Azure IPaaS Platform Assessment

Recommendation Report Sample

Digital Integration

- **Require Client's Input**
- **Assessment Output**

Azure Platform Health Checklist

Understanding the landscape

- **1-hour workshop** with platform owner to describe the importance of Azure integration platform within the organization and understand the anticipated growth of the organization and platform scalability.
- **1-hour workshop** with the operations lead to review RPO/RTO requirements and assess disaster recovery readiness.
  RTO [**Recovery Time Objective**] is the timeframe within which application and systems must be restored after an outage. RPO [**Recovery Point Objective**] is about how much data you afford to lose before it impacts business operations.
- Provide Neudesic with integration architecture diagrams and design documents.

Scope

- Connect with developers and architects to identify critical integrations that requires a Health Check. Also, understand their pain points on the platform.
- Scope is limited to couple of critical integration flows with different design patterns.
- Client to provide walkthrough of current DevOps and Software Development Life Cycle (SDLC).
- **1- hour workshop** with Client architect/developer to walkthrough existing CI/CD platform/tools.
- **1- hour workshop** with Client architect/developer to walkthrough policies and security.

Governance

- Access to Client’s best practices documentation to review standards and alignment of standards across projects.
- **1-hour workshop** with a client developer to walk through existing error handling, logging, and troubleshooting frameworks.

- Client to provide access to repository, and read-only access to Azure resource group for Neudesic to review the existing artefacts and resource optimization.

Technical platform assessment

Documentation

Evaluate if essential project documents are in place.

Example

Key documentation is missing for tracking the requirement and technical implementation for the flows.

We recommend to prepare business requirement and technical specification documents for all the existing flows in the system. Also, add this step as part of the process going forward.

Rating: 4/10

Architecture & Design

Assess architecture and solution design for integration flows and platform.

Example

Assessed flows are developed as one whole business workflow using Logic App which creates tight coupling between all connected destination systems. This also increases failure points in the business process. Flows are designed as pull model for real time transactions.

Here by converting pull-based models into event-based models for real-time transactions and implementing pub-sub architecture, we can help improve the design. It is a perfect fit for application-to-application or enterprise application integration (EAI) scenarios.

Rating: 6/10

Resource & Cost Optimization

Evaluate opportunity to reduce the cost for the platform.

Example

Overall platform resources look good. There are no unused resources and most of the function apps are hosted on App Service plan with the right workload capacity. Logic Apps are based on consumption plan which incurs billing based on each action execution.

Rating: 9/10

Coding Standards & Practices

Evaluate if coding standard and best practices are followed in the solution.

Example

There are no formal best-practices document that exists within the organization. Looking at the code in the repository, there is no clear coding or architecture and no standards relating to naming conventions, development best practices, or overall architecture. The recommendation is to define and implement coding standards and best practices for integration solutions.

Rating: 2/10

Branching Strategy & Deployment

Assessing branching strategy and deployment strategy for iPaaS platform flows. How code will be managed, in case of multiple developers are working on same code base. How deployment to different environments will be performed, and how gatekeeping process around the same is handled.

Example

With distributed version control systems like Git, you can manage and share your code in a flexible way. It is important that your team find a balance between this flexibility and consistency in collaborating and sharing code. Team members publish, share, review, and iterate on code changes through Git branches shared with others. We recommend to adopt a branching strategy for your team. Collaboration is easier and less time is spent maintaining version control, so that more time is spent developing code. Use DevOps best practices for deployment process.

Rating: 5/10

Security

Security is very important, and it has to be baked in solution design. Evaluate overall platform and integration solution security.

Example

The current implementation is based on shared access key which lacks strong security implementation. For integration flows, we recommend using Azure AD authentication and message encoding/decoding. Alternately, we can look at moving to an Integration Service Environment (ISE) for critical workflows. In addition, we can move sensitive information to Azure KeyVault.

Rating: 6/10

Business Continuity

Evaluate business continuity for integration platform in case of region failure or a data center failure in Azure.

Example

Business continuity is most critical aspect for any business.

We recommend considering active/passive resource group model to maintain the parallel system in separate region for most critical flows. Another low-cost solution is to implement DevOps pipelines to create parallel resource groups with all the necessary business flows for undesirable incidents.

Rating: 5/10

Reliability & Recoverability

A platform's ability to handle system failures and transaction failures is what determines how reliable and recoverable it is

Example

Looking at the current implementation, we recommend using Azure Service Bus in flow design which is the fastest & easiest way to achieve reliability for any process. It has features like sharing

messages across processes, enabling automatic retries during process failures, forwarding messages to the dead letters on retries failures, and resubmitting a message.

Rating: 7/10

Supportability

Evaluate platform and solution supportability.

Example

We recommend enabling log analytics and app insights for Logic App based integrations and install Logic Apps Management solution to monitor & manage the Logic apps.
Custom dashboard can also be created to monitor A2A and B2B integrations.
Use Azure Monitor for platform-level health checks and notification.

Rating: 7/10

Overall Recommendation

Here are some immediate recommendations that we have for your organization to move forward with

- Training on DevOps, CI/CD, and general SDLC practices.
- Develop integration best practices, coding standards, and perform code reviews to ensure standards are being followed.
- Consider implementing integration design patterns like pub-sub, event-based and API based approaches for real time A2A or B2B flows.
- Consider emphasizing on documentation for maintaining the knowledge base.
- From security perspective, consider different security options like Shared access policy, Azure AD authentication, OAuth authentication, PGP encryption and decryption, platform level policy etc.
- Develop strategy for business continuity.