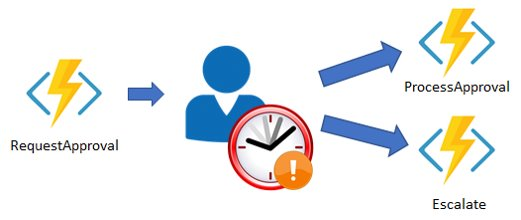
1. **Approval Workflow – Architecture**

****

|  |  |
| --- | --- |
| **20 Solution Overview** | * 1. **BlobTrigger starts the orchestration.**   2. **Durable Function orchestrator is kicked off.**   3. **Sends email approval request via SendGrid.**   4. **Email is delivered to user.**   5. **Timer is created (for timeout handling).**   6. **Email has a link (Approve/Reject).**   7. **Link hits an HTTP-triggered function to process the response.**   8. **HTTP function raises an external event to the orchestrator.**   9. **Orchestrator continues once either the user responds or timeout occurs.**   10. **Blob is moved to Approved/Rejected container.**   11. **Flow ends.** |
| **✅ Use Azure AD B2C to authenticate the approver ✅ Use System Assigned Managed Identity (auto-deletion when app is deleted, app doesn’t share identity with other services) for Function App to securely talk to Cosmos DB, Key Vault, and Storage**  **🔹 Step 1: Enable System Assigned Managed Identity on Azure Function**   1. **Go to Azure Portal → Your Function App** 2. **Navigate to Identity → System Assigned** 3. **Turn the status to On, click Save** 4. **Azure automatically creates a Managed Identity (MSI) under the same name** | |
| **Step-by-Step: Grant Access to Azure Resources**  **🗝️ A. Access Azure Key Vault (For SendGrid Key, Connection Strings, etc.)**   1. **Go to Azure Key Vault → Access Policies** 2. **Click + Add Access Policy** 3. **Configure:**    * **Secret permissions: Get**    * **Select Principal: Your Function App** 4. **Click Add, then Save**   **✅ Code Example in .NET (Azure.Identity NuGet):**  **csharp**  **CopyEdit**  **var client = new SecretClient(new Uri("https://<your-vault>.vault.azure.net/"), new DefaultAzureCredential());**  **KeyVaultSecret secret = await client.GetSecretAsync("SendGridApiKey");**  **string sendGridKey = secret.Value;**  **💾 B. Access Cosmos DB using RBAC + Managed Identity**   1. **Go to Cosmos DB Account → Access Control (IAM)** 2. **Click + Add Role Assignment** 3. **Role: Cosmos DB Built-in Data Contributor** 4. **Assign to: Your Function App**   **✅ Code Example (Azure.Identity + Azure.Cosmos):**  **var credential = new DefaultAzureCredential();**  **var cosmosClient = new CosmosClient("https://<account>.documents.azure.com", credential);**  **var container = cosmosClient.GetContainer("MyDB", "Approvals");**  **await container.UpsertItemAsync(item, new PartitionKey(item.UserId));**  **🗂️ C. Access Blob Storage with Managed Identity**  **Use RBAC if accessing via Azure.Storage.Blobs SDK, not via trigger bindings.**   1. **Go to Storage Account → Access Control (IAM)** 2. **Assign role: Storage Blob Data Contributor** 3. **Assign to: Your Function App**   **✅ Code Example:**  **csharp**  **CopyEdit**  **var blobServiceClient = new BlobServiceClient(new Uri("https://<storage-account>.blob.core.windows.net"), new DefaultAzureCredential());**  **var containerClient = blobServiceClient.GetBlobContainerClient("approved");**  **await containerClient.UploadBlobAsync("myfile.txt", stream);**  **🧠 Note: Triggered functions (e.g., BlobTrigger) use connection strings in bindings unless switching to SDK.** | |

[**https://pacodelacruzag.wordpress.com/2018/04/17/azure-durable-functions-approval-workflow-with-sendgrid/#:~:text=SendGrid%20sends%20the%20approval%20request,finishes%20the%20request%20is%20rejected**](https://pacodelacruzag.wordpress.com/2018/04/17/azure-durable-functions-approval-workflow-with-sendgrid/#:~:text=SendGrid%20sends%20the%20approval%20request,finishes%20the%20request%20is%20rejected)**.**

**A screenshot of a computer

AI-generated content may be incorrect.**

A computer screen shot of text

AI-generated content may be incorrect.

**A screenshot of a computer code

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

**Logging**

**"logging": {**

**"applicationInsights": {**

**"samplingSettings": {**

**"isEnabled": true**

**}**

**}**

**}**

A screenshot of a computer

AI-generated content may be incorrect.

**Security:**

**[Angular App] → [APIM / HTTPS] → [Azure Durable Functions]**

**↓ ↓**

**[Azure AD B2C] [Azure Key Vault, SendGrid]**

**↓**

**[Approval Link w/ Token]**

**↓**

**[Backend Verifies Token]**

**↓**

**[Durable Orchestrator continues → timeout/reject]**

|  |
| --- |
| 1. **Function-Level Security**  [FunctionName("StartApproval")]  public async Task<IActionResult> StartApproval(  [HttpTrigger(AuthorizationLevel.Function, "post")] HttpRequest req,  [DurableClient] IDurableOrchestrationClient starter)  **This restricts the function to only be called with a valid function key.** |
| **2. Combine with Azure AD (No Code Needed)**  Enable App Service Authentication in the Azure portal:   * Azure → Function App → Authentication * Enable Azure AD B2C or Azure AD * Set Action: Log in with Azure Active Directory * Enforce login on all routes |
| two app registrations  Angular frontend 🡪 Azure AD B2C → App registrations → New registration  backend (Function App).  Step 2: Enable App Service Authentication (Easy no-code security)   1. Go to Azure → Your Function App 2. Navigate to Authentication → Add identity provider 3. Select Microsoft Identity Platform (Azure AD B2C) 4. Configure with:    * Client ID of backend (function-app-api)    * Issuer URL from B2C tenant    * Allowed token audience: api://function-app-api-client-id 5. Set Action to take when request is not authenticated to: ✔ Log in with Azure Active Directory B2C |