**Azure Service Installations**

**Azure Web App**

Select Resource as Web App

Subscription: prod

RG: docai-prod

Name: docai-web-prod

Unique Default Host Name: off

Publish: Code

Runtime stack: Java 17

Java web server stack: Java SE (Embedded Web Server)

OS: Linux

Region: East US 2

Linux Plan: linux-app-service-plan-prod

Pricing Plan:

Pricing Plan: Premium V3 P1V3 (195 minimum ACU/vCPU, 8 GB memory, 2 vCPU)

Application Insights: No

Microsoft Defender for Cloud: Enable

Rest select default

**Azure Functions - Python**

Hosting Option: Flex Consumption

Subscription: prod

RG: docai-prod

Name: docai-python-func-prod

Region: East US

Runtime stack: Python

Version:3.11

Instance size: 2048 MB

Storage account: docai8func8adls8prod

App Insights: No

Rest select default

**Azure Functions - C#**

Hosting Option: Flex Consumption

Subscription: prod

RG: docai-prod

Name: docai-csharp-func-prod

Region: East US

Runtime stack: .NET

Version: 8 (LTS), isolated worker model

Instance size: 2048 MB

Storage account: docai8func8adls8prod

App Insights: No

Rest select default

**Azure Logic App**

Hosting option: Workflow Service Plan

Subscription: prod

RG: docai-prod

Name: docai-logic-app-prod

Region: East US

Windows Plan: windows-app-service-plan-prod

Storage account: docai8logicapp8adls8prod

App Insights: No

Rest select default

**Azure Document Intelligence Service**

Subscription: non-prod

RG: docai-non-prod

Name: doc-intel-non-prod

Region: East US 2

Pricing Tier: Standard S0

**Azure Storage**

Subscription: prod

RG: docai-prod

Name: docai8adls8prod

Region: East US

Primary Service: Azure Blob Storage or Azure Data Lake Storage Gen 2

Redundancy: LRS

Hierarchical Namespace: Yest

Soft Deletes: disable for blobs, containers and file shares

**Azure Storage (for Document Intelligence)**

Subscription: non-prod

RG: docai-non-prod

Name: docai8doc8intel8blob

Region: East US

Primary Service: Azure Blob Storage or Azure Data Lake Storage Gen 2

Redundancy: LRS

Hierarchical Namespace: Yest

Soft Deletes: disable for blobs, containers and file shares

**Azure AI Service**

Subscription: non-prod

RG: docai-non-prod

Name: ai-non-prod

Region: East US

Pricing Tier: Standard S0

**Azure Open AI**

Subscription: non-prod

RG: docai-non-prod

Name: aoai-non-prod

Region: East US

Pricing Tier: Standard S0

**Azure Open AI (for Video)**

Subscription: non-prod

RG: docai-non-prod

Name: aoai-vision-non-prod

Region: West US

Pricing Tier: Standard S0

**Configuration**

**Azure Logic App**

* 1. Find out the storage account for this Logic App setup during creation. In this case it is docai8logicapp8adls8prod
  2. Go that Storage account in the Portal, and then go to Settings->Configuration and enable 'Allow storage account key access'

**Azure Function App**

* 1. Find out the storage account for this Function App setup during creation. In this case it is docai8func8adls8prod
  2. Go that Storage account in the Portal, and then go to Settings->Configuration and enable 'Allow storage account key access'

**Azure Open AI**

From Azure Open AI Studio, deploy

Model: gpt-4o

Name: gpt-4o

Model Version: 2024-08-06

Deployment Type: Standard

Capacity: 150k TPM

**Azure Open AI (for video)**

From Azure Open AI Studio, deploy

Model: gpt-4 vision-preview

Name: gpt-4-vision

Model Version: vision-preview

Deployment Type: Standard

Capacity: 80k TPM

API version: 2024-02-15-preview (get this from the Sample Code in Azure Open AI Studio for this model)

**Azure Storage** docai8adls8prod

* 1. Create a blob container named **docai**
  2. Generate the SAS token

A screenshot of a computer

Description automatically generated

* 1. From MSDOS command line go to the DocAI\code\python folder and run the following:
     + env.bat
     + cd cli
     + python base64Tool.py -e "<the Blob SAS Token>"
  2. Save the base 64 encoded token returned and use it as the value for the BLOB\_STORE\_SAS\_TOKEN environment variable for the Azure Web App and the Azure Python Function App
  3. In the Portal, go to Access Control (IAM) and add 'Storage Blob Data Owner' role to your admin user. This will allow the admin to view files using the Storage browser from the Portal.
  4. In the Portal, go to Settings->Configuration and enable 'Allow storage account key access'

**Azure Document Intelligence Service**

* 1. In the Storage Account docai8doc8intel8blob, create a Blob container named

doc-intel-models

* 1. Upload all the files and folder in <GitHub DocAI project root>/data/doc-intel-models

into the doc-intel-models blob container

* 1. Go to Document Intelligence Studio
  2. Select Custom classification models
  3. Create 1 new Project

Name docai-classifier

Service resource

Subscription: non-prod

RG: docai-non-prod

Document Intelligence Service: doc-intel-non-prod

API version 2024-07-31 (Preview)

Training Data Source

Subscription: non-prod

RG: docai-non-prod

Storage account: docai8doc8intel8blob

Blob container: doc-intel-models

Folder path: blank

* 1. Now Train the model

Model Name: docai-classifier-v1

* 1. Select Custom extraction models
  2. Create 3 new Projects

Name: AutoInsuranceClaim (or CommercialInsuranceApplication-AcordForm or WorkersCompensationApplication-AcordForm)

Service resource

Subscription: non-prod

RG: docai-non-prod

Document Intelligence Service: doc-intel-non-prod

API version 2024-07-31 (Preview)

Training Data Source

Subscription: non-prod

RG: docai-non-prod

Storage account: docai8doc8intel8blob

Blob container: doc-intel-models

Folder path: sample-auto-insurance-claims-docs/training (or sample-commercial-insurance-applications/training or sample-workers-compensation-applications/training)

* 1. Now Train each one of the models

Model Names:

* 1. autoInsuranceClaimExtraction-v1
  2. commercialInsuranceApplicationExtraction-v1
  3. workersCompensationApplicationExtraction-v1

**Microsoft Graph API**

* 1. Login as admin in your tenant to the Add User website - <https://portal.azure.com/#view/Microsoft_AAD_UsersAndTenants/UserManagementMenuBlade/~/AllUsers>
  2. Add the users
     + fsi-demo@<your FQDN>
     + docai@<your FQDN>
     + docai-dev@<your FQDN>
  3. Remember the passwords
  4. Enable MFA for the above users - <https://admin.microsoft.com/AdminPortal/Home?#/mfasetupguide>

A screenshot of a computer

Description automatically generated

A screenshot of a computer

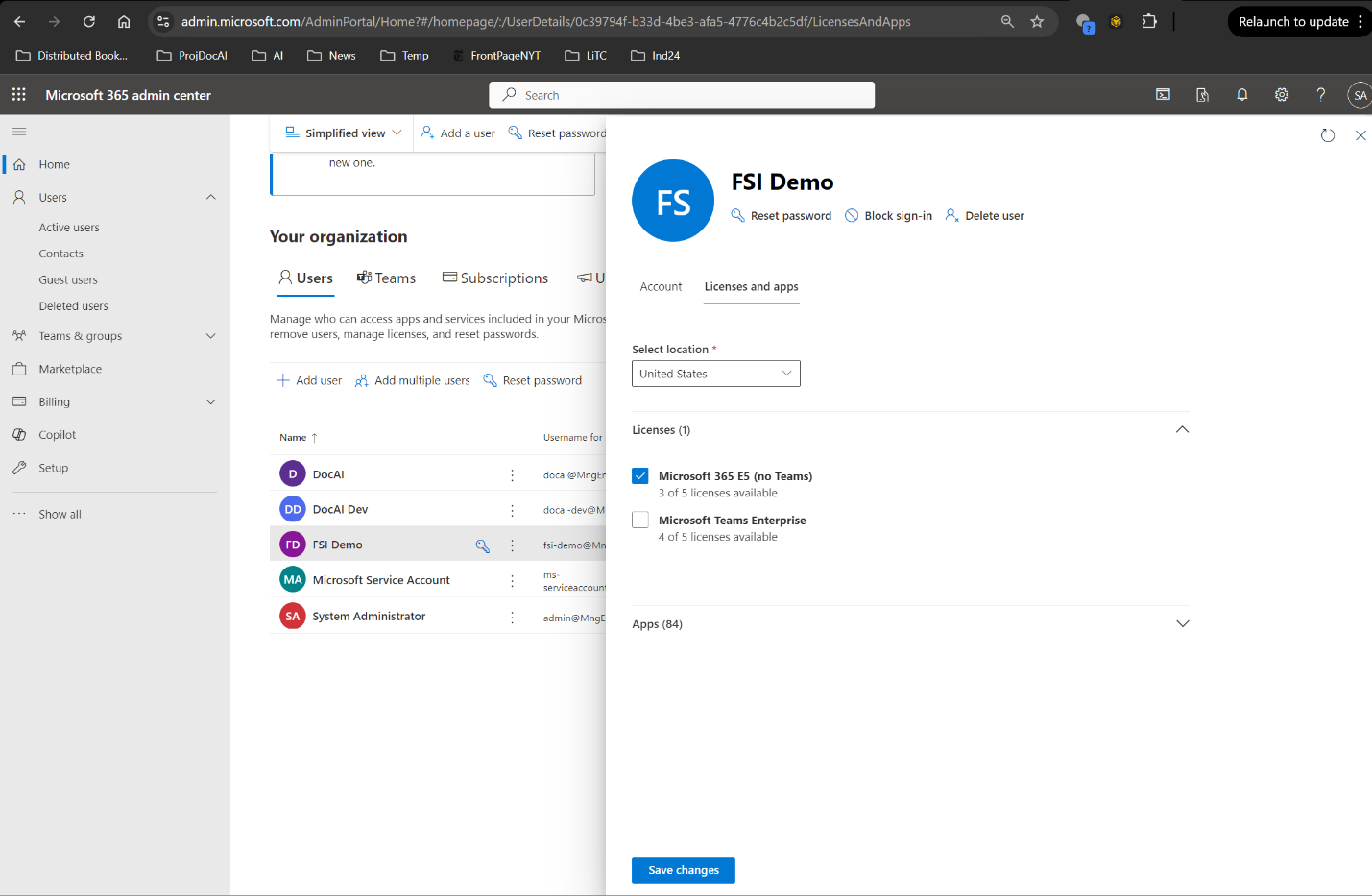
Description automatically generated

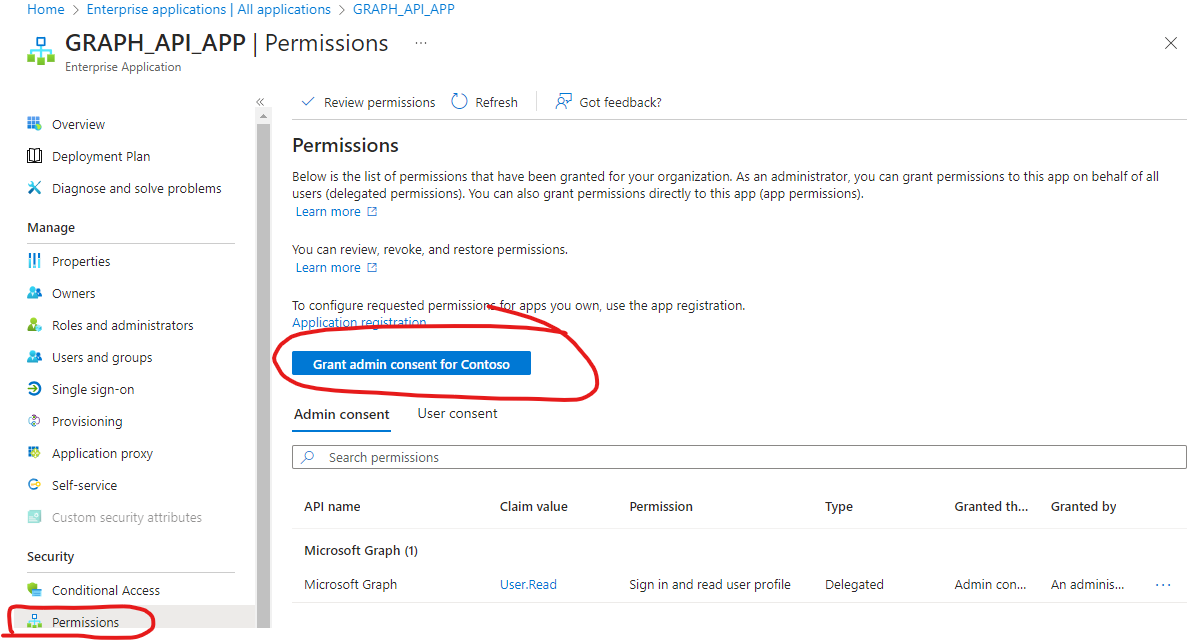
* 1. On the same above web page above, on the left side menu items go to Users -> Active Users and select Multi Factor Authentication button

A screenshot of a computer

Description automatically generated

* 1. Enable M365 license for the fsi-demo user as below:



* 1. Register your App so the java code in your DocAI Web App can send email using the Graph API. The link is here <https://learn.microsoft.com/en-us/graph/auth-register-app-v2>
     + Sign in to the MS Entra admin center - [Microsoft Entra admin center](https://entra.microsoft.com/)
     + Go to Identity > Applications > App registrations and select New registration
       1. Name : GRAPH\_API\_APP
       2. Select Accounts in any organizational directory
     + Hit Register
     + Go to Applications > Enterprise applications
       1. Select the Application GRAPH\_API\_APP
       2. A new page shows up. This page displays the details of the GRAPH\_API\_APP
       3. Go to Permissions under Security from the left side menu
       4. 
       5. Hit Grant admin consent for Contoso
     + Go to Applications > App registrations > All applications
       1. Click on GRAPH\_API\_APP in the list
       2. Click on API permissions menu item from the left side of the page
       3. Now you should see the Microsoft Graph User.Read in the list. If not, hit refresh on the dialog box in the image above
       4. You need to add the Microsoft Graph Mail.Send permission now, from Add a permission button
       5. Finally, you need to click 'Grant admin consent for Contoso' to grant permission for Mail.Send

A screenshot of a computer

Description automatically generated

* 1. Now from the M365 admin portal (<https://admin.microsoft.com/adminportal/home#/homepage>) open a Support ticket with the below message, so the fsi-demo user can send emails

"Release tenant IP not accepting traffic"

* 1. Under Certificates & secrets, add a Client secret

**Azure Web App**

* 1. Create the App Registration, so users who login to the web app are authenticated properly. The link is here <https://learn.microsoft.com/en-us/graph/auth-register-app-v2>
     + Sign in to the MS Entra admin center - [Microsoft Entra admin center](https://entra.microsoft.com/)
     + Go to Identity > Applications > App registrations and select New registration
       1. Name : DOCAI\_APP
       2. Select Accounts in any organizational directory
       3. Make sure you add the web url

(https://<hostname>.azurewebsites.net/login/oauth2/code/)

* 1. Add <http://localhost:8080/login/oauth2/code/> (needed when running in your dev machine)
  2. Hit Register
  3. Now add the Microsoft Graph Mail.Send permission under API Permissions
  4. A screenshot of a computer

     Description automatically generated
  5. Under Certificates & secrets, add a Client secret
  6. Change all the Environment variables

**Azure Bing Search 7 API**

Subscription: non-prod

RG: docai-non-prod

Name: bing-search-non-prod

Region: Global

Pricing Tier: Free F1

Gazu3498

**Deployment**

**Azure Web App**

Go to DocAI\code\web-apps\DocumentAIManagementApp and run:

* 1. env.bat prod
  2. copy pom.xml.runLocal pom.xml
  3. mvn package spring-boot:run
  4. In Eclipse IDE go to Run > Debug Configurations > and hit Debug

A screenshot of a computer

Description automatically generated

Then from the browser go to localhost:8080

If the web app is working then you are ready to deploy in production

Then run the following to deploy in production

* 1. copy pom.xml.runLocal pom.xml
  2. mvn install "-DRESOURCEGROUP\_NAME=DocAI" "-DWEBAPP\_NAME=tr-docai-web" "-DREGION=East US" azure-webapp:deploy
  3. This should pop up an Azure Login browser window. Login to your Azure portal.
  4. This will now deploy the web app in production

Once complete, make sure you configure all the environment variables for the Web App correctly

**Azure Function App (python)**

Start VSCode and open folder DocAI\code\python\function

Make sure the variables in local.settings.json are correct. Example below:

{

  "IsEncrypted": false,

  "Values": {

    "DOCUMENT\_EXTRACTION\_MODEL\_CLASS\_MAP": "[{'unknown':'unknown'},{'auto-insurance-claim':'autoInsuranceClaimExtraction-v1'},{'commercial-insurance-application':'commercialInsuranceApplicationExtraction-v1'},{'workers-compensation-application':'workersCompensationApplicationExtraction-v1'}]",

    "DOCUMENT\_CLASSIFIER\_ID": "docai-classifier-v1",

    "DOCUMENT\_CONFIDENCE\_THRESHOLD": "0.9",

    "CosmosDbConnectionString": "AccountEndpoint=https://<fqdn for cosmosdb>:443/;AccountKey=<key>;",

    "COGNITIVE\_SERVICE\_ENDPOINT": "<https://ai-vision-non-prod.cognitiveservices.azure.com/>",

    "COGNITIVE\_SERVICE\_KEY": "<key>",

    "FUNCTIONS\_WORKER\_RUNTIME": "python"

  }

}

Hit F5 to run the functions locally.

Once it works, deploy the functions in Azure.

Now, make sure all the environment variables are set in Azure Functions instance

**Azure Function App (python)**

Start VSCode and open folder DocAI\code\csharp\function

Hit f5 to test locally

If it works, then deploy in Azure.

**Azure Logic App**

* 1. From portal expand the Workflows menu and click on [@] Parameters and add the following:

{

    "SUBJECT\_PREFIX\_LOGIC\_APP\_TRIGGER": {

        "type": "String",

        "value": "docai"

    }

}

* 1. Save Parameter
  2. Create a workflow named test
  3. Select Stateful
  4. Add Office 365 when new email arrives as the trigger and set the connection to
     + docai@...
  5. Add an action to setup the connection to the Bob Store from above. Use Managed Identity
  6. Create a workflow named *docAIEmailProcessingLogicApp*
  7. Select Stateful
  8. Open it under workflow -> *docAIEmailProcessingLogicApp -> Code ->*
  9. Copy and paste the code from the github repo under <repo home>\code\logic-apps\docAIEmailProcessingLogicApp\workflow.json
  10. Replace in the workflow.json the name of the placeholder blob store with *docai8adls8prod*
  11. Create the connections for Office365, Blob and Functions and fix the workflow in designer
  12. Check all the functions and make sure the urls for blob store have the right hostname
  13. Make sure the trigger condition for New Email module matches what is in the Web APP application.properties (or the respective environment variable)

**Setup Managed Identity**

* 1. Azure Open AI Services (aoai-non-prod):

Add **Cognitive Services OpenAI Contributor role** for the Management Identity for

* 1. Azure Function App (docai-python-func-prod)
  2. Azure Open AI Services (aoai-vision-non-prod):

Add **Cognitive Services OpenAI Contributor role** for the Management Identity for

* 1. Azure Function App (docai-python-func-prod)
  2. Azure Storage (docai8adls8prod):

Add **Storage Blob Data Contributor role** for the Management Identity for

* 1. Azure Open AI (aoai-non-prod)
  2. Azure Document Intelligence Service (doc-intel-non-prod)
  3. Azure Logic App (docai-logic-app-prod)