CLOUD1

INTEGRATION PLATFORM

TABLE OF CONTENTS

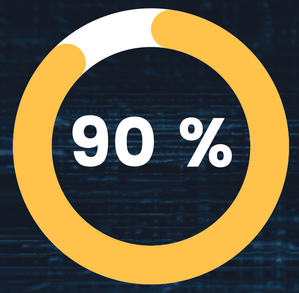
INTEGRATION PLATFORM BLUEPRINT

IMPORTANCE OF INTEGRATION PLATFORMS

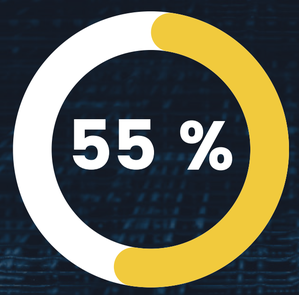
Making data accessible is a key factor for any organization.

The most important role of an integration platform is to make data accessible. This is achieved by minimizing disruptions caused by deficient system features and incompatibility issues with integrations. By implementing a shared integration platform, we can also decrease the overall complexity of organization's integration infrastucture by creating a centralized hub for integrations instead of implementing integrations in or between each system separately. This lowers overall costs by eliminating overlapping development efforts and maintenance costs.

Integration Platform as a Service (iPaas) is an integration platform created using cloud-based tools. Our integration platform concept is built on Azure Integration Services.



90% of leaders agree that every organization will need to extract value from data to be successful in the future.



55% of an organization's data is unqualified and/or underutilized.

*1 https://www.splunk.com/pdfs/dark-data/the-state-of-dark-data-report.pdf (2017)

TRADITIONAL DRIVERS FOR MODERN INTEGRATION PLATFORMS

Ability to provide data internally and externally via API Management for various clients to support business cases. Data can be in multiple different backend systems, and it should be readily accessible.

TOP 3 types of API's people work on

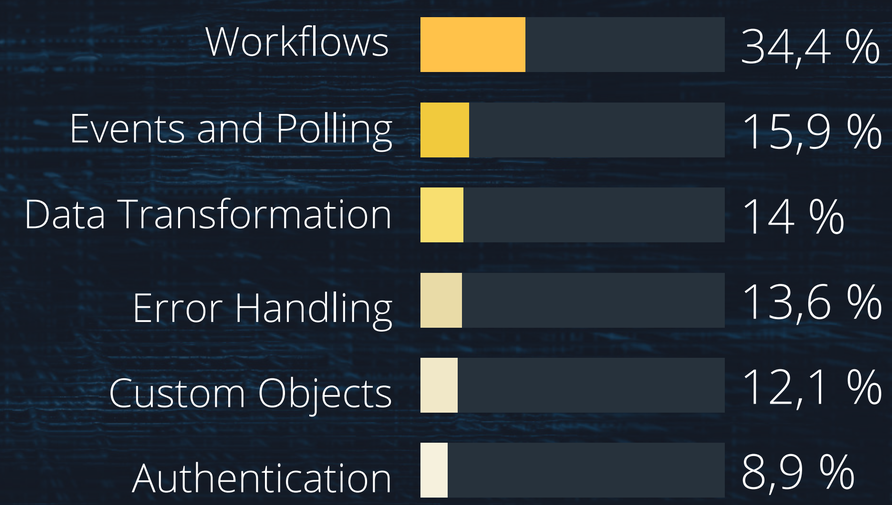
*2 https://stateofapis.com/#api-source

Speed is one of the biggest drivers when selecting a modern integration platform. Modern platforms are most often iPaaS platforms with low-code development environments, pre-built connectors, and abilities to allocate resources efficiently. These features enable fast prototyping and increases speed to market.

Centralized management for development, governance and monitoring in a single platform gives better visibility to APIs, data flows, integrations, and data access for developmental and operational purposes. With a modern platform, you can easily set up access control and configure access to different services effortlessly.

Built-in capabilities, such as adapters and connectors, support development speed and make integrating between different systems easier, as there is no need to build connectors from scratch.

Most time-consuming parts of integration development



*3 https://cdn2.hubspot.net/hubfs/440197/2020-04%20-%20SOAI%20Report.pdf

Multiple applications need to interchange data in real-time or periodically. A modern integration platform should connect applications seamlessly, regardless of whether applications are in the cloud or on-premise.

BUSINESS PROCESS INTEGRATION

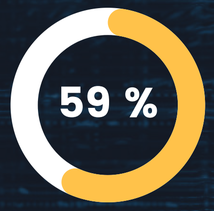
Automation tools have become more popular, and they can be used in a lot of areas, all the way from data integrations to process integrations. Organizations may need to automate processes internally, but also externally - this is where Business Process Integration (BPI) steps in. BPI is synchronization of internal processes with internal and external parties by connecting systems and services, according to predefined processes. It enables automation of management, operational and support related processes within and between organizations.

BPI provides additional value by reducing errors and delays in processing in addition to decreasing the amount of manual work by automating data transfers. These integration solutions are able to track processes and to generate real-time data, for example in the context of Service Level Agreement monitoring. In addition, with BPI you're able to audit what was done during each step of any given process. Automating processes can also lead to faster processing time and therefore, better customer experience.

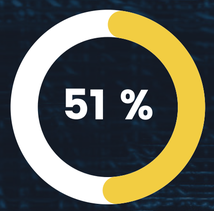
BPI is enabled by different architectural approaches, such as API Driven and Event Driven Integrations. Combining these with BPI, enables getting the most value from integration platform in Business Process Automation. Process automations can be triggered by an event, for example new purchase order, or updating customer address via API.

Digital Transformation and Cloud App Adoption are the most important drivers for integrations.

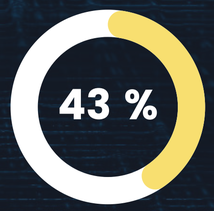
Top benefits resulted by applying APIs reported by survey respondents:



Increased productivity 59%



Increased innovation 51%



Direct increase in revenue 43%

*4 https://cdn2.hubspot.net/hubfs/440197/2020-04%20-%20SOAI%20Report.pdf

LOW-CODE APPLICATIONS ARE A GROWING CATECORY OF API CONSUMERS

"By 2025, 70% of new applications developed by enterprise will use low-code or no-code technologies."[*5]

70% is a huge growth figure. Back in 2020, this same figure was less than 25%. Growth is speeding up, and doing so rapidly. This is not a surprising considering the benefits of using low-code tools. According to one survey, organizations reported 43% greater agility and 43% reduced cost from using these tools and a considerable potential of 90% in development time reduction.[*6]

It is no wonder that demand for these tools and community development in general is on the rise.

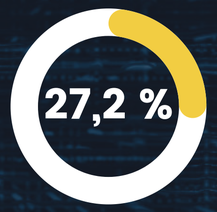
A noticeable point in Gartner's statement from the integrations platforms' point of view, is that applications will use more and more low-code technologies, but they almost never can solely rely on them. Low-code applications, as well as traditional code-based applications require external data to be useful. This data in many cases, however, is hard to reach. Typical issues include that the data is in a problematic format or not accessible to developers directly from source systems. There are security-, network-, and capability limitations that need to be solved for low-code application development to be possible. These data requirements are starting to show in the latest surveys as well.

On average, 83% of respondents consider API integration a critical or very critical part of their business strategy. [*7]

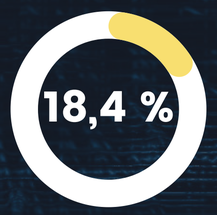
The importance of API integrations means that there is a demand for better suited and more user-friendly APIs from the application point of view. There is also demand for better documentation, as more people with less technical competence are looking to utilize the data. Customized API's without an intermediary system such as an Integration Platform is not a real option anymore. Legacy systems might not provide API's at all. Also some of the modern source systems, especially SaaS sources, eventhough they might have a good set of APIs, might not provide option to add custom API's for client specific needs. SaaS source modifiability can be restricted as their development efforts are not based on a single customer's requirements but instead are focusing to please the audience as whole.

Highest demand for API integrations were

Customized APIs that fit a specific business need 42,4%



Better documentation 27,2%



"No code" integration templates 18,4%

*8 https://f.hubspotusercontent40.net/hubfs/440197/Cloud-Elements-2021-SOAI.pdf

Additionally, low-code application development isn't solely about ease of development or increased efficiency; it increases motivation and innovation. As it becomes easier are to create new applications to benefit from existing data it inevitably leads to better work satisfaction. Not demanding but enabling the use of citizen development by providing good set of low-code development tools like Power Platform and providing necessary data through an Integration Platform to the business can be seen as s competitive edge in the recruitment market in the near future. According a survey, 83 percent of low-code users experienced positive impact on their work satisfaction. *9

WHAT MAKES iPaaS DIFFERENT?

INTEGRATION PLATFORM AS A SERVICE

Structure and flexibility in the same package

CHALLENGES WITH INTEGRATION SOLUTIONS

Many integration systems work better in certain scenarios than in others

If a platform can utilize only single-use technology, it is, by definition, doomed to fall short in edge-case scenarios. With some products you can use low-code or script implementation to extend the compatibility of the software, but when even a frankensteinian solution is no longer is enough, a suboptimal solution must be created.

Licensing and operating costs are hard to forecast and license price changes remain a risk

When a system has a licensing model based on registered users or applications it might become an issue to forecast costs.

Systems do not provide access management or support single-sign-on

Lack of Azure AD access models as well as data product and role-based access models can make it a complex task to manage access to API layers.

Integrations do not scale, or latencies are too large

If scaling is possible only by adding more memory or processing power instead of scaling to multiple instances, it can create problems. Furthermore, if a core service is not suitable for a use-case and cannot be selected per use-case, that specific use-case might be impossible to implement.

Self-service / multi-team development is not supported

If the individual integrations can be deployed using only a single code base it might create problematic bottlenecks in multi-team development environments.

CHALLENGES WITH INTEGRATION SOLUTIONS

Failures are logged but the system does not self-recover

With well-designed integration patterns and modular architecture that includes processing queues and error logging the processes can alert developers in real-time and self-recover from common failures.

Restricted integration platform function extendibility

Some tools do not provide extendibility outside of the out-of-the-box low-code and scripting functionality. When an extension means changes to the system base code rather than developing an additional module into your platform it might create huge problems with the testing and deployment processes.

Data is not understood, owned or overseed

If a solution doesn’t provide API definitions and documentation it might be very complex to utilize the data. Development efforts go into first finding the correct interface and secondly trying to figure out how to use it.

Integration platform is used for a tasks it's not suitable for

Sometimes there is an incentive for the vendor to use its product for everything. However, integration platforms are best suited for actual integrations. There are other solutions that are better suited for analytical data platforms, IoT platforms, AI platforms and others. These solutions have functionalities and architecture designed to work best on their dedicated use-cases. Moreover, finding an individual developer that has deep understanding of developing many different kinds of solutions is next to impossible.

Lack of skilled developers

If a tool is not widely used it might be hard to scale its usage because a lack of skilled developers.

DIFFERENT TYPES OF INTEGRATION SOLUTIONS

Azure iPaaS

iPaaS solutions built on serverless Azure services, such as API Management and Logic Apps, allow efficient elastic and horizontal scaling. Licensing is based on a pay-as-you-go model without upfront fee, so you can start testing services easily. Azure iPaaS supports low-code, no-code and code-based development. Platform use can be extended by either coding functions, utilizing existing connectors, or by starting to use new Azure based services as part of iPaaS stack. Azure offers lots of services that can be used as part of integration architecture, but this means that there is no pre-defined architecture either. Azure may not offer ready-made architecture but offers best practices and guidelines for architecture and services, including which service suits what kind of use case.

iPaaS Products

iPaas Products are out-of-the-box platforms that are offered as a service and can include capabilities for API Management. Often, iPaaS products have agent-based architecture, that can be in the cloud or on-premise. Additionally, they may support horizontal scaling by setting up new agent servers. Licensing models vary between agent-based and connector-based pricing, which make costs simple to calculate. They support connectors and low-code, no-code and code development, but coding capabilities may be limited to one-liners or to a pre-defined set of libraries and languages. Getting started with iPaaS products is easy thanks to the hosting capabilities in the vendor's cloud or supported cloud and out-of-the-box design, but it also limits how solution should be used.

Integration Middlewares

Integration Middlewares are running on servers or virtual machines, usually on-premise, supporting vertical scaling. Licensing is monthly or yearly fee for the whole platform with possible up-front fee for the initial setup. Development is done by using offered built-in functionalities or coding, without support of extensions. They are targeted for large enterprises and may not suit small and medium sized enterprises due to costs and steep learning curve to start utilizing the platform properly. Integration Middlewares' are out-of-the-box solutions, that do not offer much room for customization, other than as code, but do provide the development framework.

	Azure iPaaS	iPaaS Products	Integration Middlewares
Type	Cloud	Cloud/ Hybrid	Typically on-premises
Licensing	Pay-as-you-go, per service	Monthly/yearly Different pricing models	Monthly or yearly
Development	Low-code, No-code, Code	Low-code, No-code, Code	Code
Extentibility	Code, connectors and other services	Code and connectors	Code
Scaling	Horizontal and elastic scaling	Horizontal scaling	Vertical scaling
Architecture	Serverless	Serverless/Agent based	Server based
Target	Small to large enterprises	Small to large enterprises	Large enterprises

WHAT IS AZURE iPaaS?

Integration Platform as a service (iPaas) is a suite of cloud services enabling development, execution and governance of integration flows.

Connects on-premises and cloud-based processes, services, applications and data within individual or across multiple organizations.

(E)iPaaS offers enterprise level scalability, high availability, disaster recovery and security.

iPaaS use cases

Different integration use cases in which iPaaS can be utilized:

- Event driven integrations
- APIs / API driven integrations
- Managed file transfers
- Process integration (stateless/stateful)
- Recurring/Scheduled jobs / Automating recurring jobs
- System integration / Application-to-Application integration
- Business-to-Business integration
- Automating business processes

Process examples in which iPaaS can be utilized:

- Users from AD to HR
- Hour entries from time tracking to invoice system

Benefits of Azure iPaaS

- Centralized development and monitoring environments provide an overall view of all your data flows. You can use Azure services to monitor your data flows and send alerts if errors occur. Additionally, you can use APIs to gain a better overview of utilization level.
- An integration with DevOps allows you to track changes done to iPaaS via code repositories, and set up a CI/CD pipeline to automate your deployments from dev to test and test to production, thus minimizing human error.
- A wide selection of built-in connectors helps to connect applications and transform data in the workflows. This makes development easier and faster compared to building everything from scratch
- Serverless solutions allow fast deployments and scaling flexibility, without the burden of maintaining infrastructure. Develop and test locally and deploy to the cloud.
- With an Infrastructure-as-Code approach, your infrastructure is in the configuration files and enables configuring Azure resources in a repeatable manner, minimizing potential human errors and manual work. You can set up temporary environments effortlessly when needed. Thus there's no need to run temporary environments when they are not actively in use
- Cost efficient and optimized: Pay only for resources that you use when you use them. Disable services that are not used and scale in resources if there is no heavy traffic
- Scale in and scale out resources automatically when more processing power is needed. No need to manually spin up another Virtual Machine, or disable instances when less processing power is enough

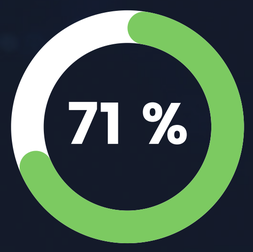
INTEGRATION USE CASES

Integrations are often based on integration patterns, which are a set of best practices for developers to design and build integrations between systems. Modern applications offer APIs and webhooks to support different use cases, but a modern integration environment can consist of multiple systems that should be connected to each other. There might be a need to synchronize data between systems once a day or when an entity is created.

A common use case is when clients or other systems pull data from a system to synchronize data. In the case of a single client it could be possible for a client to connect directly to the system that provides the needed data, but when there are multiple clients and multiple systems it may not be feasible. Therefore, Instead of direct connection, systems are exposed via API Management as APIs to limit the number of different kinds of connections and credentials to different systems. This is called API Driven Integration.

Another common use case would be when a user does a specific operation, such as saving a record in a system, the system can trigger sending an event. Events can be used to start process flows, that consist of multiple actions across one or more systems creating new entities, synchronizing data, or starting other process flows. This is called Event Driven Integration.

In event driven architecture the consumer registers in to receive updates and the provider keeps track of the state. This differs from standard API driven architecture where the consumer handles the state and makes API calls accordingly. Event driven architecture takes more effort to implement and as such is not an ideal solution for all use cases.



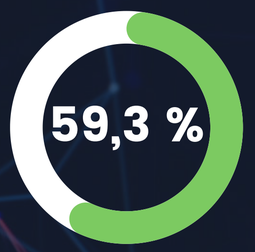
However 71% of businesses see the benefits of EDA outweighting the costs and it is already considered one of the standard capabilities of a modern iPaaS solution.

*10 https://www.gartner.com/document/3867165

INTEGRATION ARCHITECTURE PATTERNS

API Driven Integration

Sometimes data is needed from multiple systems, and there is a chance that each of these systems offer data via different APIs to which connections are opened using different protocols and data is returned in various formats. API Driven Integration is an approach, where software interfaces are developed to expose backend data and application functionality for clients to use more easily.



59.3% of respondents are using REST APIs in production which illustrates the extent of API driven integrations adoption in organizations.

*2 https://stateofapis.com/#api-source

API Driven integration architectures simplify application complexity by allowing clients to connect to an API gateway, which can work as a front to multiple backend applications. API gateways expose all of these as resources, often with similar data formats, which simplifies consumption of APIs on the client's end. In short, this approach decouples frontend and backend, removing dependencies that may exist between these two.

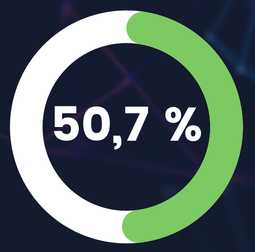
This approach can act as an extra "access control" layer between the client and backend system. APIs may have more logic on the background to filter data from the backend system, define which resources are accessible with given credentials, or even limit amount of allowed requests within specified timespan.

Event Driven Integration

In some integration cases, the challenge is to enable real-time data integration, where a change in one system needs to reflected to one or more other systems. Achieving a synchronous data state can get difficult with REST-based API integration patterns, which are based on requests and responses. Therefore, changes must be polled, and this can cause a delay in data synchronization between systems. An event driven integration architecture is made of highly decoupled event processing components, that receive and process events asynchronously, therefore being resilient to connection issues. This is usually important in transaction-based solutions, where the order of messages is crucial.

An event can represent something that happens in a system, a user action or a signal of an action has finished. An event can trigger a new action within the system or send the payload to the integration layer, where different event types may have different processing rules or workflows.

One example of such an event driven process could be processing an order on an Ecommerce website. When a customer orders a product, it triggers an event to the ERP as a new order, and to the CRM to add a new entry to the customer's order history. An event driven approach can also coexist with an API Driven Integration. The API layer could handle incoming events without the need to involve the logical layer, enabling changes to routing and workflows without affecting clients.



50.7% of respondents in Azure iPaaS survey had adopted Event Hub. This aligns with the growing popularity of modernizing via event-driven architectures for key systems.

*11 https://azure.microsoft.com/en-us/resources/how-modern-enterprise-integration-platform-as-a-service-ipaas-enables-business-strategy/

CLOUD1 iPaaS CONCEPT

PROVEN INFRASTRUCTURE

Patterns and best practices for iPaas on Azure

ARCHITECTURAL LAYERS ON iPaaS

API Management is responsible for handling incoming requests through published APIs. It functions as a front-end in the integration architecture: Requests from clients are authenticated and payload schemas can be validated in the API Management. Therefore, it offers access control and security functionalities before the actual consumption and processing of the request payloads.

Messaging and Event solutions can temporarily store incoming messages until workflows are ready to process new messages or broadcast a single event to one or more workflows. Messaging and Event solutions increase reliability by decoupling the architecture. It removes the direct connection from API Management to Orchestrations and workflows, when this connection is not mandatory by system or process requirements.

Orchestration and Workflows are responsible for processing events and messages. Processing can be simple data transformations from XML to JSON or identifying the destination system based on incoming data and transferring the data to the destination system. Orchestration and workflows have a low-code/no-code development environment with various built-in connectors to handle most cases. If more complex logic is needed, it's possible to utilize customizable Serverless computation solutions with Orchestration and workflows.

Serverless Compute Solutions offer functionalities for Orchestration and workflows by providing capability to create more customized data processing tools to support the most complex use cases. In some cases, low-code and no-code solutions would become too complex or lack performance. In these cases it would be better to write the required logic as plain code with Serverless computation solutions.

Logging and Monitoring is generally used by all layers. It's responsible for knowing what's happening and when, and whether the operation was successful or failed. A well-designed Logging and Monitoring layer provides valuable information about the whole integration environment, such as which services are used, when they're used the most, and how they are used. This is especially important when dealing with transactional workflows.

Identity and Access Management control user access and permissions to different resources. Access can be restricted according to Role-Based Access Control, enabling granular access management to services and services' components, such as APIs or Workflows.

Secret Management allows storing credentials and secrets in safe manner, while enabling services to use stored credentials and secrets. Secret Management combined with API Management enables storing API keys into safe vaults. Additionally, if a Workflow needs access to an external API to enrich data, API credentials can be store under Secret Management.

CLOUD1 iPaaS IMPLEMENTATION PRICIPLES

Cloud1 iPaaS is built on selected Azure Integration and core services. The selected services provide several benefits.

First benefit is scaling. The most general aspect of scaling is availability. Azure provides high availability for all used services as standard. Compared especially to integration platforms built on middleware, this is a considerable benefit. Most Azure services are priced depending on how much they are used. This is also the case with Azure Integration services, meaning that up-front investments are not needed; costs are not linked to the scope of the platform but instead to capacity and storage requirements. This pay-as-you-go model makes it possible to optimize patterns that require process power and to be able to use less optimized low-code implementations for the majority of the patterns. And the last perspective in scaling is of course the actual hard performance. This is where Azure iPaaS really shines by providing the possibility to boost processing power for individual service instances, like adding more CPU to a server or dividing the load to multiple instances, like adding more computers to balance the different types of integrations. Of course this balancing is just part of the IaC and doesn't require actual server maintenance.

On the other hand, flexibility is achieved by the different kinds of services available in Azure. Key services are selected as part of the Cloud1 iPaaS concept but it can easily be extended with many other services. Thus, if the most optimal implementation of a use case requires a pattern that would be better suited for some other service type it is possible to go the most optimal route. This means that the architecture, while providing selected commonly used patterns, is not restricted to use only these ones. At Cloud1, we also see the integration platform as part of the larger Cloud1 Data Hub concept and benefits achieved from capabilities outside of the integration platform are common.

Working with both on-premise and external cloud systems is common in today's integration world. Azure provides many options to implement secure networking solutions on top of which iPaaS can work. The Cloud1 iPaaS concept features a predefined networking architecture that is suitable for the vast majority of hybrid solution requirements.

The Cloud1 concept also includes a set of best practice patterns and capabilities based on selected key services. Combined with infrastructure as code, version control and automated deployment pipelines, it provides a system that can be developed by anyone that is familiar with the architecture and has the required technical know-how of the services used to create the actual integrations. And, when combined with the Cloud1 governance model the concept provides multivendor development capabilities. This means that there is zero vendor lock beyond Azure cloud.

One notable functionality of the Cloud1 iPaaS architecture is logging patterns. It of course provides out-of-the-box operational logging and monitoring functionalities, but beyond that it also features an improved process level monitoring capability. Process-level monitoring makes it possible to easily see which individual event has passed through the system and, furthermore, which events have produced errors and why. Especially in a low-latency event integration it is important that problems can be identified right away so decisive action can be taken without delay.

CLOUD1 iPaaS
CORE CAPABILITIES

Scaling – All of the benefits of utilizing a modern cloud service and platform

- Pricing – Start small and increase your capacity accordingly. Pay as you go - no license costs.
- Performance – Ability to scale required capacity up and down.
- Availability – Built-in feature of Azure services. Differences and options per service.

Hybrid connectivity

- Connectivity to on-premise systems or other cloud services as needed

Flexible and extendable

- Use the best available services to implement your integration patterns
- Extendable based on new requirements

Standard components/services

- No vendor locks. Using the Microsoft best practices with general Terraform IaC automation.
- Version control and solution deployment model (CI/CD)

Complete Infrastructure as Code and CI/CD model

- Best-practice models and templates to manage Azure Integration services using automation

Built-in multivendor capabilities

- Flexible modular structure for management

Centralized technical and process-level logging

- Best-practice models for monitoring, alert configuration and management

CLOUD1 iPaaS
BENEFITS OF AZURE

Modernize integrations by utilizing serverless services such as Azure API Management, Azure Service Bus, Azure Event Grid and Azure Logic Apps, that can be seamlessly integrated with each other, as part of the integration architecture. Serverless architecture enables easy and fast scaling of services when integration needs change and grow.

Boost development with Azure Logic Apps' low-code development environment with out-of-the-box connectors for various operations to build complex workflows to support business needs. Use Azure Functions to support Azure Logic Apps in more complex data processing cases with various coding language options, and expose backend systems via Azure API Management as resources with unified data formats instead of each client connecting directly to backend system, such as a database.

Build enterprise integrations with messaging services, such as Azure Service Bus and Azure Event Hub, to decouple services for scalability and reliability. Processing Logic Apps polls messages from messaging services, once they're ready to process new messages, and processes them according to workflow. In case of errors, messages stay in the messaging service and no data is lost.

Ease operational work with monitoring services, like Application Insights and Azure Monitor, to gain better insight on integration flows and processes. Get detailed information on how long requests take, how many processes were run during any given day, why a workflow failed and what caused the issue.

Protect applications and data by storing credentials and secrets to Key Vault and setting Role Based Access Control to Storage Accounts and APIs to prevent unauthorized access. Secure connections and data flows with Azure Firewall and Azure Virtual Networks within cloud and Azure VPN Gateways between cloud and on-premise.

CLOUD1 iPaaS
KEY ARCHITECTURE COMPONENTS

Logic Apps
Azure Logic Apps is a cloud-based platform for creating and running automated workflows that integrate your apps, data, services and systems.

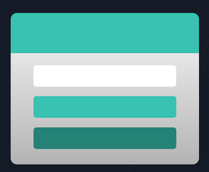
Azure Functions
Azure Functions is a cloud service available on an on-demand basis that provides all of the continually updated infrastructure and resources needed to run your applications.

Event Grid
Event Grid is a fully managed message delivery service at massive scale. It provides functionality to support reactive, event-driven apps in a modern, serverless architecture - eliminating polling and its associated cost and latency.

API Management
offers a scalable, multi-cloud API management platform for securing, publishing and analyzing APIs.

Service Bus
Azure Service Bus is a fully managed enterprise message broker with message queues and publish-subscribe topics (in a namespace).

Log Analytics
Log Analytics is a tool to edit and run log queries from data collected by Azure Monitor logs and interactively analyze their results.



Storage Account
Azure Storage Account contains all of your Azure Storage data objects, including blobs, file shares, queues, tables and disks. The storage account provides an unique namespace for your Azure Storage data that is accessible from anywhere in the world over HTTP or HTTPS.

Key Vault
Azure Key Vault is a cloud service that provides a secure store for secrets. You can securely store keys, passwords, certificates, and other secrets. Azure Key Vaults may be created and managed through the Azure portal.

Application Insight
Application Insights is a feature of Azure Monitor that provides extensible application performance management (APM) and monitoring.

Azure DevOps
Azure DevOps provides developer services for allowing teams to plan work, collaborate on code development, and build and deploy applications.

Terraform
Terraform is an open-source infrastructure as code software tool that enables you to safely and predictably create, change and improve infrastructure.

CLOUD1 iPaaS
DEPLOYMENT AUTOMATION

The Cloud1 concept is based on an IaC (infrastructure as code) template that is used to deploy the concept architecture model out-of-the box. This enables a rapid start for the development as the underlying platform can be setup in a way that it is fully functional and ready for development efforts to start. Compared to the use of Azure Integration services as they are, our solution brings the platform much closer to an iPaaS product without losing the edge of extend functionalities with other Azure Services and custom code modules.

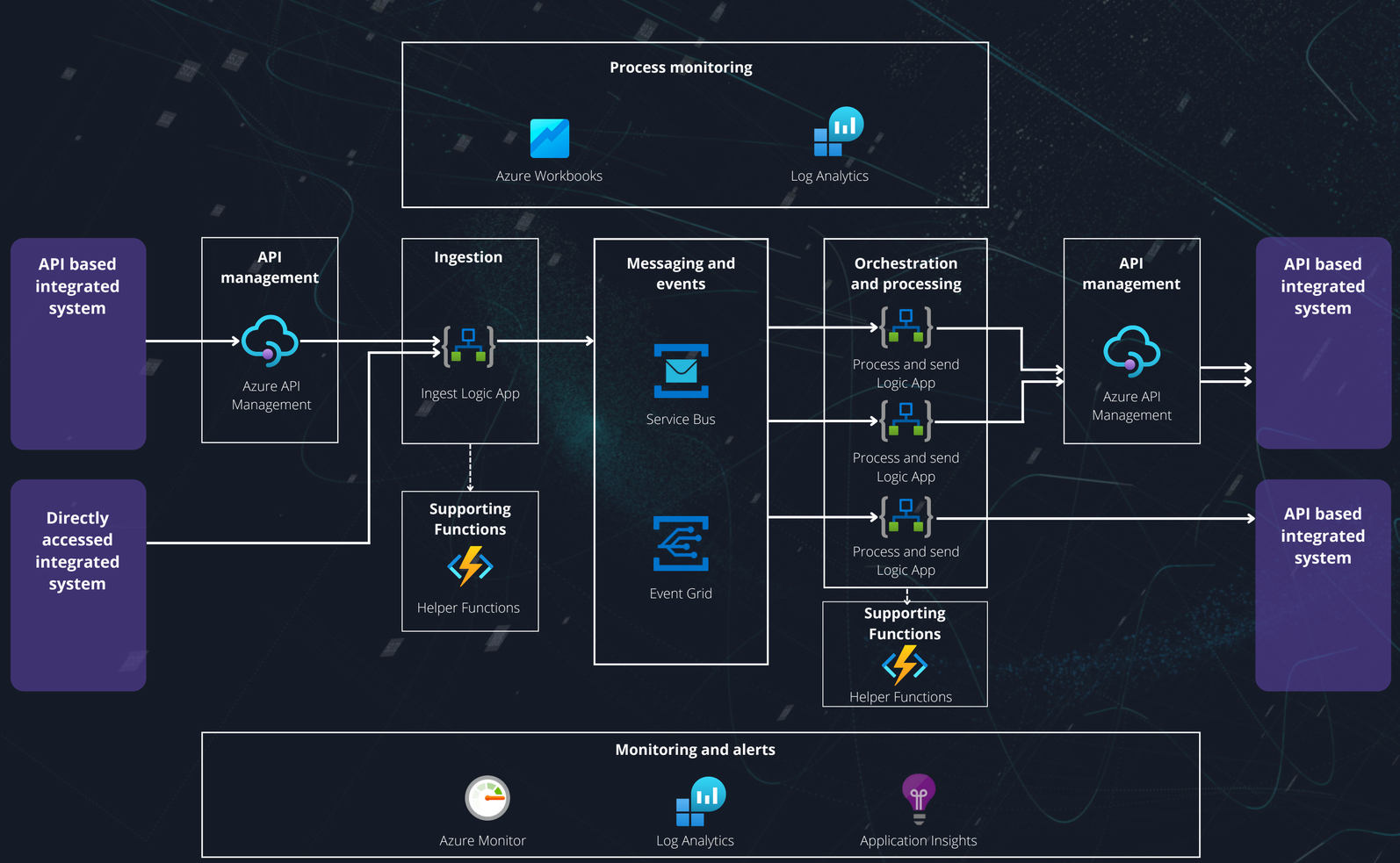
Continuous integrations and deployments are achieved using Azure Dev Ops (or GitHub). For an integration platform it is crucial that we are able to deploy incremental changes to the platform created by multiple developers or even teams. As described at the start of this blueprint, the need to rapidly provide data is seen as one of the major requirements for an integration platform. The speed to develop new features was the second most important driver for iPaaS solutions. As this is also a project management challenge that technology alone cannot solve, it is important to fit deployment patterns and best work management practices together.

For IaC and CI/CD to work beyond the initial deployment we need version controlling. Both changes in infrastructure as well as in the implementation need to be done using version control. To achieve that in a world of multi-developer environments, we provide and enforce version control best practices. This ensures that incremental changes can be deployed with minimal risks and conflicts between code branches stays at minimum as solving conflicts does not provide additional value. Using fully version-controlled development practices it is possible to fall back to the previous version in most cases. For this to be possible everything from infrastructure and configurations to the actual integration pattern needs to be managed and deployed through version control.

CLOUD1 iPaaS
CONCEPT ARCHITECTURE

API Management layer is responsible for handling incoming requests via published APIs. It functions as a front-end in the integration architecture: requests from clients are authenticated and schemas of the payloads are validated here. Therefore, it offers access control and security functionalities in front of the Ingestion layer. API management is also used to manage outbound API calls.

Ingestion layer is the back-end of the API Management layer. APIs route requests to ingestion layer, which write requests as events and messages to Event and Message layer creating asynchronous integration. Ingestion layer can also open connections to source systems to fetch data by schedule.



Event grid and Service Bus are the main messaging and event components.

Service Bus is a transactional enterprise scale messaging solution but can also be used in a less demanding scenarios for queues and once only delivery publish-subscribe patterns.

Event grid is a lightweight messaging solution for situations where at least once delivery is an acceptable outcome. Event grid is very scalable and fast and can also easily be used where massive amount of subscribers are needed.

Orchestration and Processing processes events and messages. Processing can be data transformations or identifying destination systems based on incoming data and transferring data to the destination system s

Supporting functions can be used for computing intense workloads that are not feasible to be run as logic app workflow solutions.

Platform monitoring and logging is done at two different levels of detail.

- Technical logging and monitoring for all platform services with alerts on technical level.
- Process level logging and monitoring where business data from integration process can analyzed and monitored.

CLOUD1 iPaaS
BENEFITS AND FUNCTIONALITIES

Enterprise Integration platform for near real-time system-to-system & user integrations with centralized and shared processing modules and governance.

Functionalities

- Ingest (Pull batches of data with schedules, listen to asynchronous APIs and receive data trough provided API layers that source systems can use to push data)
- Reusable transformation modules (Architectural and pattern level modular implementation makes it possible to create reusable and widely utilized processing modules)
- Serve processes (Platform can serve data by directly pushing it into destinations using event-by-event or aggregated batch patterns to multiple destinations without copying data)
- APIs and API Management (Include both receive and serve API layers with API Management that provides out-of-the box developer portal with API documentation through role-based management layer)
- Access management (Monitored user access including token-based, certificate-based and Azure AD-based authentication models)
- Process and data quality monitoring (Real time monitoring provides information for example of processes success rates, timelines and data silence)
- Event disturbance handling (Using well designed patterns, recovery is automatic from common error scenarios like network errors)

Benefits

- Initial setup and further development in an IaC and CI/CD supported manner (By using Terraform template and Azure DevOps CI/CD pipelines with GIT source control repositories. By utilizing IaC and team dedicated development environments parallel development by multiple teams comes a real option)
- Integrates seamlessly with Azure IoT and Data Platform services (Cloud1 Integration Platform is a part of the larger Cloud1 Data Hub solution and can share same resources and serve data in and out of the other parts of general system without any additional infrastructure changes)
- Supports a best-of-all-breeds methodology by implementing a module-based architecture (When additional or better suited services emerges, they can be easily benefitted. Also using services like Azure ML Services that are not part out-of-the-box solution can be easily be added as addition processing modules into the architecture)
- Static pricing - Microsoft has proven to be a trustworthy partner and as Azure based iPaaS does not include user or client licenses future costs should remain constant and directly proportional to processing and data usage.
- Technical skills (Used Azure services are widely used and know-how is easy to find, training courses and opportunities in Finland are excellent)
- An industry proven Azure Integration stack with production tested best-practice patterns provides a scalable, stable and highly self-recoverable solution (Cloud1 has 10 years experience in creating iPaaS solutions using Azure Integration Services and all of this knowledge is poured into this architecture, while we never stop developing new and better solutions as technologies evolve)

CLOUD1 iPaaS
PLATFORM SETUP PROJECT

Cloud1 is an expert at creating Azure platforms that speed up development. We focus on creating value for the business instead of putting unnecessary effort in the platform.

Azure integration platform is a scalable integration platform that uses Azure services and development tools that are generally available. These technologies are open to everyone and the way the platform is created is open for everyone.

Platform development and management is done by Microsoft best practices with the most modern available technology. Automation is built in, and the platform is development ready with the basic documentation.

Developers should be able to start the development using this platform if they have basic expertise on how to work with standard Azure integration services.

Platform creation price does not include the definition phase for the platform which needs to be done before the platform can be created: Customer specific Azure governance, network and security architecture specifications and other needs for the platform (GDPR, multivendor model etc.

Deliverables:

- Terraform automation templates for the Azure integration platform, customized for the customer Azure architecture
- Development ready environments (dev, test, prod)
- With the basic Azure Integration services (Logic Apps, Functions, Key Vault, App Service)
- Azure integration platform & architecture documentation in the integration scope
- Basic level Developer handbook for Azure integrations
- Azure DevOps CI/CD pipeline configurations ready and version control for the platform
- Development ready platform with the deployment model and automation in place
- Using current Azure best practices and customer specific needs
- Customer-specific configuration options need to be added to the fixed price. These are typically API Management, BizTalk hybrid architecture or some other custom specific needs. These options are defined in the definition & requirements phase.

Azure Integration Platform fixed price without any options: Starting from 15 000 euros.

Platform creation needs typically 1-2 months calendar time depending the definition or pre-study phase.

TYPICAL iPaaS PROJECT TIMELINE

Contracting

- Agreeing the scope
- Listing options and extra requirements

Definitions & requirements

- Workshops & documents to understand customer cloud architecture
- Typically networking, naming conventions
- Other possible requirements - can we see some non-typical use cases
- Approving plans with the customer

Access and user rights

- Customer organization is typically resposible for this phase
- It's very important that access and user rights are provided on time

Platform building

- Creating the customer specific templates and configurations with the needed Azure services
- Pipelines, deployments and integration for development process
- Basic documentation for the platform
- Test case to check the platform configuration

Platform review and acceptance

- Review for the platform and acceptance
- Ready for development - support also available
- Closing the creation project

In an iPaaS project the first thing to do is to define the scope for the first phase or iteration, depending on how agile the development needs to be to respond to changing requirements. The same process is repeated during each iteration, but it will get lighter as the process evolves and adapts to the specific business environment. This is also an excellent step to assess the overall strategy and implementation order.

After the high-level requirements have been defined, usually referred to as an epic or user story, more detailed requirements will be documented. For the very first iteration where the platform itself is initialized, there is a higher need for governance, networking and architectural decisions and discussion. However, during development some refinement to these aspects might be needed. Needs on architectural evolution should be discussed during this step.

Once the plan for the iteration or phase is approved, access in terms of user rights as well as network access needs to be tested. This is one the most common external issues for an integration implementation and as such it deserves to have a dedicated step in the process.

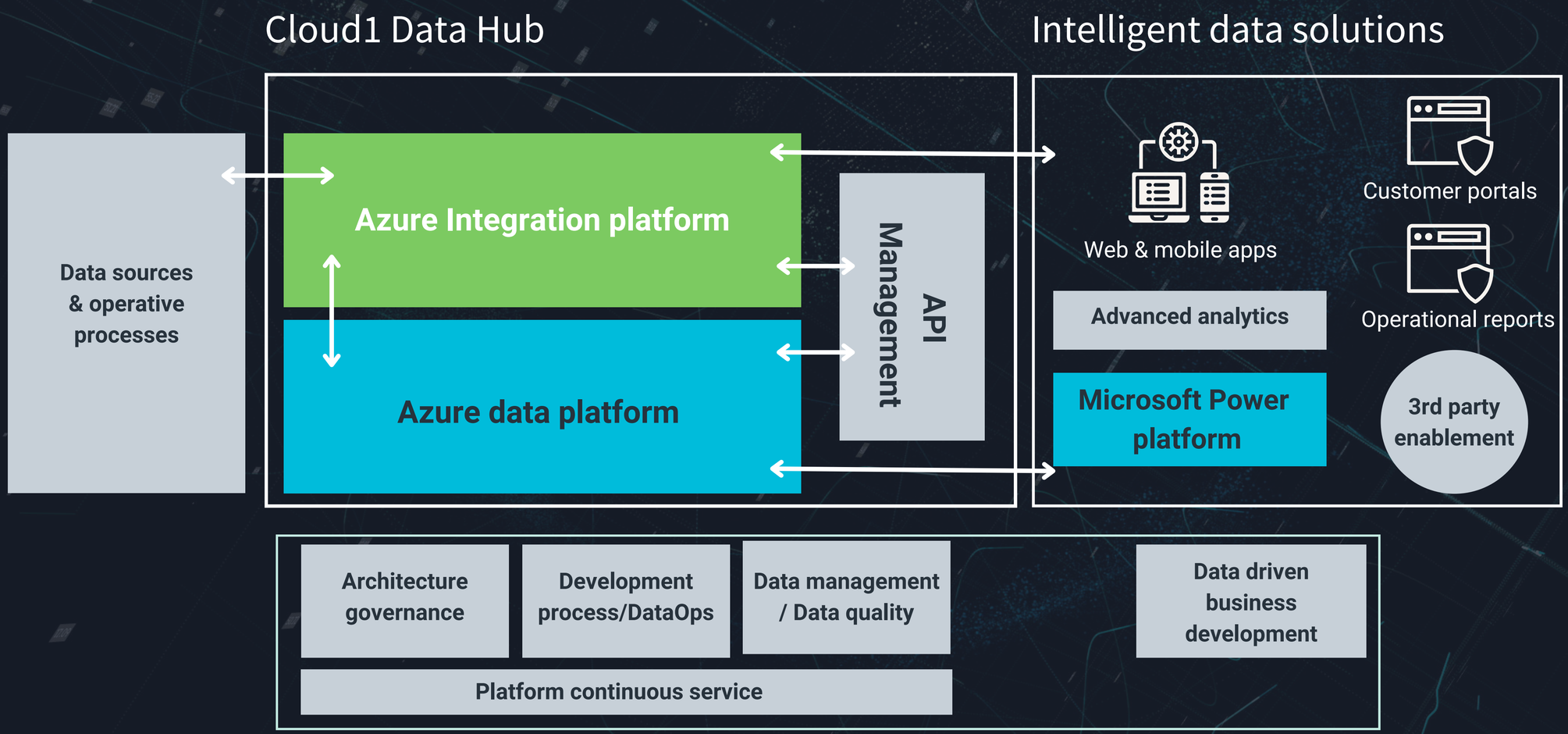
Once access is available and tested, the platform infrastructure is set up. For the initial platform build phase this part is naturally large compared to following phases. But at the beginning of each phase, it is a good practice to provide infrastructure changes upfront before the implementation work.

In the final phase the platform functionality will be validated. Maintenance support and requirements for added features are decided and provided for by the support team.

Once the initial setup phase is completed implementation iterations can begin.

CLOUD1 DATA HUB

Cloud1 IIoT platform is based on Cloud1 Data Hub architecture and is fully compatible with its additional features.



SOURCE LIST

USED SOURCES

We don't make stuff up, here are the sources to the statistics.

1. https://www.splunk.com/pdfs/dark-data/the-state-of-dark-data-report.pdf (2017)
2. https://stateofapis.com/#api-source
3. https://www.mulesoft.com/press-center/march-2021-connectivity-benchmark-report
4. https://cdn2.hubspot.net/hubfs/440197/2020-04%20-%20SOAI%20Report.pdf
5. Gartner - Magic Quadrant for Enterprise Low-Code Application Platforms 2021.
6. https://www.integrate.io/blog/the-top-benefits-of-low-code-development-platforms/
7. https://cdn2.hubspot.net/hubfs/440197/2020-04%20-%20SOAI%20Report.pdf
8. https://f.hubspotusercontent40.net/hubfs/440197/Cloud-Elements-2021-SOAI.pdf
9. https://cloudblogs.microsoft.com/powerplatform/2022/05/24/low-code-trend-report-2022-building-a-learning-culture-on-a-low-code-platform/
10. https://www.gartner.com/document/3867165
11. https://azure.microsoft.com/en-us/resources/how-modern-enterprise-integration-platform-as-a-service-ipaas-enables-business-strategy/