Event-Driven Invoice PDF Generation

using Azure Function (.NET 8 In-Process), Service Bus & Blob Storage

1 Problem Scenario

Business Requirement

When a customer places an order:

- An invoice must be generated as a PDF
- PDF should be stored securely
- Process must not block the main application
- System should be scalable and reliable

Traditional Problem

If invoice generation happens inside the main API:

- API becomes slow
- Heavy PDF processing blocks request thread
- Poor scalability
- System tightly coupled

Proposed Cloud Solution

We use:

- **Azure Service Bus Queue**
- **Azure Function (Queue Trigger)**
- **QuestPDF for PDF generation**
- **Azure Blob Storage**

2 Architecture

```
Client / Order API
        ↓
Service Bus Queue (p1-invoicepdf)
        ↓
Azure Function (GenerateInvoicePdf)
```

```
        ↓
Generate PDF using QuestPDF
        ↓
Upload PDF to Azure Blob Storage
        ↓
Invoice Ready
```

Why This Architecture?

- Event-driven
- Asynchronous processing
- Loose coupling
- Scalable
- Retry + Dead Letter support
- Cloud-native

3 Technology Stack

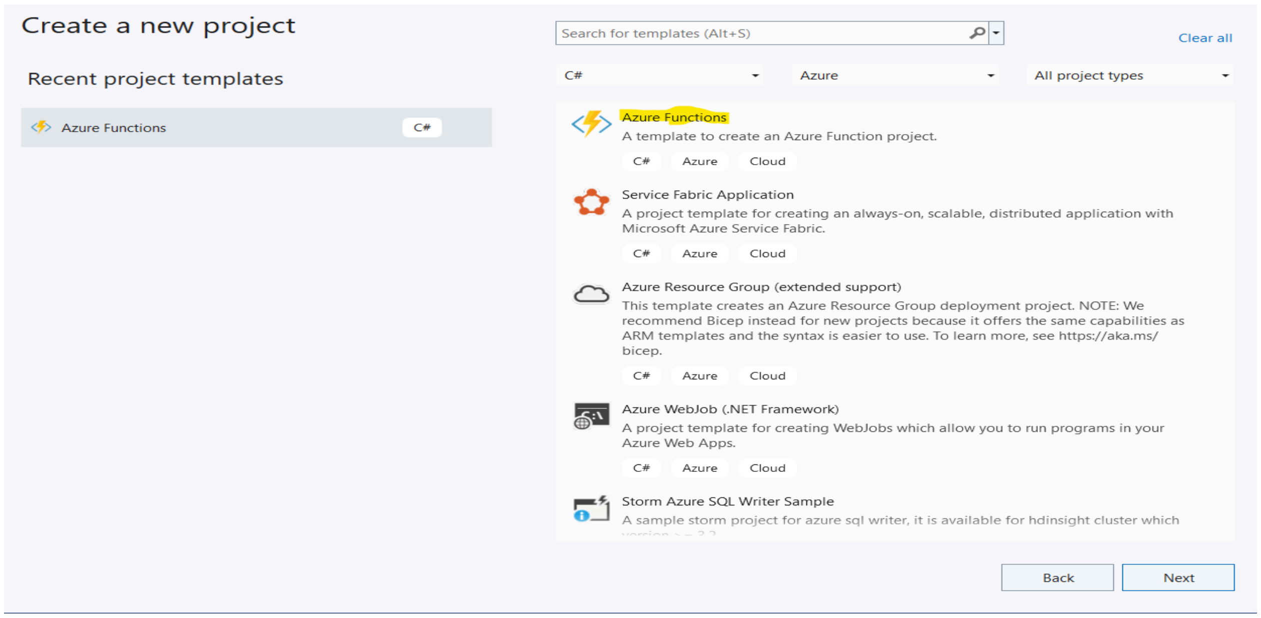
- .NET 8 (In-Process – LTS)
- Azure Functions
- Azure Service Bus
- Azure Blob Storage
- QuestPDF
- C#

4 Create Azure Function App (Visual Studio)

Step 1: Create Project

1. Open Visual Studio
2. Create New Project
3. Select **Azure Functions**
4. Target Framework → .NET 8.0
5. Worker Model → **In-Process**
6. Trigger → Service Bus Queue Trigger
7. Function Name → GenerateInvoicePdf
8. Queue Name → p1-invoicepdf
9. Connection → ServiceBusConnection

Click Create.


Create a new project

Recent project templates

Azure Functions C#

Search for templates (Alt+S)

Clear all

C# Azure All project types

Azure Functions
A template to create an Azure Function project.
C# Azure Cloud

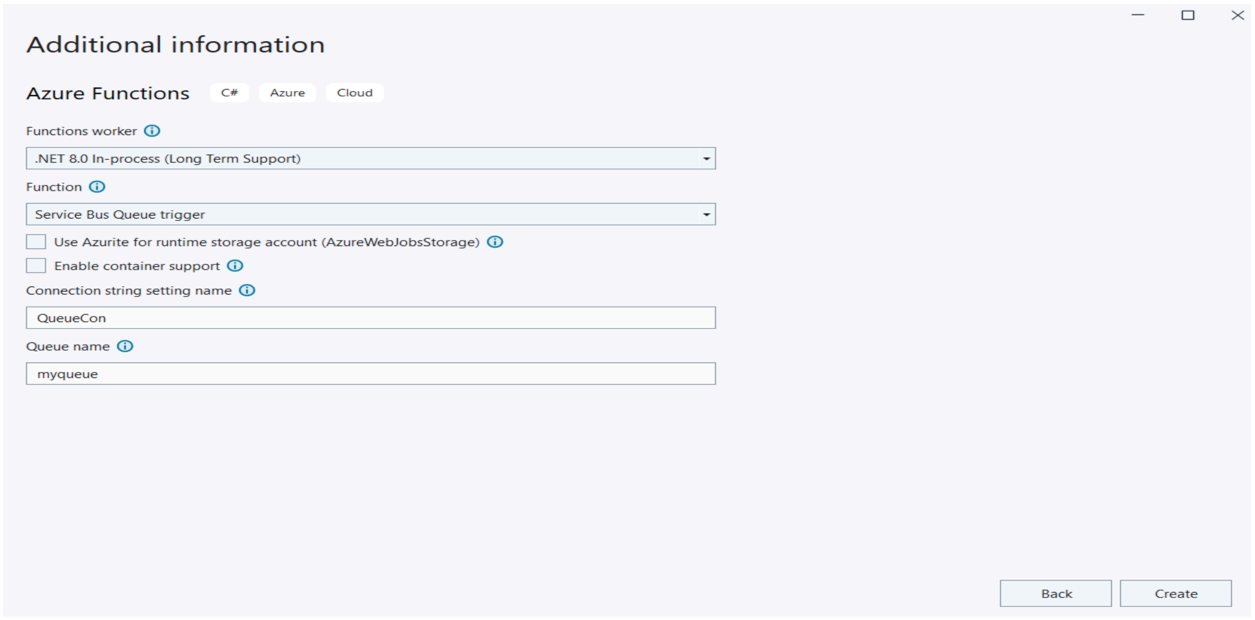
Service Fabric Application
A project template for creating an always-on, scalable, distributed application with Microsoft Azure Service Fabric.
C# Azure Cloud

Azure Resource Group (extended support)
This template creates an Azure Resource Group deployment project. NOTE: We recommend Bicep instead for new projects because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see https://aka.ms/bicep.
C# Azure Cloud

Azure WebJob (.NET Framework)
A project template for creating WebJobs which allow you to run programs in your Azure Web Apps.
C# Azure Cloud

Storm Azure SQL Writer Sample
A sample storm project for azure sql writer, it is available for hdinsight cluster which

Back Next


Additional information

Azure Functions C# Azure Cloud

Functions worker: .NET 8.0 In-process (Long Term Support)

Function: Service Bus Queue trigger

☐ Use Azurite for runtime storage account (AzureWebJobsStorage)

☐ Enable container support

Connection string setting name: QueueCon

Queue name: myqueue

Back Create

5 Folder Structure

```
InvoiceFunctionApp
│
├── Functions
│    └── GenerateInvoicePdf.cs
│
├── Models
│    ├── InvoiceRequest.cs
│    └── InvoiceItem.cs
│
├── Services
│    ├── PdfService.cs
│    └── BlobService.cs
│
├── Templates
│    └── CompanyLogo.png
│
├── Program.cs
├── local.settings.json
└── InvoiceFunctionApp.csproj
```

6 Required NuGet Packages

Install via Package Manager Console:

```
Install-Package Azure.Messaging.ServiceBus
Install-Package Azure.Storage.Blobs
Install-Package QuestPDF
Install-Package Microsoft.Azure.WebJobs.Extensions.ServiceBus
```

7 Sample Request JSON (Queue Message)

```
{
  "invoiceNo": "INV-101",
  "customerName": "Renuka N",
  "items": [
    { "itemName": "Laptop", "quantity": 1, "price": 50000 },
    { "itemName": "Mouse", "quantity": 2, "price": 500 }
  ]
}
```

8 Create Model

Models/InvoiceMessage.cs

```
namespace InvoicePdfProcessor.Function.Models
{
 public class InvoiceRequest
 {
     public string InvoiceNo { get; set; }
     public string CustomerName { get; set; }
     public List<InvoiceItem> Items { get; set; }
 }
}
 public class InvoiceItem
{
    public string ItemName { get; set; }
    public string HSNCode { get; set; }
    public int Quantity { get; set; }
    public decimal Price { get; set; }
}
```

9 Create PDF Service (Using QuestPDF)

Services/PdfService.cs

```
using InvoicePdfGenerationQueue.Processor.Function.Models;
using QuestPDF.Fluent;
using QuestPDF.Helpers;
using QuestPDF.Infrastructure;

namespace InvoicePdfGenerationQueue.Processor.Function.Services;

public class PdfService
{
    public MemoryStream GenerateInvoice(InvoiceRequest invoice)
    {
        QuestPDF.Settings.License = LicenseType.Community;

        var stream = new MemoryStream();

        Document.Create(container =>
        {
            container.Page(page =>
            {
                page.Size(PageSizes.A4);
                page.Margin(25);
                page.DefaultTextStyle(x => x.FontSize(10));
                page.Header().Element(c => ComposeHeader(c, invoice));
                page.Content().Element(c => ComposeContent(c, invoice));
                page.Footer().AlignCenter().Text("This is a computer
generated GST Invoice");
            });

        }).GeneratePdf(stream);

        stream.Position = 0;
        return stream;
    }
    private void ComposeHeader(IContainer container, InvoiceRequest
invoice)
    {
        container.Row(row =>
        {
            row.RelativeItem().Column(col =>
            {
                col.Item().Text("ABC TECHNOLOGIES PVT LTD")
                    .FontSize(16).Bold();

                col.Item().Text("GSTIN: 27ABCDE1234F1Z5");
                col.Item().Text("Place of Supply: Maharashtra");
                col.Item().Text("Email: support@abc.com");
            });

            row.ConstantItem(100)
                .Height(60)
                .Element(e =>
                {
                    var logoPath = Path.Combine(
                        Directory.GetCurrentDirectory(),
                        "Templates",
                        "companylogo.png"
                    );

                    if (File.Exists(logoPath))
                    {
                        e.Image(logoPath);
                    }
                    else
                    {
                        // Prevent function crash if image missing
                        e.AlignCenter().AlignMiddle()
                         .Text("Logo")
                         .FontSize(10);
                    }
                });
        });
    }

      private void ComposeContent(IContainer container, InvoiceRequest
invoice)
    {
        decimal subtotal = invoice.Items.Sum(x => x.Price * x.Quantity);
        decimal cgst = subtotal * 0.09m;
        decimal sgst = subtotal * 0.09m;
        decimal grandTotal = subtotal + cgst + sgst;

        container.Column(column =>
        {
            column.Spacing(5);

            // Invoice Info
            column.Item().Row(row =>
            {
                row.RelativeItem().Text($"Invoice No:
{invoice.InvoiceNo}").Bold();
                row.RelativeItem().AlignRight()
                    .Text($"Date: {DateTime.Now:dd-MM-yyyy}");
            });

            column.Item().Text($"Bill To: {invoice.CustomerName}");

            column.Item().PaddingVertical(10);

            // Item Table
            column.Item().Table(table =>
            {
                table.ColumnsDefinition(columns =>
                {
                    columns.RelativeColumn(3);
                    columns.ConstantColumn(70);
                    columns.ConstantColumn(50);
                    columns.ConstantColumn(80);
                    columns.ConstantColumn(80);
                });

                table.Header(header =>
                {
                    header.Cell().Text("Item").Bold();
                    header.Cell().Text("HSN/SAC").Bold();
                    header.Cell().Text("Qty").Bold();
                    header.Cell().Text("Rate").Bold();
                    header.Cell().Text("Amount").Bold();
                });

                foreach (var item in invoice.Items)
                {
                    table.Cell().Text(item.ItemName);
                    table.Cell().Text(item.HSNCode);
                    table.Cell().Text(item.Quantity.ToString());
                    table.Cell().Text($"₹{item.Price:N2}");
                    table.Cell().Text($"₹{item.Price * item.Quantity:N2}");
                }
            });

            column.Item().PaddingTop(10);

            // GST Summary
            column.Item().AlignRight().Column(totals =>
            {
                totals.Item().Text($"Taxable Amount: ₹{subtotal:N2}");
                totals.Item().Text($"CGST (9%): ₹{cgst:N2}");
                totals.Item().Text($"SGST (9%): ₹{sgst:N2}");
                totals.Item().Text($"Grand Total: ₹{grandTotal:N2}")
                    .FontSize(12).Bold();
            });

            column.Item().PaddingTop(10);
            column.Item().Text($"Amount in Words:
{NumberToWords((int)grandTotal)} Only")
                .Italic();

            column.Item().PaddingTop(20);
            column.Item().Row(row =>
            {
                row.RelativeItem().Text("Bank Details:\nA/C No:
1234567890\nIFSC: HDFC0000123");

                row.RelativeItem().AlignRight().Column(sig =>
                {
                    sig.Item().Text("For ZyberPlus Technologies Pvt Ltd");
                    sig.Item().Height(50);
                    sig.Item().Text("Authorized Signatory").Bold();
                });
            });

            // Page Break for Page 2
            column.Item().PageBreak();

            column.Item().Text("Terms & Conditions").FontSize(14).Bold();

            column.Item().Text("1. Goods once sold will not be taken
back.");
            column.Item().Text("2. Payment due within 15 days.");
            column.Item().Text("3. Subject to Mumbai jurisdiction.");
        });
    }

    // Convert number to words (basic)
    private string NumberToWords(int number)
    {
        if (number == 0) return "Zero";

        var units = new[]
        {
            "", "One", "Two", "Three", "Four", "Five", "Six",
            "Seven", "Eight", "Nine", "Ten", "Eleven",
            "Twelve", "Thirteen", "Fourteen", "Fifteen",
            "Sixteen", "Seventeen", "Eighteen", "Nineteen"
        };

        var tens = new[]
        {
            "", "", "Twenty", "Thirty", "Forty",
            "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"
        };

        if (number < 20)
            return units[number];

        if (number < 100)
            return tens[number / 10] + " " + units[number % 10];

        if (number < 1000)
            return units[number / 100] + " Hundred " + NumberToWords(number
% 100);

        if (number < 100000)
            return NumberToWords(number / 1000) + " Thousand " +
NumberToWords(number % 1000);

        return number.ToString();
    }
}
```

10 Blob Storage Service

Services/BlobStorageService.cs

```
using Azure.Storage.Blobs;
using Azure.Storage.Blobs.Models;

namespace InvoicePdfGenerationQueue.Processor.Function.Services
{
    public class BlobStorageService
    {
        private readonly BlobContainerClient _containerClient;

        public BlobStorageService(string connectionString, string
containerName)
        {
            var blobServiceClient = new
BlobServiceClient(connectionString);
            _containerClient =
blobServiceClient.GetBlobContainerClient(containerName);
        }
        public async Task UploadPdfAsync(string fileName, Stream content)
        {
            await _containerClient.CreateIfNotExistsAsync();

            var blobClient = _containerClient.GetBlobClient(fileName);

            content.Position = 0;

            await blobClient.UploadAsync(
                content,
                overwrite: true
            );

            await blobClient.SetAccessTierAsync(AccessTier.Cool);
        }
    }
}
```

11 Service Bus Trigger Function

GenerateInvoicePdf.cs

```
using InvoicePdfGenerationQueue.Processor.Function.Models;
using InvoicePdfGenerationQueue.Processor.Function.Services;
using Microsoft.Azure.WebJobs;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;
using System.Text.Json;


namespace InvoicePdfGenerationQueue.Processor.Function
{
    public class GenerateInvoicePdf
    {
        private readonly PdfService _pdfService;
       // private readonly InvoiceTemplateService _templateService;
        private readonly BlobStorageService _blobStorageService;
        public GenerateInvoicePdf(
            PdfService pdfService,
            BlobStorageService blobStorageService)
        {
            _pdfService = pdfService;
            _blobStorageService = blobStorageService;
        }


        [FunctionName("GenerateInvoicePdf")]
        public async Task Run([ServiceBusTrigger("p1-invoicepdf",
Connection = "QueueCon")] string message, ILogger log)
        {
            log.LogInformation("Service Bus message received");

            var invoice =
JsonConvert.DeserializeObject<InvoiceRequest>(message);

            using var pdfStream = _pdfService.GenerateInvoice(invoice);

            await _blobStorageService.UploadPdfAsync(
                $"Invoice_{invoice.InvoiceNo}.pdf",
                pdfStream
            );

            log.LogInformation("Invoice PDF generated and uploaded
successfully");
        }
    }

}
```

12 local.settings.json

```
{
    "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "",
    "FUNCTIONS_INPROC_NET8_ENABLED": "1",
    "FUNCTIONS_WORKER_RUNTIME": "dotnet",
    "QueueCon": "YOUR_SERVICE_BUS_CONNECTION_STRING",
    "BlobStorageConnection": "YOUR_BLOB_CONNECTION_STRING",
    "InvoiceContainerName": "invoices"
  }
}
```

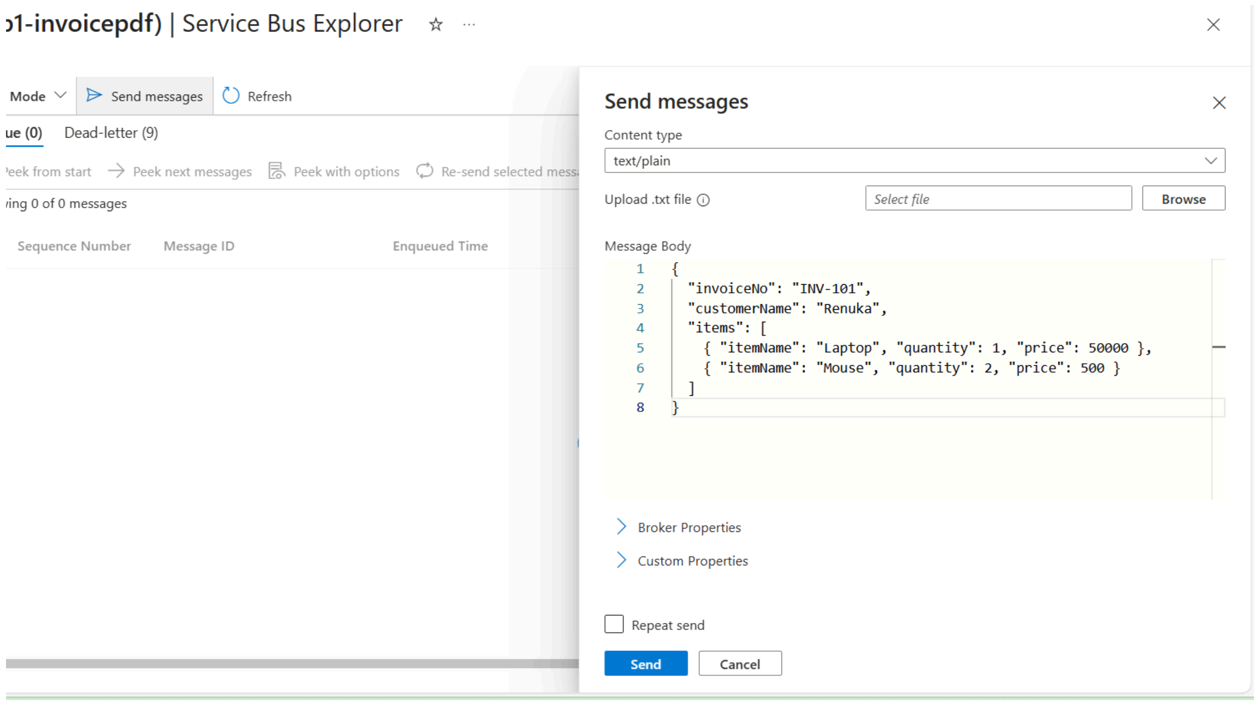
13 How to Run

Step 1:

F5 (Run in Visual Studio)

Step 2:

Send message to Service Bus queue:
Queue Name: p1-invoicepdf
Paste Sample JSON.



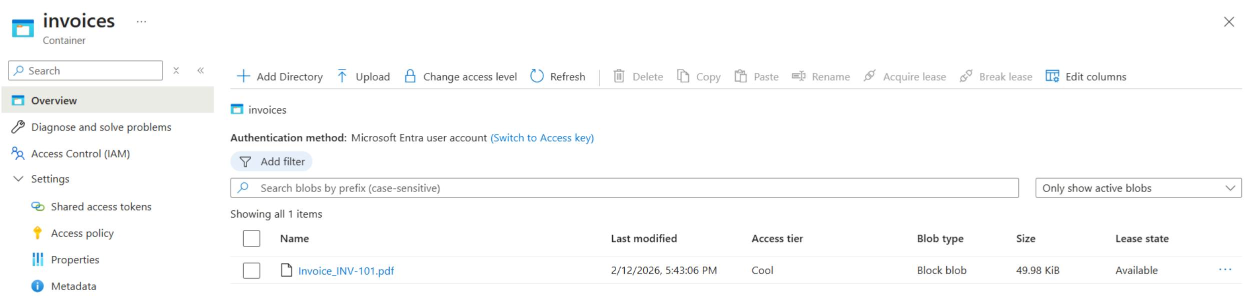
Content type: text/plain

Message Body

```
{
  "invoiceNo": "INV-101",
  "customerName": "Renuka",
  "items": [
    { "itemName": "Laptop", "quantity": 1, "price": 50000 },
    { "itemName": "Mouse", "quantity": 2, "price": 500 }
  ]
}
```

Step 3:

Check Blob Storage → invoices container → PDF generated.



Authentication method: Microsoft Entra user account (Switch to Access key)

Showing all 1 items

Name	Last modified	Access tier	Blob type	Size	Lease state
Invoice_INV-101.pdf	2/12/2026, 5:43:06 PM	Cool	Block blob	49.98 KiB	Available

14 Overview

I implemented an event-driven invoice generation system using Azure Service Bus and Azure Functions (.NET 8 In-Process).
The function listens to queue messages, generates dynamic invoice PDFs using QuestPDF, and uploads them to Azure Blob Storage.
This ensures asynchronous processing, scalability, retry handling, and loose coupling.