1. Creación de la cuenta y del entorno en Azure

1.1 Objetivo

Describir paso a paso, desde cero, cómo crear una cuenta gratuita en Azure y preparar el entorno básico para desplegar un portal web estático (App Service + App Service Plan) sin incurrir en costos.

1.2 Crear la cuenta gratuita de Azure

1. Accede a: https://azure.microsoft.com/es-es/free/

2. Haz clic en **Empieza gratis** y **Inicia sesión** con una cuenta Microsoft (Outlook, Hotmail, etc.).

3. Rellena los datos personales y valida con tarjeta (solo verificación; no se te cobrará mientras uses el plan gratuito y servicios gratuitos).

4. Obtendrás créditos (por ejemplo: 200 USD por 30 días, sujeto a la oferta vigente) y acceso limitado a servicios gratuitos.

Importante: *usa la suscripción de "Free Trial" o la suscripción que aparezca como gratuita en tu portal.*

1.3 Crear el grupo de recursos

1. En el portal (portal.azure.com) busca **Resource groups** (Grupos de recursos).

2. Clic en **Create**.

3. Datos:
    - **Subscription:** Azure subscription 1

    - **Resource group name:** rg-innova-retail (o innovaretail-proyecto)

    - **Region:** Canada Central (o la que elegiste antes)

4. Crear.

El grupo de recursos es como una carpeta para organizar todos los recursos del proyecto.

1.4 Crear App Service Plan (hosting)

1. Busca **App Service plans** en el buscador del portal.
2. Clic en **Create** y completa:
    - **Subscription:** Azure subscription 1
    - **Resource group:** rg-innova-retail
    - **Name:** asp-innova-retail
    - **Operating System:** Linux
    - **Region:** Canada Central
    - **Pricing tier: F1 (Free)** si aparece (si no, B1)
3. Revisar y crear → Create.

1.5 Crear App Service (la aplicación web)

1. Busca **App Services** → Create → **Aplicación web**.
2. Completa:
    - **Subscription:** Azure subscription 1
    - **Resource group:** rg-innova-retail
    - **Name:** portal-innova-retail
    - **Publish:** Code
    - **Runtime stack:** PHP 8.2 (recomendado para HTML estático)

- **Operating System:** Linux
- **Plan:** asp-innova-retail (F1)

3. Revisar y crear → Create.
4. Al entrar al recurso, localiza el **Dominio predeterminado** (algo.azurewebsites.net) y pruébalo: si aparece el mensaje “La aplicación web se está ejecutando y está esperando el contenido”, significa que el App Service está activo pero vacío.

2. Publicación: opciones y flujo recomendado

2.1 Opciones para subir la web (resumen)

- **ZIP Deploy (desde el portal)** — la más simple para sitios estáticos.
- **FTP** — tradicional, manual.
- **GitHub Actions / CI** — recomendable para proyectos académicos (automático).
- **Azure Static Web Apps / Blob Static Website** — alternativas para sitios estáticos, con ventajas de coste y CDN.

2.2 Recomendación para el proyecto

- Si el equipo usará GitHub (repo con /web dentro de main) — configurar GitHub Actions para desplegar solo la carpeta /web.
- Si no, usar ZIP Deploy: empaquetar los archivos HTML/CSS/JS directamente en la raíz del ZIP (sin subcarpetas anidadas que contengan la raíz) y subir.

2.3 Estructura recomendada del repo (manteniendo PDFs en root)

MAIN (rama principal)
│── PDF1.pdf
│── PDF2.pdf

```
├── otro_archivo.pdf
├── /web
    ├── index.html
    ├── estilos.css
    ├── /imagenes...
```

Con esta estructura, la acción de GitHub o el ZIP Deploy debe publicar **el contenido de /web como la raíz** de wwwroot en App Service.

2.4 GitHub Actions (ejemplo mínimo para desplegar carpeta web)

Asegúrate de que tu archivo de workflow (por ejemplo .github/workflows/deploy-azure.yml) tenga la siguiente configuración en la parte del deployment (nota en español y usando package: './web'):

```
name: Deploy HTML site to Azure Web App

on:
  push:
    branches: [ main ]
  workflow_dispatch:

jobs:
  deploy:
    runs-on: ubuntu-latest
    permissions:
      id-token: write
      contents: read

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Login to Azure
        uses: azure/login@v2
        with:
          client-id: ${{ secrets.AZUREAPPSERVICE_CLIENTID }}
          tenant-id: ${{ secrets.AZUREAPPSERVICE_TENANTID }}
          subscription-id: ${{
secrets.AZUREAPPSERVICE_SUBSCRIPTIONID }}
```

```
- name: Deploy to Azure Web App
  uses: azure/webapps-deploy@v3
  with:
    app-name: 'portal-innova-retail'
    slot-name: 'Production'
    package: './web'
```

Importante: configura los secrets en GitHub (AZUREAPPSERVICE_CLIENTID, AZUREAPPSERVICE_TENANTID, AZUREAPPSERVICE_SUBSCRIPTIONID) que Azure genera cuando conectas el App Service en el Centro de implementación.

3. Validación inicial y solución de problemas comunes

3.1 Mensaje de App Service: “La aplicación web se está ejecutando y está esperando el contenido”

Significa que el App Service no tiene contenido en /site/wwwroot. Solución: subir index.html (por ZIP, FTP o GitHub Actions apuntando a ./web).

3.2 Verificar archivos con Kudu

1. En el recurso App Service → Herramientas avanzadas (Kudu).
2. Debug console → CMD → navegar a /site/wwwroot y comprobar presencia de index.html, css y carpetas.

3.3 Registros y diagnóstico

- En App Service: **Supervisión** → **Registros de la aplicación** → activar logging para ver errores.
- En despliegues por GitHub Actions: ver logs en la pestaña Actions del repo.

4. Supervisión y Application Insights

4.1 Objetivo

Proveer visibilidad del comportamiento, rendimiento y errores de la aplicación para detectar fallos, medir uso y optimizar la experiencia del usuario.

4.2 ¿Qué es Application Insights?

Application Insights (AI) es un servicio de telemetría que captura métricas (latencia, peticiones/s), trazas, excepciones, dependencias (llamadas a bases de datos, APIs) y eventos de usuario. Permite diagnosticar problemas y construir alertas.

4.3 Opciones de instrumentación según tipo de app

- **Sitio estático (HTML/CSS/JS):** se usa el *JavaScript SDK* de Application Insights y se añade un fragmento <script> en index.html.
- **App con backend (Node, PHP, .NET, Java):** usar el SDK del servidor (o el agente). En App Service puedes activar la integración y enlazar al recurso AI creado; además conviene instalar/instrumentar a nivel de aplicación.

4.3.1 Fragmento JavaScript (ejemplo para index.html) — **Sustituye la connectionString**

```
<!-- Application Insights - snippet (ejemplo para sitio estático) -->
<script>
  (function(c,l,a,r,i,t,y){
    c[i]=c[i]||function(){(c[i].q=c[i].q||[]).push(arguments)};
    t=l.createElement(a);t.async=1;t.src=r;
    y=l.getElementsByTagName(a)[0];y.parentNode.insertBefore(t,y);
  })(window, document, "script",
"https://js.monitor.azure.com/scripts/b/ai.2.min.js", "appInsights");

  appInsights("setAuthenticatedUserContext", "usuario-demo");
  appInsights("config", {
    connectionString: "CONNECTION_STRING_AQUI"
  });

  // Ejemplo: trackear una página y evento
  appInsights("trackPageView");
```

```
  appInsights("trackEvent", { name: "InicioCargado" });
</script>
```

Pasos para habilitar desde el portal (resumen): 1. Crear el recurso *Application Insights* en la misma suscripción y grupo de recursos (ya lo hiciste).
2. Si App Service no muestra la opción habilitable, abre el recurso Application Insights → **Overview** → copia la *Connection String*.
3. Inserta el snippet en index.html con la connectionString.
4. Despliega la web y valida en Azure Portal → Application Insights → *Live Metrics* y *Logs*.

4.4 Limitaciones y costes

- Application Insights tiene un nivel gratuito con volumen limitado; si se excede, hay coste por GB.

- Para sitios estáticos, la instrumentación se hace en el frontend (snippet JS) y genera telemetría mínima.

5. Almacenamiento (para assets estáticos)

5.1 Opciones y cuándo usar cada una

- **Azure Blob Storage (Static website - contenedor $web):** recomendado para servir imágenes, videos y archivos estáticos grandes y, si se desea, servir todo el sitio (Static Website).

- **Azure Files:** uso SMB para compartir entre VMs; no recomendado para esta arquitectura simple.

- **Tablas/Colas:** para datos ligeros o mensajería; no para assets.

5.2 Recomendación práctica

- Habilitar una **Storage Account (StorageV2)** y activar **Static website** si quieres reducir coste y servir assets estáticos desde allí.

- Subir los assets al contenedor $web o a un contenedor normal y configurar permisos.

5.3 Pasos rápidos para Static Website

1. Crear Storage Account → tipo StorageV2.

2. En el recurso Storage → Static website → Habilitar → Index document: index.html.

3. Subir archivos al contenedor web.

6. Certificados SSL/TLS

6.1 Opciones

1. **Certificado gestionado por App Service:** gratuito y sencillo (ideal para www.tudominio.com).

2. **Subir certificado PFX:** para certificados comprados.

3. **Usar Front Door o CDN con TLS:** para WAF y rendimiento global.

6.2 Pasos (resumen)

- Para dominio por defecto *.azurewebsites.net ya hay HTTPS.

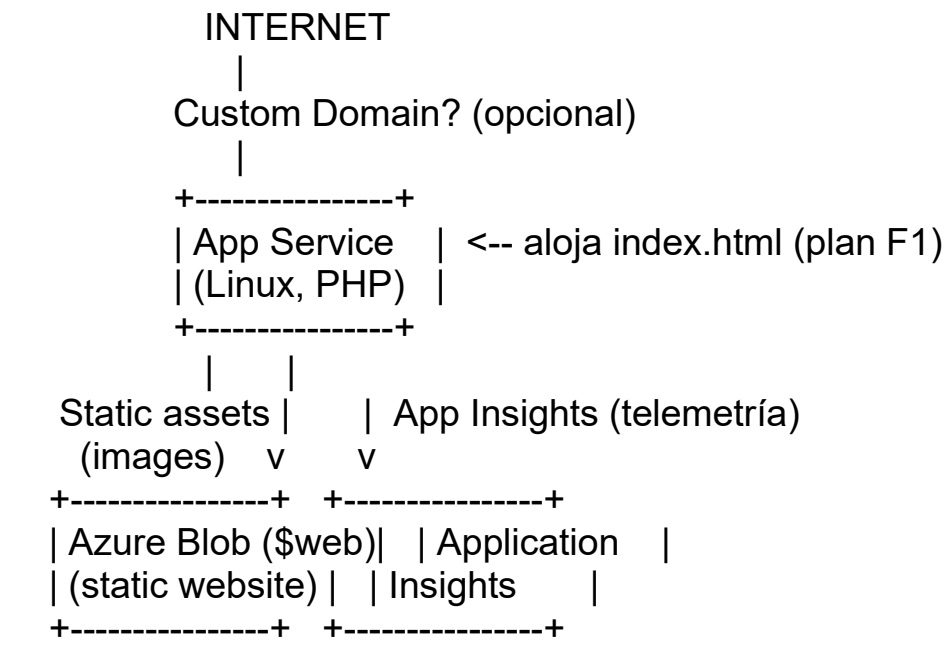
- Para dominio personalizado: App Service → Dominios personalizados → Añadir dominio (validar con DNS) → TLS/SSL → Certificados gestionados → Solicitar y enlazar.

6.3 Coste

- Certificado gestionado por App Service: gratuito.

- Otros servicios (Front Door, CDN avanzado): tienen coste.

Diagrama de alto nivel



TLS/SSL: App Service Managed Certs o Front Door

Checklist final (paso a paso - desde cero)

1. Crear cuenta Azure (Free Trial).
2. Crear grupo de recursos rg-innova-retail.
3. Crear App Service Plan asp-innova-retail (Linux, F1).
4. Crear App Service portal-innova-retail (PHP 8.2, Linux, usar plan F1).
5. Preparar repo GitHub: mantener PDFs en raíz y /web con index.html.
6. Configurar GitHub Actions apuntando a package: './web' o usar Zip Deploy con contenido de /web.

7. Crear Application Insights y añadir snippet JS con la connection string en index.html.

8. (Opcional) Crear Storage Account y habilitar Static website para assets pesados.

9. (Opcional) Configurar dominio y solicitar certificado gestionado en App Service.

Anexos

- Snippet JS para Application Insights (poner la connection string real).

- Ejemplo de workflow YAML (usar package: './web').