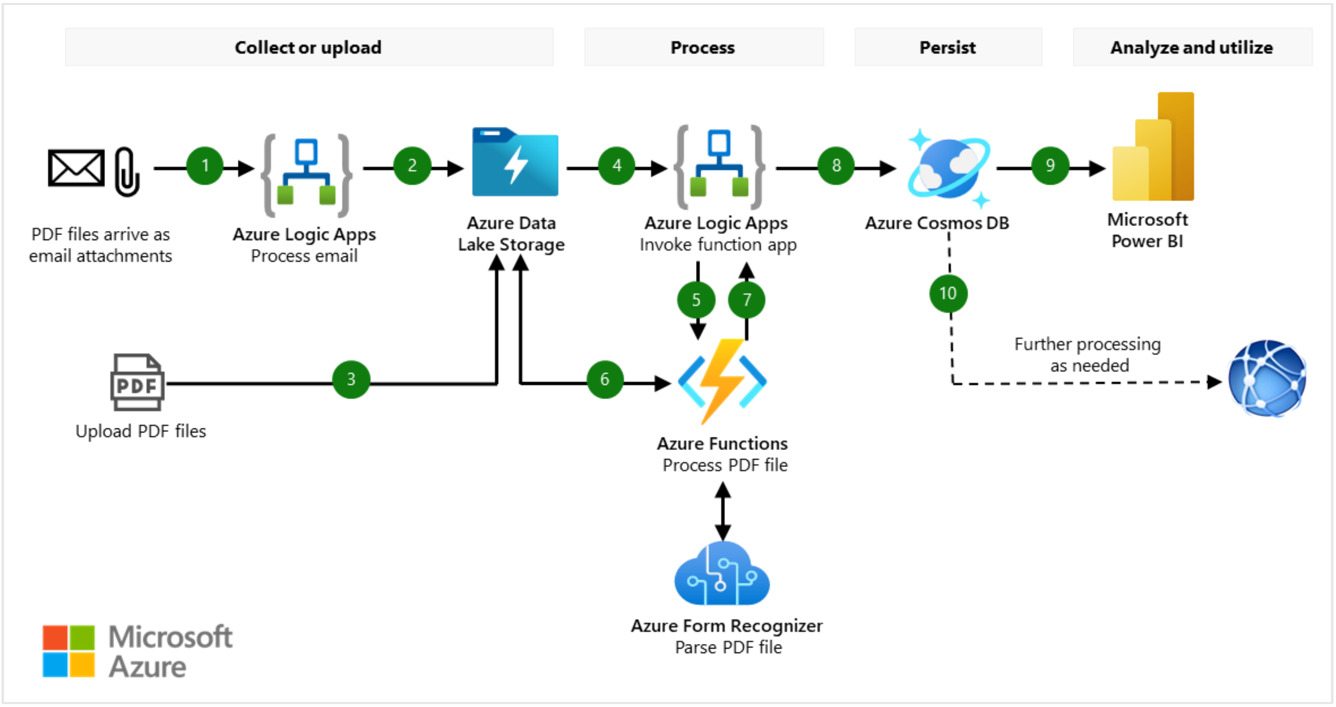
Automate PDF forms processing

Azure AI Document Intelligence   Azure AI services   Azure Logic Apps   Azure Functions

This article describes an Azure architecture that you can use to replace costly and inflexible forms processing methods with cost-effective and flexible automated PDF processing.

Architecture


Collect or upload
Process
Persist
Analyze and utilize
PDF files arrive as email attachments
Azure Logic Apps Process email
Azure Data Lake Storage
Azure Logic Apps Invoke function app
Azure Cosmos DB
Microsoft Power BI
Upload PDF files
Azure Functions Process PDF file
Further processing as needed
Azure Form Recognizer Parse PDF file
Microsoft Azure
1
2
3
4
5
6
7
8
9
10

Download a PowerPoint file of this architecture.

Workflow

1. A designated Outlook email account receives PDF files as attachments. The arrival of an email triggers a logic app to process the email. The logic app is built by using the capabilities of Azure Logic Apps.

2. The logic app uploads the PDF files to a container in Azure Data Lake Storage.
3. You can also manually or programmatically upload PDF files to the same PDF container.
4. The arrival of a PDF file in the PDF container triggers another logic app to process the PDF forms that are in the PDF file.
5. The logic app sends the location of the PDF file to a function app for processing. The function app is built by using the capabilities of Azure Functions.
6. The function app receives the location of the file and takes these actions:
   a. It splits the file into single pages if the file has multiple pages. Each page contains one independent form. Split files are saved to a second container in Data Lake Storage.
   b. It uses HTTPS POST, an Azure REST API, to send the location of the single-page PDF file to AI Document Intelligence for processing. When Azure AI Document Intelligence completes its processing, it sends a response back to the function app, which places the information into a data structure.
   c. It creates a JSON data file that contains the response data and stores the file to a third container in Data Lake Storage.
7. The forms processing logic app receives the processed response data.
8. The forms processing logic app sends the processed data to Azure Cosmos DB, which saves the data in a database and in collections.
9. Power BI obtains the data from Azure Cosmos DB and provides insights and dashboards.
10. You can implement further processing as needed on the data that's in Azure Cosmos DB.

Components

- Azure AI services is a category of Azure AI products that use Azure AI services, task-specific AI, and business logic to provide turnkey AI services for common business processes. One of these products is Azure AI Document Intelligence , which uses machine learning models to extract key-value pairs, text, and tables from documents.
- Azure Logic Apps is a serverless cloud service for creating and running

automated workflows that integrate apps, data, services, and systems.
- Azure Functions is a serverless solution that makes it possible for you to write less code, maintain less infrastructure, and save on costs.
- Azure Data Lake Storage is the foundation for building enterprise data lakes on Azure.
- Azure Cosmos DB is a fully managed NoSQL and relational database for modern app development.
- Power BI is a collection of software services, apps, and connectors that work together so that you can turn your unrelated sources of data into coherent, visually immersive, and interactive insights.

Alternatives

- You can use Azure SQL Database instead of Azure Cosmos DB to store the processed forms data.
- You can use Azure Data Explorer to visualize the processed forms data that's stored in Data Lake Storage.

Scenario details

Forms processing is often a critical business function. Many companies still rely on manual processes that are costly, time consuming, and prone to error. Replacing manual processes reduces cost and risk and makes a company more agile.

This article describes an architecture that you can use to replace manual PDF forms processing or costly legacy systems that automate PDF forms processing. Azure AI Document Intelligence processes the PDF forms, Logic Apps provides the workflow, and Functions provides data processing capabilities.

For deployment information, see Deploy this scenario in this article.

Potential use cases

The solution that's described in this article can process many types of forms, including:

- Invoices
- Payment records
- Safety records
- Incident records
- Compliance records
- Purchase orders
- Payment authorization forms
- Health screening forms
- Survey forms

Considerations

These considerations implement the pillars of the Azure Well-Architected Framework, a set of guiding tenets that you can use to improve the quality of a workload. For more information, see Microsoft Azure Well-Architected Framework.

Reliability

Reliability ensures that your application can meet the commitments that you make to your customers. For more information, see Overview of the reliability pillar.

A reliable workload is one that's both resilient and available. *Resiliency* is the ability of the system to recover from failures and continue to function. The goal of resiliency is to return the application to a fully functioning state after a failure occurs. *Availability* is a measure of whether your users can access your workload when they need to.

This architecture is intended as a starter architecture that you can quickly deploy and prototype to provide a business solution. If your prototype is a success, you can then extend and enhance the architecture, if necessary, to meet additional requirements.

This architecture utilizes scalable and resilient Azure infrastructure and technologies. For example, Azure Cosmos DB has built-in redundancy and global coverage that you can configure to meet your needs.

For the availability guarantees of the Azure services that this solution uses, see Service-level agreements (SLAs) for Online Services .

Security

Security provides assurances against deliberate attacks and the abuse of your valuable data and systems. For more information, see Overview of the security pillar.

The Outlook email account that's used in this architecture is a dedicated email account that receives PDF forms as attachments. It's good practice to limit the senders to trusted parties only and to prevent malicious actors from spamming the email account.

The implementation of this architecture that's described in Deploy this scenario takes the following measures to increase security:

- The PowerShell and Bicep deployment scripts use Azure Key Vault to store sensitive information so that it isn't displayed on terminal screens or stored in deployment logs.
- Managed identities provide an automatically managed identity in Microsoft Entra ID for applications to use when they connect to resources that support Microsoft Entra authentication. The function app uses managed identities so that the code doesn't depend on individual principals and doesn't contain sensitive identity information.

Cost optimization

Cost optimization is about looking at ways to reduce unnecessary expenses and to improve operational efficiencies. For more information, see Overview of the cost optimization pillar.

Here are some guidelines for optimizing costs:

- Use the pay-as-you-go strategy for your architecture, and scale out as needed rather than investing in large-scale resources at the start.

- The implementation of the architecture that's described in Deploy this scenario deploys a starting solution that's suitable for proof of concept. The deployment scripts create a working architecture with minimal resource requirements. For example, the deployment scripts create a smallest serverless Linux host to run the function app.

Performance efficiency

Performance efficiency is the ability of your workload to scale in an efficient manner to meet the demands that are placed on it by users. For more information, see Performance efficiency pillar overview.

This architecture uses services that have built-in scaling capabilities that you can use to improve performance efficiency. Here are some examples:

- You can host both Azure Logic Apps and Azure Functions in a serverless infrastructure. For more information, see Azure serverless overview: Create cloud-based apps and solutions with Azure Logic Apps and Azure Functions.
- You can configure Azure Cosmos DB to automatically scale its throughput. For more information, see Provision autoscale throughput on a database or container in Azure Cosmos DB - API for NoSQL.

Deploy this scenario

You can deploy a rudimentary version of this architecture, a *solution accelerator*, and use it as a starting point for deploying your own solution. The reference implementation for the accelerator includes code, deployment scripts, and a deployment guide.

The accelerator receives the PDF forms, extracts the data fields, and saves the data in Azure Cosmos DB. Power BI visualizes the data. The design uses a modular, metadata-driven methodology. No form fields are hard-coded. It can process any PDF forms.

You can use the accelerator as is, without code modification, to process and visualize any single-page PDF forms such as safety forms, invoices, incident records, and many others. To use it, you only need to collect sample PDF forms, train a new model to learn the layout of the forms, and plug the model into the solution. You also need to redesign the Power BI report for your datasets so that it provides the insights that you want.

The implementation uses Azure AI Document Intelligence Studio to create custom models. The accelerator uses the field names that are saved in the machine learning model as a reference to process other forms. Only five sample forms are needed to create a custom-built machine learning model. You can merge as many as 100 custom-built models to create a composite machine learning model that can process a variety of forms.

Deployment repository

The GitHub repository for the solution accelerator is at Azure PDF Form Processing Automation Solution Accelerator which contains the deployment guide for this solution.

Deployment prerequisites

To deploy, you need an Azure subscription. For information about free subscriptions, see Build in the cloud with an Azure free account.

To learn about the services that are used in the accelerator, see the overview and reference articles that are listed in:

- AI Document Intelligence documentation
- Azure Logic Apps documentation
- Azure Functions documentation
- Introduction to Azure Data Lake Storage Gen2
- Azure Cosmos DB documentation
- Power BI documentation

Deployment considerations

To process a new type of PDF form, you use sample PDF files to create a new machine learning model. When the model is ready, you plug the model ID into the solution.

This container name is configurable in the deployment scripts that you get from the GitHub repository.

The architecture doesn't address any high availability (HA) or disaster recovery (DR) requirements. If you want to extend and enhance the current architecture for production deployment, consider the following recommendations and best practices:

- Design the HA/DR architecture based on your requirements and use the built-in redundancy capabilities where applicable.
- Update the Bicep deployment code to create a computing environment that can handle your processing volumes.
- Update the Bicep deployment code to create more instances of the architecture components to satisfy your HA/DR requirements.
- Follow the guidelines in Azure Storage redundancy when you design and provision storage.
- Follow the guidelines in Business continuity and disaster recovery when you design and provision the logic apps.
- Follow the guidelines in Reliability in Azure Functions when you design and provision the function app.
- Follow the guidelines in Achieve high availability with Azure Cosmos DB when you design and provision a database that was created by using Azure Cosmos DB.
- If you consider putting this system into production to process large volumes of PDF forms, you can modify the deployment scripts to create a Linux host that has more resources. To do so, modify the code inside deploy-functionsapp.bicep

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Gail Zhou | Sr. Architect

Other contributors:

- Nalini Chandhi | Principal Technical Specialist
- Steve DeMarco | Sr. Cloud Solution Architect
- Travis Hilbert | Technical Specialist Global Black Belt
- DB Lee | Sr. Technical Specialist
- Malory Rose | Technical Specialist Global Black Belt
- Oscar Shimabukuro | Sr. Cloud Solution Architect
- Echo Wang | Principal Program Manager

To see nonpublic LinkedIn profiles, sign in to LinkedIn.

Next steps

- Video: Azure PDF Form Processing Automation .
- Azure PDF Form Processing Automation Solution Accelerator
- Azure invoice Process Automation Solution Accelerator
- Business Process Automation Accelerator
- Tutorial: Create workflows that process emails using Azure Logic Apps, Azure Functions, and Azure Storage

Related resources

- Custom document processing models on Azure
- Index file content and metadata by using Azure Cognitive Search
- Automate document identification, classification, and search by using Durable Functions
- Automate document processing by using Azure AI Document Intelligence

Feedback