window) to complete
   7. Switch back to Visual Studio
   8. Open the “appsettings.json” file and replace the text [BlobConnectionString] with the same connection string used for the “JFKFilesBlobStorageAccountConnectionString” setting in the JfkInitializer appsettings.json file and save the file
   9. Right-click on the “ProxyBlobFile” project and select “Publish”
   10. In the publish wizard, choose the “Select existing” option, and click Publish
   11. Select the appropriate subscription, resource group and the web app that is in your resource group then click ok
   12. We need to change the defaults so either cancel the build or allow it to complete
   13. Once complete, click “Edit” and in the pop-up add “/blob-proxy to the destination URL and the site name as follows:



* 1. Click save to close this window, then click publish to start the publishing process
  2. Ensure it is completed as indicated by the output window

1. **Run JfkInitialiser:**
   1. In the Azure portal, navigate to the storage account in your resource group
   2. In the Blobs section create a new container named “imagestoreblob”, and it should be set to private access
   3. In Visual studio, right-click on the JfkInitializer project, and select “Set as startup project”
   4. Open the app.config file again, and hit Ctrl-H to search and replace any “&” with “&amp”, then click replace all
   5. Hit F5 to start the build and rub process
   6. The process will pause at a specific point, and display “Website keys have been set. Please build the website then return here and press any key to continue”
   7. At that point, open a command prompt and navigate to the “frontend’ folder in the source code that was cloned
   8. While in the frontend folder, enter the following commands:
      1. npm install
      2. npm run build:prod
   9. Once completed, return to the JfkInitializer console app and hit a key to continue the deployment.
   10. Once completed you should now be able to open the web app and start browsing the indexed content.

**7. Acquiring client ID and client secret**

* 1. Submit a request to the AD Ops team for service principal using PowerApps available at: <https://web.powerapps.com/webplayer/app?source=email&tenantId=7369e6ec-faa6-42fa-bc0e-4f332da5b1db&appId=/providers/Microsoft.PowerApps/apps/c8c6127d-858e-4eb5-a9db-2e959aec4207>
  2. Click on New request
  3. From the top dropdown list, choose the second option (Azure Application Registration)
  4. Integration Type choose Web /API
  5. Fill in sign-on-URL
  6. Fill in APP ID URL
  7. Fill in application owner

1. **Configure your App Service to use Azure AD**
   1. the [Azure portal](https://portal.azure.com/), search for and select “**App Services”**, and then select your app.
   2. From the left navigation, select “**Authentication / Authorization”** > “**On”**.
   3. Action to take when request is not authenticated? Select “log in with Azure Active Directory.
   4. Select “**Azure Active Directory”** > “**Express”.**
   5. If you want to choose an existing app registration instead:
   6. Choose “Select Existing AD app”, then click “Azure AD App”.
   7. Choose an existing app registration and click “OK”.
   8. Select “**OK”** to register the App Service app in Azure Active Directory. A new app registration is created.