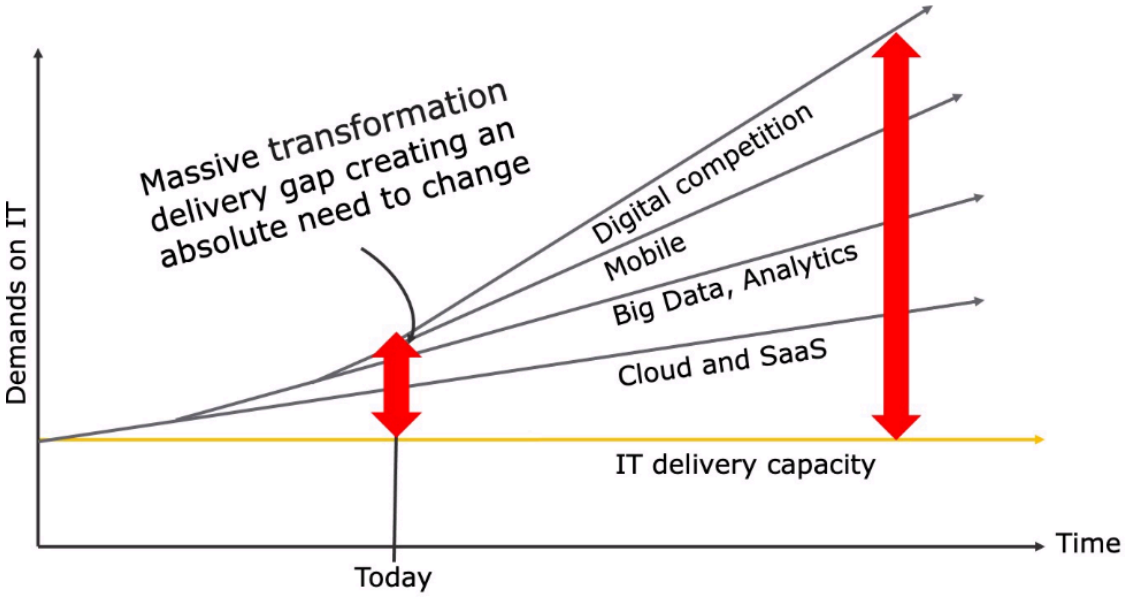
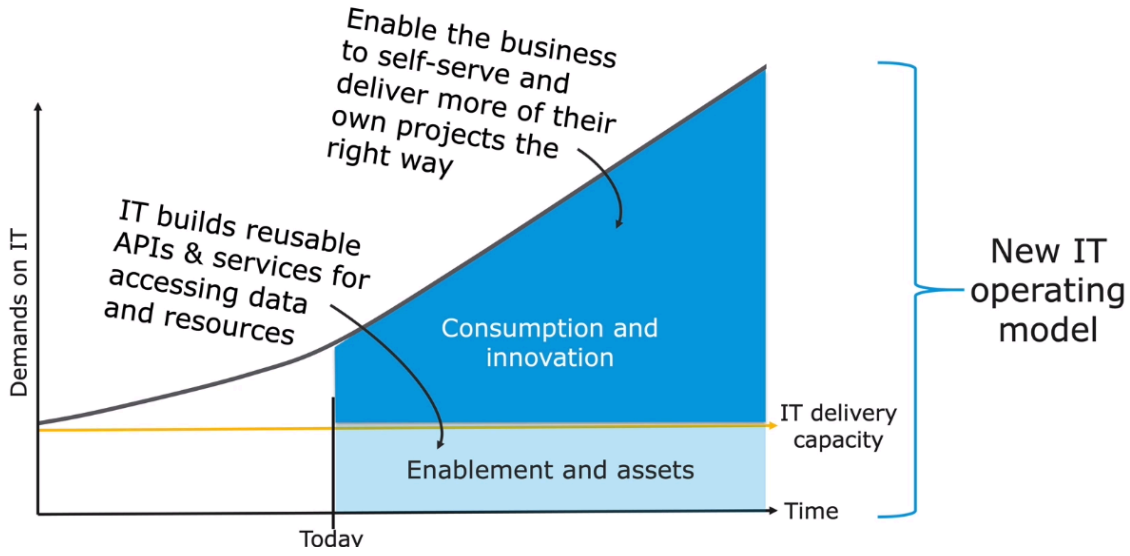
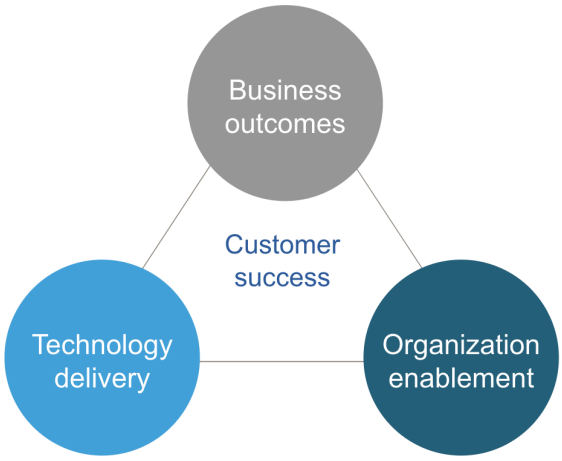
Identify the roles, responsibilities, and lifecycle phases of a typical integration project

- Identify the common reasons that IT integration projects frequently fail
- Define the IT delivery gap and describe MuleSoft's approach to closing it
- Describe the characteristics and roles of an API-led IT delivery model that emphasizes both production and consumption
- Describe the common delivery methodologies for integration projects
- Identify key DevOps practices and tools for building, testing, deploying, and delivering integration solutions
- Identify and describe the steps of the design, implement, and management stages of MuleSoft's recommended product-centric API lifecycle
- Describe the roles and responsibilities within a typical MuleSoft integration project team

1. Identify the roles, responsibilities, and lifecycle phases of a typical integration project
    1.1. MuleSoft Catalyst Playbooks
        1.1.1. Discover the MuleSoft IT Operating Model



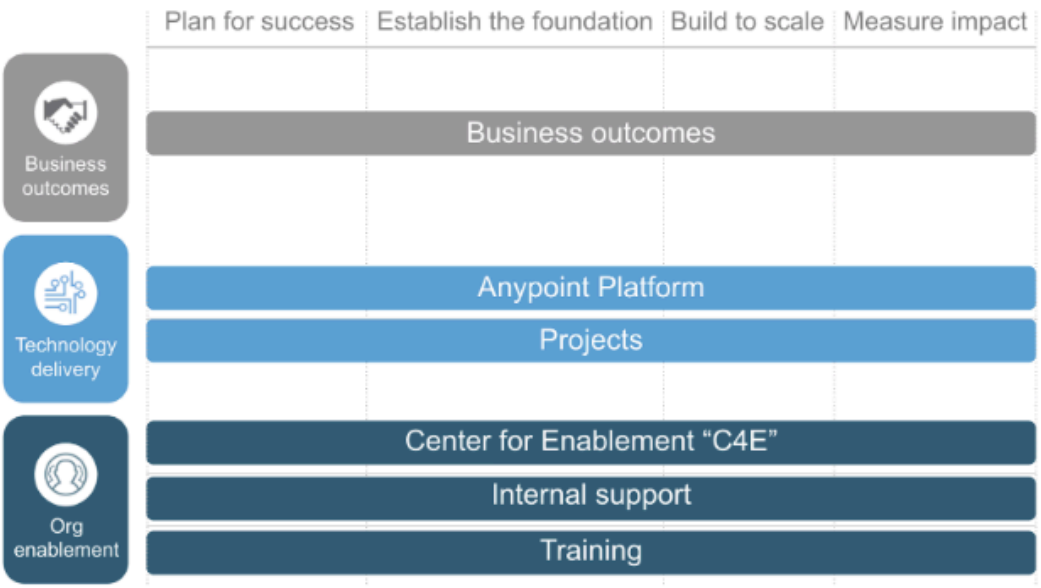




- Business outcomes: Define clear outcomes and KPIs with stakeholder alignment.

- Organizational enablement: Ensure organizational readiness with the Anypoint Platform.
- Technology delivery: Enable platform availability and team readiness to build APIs and integrations.
- https://knowledgehub.mulesoft.com

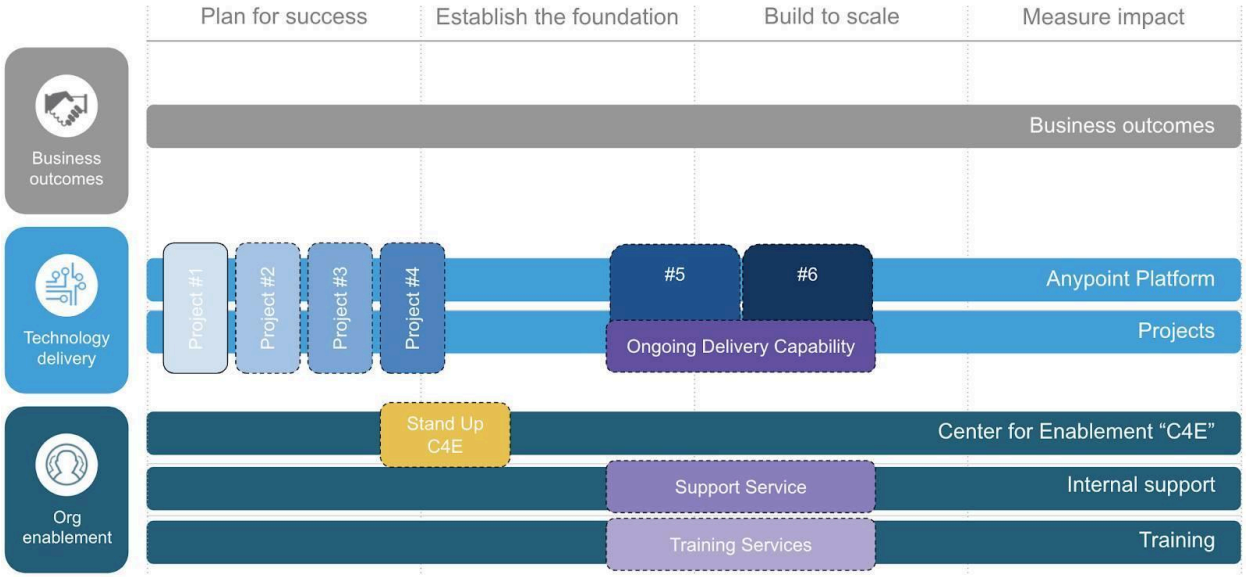
1.1.2. Learn about the MuleSoft Catalyst Delivery Methodology

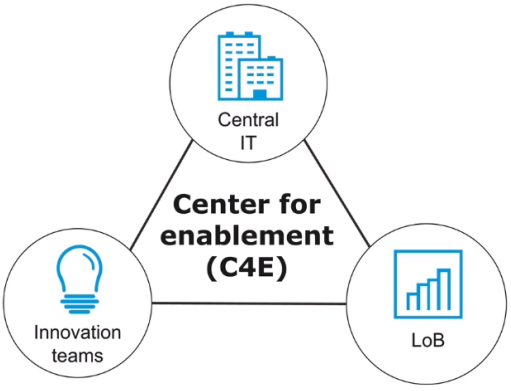


- Business Outcomes playbook: Information about how to identify and measure outcomes and align them to KPIs and stakeholders
- Anypoint Platform playbook and Projects playbook: Demonstrate the path to operate MuleSoft's Anypoint Platform
- C4E playbook: Enables organizations to maximize their results through best practices, reuse, and self-service
- Internal Support playbook: Helps customers build support models for projects involving Anypoint Platform
- Training playbook: Shows customers how to use enablement resources to provide training and certification

	Plan for success	Establish the foundation	Build to scale	Measure impact
Business outcomes	Agree on business outcomes and KPIs Develop the overall success plan	Monitor and manage	Refresh the success plan	Measure business outcomes
Technology delivery	Define Anypoint platform vision and roadmap Design Anypoint platform architecture and implementation plan	Deploy Anypoint Platform	Refine and scale Anypoint Platform	Measure Anypoint platform KPIs
	Prioritize IT projects and quick wins Staff and onboard the project teams	Define reference architecture Launch initial projects and quick wins	Onboard additional project teams Launch additional projects	Measure project KPIs
Org enablement	Assess integration capabilities Establish the C4E operating model	Build and publish foundational assets Evangelize	Drive consumption	Measure C4E KPIs
	Onboard MuleSoft Determine the internal support operating model	Staff, train and launch team Publish support guidance and self-serve materials	Monitor Anypoint Platform	Measure support KPIs
	Agree on initial roles Train the initial team(s)	Develop the broader training plan Launch experiential learning opportunities	Update training plan	Conduct skills assessment

 Note: For details about individual steps, contact your MuleSoft Customer Success Manager





- C4E is a cross-functional team
- C4E ensures that assets are
  - Productized and published
  - Consumable
  - Consumed broadly
  - Fully leveraged
- Success of C4E measured on asset consumption

1.2. Closing the IT delivery gap

Central IT will spend its time creating reusable assets and enabling the rest of the organization to use them.

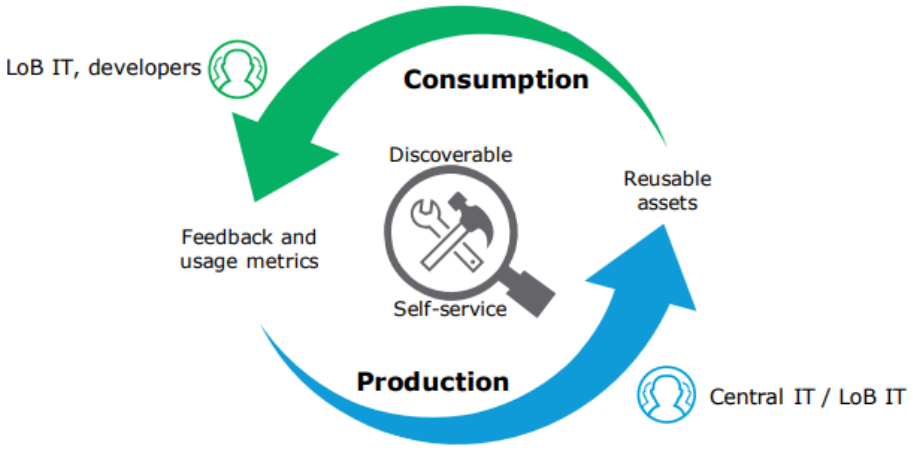


Figure 3. The production + consumption model

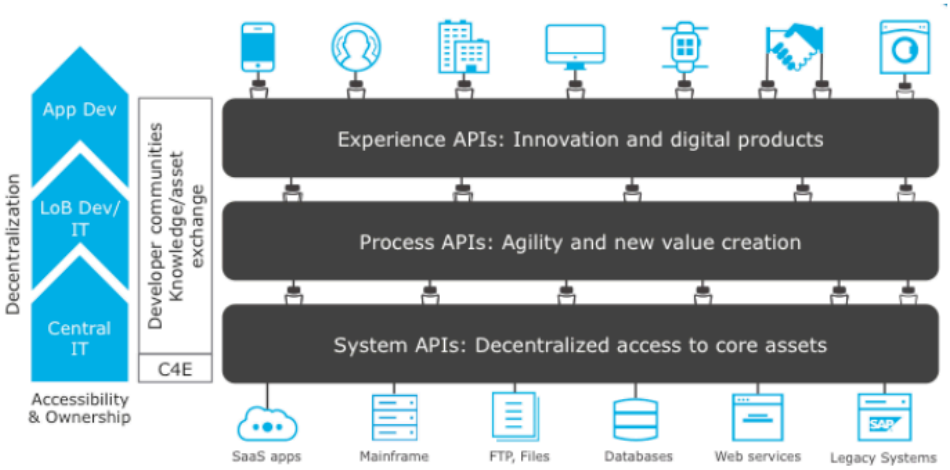


Figure 6. The API-led approach to connectivity

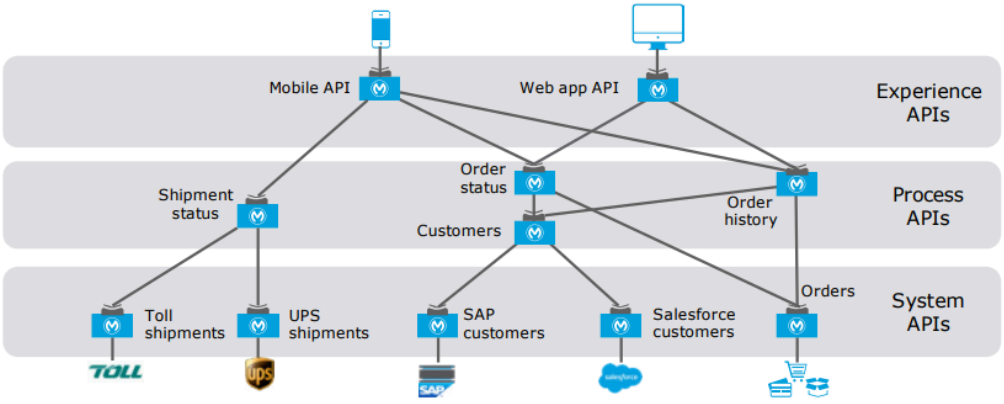


Figure 7. Web and mobile apps built with an API-led approach

C4E: APIs, templates, common platform capabilities, like logging, caching, and error handling as well as documentation, code samples, videos

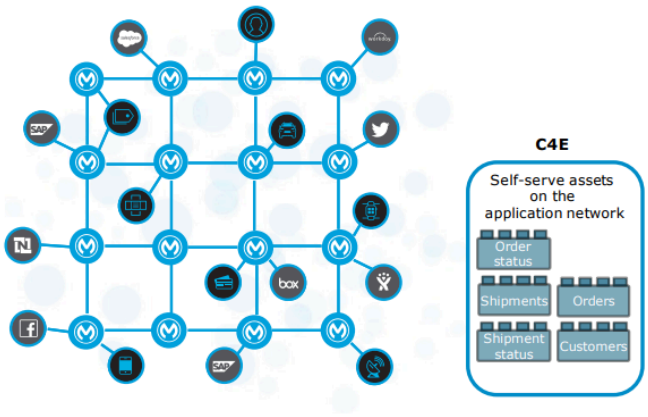
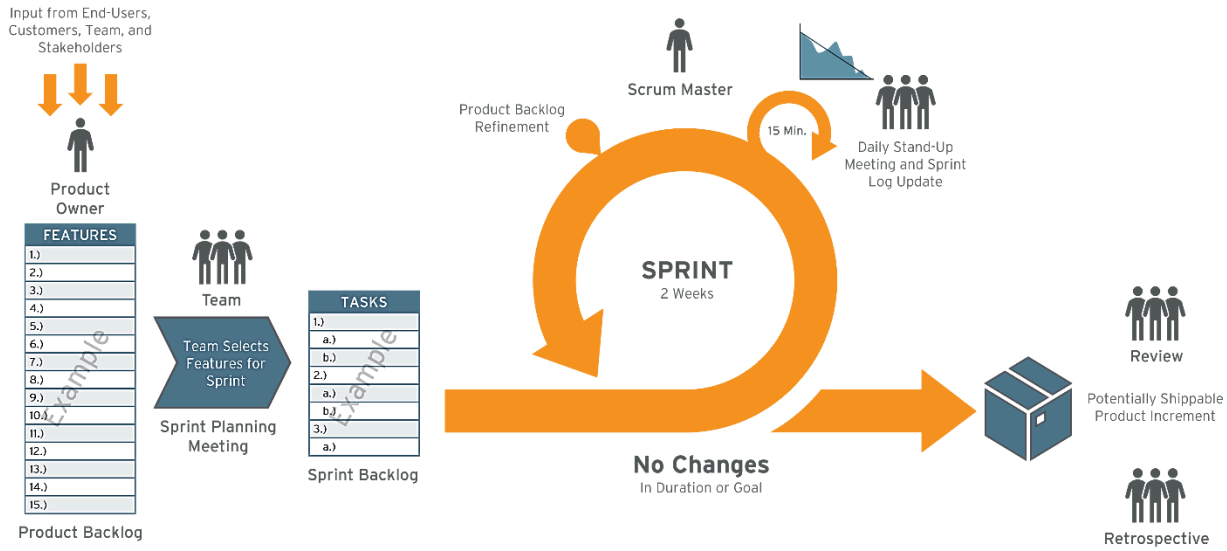


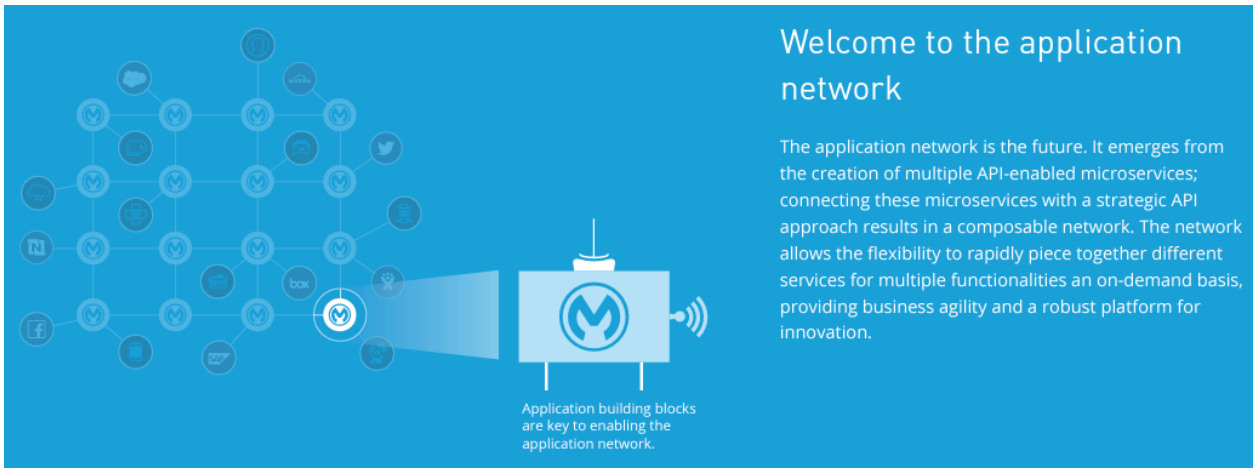
Figure 8. Emergence of an Application Network

1.3. What is agile project development?

- Kanban
- Scrum
- Extreme Programming
- Feature-Driven Development

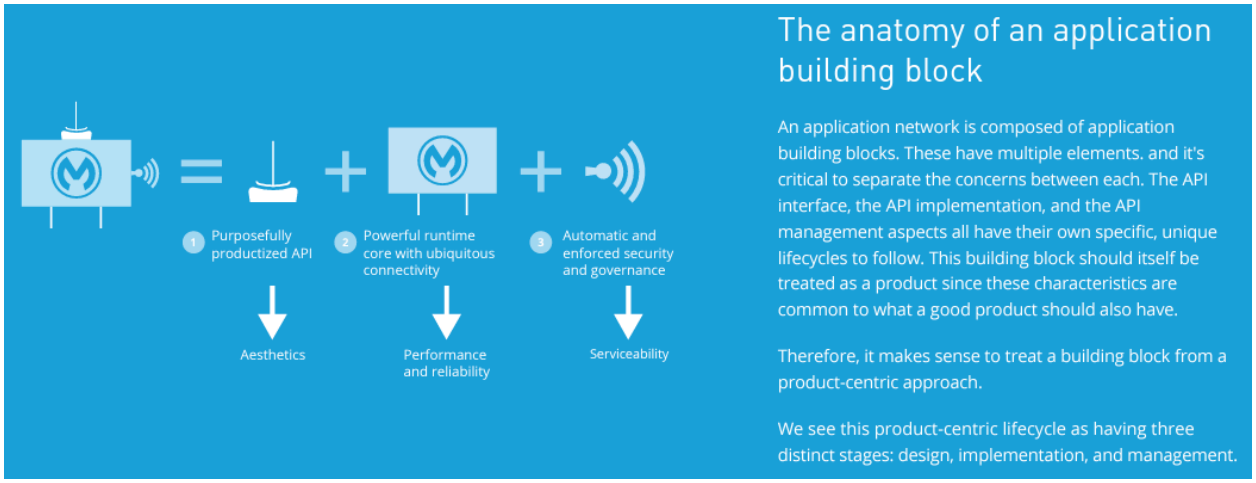


1.4. API lifecycle management



Welcome to the application network

The application network is the future. It emerges from the creation of multiple API-enabled microservices; connecting these microservices with a strategic API approach results in a composable network. The network allows the flexibility to rapidly piece together different services for multiple functionalities an on-demand basis, providing business agility and a robust platform for innovation.

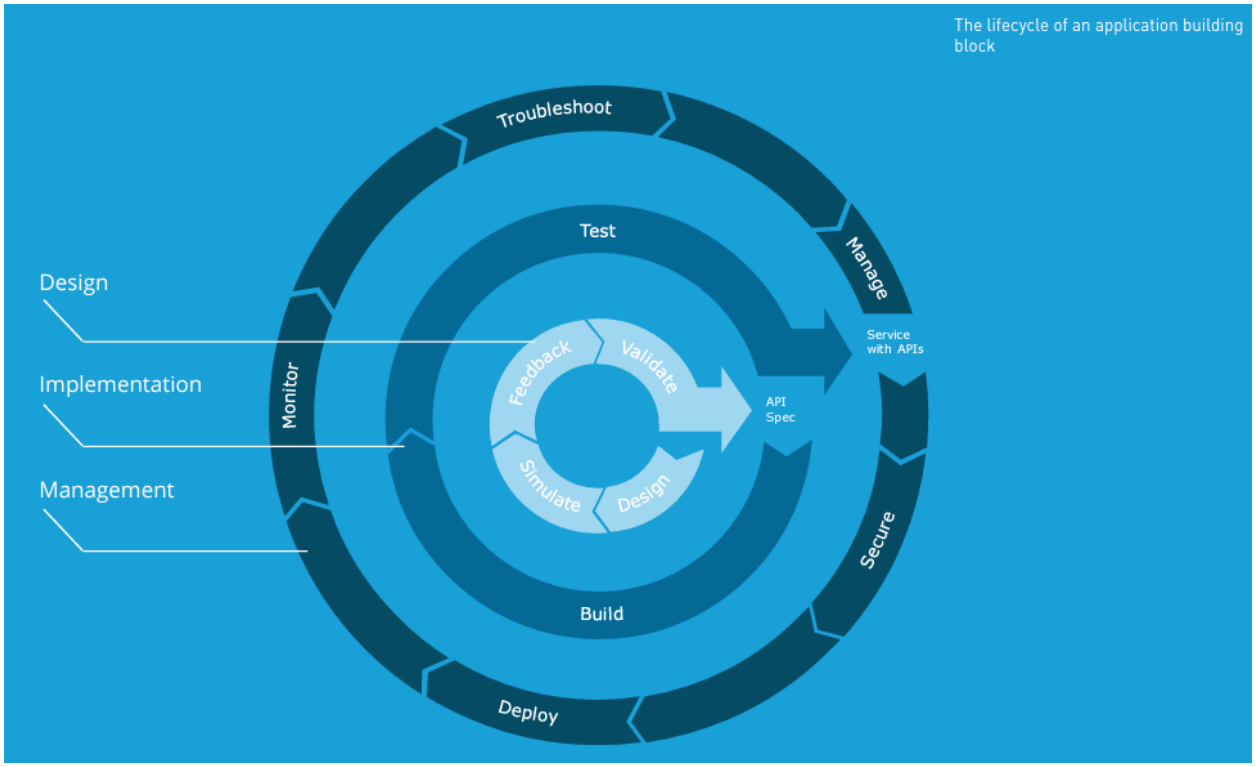


The anatomy of an application building block

An application network is composed of application building blocks. These have multiple elements. and it's critical to separate the concerns between each. The API interface, the API implementation, and the API management aspects all have their own specific, unique lifecycles to follow. This building block should itself be treated as a product since these characteristics are common to what a good product should also have.

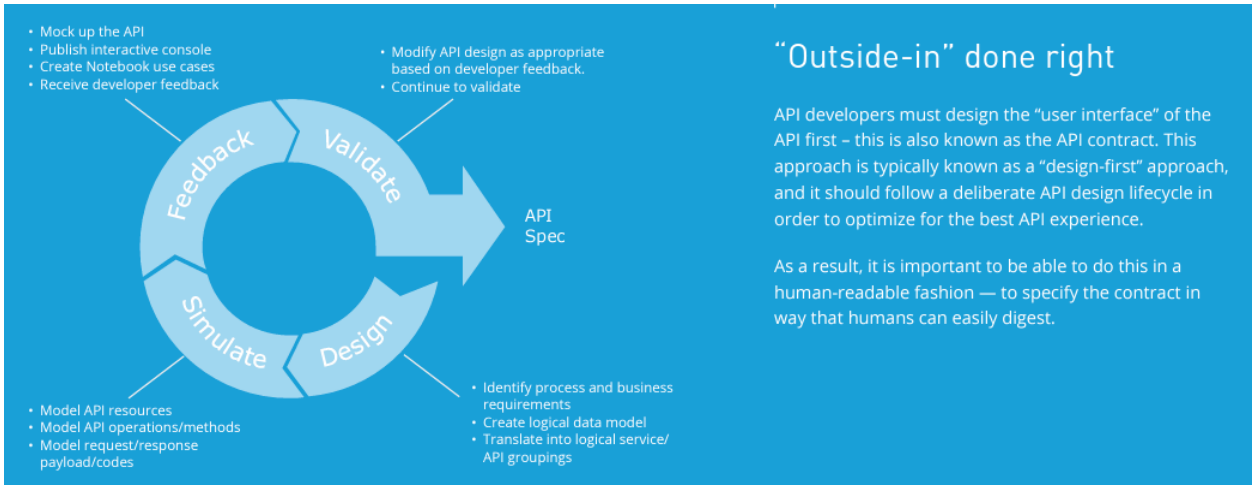
Therefore, it makes sense to treat a building block from a product-centric approach.

We see this product-centric lifecycle as having three distinct stages: design, implementation, and management.

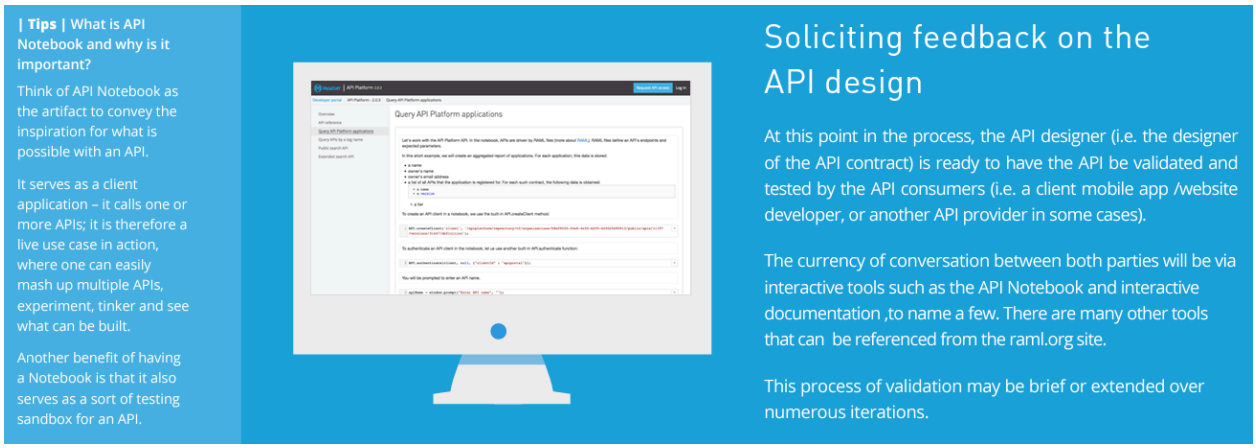


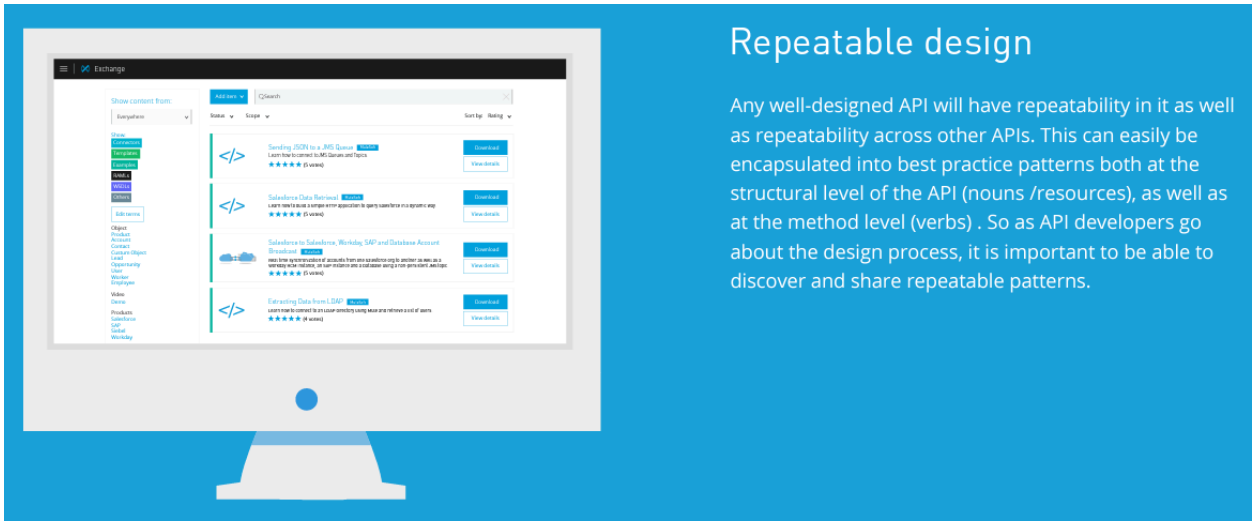
Design - DSFV, Implementation - BT, Management - DMTMS

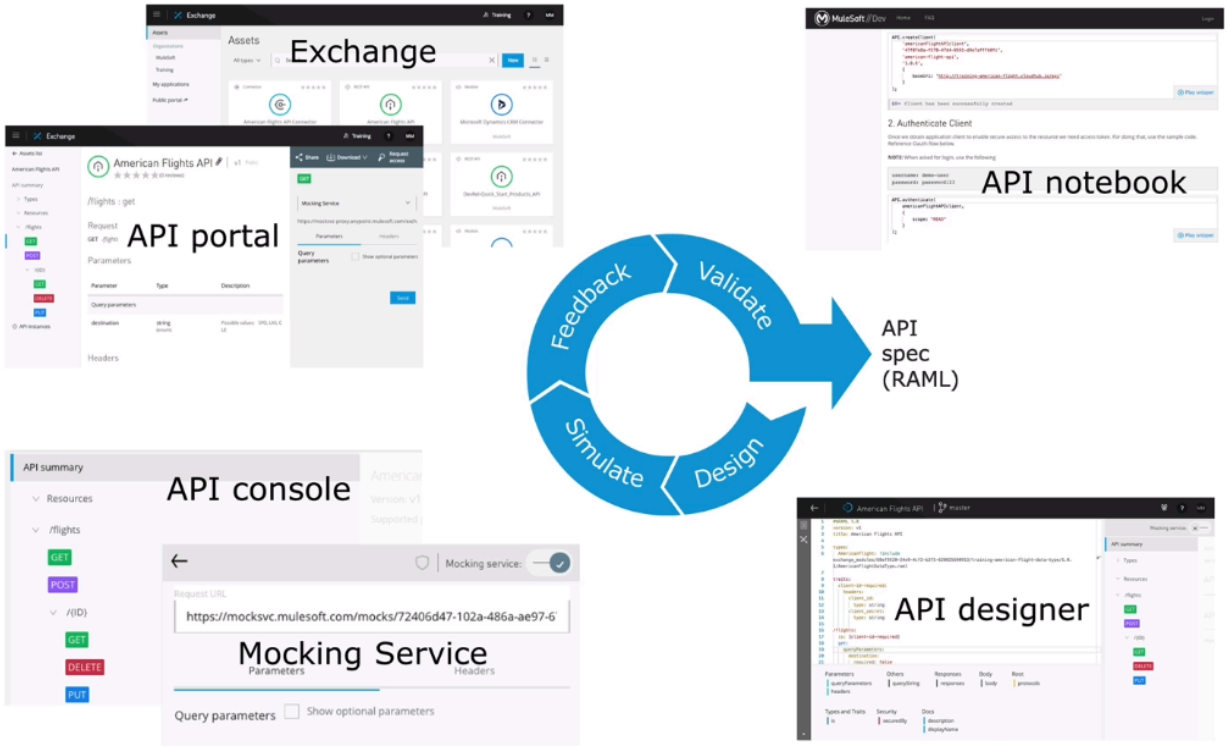
1.4.1. Design



API Spec – API Contract – Design-first approach






Exchange
API portal
API notebook
Feedback
Validate
Simulate
Design
API
spec
(RAML)
API console
API designer
Mocking Service
https://mocksvc.mulesoft.com/mocks/72406d47-102a-486a-ae97-6
Parameters
Headers
Query parameters
Show optional parameters

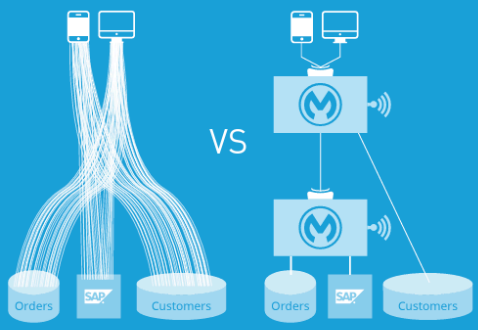
1.4.2.Implementation

| Tips | What do we mean by 'systematic' in this case? We see it as having the following architectural patterns easily available to the API developer:

- Orchestration
- Transformation
- Routing
- Data mapping
- Connectivity to popular SaaS, on-prem systems and data, file and other protocols

It also must be:

- Out of the box
- Pattern based
- Extendable and based on best practices



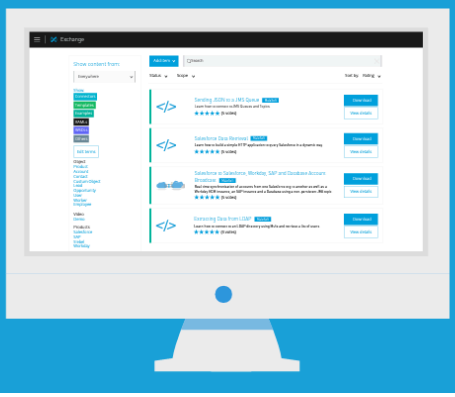
Connect systematically, not in an ad-hoc fashion

API implementation is a critical piece of enabling a next generation enterprise. Enabling for dozens, potentially even hundreds or thousands of APIs to be connected down to a backend and connected to each other will be key.

This must be done in a systematic manner (as opposed to point-to-point code).

| Tips | Best practice patterns to be leveraged when creating new building blocks.

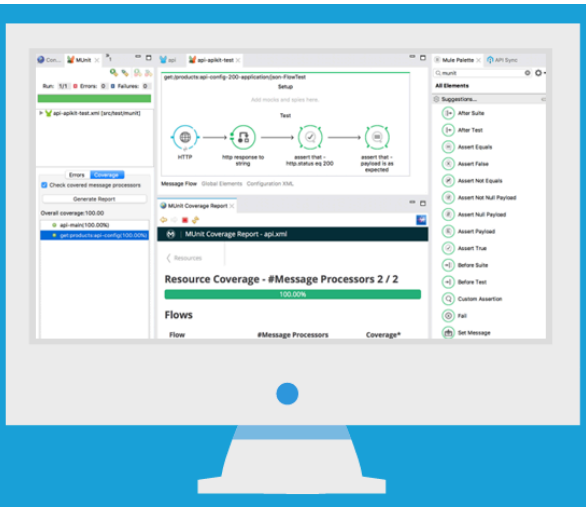
- popular database calls
- most commonly done transformations
- most popular workflows across multiple systems
- connections to popular SaaS and on-premise endpoints
- REST-SOAP transformations



Put API design principles and best practices in a common repository

Benefits of a best practices repository:

- Increase business agility
- Share best practices with reusable templates and logic
- Leverage best practice patterns
- Rapidly deploy APIs: fail fast, succeed faster
- Minimize point-to-point logic, and future proof for stability

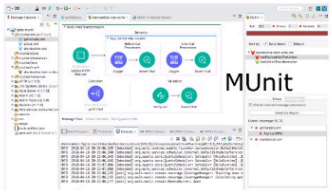


Testing the API implementation

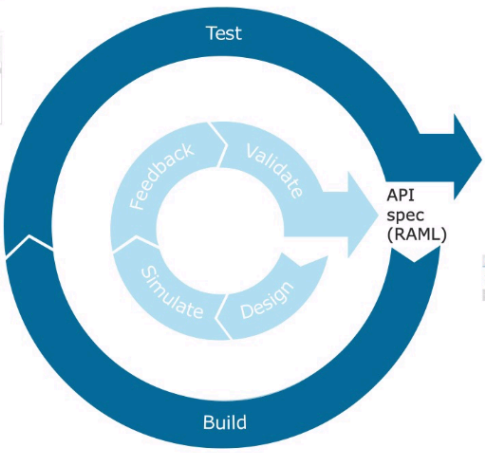
At this point in the process, the API provider (i.e. the developer behind the API implementation) is ready to have the 'guts' of the API tested.

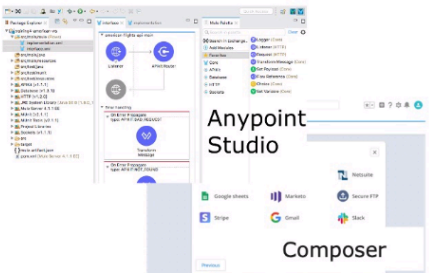
MUnit is MuleSoft's testing solution, which is incorporated into the full application building block lifecycle.

Test automation tools are critical here, as this integrates into the DevOps processes of continuous delivery and deployment.

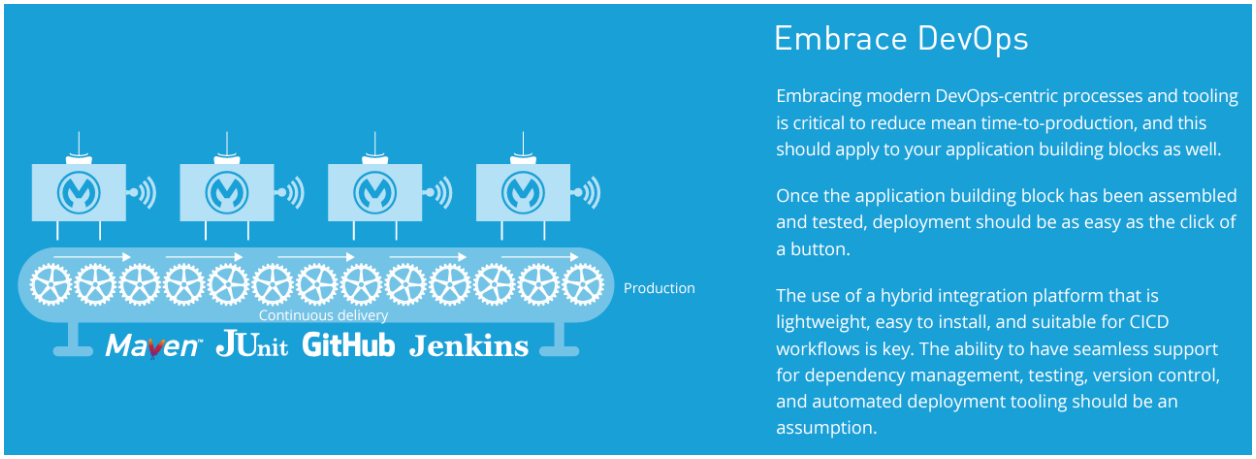


MUnit





1.4.3.Management



Embrace DevOps

Embracing modern DevOps-centric processes and tooling is critical to reduce mean time-to-production, and this should apply to your application building blocks as well.

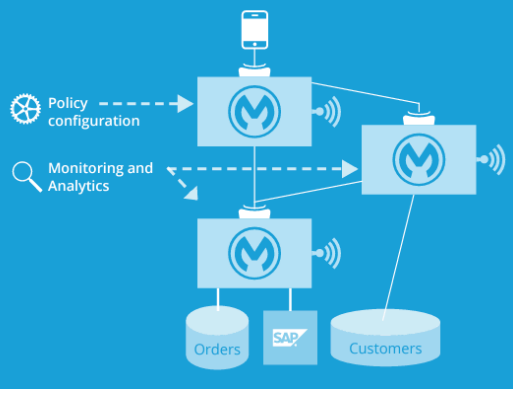
Once the application building block has been assembled and tested, deployment should be as easy as the click of a button.

The use of a hybrid integration platform that is lightweight, easy to install, and suitable for CICD workflows is key. The ability to have seamless support for dependency management, testing, version control, and automated deployment tooling should be an assumption.

| Tips |

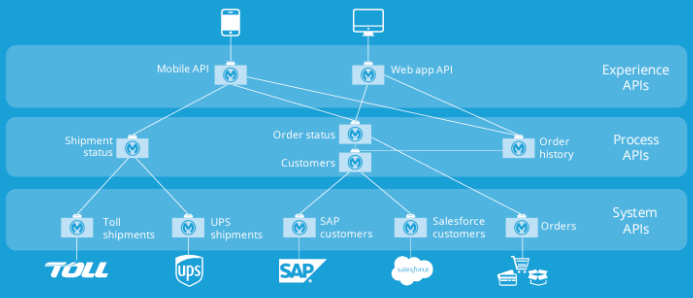
Policy configuration
Examples
- Traffic management (eg, rate limit)
- Access Policies (eg., OAuth2)
- Identity policies
- Custom policies

Monitoring and Analytics
Examples
- Infrastructure logs
- Service uptime analysis
- Client consumption data
- Provider analytics



Govern and secure all traffic

It is critical to ensure your application building blocks are following best practices in security and architectural governance by applying policies to them at runtime. Monitoring all traffic is equally critical because it just takes just one weak link to bring the ship down.

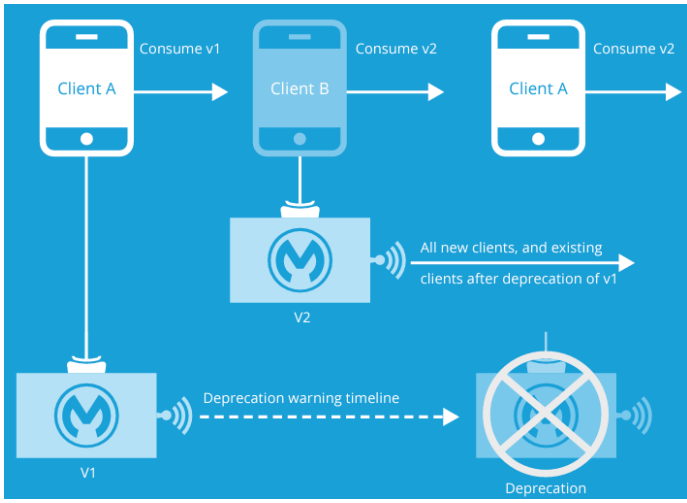


Don't forget about the discoverability and onramp

Imagine your company with hundreds — if not thousands — of APIs in your expansive application network. Imagine you're adding several new ones every day.

Being able to appropriately publish them so the consuming developer can find, research, and understand them easily could make or break your entire program.

There is no point in building something that won't ever be found, let alone used.

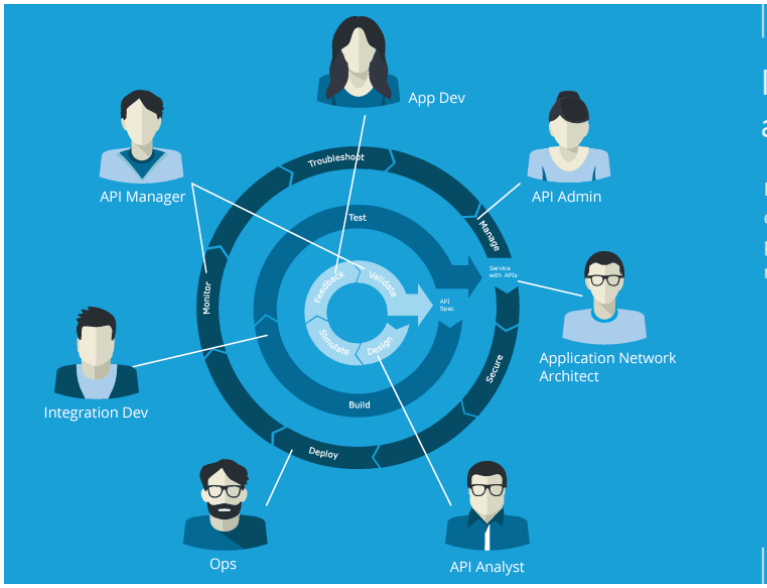


Just like any product, application building blocks change

Building blocks will change. It's a WHEN, not an IF.

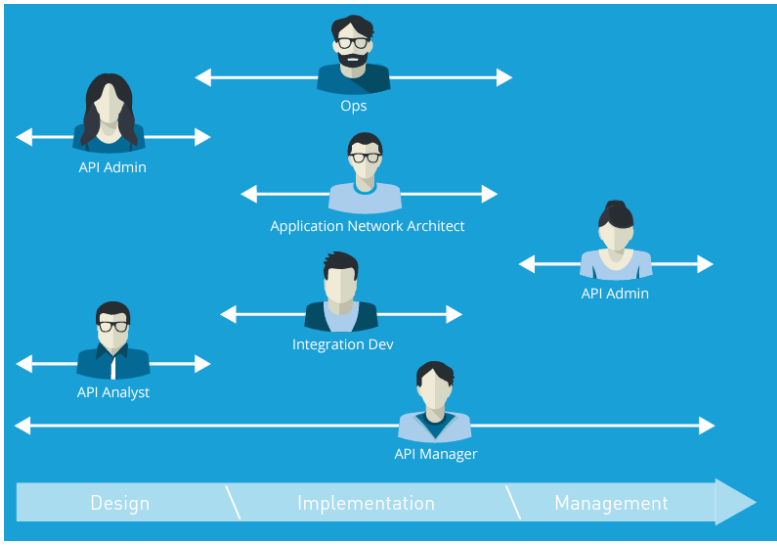
So be ready for it, which requires a carefully planned set of policies, procedures and the right platform to seamlessly migrate clients across new versions of APIs.

Getting this migration wrong will affect your customers. That's not a risk worth taking.



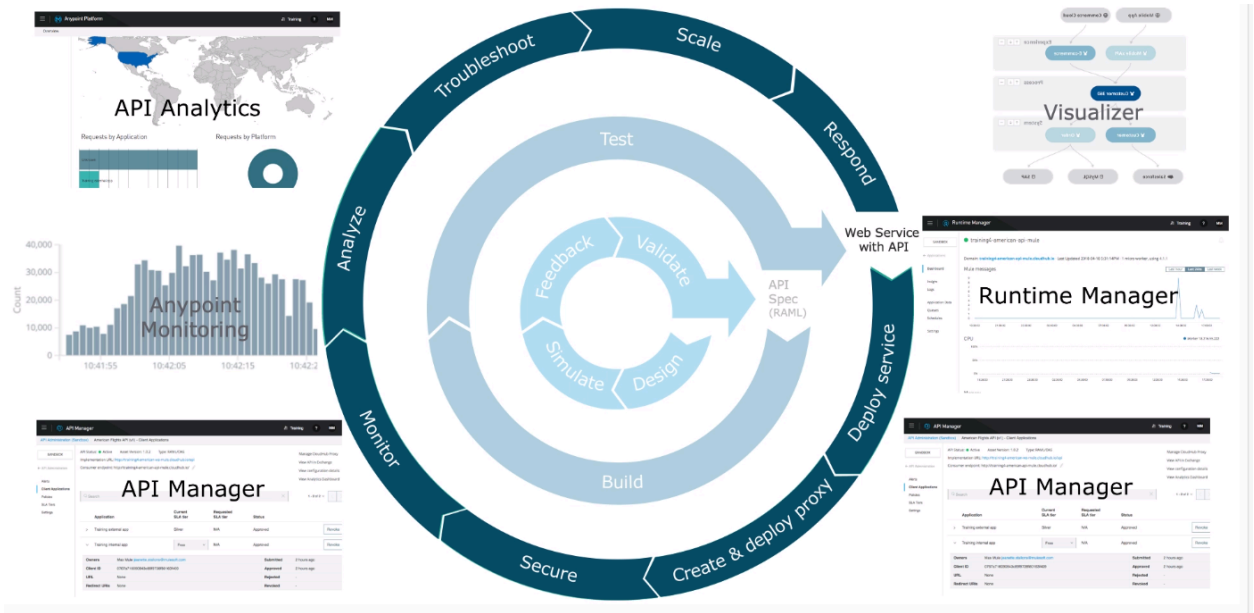
It takes a village to have an application network

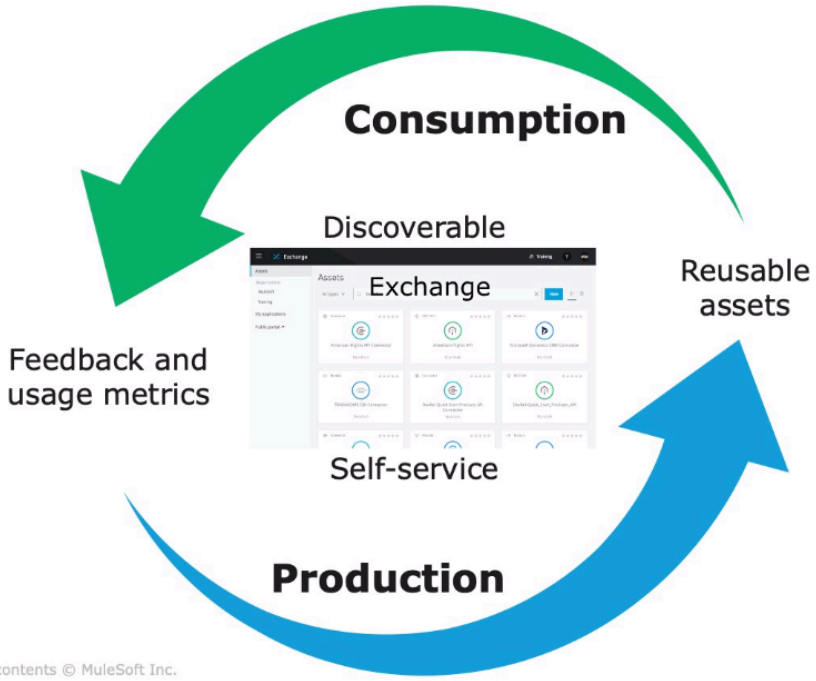
It's key to have the ability to adapt towards the new operating model, one where DevOps and lean practices are adopted, as well as the creation of new roles and responsibilities to support this new

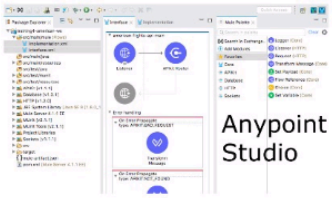


Where people play in the lifecycle

Depending on the maturity of your organization, there may be one person handling all these responsibilities or there may be multiple people doing so. The important thing to note is that each stage in the lifecycle provides specific value, which ensures application building blocks provide desired business outcomes.

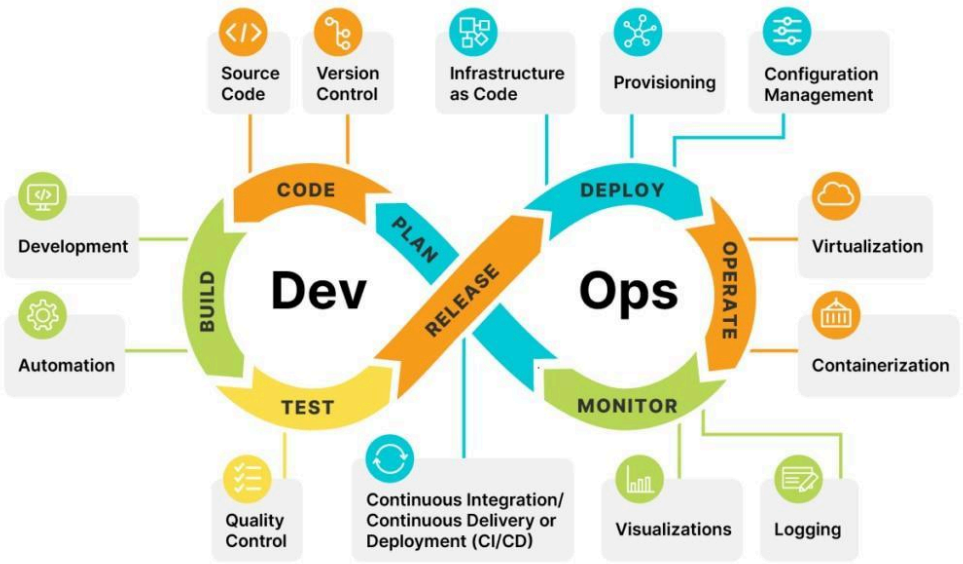


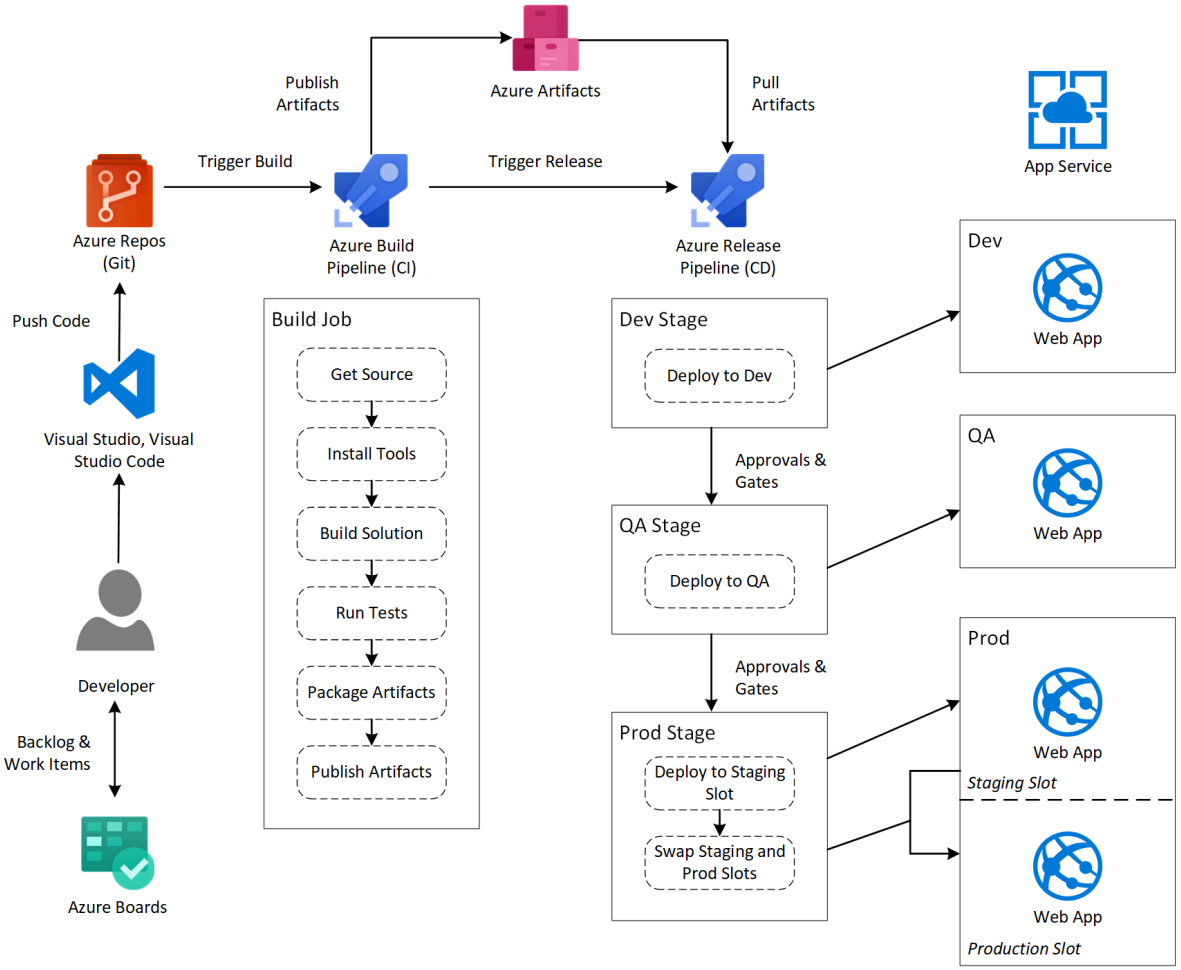




1.5. DevOps Resources

- Collaboration between IT operations and Development teams
- Continuous Integration (CI): Integrating code into a shared control repository regularly, preferably daily.
- Continuous Delivery and Deployment (CD): Delivering and deploying changes and fixes daily in a controlled, rapid manner.
- Automated Testing: Testing deployments and deliveries in a continuous, automated fashion.
- Active Monitoring: Conducting “health-checks” to ensure platforms, applications, and solutions are running appropriately.





1.6. Learning Paths

- Developer: You turn business requirements into code, conduct unit testing, as well as deploy, monitor, and troubleshoot integrations and APIs. You write software to solve problems and want to be empowered to go fast and get stuff done easily.
- Operations: You are an IT professional, focusing on deploying, managing, capacity planning, monitoring, and/or troubleshooting integrations, APIs, or both. You care about automating and streamlining the integration and deployment processes.
- Integration Architect: You make project-level design decisions and are the bridge between solutions or enterprise architect managers and developers. You care about architectural repeatability and ensure new projects are delivered according to standards.
- Platform Architect: You are in charge of cross-project design decisions — always keeping the big picture in mind — and are building towards an application network. You care about visibility across systems and clouds and focus on identifying and addressing issues before they impact the business.

Recognize and interpret essential integration concepts and terminology used by MuleSoft architects and developers
• Distinguish between Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) • Identify the types of virtualization, computing, and storage infrastructure required by enterprise systems and describe the principles of system scalability • Classify and describe common networking protocols used in system communication • Recognize the differences between common data formats (e.g., XML, YAML, and JSON) used in transformations and configuration files • Define and describe the core concepts of API and enterprise system security • Identify and describe the HTTP components that enable RESTful web services • Define and correctly use the terms API implementation, API proxy, API interface, API client/consumer, and API invocation • Identify and classify RESTful, SOAP, AsyncAPI, and GraphQL APIs

2. Recognize and interpret essential integration concepts and terminology used by MuleSoft architects and developers

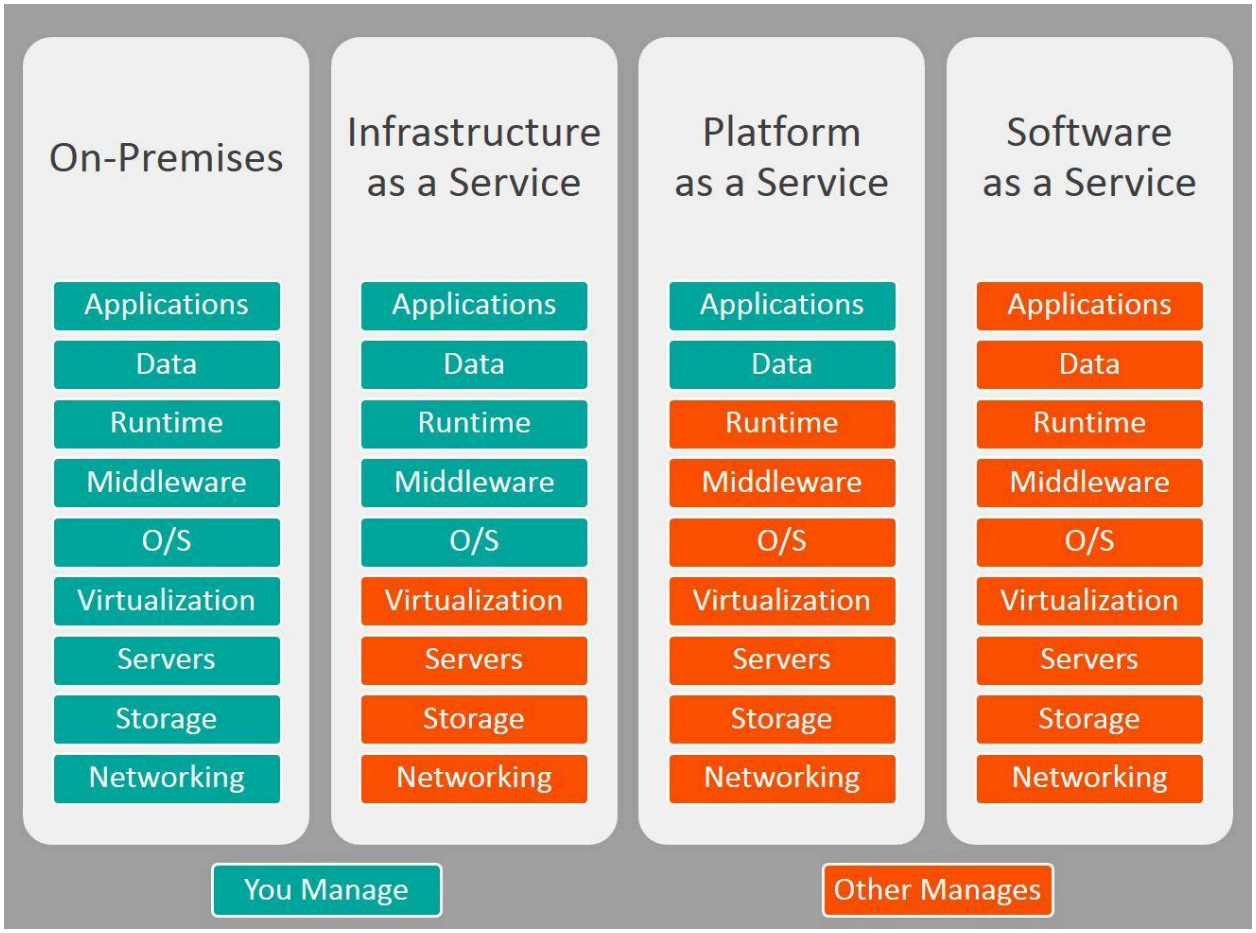
2.1. About the Cloud

- Benefits:
    - On-demand resources and rapid elasticity
    - Broad network access
    - Resource pooling
- Cloud Service Providers (CSP): AWS, GCP, Azure, etc
- Offerings: Compute, Storage, Databases, etc
- Hosting: Private, Public, Hybrid, Multi-cloud
- Computing Model: IaaS, PaaS, SaaS, iPaaS
- Scalability: Horizontal, Vertical, Diagonal
- Network Security: CIA - confidentiality, integrity, and availability

Cloud Service/Offerings	Description
Compute	Using servers/serverless resources to support workloads and provide bandwidth that can be easily scaled, allowing you to build, deploy, and manage applications efficiently.
Storage	Storing data online so that it can be accessed from multiple distributed and connected resources, allowing easy accessibility, increased reliability, and quick deployment of applications.
Databases	Software development is moving to the cloud, and databases are no exception. Performing transactions, searching, analyzing, indexing, querying, reading, and writing data are just some of the things you can do with databases in the cloud.
Migration	Moving applications and data from on-premise hosting to the cloud, including providing backup and restoration services.
Software-defined networking	Using a network architecture approach that enables the network to be intelligently and centrally controlled using software applications. This provides admins the ability to dynamically adjust network-wide traffic flow to meet changing needs.
Public key infrastructure (PKI)	Making use of a set of roles, policies, hardware, software, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption.
Key management service (KMS)	Employing a service that makes it easy for you to create and manage cryptographic keys and control their use across a wide range of services in your applications.
Secrets management	Operating tools for managing digital authentication credentials (secrets) including passwords, keys, and tokens for use in applications, services, privileged accounts, and other sensitive parts of the IT ecosystem.

Cloud Computing Model	Description
Infrastructure as a Service (IaaS)	IaaS lets customers run virtualized applications on a rented infrastructure instead of their own hardware. An example is virtual machines, which use software instead of a physical computer to allow you to operate multiple operating systems at the same time.

Cloud Computing Model	Description
Platform as a Service (PaaS)	Sitting between IaaS and Software as a Service (SaaS) in terms of functionality and responsibility is PaaS. This includes services where the cloud provider manages much more of the underlying infrastructure, such as operating system patching, and abstracts away a lot of the work for users, who in this case acquire a stable environment to run containers. PaaS is becoming increasingly prevalent.
Software as a Service (SaaS)	Some cloud providers offer up the infrastructure to run applications with managed services, like databases that a customer does not need to patch and maintain, or even complete application environments themselves. This is known as SaaS. If you've ever used Gmail, or something like it, then you've used SaaS.

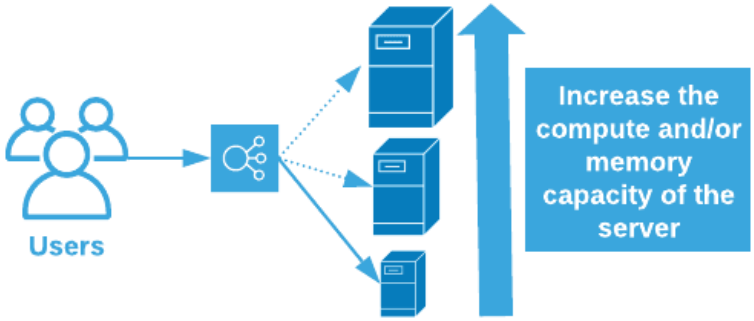


2.2. iPaaS: Integration for the Cloud

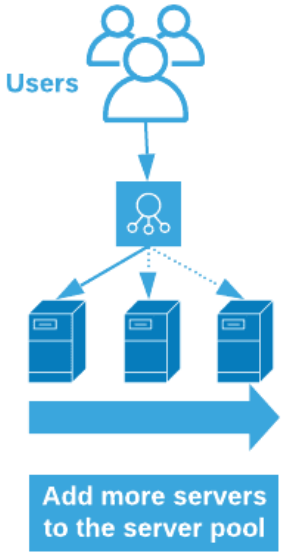
- integration Platform as a Service
- CloudHub Anypoint Platform
- End-to-end solution

2.3. What does scalability in cloud computing mean?

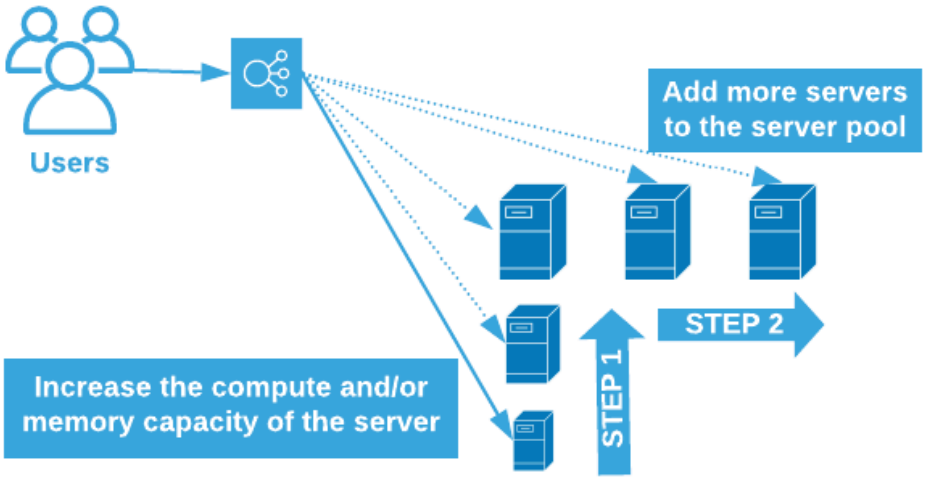
- Vertical scale or scale up



- Horizontal scale or scale out

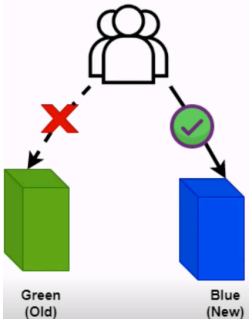


- Diagonal scaling

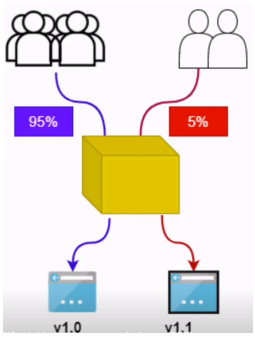


- Benefits:
  - auto-scaling
  - only pay for resources being used and for the duration it is being used
  - provision new DEV, TEST, and QA instances by simply replicating the existing instance

- advanced deployment techniques
  - big bang (recreate/rebuild)
    - rebuild the full server stack with downtime
  - rolling update (incremental updates)
    - remove and add servers in the stack in a serial fashion with no downtime
  - blue/green
    - build a new server in the stack, test, and repoint the main server to a new server with no downtime



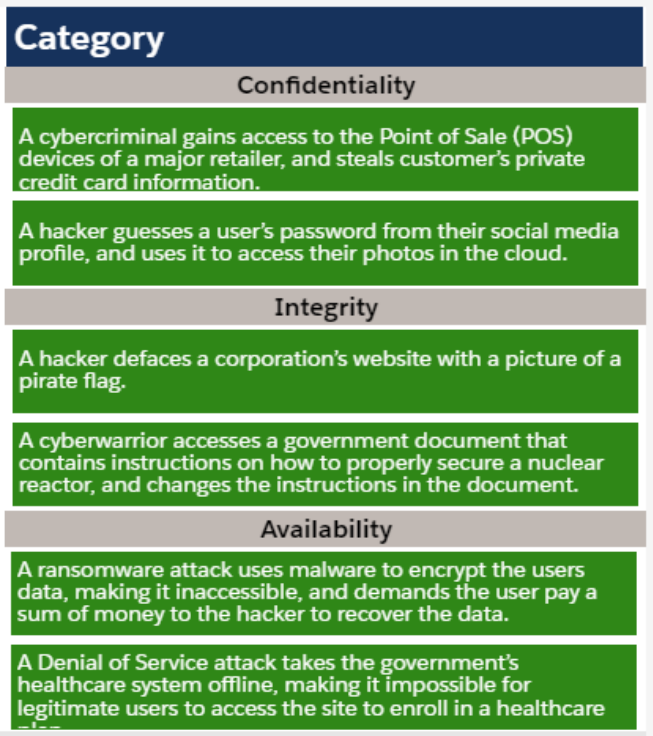
  - canary
    - blue/green in the production environment
  - a/b testing
    - latest features only to a set of users in the production environment



  - shadow

2.4. Network Security

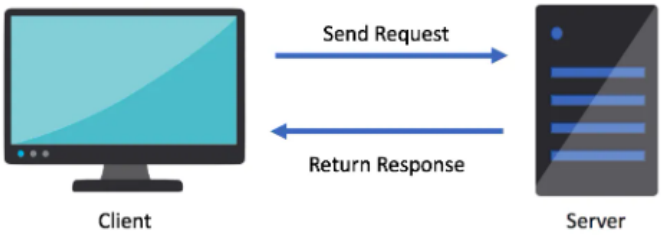
- Public Network: Public Wi-Fi
- Private Network: Home/Office Wi-Fi
    - Encryption
        - Encoded with Hypertext Transfer Protocol Secure (HTTPS)
    - Authentication
        - Username, Password, Personal Identification Number (PIN), Multi-factor Authentication (MFA)
    - CIA breach:
        - Confidentiality: Stealing the data
        - Integrity: Change the Procedure
        - Availability: Locking the data



2.5. API Basics

2.5.1. Make APIs for You and Me

- Application Programming Interface
- An API is the machine equivalent to a User Interface
- Common usage of APIs
    - Ordering groceries for your human
    - Displaying restaurant reviews based on your location



- An **API** is an **A**pplication **P**rogramming **I**nterface

- It provides the info for **how to communicate with a software component**, defining the
  - Operations (what to call)
  - Inputs (what to send with a call)
  - Outputs (what you get back from a call)
  - Underlying data types

- It defines **functionalities independent of implementations**
  - You can change what's going on behind the scenes without changing how people call it

1. **An API interface definition file (API specification)**
   - Defines what you can call, what you send it, and what you get back

2. **A web service**
   - The actual API implementation you can make calls to or the interface of that API implementation

3. **An API proxy**
   - An application that controls access to a web service, restricting access and usage through the use of an API gateway

2.5.2. Learn the Benefits of APIs

- API endpoint: GET https://api.fitbit.com/1/user/[user-id]/activities/date/[date].json
- Abstraction: A way for applications to retrieve data without having to directly engage with the source system's logic

HTTP Verbs	CRUD	Descriptions
POST	Create	Submit requested data to a server for processing
GET	Read	Retrieve requested data from a server
PUT	Update	Update and replace existing data with new data being sent in the request
PATCH	Update	Partially updates a resource
DELETE	Delete	Remove the requested data from the server

2.5.3. Put the Web in Web API

- When creating APIs you are limited to one application per endpoint – False
- Streaming API: A type of API that sends data to the consuming application as soon as that data is updated or becomes available.
- Web approach: GraphQL, gRPC
- Web services: SOAP (XML), RESTful (HTTP)

- Different software systems often need to exchange data with each other
  - Bridging protocols, platforms, programming languages, and hardware architectures

- A **web service** is a method of communication that allows two software systems to exchange data over the internet

- Systems interact with the web service in a manner prescribed by some defined rules of communication
  - How one system can request data from another system, what parameters are required, the structure of the return data, and more

	First released	Formatting type	Key strength
SOAP	Late 1990s	XML	Widely used and established
REST	2000	JSON, XML, and others	Flexible data formatting
JSON-RPC	mid-2000s	JSON	Simplicity of implementation
gRPC	2015	Protocol buffers by default; can be used with JSON & others also	Ability to define any type of function
GraphQL	2015	JSON	Flexible data structuring
Thrift	2007	JSON or Binary	Adaptable to many use cases

- SOAP web services
  - Traditional, more complex type
  - The communication rules are defined in an XML-based WSDL (Web Services Description Language) file

- **RESTful web services**
  - Recent, simpler type
  - Use the existing HTTP communication protocol

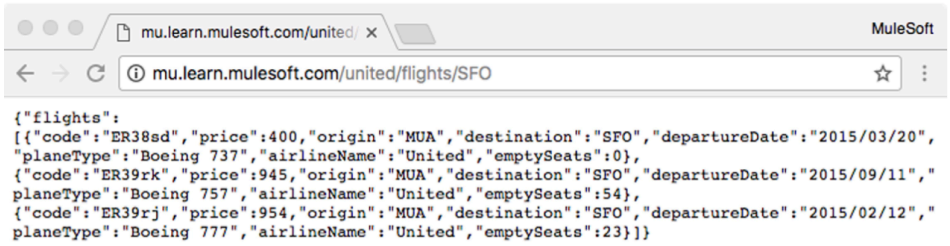
- REST stands for **R**epresentational **S**tate **T**ransfer
  - An architectural style where clients and servers exchange representations of resources using the standard HTTP protocol
  - Stateless: The server does not remember any client state from previous requests
  - Clients can cache previous responses to avoid repeated network calls

- Other systems interact with the web service using the HTTP protocol
- The HTTP request method indicates which operation should be performed on the object identified by the URL

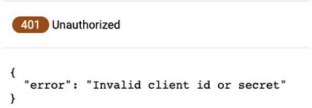
- Data and resources are represented using URIs
- Resources are accessed or changed using a fixed set of operations

- (GET)/companies
- (GET)/companies?country=France
- (GET)/companies/3
- (POST)/companies with JSON/XML in HTTP body
- (DELETE)/companies/3
- (PUT)/companies/3 with JSON/XML in HTTP body

- JSON (JavaScript Object Notation)
  - A lightweight data-interchange format (without a lot of extra XML markup)
  - Human-readable results (usually JSON or XML)
  - Supports collections and maps



- **Unsecured APIs**
  - The API may be public and require no authentication



- **Secured APIs**
  - The API may be secured and require authentication
  - You may need to provide credentials and/or a token
  - Often a proxy is created to govern access to an API
  - We will call and then later create an API secured by credentials
  - You can also secure an API with other authentication protocols
    - OAuth, SAML, JWT, and more

- RESTful web services return an HTTP status code with the response
- The status code provides client feedback for the outcome of the operation (succeeded, failed, updated)
  - A good API should return status codes that align with the HTTP spec

```
post:
  body:
    application/json:
      type: AmericanFlight
      examples:
        input: !include examples/Ameri
  responses:
    201:
      body:
        application/json:
          example:
            message: Flight added (but
```

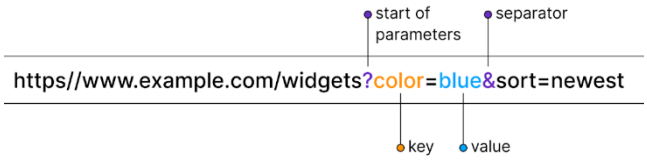
201 Created

```
{
  "message": "Flight added (but not really)"
}
```

Code	Definition	Returned by
200	OK – The request succeeded	GET, DELETE, PATCH, PUT
201	Created – A new resource or object in a collection	POST
304	Not modified – Nothing was modified by the request	PATCH, PUT
400	Bad request – The request could not be performed by the server due to bad syntax or other reason in request	All
401	Unauthorized – Authorization credentials are required or user does not have access to the resource/method they are requesting	All
404	Resource not found – The URI is not recognized by the server	All
500	Server error – Generic something went wrong on the server side	All

2 – Success RUD
3 – No change C
4 – Failure U
5 – Server Error

URL Format



https – Schema (protocol)
www.example.com – Host
widgets – Path
? – Query and Parameters

XML vs JSON vs YAML

XML	JSON	YAML
eXtensible Markup Language (data interchange)	JavaScript Object Notation (serialization format)	YAML Ain't Markup Language (configuration)
<Configurations> <Config> <Name>Ingress</Name>	{ "configurations":[{ "name": "Ingress",	--- configurations: - name: Ingress value: data/input

<Value>data/input</Value>
</Config>
 	
<Config>
<Name>Egress</Name>
<Value>data/output</Value>
</Config>
 	
</Configurations>
	"value": "data/input"
	},
	{
	"name": "Egress",
	"value": "data/output"
	}
	]
	}
		- name: Egress
		value: data/output

RAML vs OpenAPI vs AsyncAPI

RAML: Provides information to describe RESTful APIs, and can include any file content in its documentation. RAML can also support the entire API lifecycle and improve API-led connectivity.
OpenAPI: A specification for designing and documenting RESTful APIs. OpenAPI's goal is to keep documentation, client libraries, and source code in sync. OpenAPI has more adoption, a richer ecosystem, and more community support than RAML.
AsyncAPI: A specification based on Open API that describes event-driven APIs. AsyncAPI's goal is to standardize asynchronous and event-driven APIs across the industry. AsyncAPI is a solution for message-driven architectures.

API implementation, proxy, interface, client/consumer, invocation

API implementation: A program that follows the API's rules
API proxy: An interface that sits between a client and an API, providing access to the API with additional functionality such as security, caching, or rate limiting. Common types of proxies include reverse proxies, SSL proxies, and transparent proxies.
API client: Code that calls API invocations and processes requests
API consumer: Creates an API client and sends API invocations to an API
Invocation API: Part of the Java Native Interface (JNI), allows non-Java code to create a Java virtual machine, and load and use Java classes

2.6. Getting Started with Anypoint Platform – Module 1

- Refer previous sections

Recognize common integration problems, deconstruct them into their fundamental integration use cases, and identify the appropriate technologies to solve them
• Classify and describe the characteristics of common enterprise systems • Classify and describe the tradeoffs of legacy and modern integration approaches • Given a complex business problem, identify the fundamental integration use cases that can deliver an end-to-end business solution • Describe the purpose and function of the different classes of integration technologies • Identify the types of integration technologies that are most suitable to realize different integration use cases and business scenarios • Deconstruct an integration solution into its integration system constituents

3. Recognize common integration problems, deconstruct them into their fundamental integration use cases, and identify the appropriate technologies to solve them

   3.1. Connect Sales Cloud to ERP with Anypoint Platform

   - Enterprise Resource Planning (ERP): Inventory data
   - Customer Relationship Management (CRM): Customer and Account data
   - Human Capital Management (HCM): HR data
   - Configure, Price, Quote (CPQ): Solution, Pricing

3.2. API-led connectivity

Anatomy of API-led Connectivity

API-led connectivity is governed by three components:

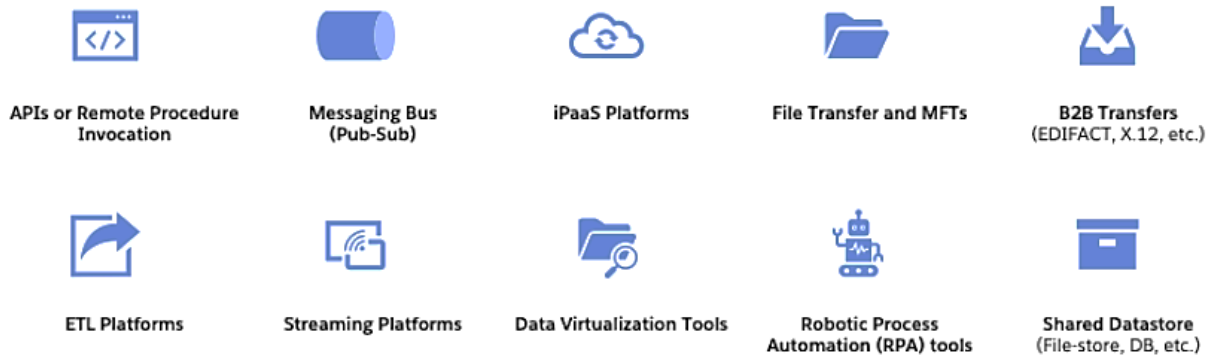
1. **Interface:**
   Presentation of data in a governed and secured form
2. **Orchestration:**
   Application of logic to that data, such as transformation and enrichment
3. **Connectivity:**
   Access to secure data from physical systems or external devices

Layer	Ownership	Frequency of Changes
System Layer	Central IT	6–12 months
Process Layer	Central IT and line of business IT	3–6 months
Experience Layer	Line of business IT and application developers	4–8 weeks; more frequently for more mature companies

3.3. iPaaS vs. full lifecycle API management: Why you need both

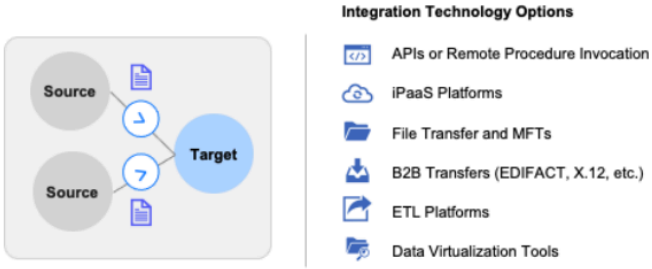
- iPaaS
    - Cloud platform
    - Users can develop and deploy integration flows between the systems without installing or managing any hardware or middleware
    - Enable connectivity to cloud-based software, SaaS, on-premises, and legacy applications
- Full lifecycle API management
    - Designing, publishing, documenting, analyzing, etc
    - Govern, secure, manage, etc

3.4. Top 10 integration patterns for enterprise use cases



#1 Aggregation of data

Businesses consolidate data from multiple sources for semantic completeness and contextualization purposes. Within these use cases, data from multiple applications are copied into a central location for further analysis and processing, for example, data warehouses, data lakes, or dashboards.



Some of the common examples of business scenarios are:

- Building a single view of customer
- Migrating data from on-prem applications to a cloud-based data warehouse
- Consolidating sensor data and ERP data for advanced analytics on usage patterns
- Preparation of executive sales dashboards

#2 Streaming data ingestion

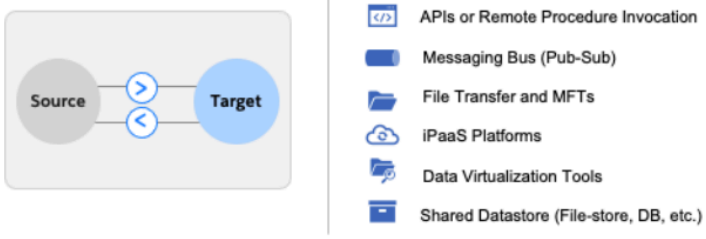
Most organizations have modern devices and objects that are capable of instrumenting and generating data to reflect the current or past state of the device. This data is vital for an organization's operations and needs to be ingested into a platform for downstream consumption and usage. Typically, such devices can generate data either at a slow pace (every minute or hour) or at a fast pace (sub-milli-second).

Some of the common examples of the business scenarios are:

- Processing IoT sensor data at edge locations
- Real-time consolidation of inventory across stores
- Consolidating user click data on retailer websites
- Collecting and analyzing machine vibration data to predict failure

#3 Data sync between multiple applications

For a given piece of data, there is generally a well-defined system of record where its lifecycle is managed. However, that data is often required within other systems to complete the business transactions within those applications. The data changes faster in the system of record and is required near real-time in the consuming applications. Consuming applications may further enrich or update certain information. More often than not, these consuming systems should not act as systems of record.

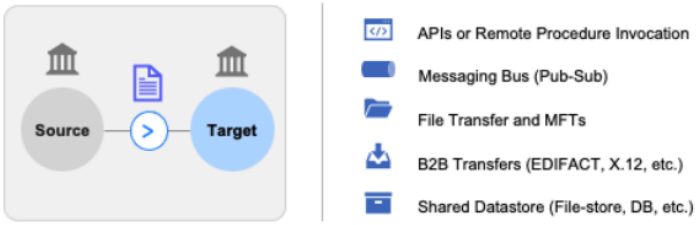


Some of the common examples of the business scenarios are:

- Upon creation of a new customer account, syncing the account details to ERP, CRM, and Financial applications. Updates to customer accounts can be made in the CRM applications and those updates are sent back to the system of record.
- Synchronizing customer data with the customer support applications so your front-line agents can better serve your customers.

#4 Sharing data with external partners

Any given organization typically works with multiple external business partners, such as suppliers, contractors, government agencies, etc. These business partners are organizations on their own with different systems, processes, and applications. Data exchange between your organization and the partner organization, however, is critical to complete business transactions.



Some of the common examples of the business scenarios are:

- Sharing purchase orders with your suppliers
- Sharing invoices for payment purposes with your suppliers
- Sending tax data with the government
- Sharing machine's operating parameters with the regulatory organization

#5 Broadcasting events

Events occur all the time within an organization. New hires, machine shutdowns or failures, expense approvals, payment issues, website traffic exceeding a certain threshold – all of these are classic examples of meaningful events that have downstream impacts and demand further actions. Identifying, capturing, and notifying these events to downstream consumers are critical aspects in these use cases.

Integration Technology Options

- APIs or Remote Procedure Invocation
- Messaging Bus (Pub-Sub)
- iPaaS Platforms
- Streaming Platforms

Some of the common examples of the business scenarios are:

- Tracking a consignment's shipment status
- Sending notifications for customer order fulfillment status
- Tracking the inventory updates in real-time
- Experiencing a retailer website crash after a surge in user clicks

#6 Bulk or batch data movement

There is an ever-growing need to see data as quickly as possible, however, there are valid scenarios where delays in presenting information is acceptable or presenting information in real-time is not required. Such scenarios optimize and provide better utilization of the resources usage in a way that constant or online availability of processing resources is not required.

Integration Technology Options

- APIs or Remote Procedure Invocation
- Messaging Bus (Pub-Sub)
- ETL Platforms

Some of the common examples of the business scenarios are:

- One-time migration of customer accounts from an old CRM to Salesforce
- Periodic migration of invoices from ERP to payment systems
- Nightly reconciliation and analysis of financial transactions from multiple applications
- Daily summarization of shipping and receiving transactions

#7 Synchronized data transfers or process triggers

Synchronous data transfers are scenarios where the receiver of the information makes a request for data transfer and waits until the sender has transmitted the requested information. During the time the data is not received, any further processing at the receiver is suspended. Such scenarios are valid when there are strong dependencies between the requested data and subsequent transactions at the receiver.

Integration Technology Options

- APIs or Remote Procedure Invocation
- iPaaS Platforms
- Robotic Process Automation (RPA) Tools

Some of the common examples of the business scenarios are:

- Retrieving last year's sales report from the CRM system
- Searching for a customer's invoice within the ERP system
- Checking a prepaid balance for your cell phone SIM card
- Retrieving a patient's prescription record before searching for medicine availability

#8 Asynchronous (fire and forget) data transfer or process triggers

Asynchronous data transfer or process triggers are scenarios where the data is sent to a target without an expectation of acknowledgment or confirmation back to the source for success or failure of transfer or process trigger. The source continues subsequent activities are sending the data or command to trigger the process. The consumers of source applications do not experience any wait time as a result of the data transfer of processing.

Integration Technology Options

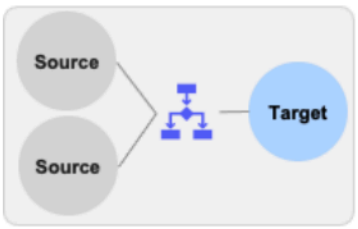
- APIs or Remote Procedure Invocation
- Messaging Bus (Pub-Sub)
- Streaming Platforms
- iPaaS Platforms
- Robotic Process Automation (RPA) Tools

Some of the common examples of the business scenarios are:

- Sending promotional texts or emails to prospective customers
- Triggering an expense approval workflow
- Processing of your insurance claim receipts

#9 Orchestration and data processing

Significant semantic clarity is built when data from multiple sources are brought together and orchestrated. Orchestration of data provides additional context or clarity. In this use case, a layer of abstraction is developed between the source and target where deeper business logic is embedded. Often the orchestration use cases are heavily tied and specific to the underlying business domains.



Integration Technology Options

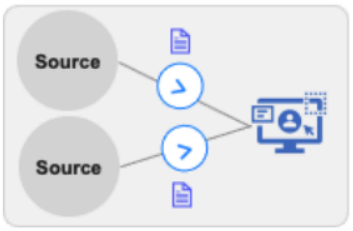
- iPaaS Platforms
- Robotic Process Automation (RPA) Tools

Some of the common examples of the business scenarios are:

- Activating a customer account – credit check, background verification, employment validation, etc.
- Processing a new hire employee – salary account setup, IT assets delivery, work locations, etc.

#10 User interface integration and mashups

Integration is often thought of as the transfer of data among applications for machine-level consumption, as described in previous use cases, however, presentation of information in a consolidated format for human consumption is a widely known use case. Information is aggregated in a central location for human consumption.



Integration Technology Options

- APIs or Remote Procedure Invocation

Some of the common examples of the business scenarios are:

- Portals, Intranets
- Public Websites
- Mobile apps

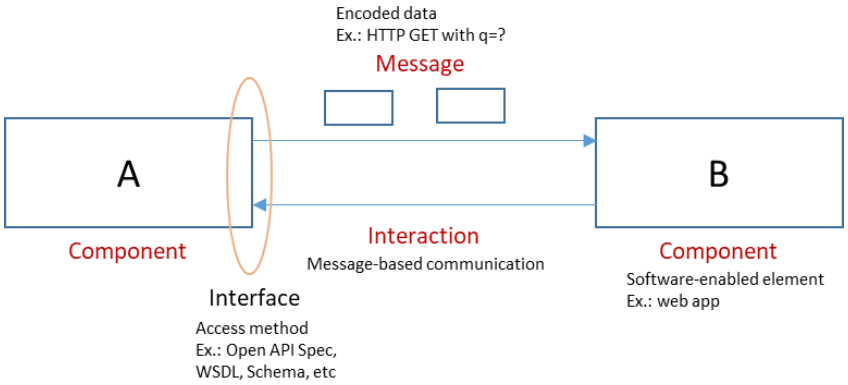
Explain the common technical complexities and patterns that are central in integration development
• Describe the differences between the request-reply, one-way, multicast, batch, and stream interaction patterns • Explain the differences between the aggregation, orchestration, and choreography composition patterns • Describe the purpose of an API specification and the benefits of following a design-first approach to API development • Describe and compare observability approaches for integration solutions including logs, metrics, and tracing • Describe the differences between cloud, hybrid, and on-premise deployment architectures • Describe the differences and tradeoffs between monolithic and microservices application architectures • Describe the difference between a service mesh and an API gateway

4. Explain the common technical complexities and patterns that are central in integration development

4.1. A primitive look at digital integration

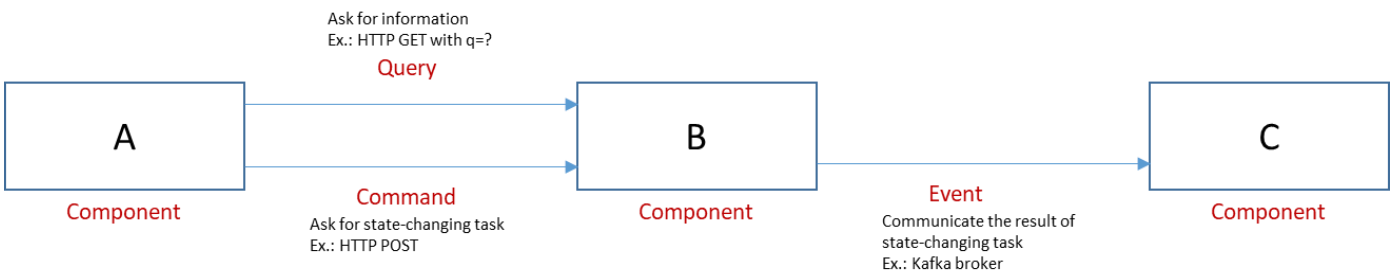
- integration system primitives

Primitive	Definition	Examples
Component	Software-enabled element of the system to be integrated	• Back-end application that exposes its functionality as a web API • Microservice publishing events to a broker • Mobile app consuming back-end GraphQL data
Interaction	Message-based communication between components	• Mobile banking app retrieving recent credit card transactions from account system • Distributor placing EDI order to supplier • Investment application producing stock price change event, consumed by multiple
Message	Encoded data sent from one component to another as part of an interaction	• HTTP GET with customer identifier encoded as a query parameter • New customer event serialized in Avro and sent over Kafka
Interface	Access method for a component that specifies the protocol and data rules for messages	• Web API with OpenAPI specification • Web service with WSDL • GraphQL API with schema



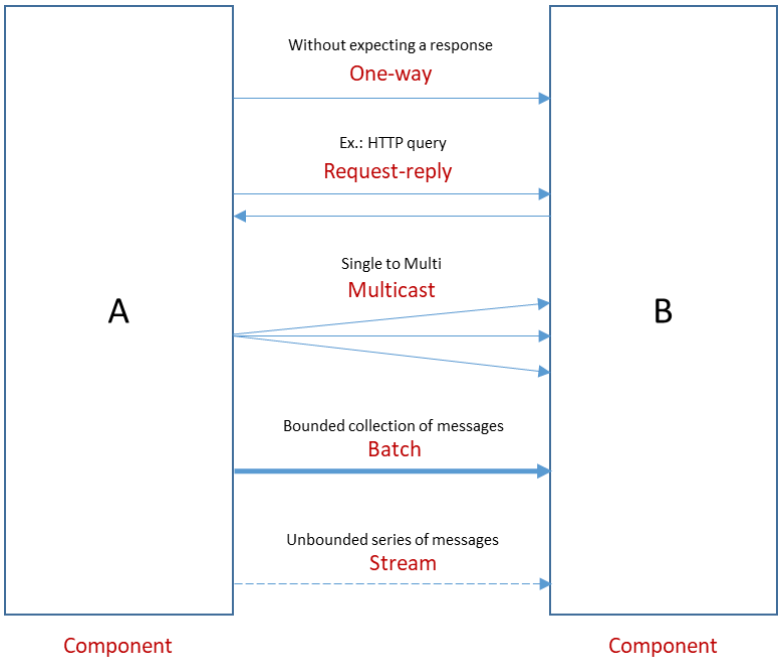
- interaction type primitives

Primitive	Definition	Examples
Query	Interaction where a component asks another component to provide specific information	• HTTP GET of store hours • SOAP call to getAccountBalance • GraphQL query
Command	Interaction where a component asks another component to complete a state-changing task	• HTTP POST to add a new store location • SOAP call to transferFunds • GraphQL mutation
Event	Interaction where a component communicates the result of a state-changing task to one or more other components	• Callback on HTTP webhook to notify that a particular store location just opened • Event published to Kafka broker that a high value customer just signed on to online banking • GraphQL subscription



- interaction pattern primitives

Primitive	Definition	Examples
Request-reply	Component sends a message to another component and expects a response message, either immediately or delayed	• Any HTTP query • Update to customer record with returned acknowledgment via GraphQL
One-way	Component sends a message to another component without expecting a response	• Cancelled point-of-sale purchase (store and forward)
Multicast	Component sends a single message to multiple interested components	• Updated price for product (over JMS)
Batch	Component sends a bounded collection of messages to another component	• Daily transaction file sent from online payments application to settlement and reconciliation batch process
Stream	Component sends an unbounded series of messages to another component	• Audit of user actions during a signed on session to a web application (Kafka)



- interaction composition primitives

Primitive	Definition	Examples
Aggregation	Stateless composition, fan out and fan in	• Mobile app back-end aggregates multiple back-end calls to format single "customer profile" response
Orchestration	Composition coordinated by a central component	• Address change process that includes data validation and propagation to multiple product applications
Choreography	Reactive composition based on event triggers	• Order is received, inventory service notified, fulfillment process triggered

- **A**ggregation
  - o stateless composition
  - o fan-in, fan-out
  - o receives a single query request, executes multiple queries to other components, assembles their responses to formulate a reply
- **O**rchestration
  - o stateful coordination of interactions by a central component
  - o flow may change depending on the intermediate results
- **C**horeography

- reactive coordination of component actions based on interaction triggers and resulting events

4.2. Choosing a great API spec saves time and hassle

- Spec-Driven Development
- REST API you build will be long-term focused
- RESTful API Modeling Language (RAML)

4.3. Everything you need to know about observability in Anypoint Platform

- logging, analytics, monitoring, troubleshooting, and measurement
- System = platform, runtimes, applications
- Provides real-time pulse of a system
- “you can’t improve, what you can’t measure,” “you can’t analyze if you can’t collect data,” and “you can’t solve unless you troubleshoot”

Pillar #1: **L**ogs

- messages logged by an application, system, or OS when an event occurs
    - Ex.: transaction processing, start or stop of the application, user audit

Pillar #2: **M**etrics

- measurements
- logged periodically
    - Ex.: Sensor sending temperature reading

Pillar #3: **T**races

- complete transaction journey from beginning to end

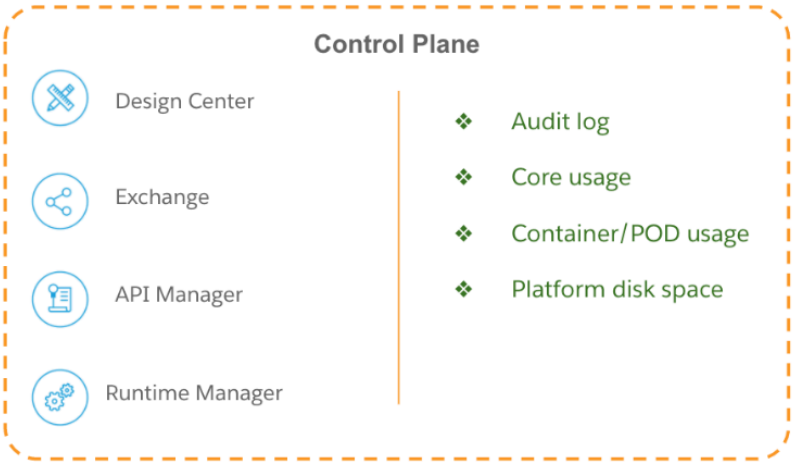


Figure 1: Observables in Anypoint Control Plane

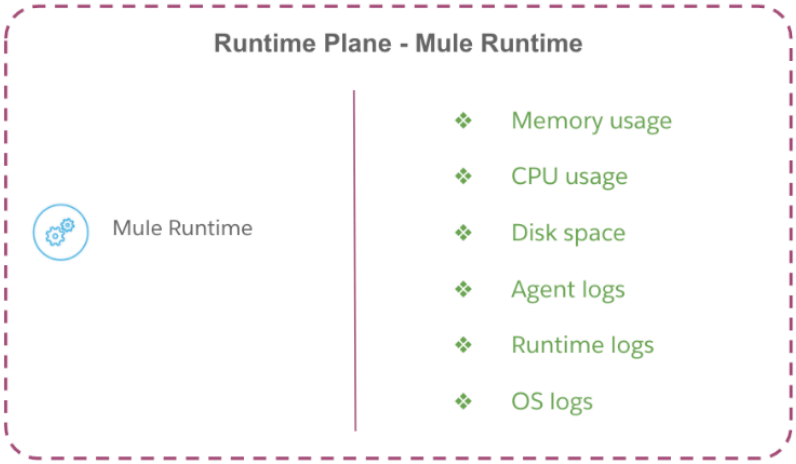


Figure 2: Observables in Anypoint Mule runtime

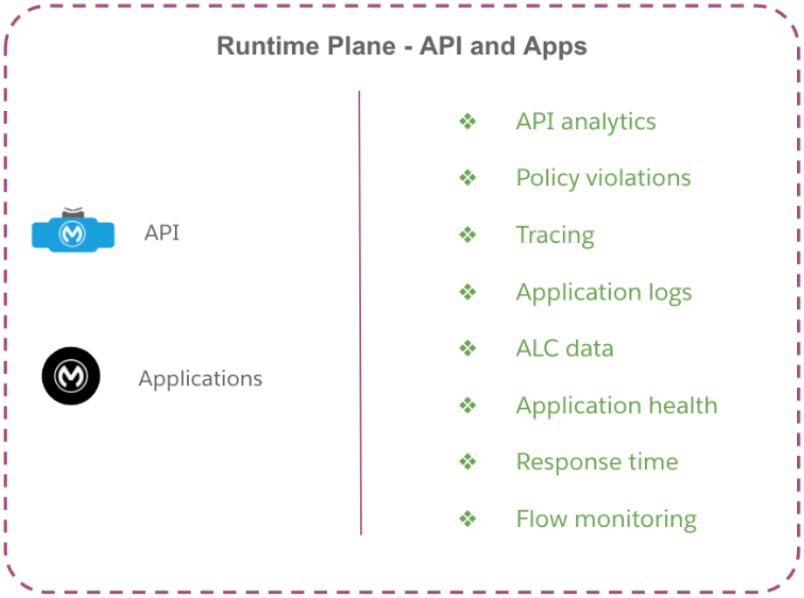


Figure 3: Observables in Anypoint runtime APIs and applications

Cloud

All cloud deployment option with two flavors. Anypoint Commercial solution and Anypoint Government solution

Hybrid

Hybrid with Control plane in Cloud. Offers two flavors for runtimes. Classic Mule runtimes and Container based Runtime Fabric

Private Cloud

Customer's own Anypoint private cloud deployed in their datacenter

Observed Component	Metric/log	Applicable to this deployment option?		
		Cloudhub	Hybrid	Onprem(PCE)
API & Application	Application logs	✅	✅	🟧
	API Analytics	✅	✅	🟧
	Policy violations	✅	✅	🟧
	Tracing	✅	✅	🟧
	Application lifecycle	✅	✅	🟧
	Application health	✅	✅	🟧
	Response time	✅	✅	🟧
	Flow monitoring	✅	✅	✅
Runtime	Memory	✅	✅	🟧
	CPU	✅	✅	🟧
	Disk space	NA	✅	🟧
	Agent logs	NA	✅	🟧
	Runtime logs	✅	✅	🟧
	OS logs	NA	🟧	🟧
Platform Control Plane	Audit logs	✅	✅	✅
	Core usage	✅	NA	NA
	POD/Container logs	NA	NA	✅
	Disk space	NA	NA	🟧

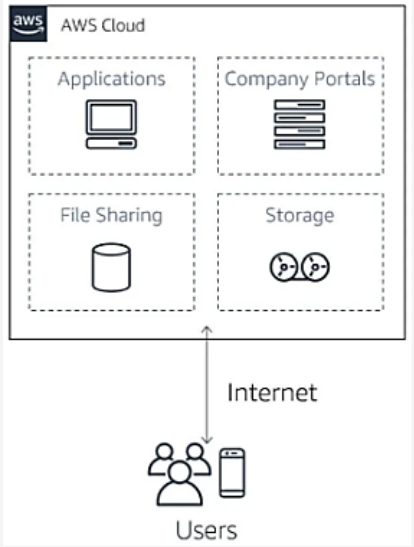
✅ Feature applicable for this deployment model and is offered OOTB by Mulesoft

🟧 Feature applicable for this deployment model but may need a 3rd party tool for easy access

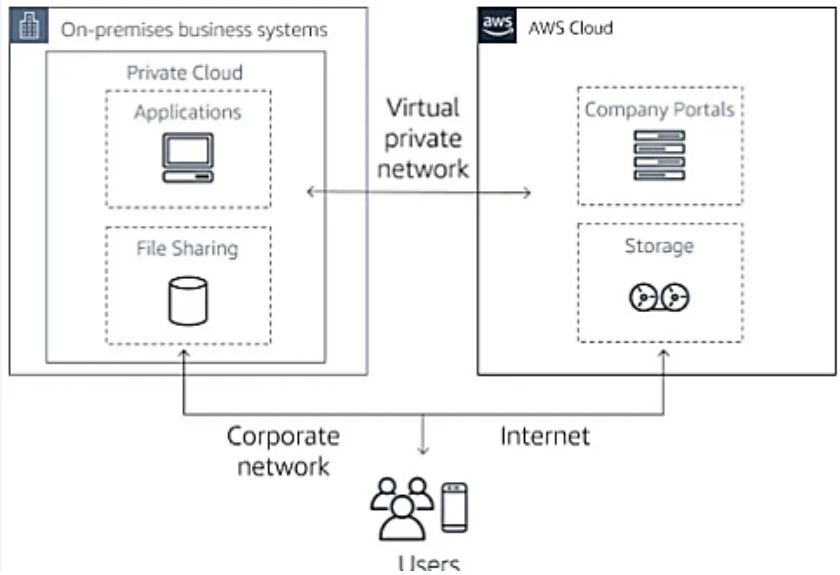
NA Feature not applicable for this deployment model

4.4. Understand the Different Cloud Computing Deployment Models

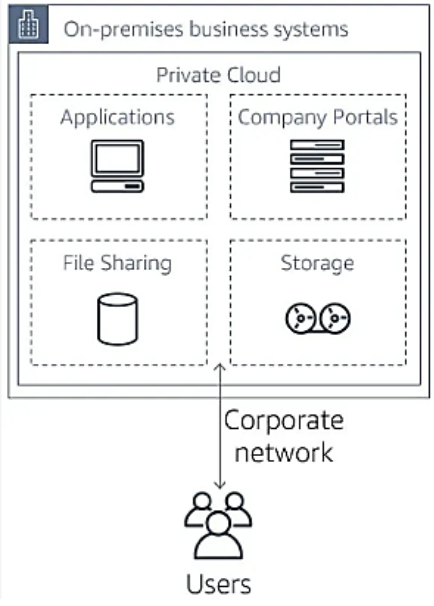
1. Cloud



2. Hybrid

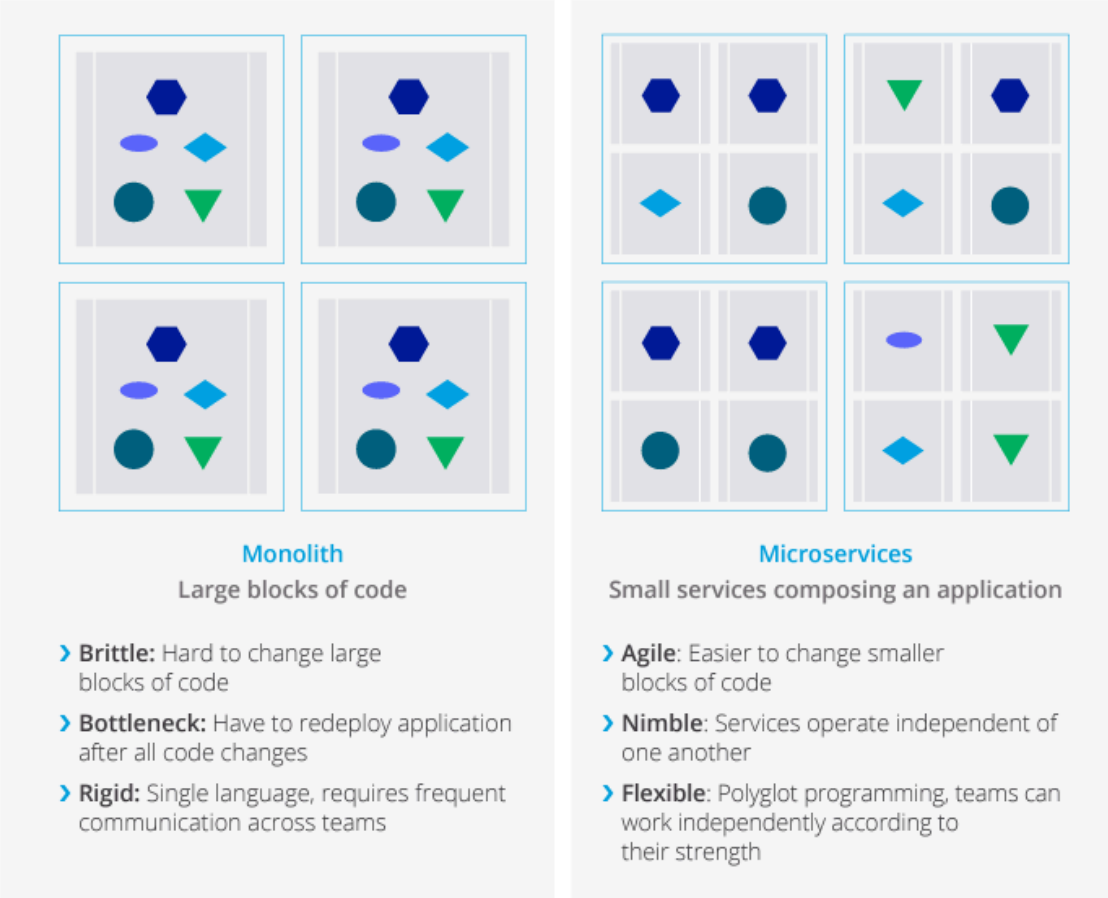


3. On-premises or private cloud



4.5. Service mesh together with API management

- Service Mesh is an architectural pattern for microservices deployments
- Solves security and governance challenges



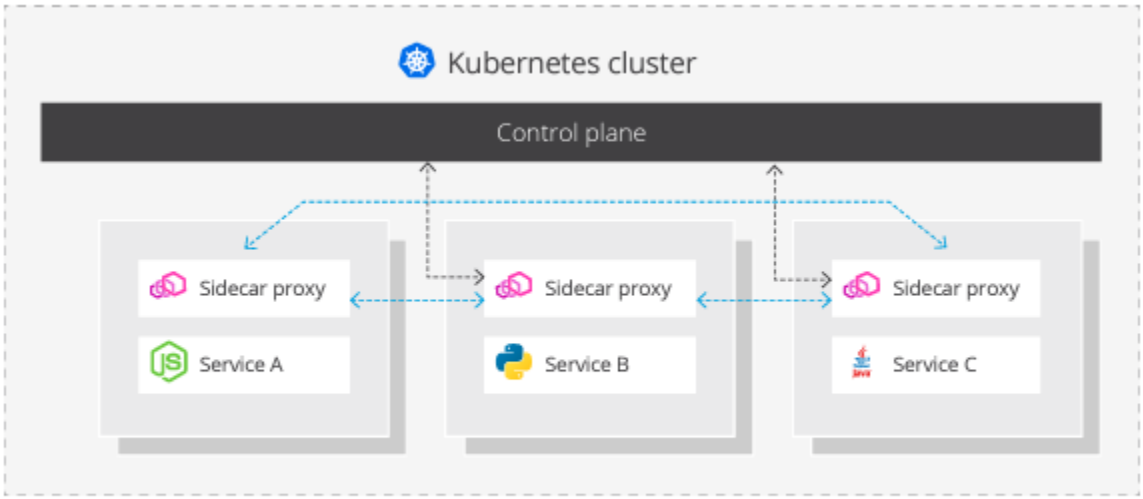
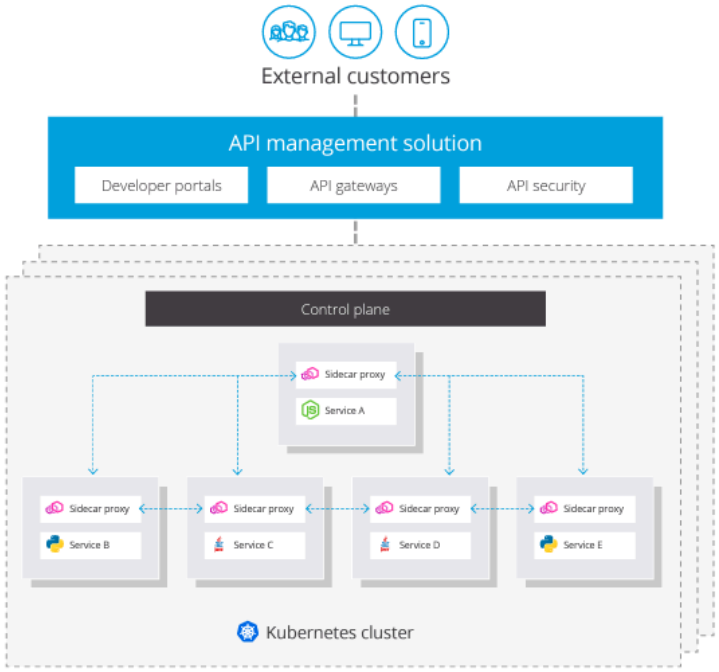


Figure 3: The sidecar pattern for microservices deployments.

- Common policies: circuit breakers, timeout implementation, load balancing, service discovery, and security (transport layer security and mutual authentication)



What role does API management play?

- **API gateways** protecting north-south traffic out of the Kubernetes cluster
- **API security** applied on the gateway to identify malicious clients and payloads
- **Developer portals** to make services reusable to accelerate future projects

The north star

Discover, manage, and secure any service in a single, unified platform

Figure 4: The role of API management.

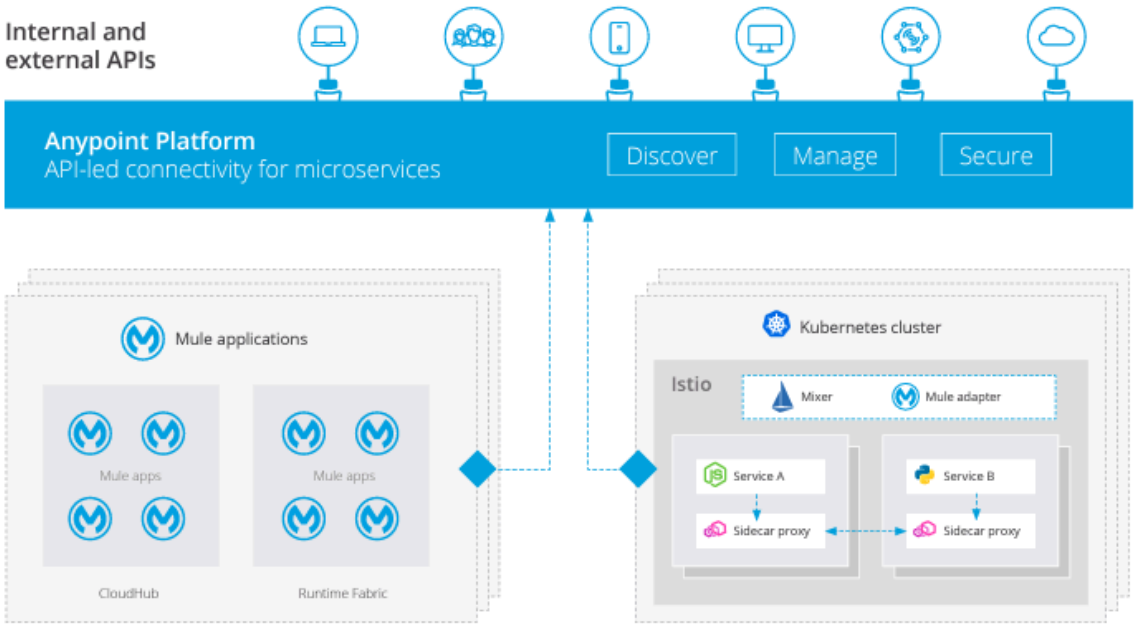


Figure 9: Extend the benefits of an application network with Anypoint Service Mesh.

Describe the components and benefits of Anypoint Platform for system integration
• Identify the primary components of Anypoint Platform and their benefits for system integration • Identify and describe the common characteristics of popular Anypoint Connectors for connecting to software applications, databases, and protocols • Identify the components and describe the benefits of the Anypoint Platform runtime planes and control planes • Describe the MuleSoft-hosted and customer-hosted deployment options for Anypoint Platform • Describe the uses and benefits of the Anypoint Platform development tools and languages for integration developers and DevOps teams • Describe and classify the types of reusable assets in Anypoint Exchange that form the building blocks of integration solutions

5. Describe the components and benefits of Anypoint Platform for system integration

5.1. API Lifecycle Management with Anypoint Platform

5.1.1. Explore the Application Network

- API-led connectivity
  - Clear contract between systems
  - Reusability
  - Discoverability
  - Visibility and security
  - Availability and resiliency

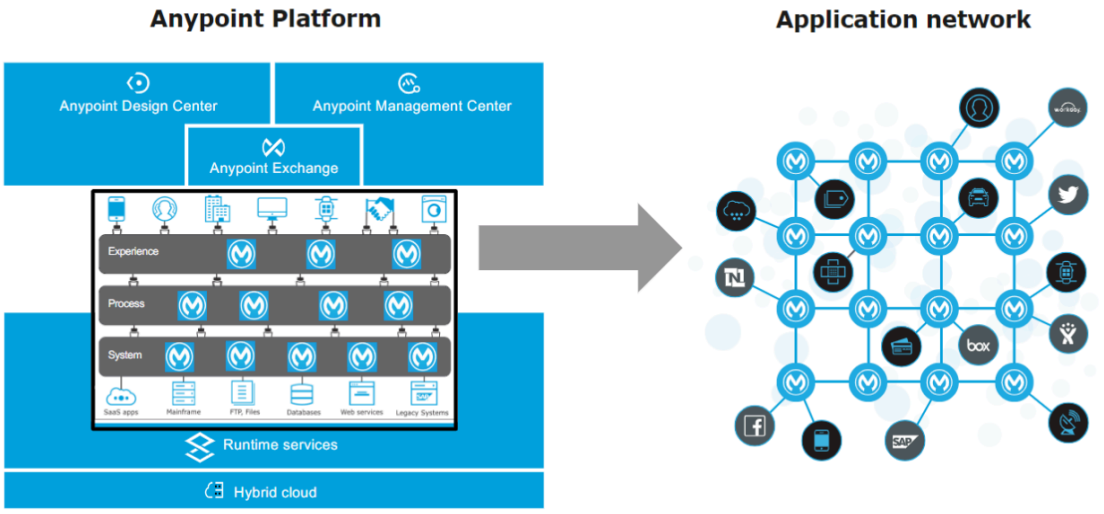
API-led Connectivity	A methodical way to connect applications, data, and devices through reusable and purposeful APIs; the opposite of point-to-point integration.
Application Network	A network of applications, data, and devices connected by reusable APIs, each built with the principles of API-led connectivity.
Anypoint Platform	MuleSoft's platform that provides many tools to design, build, deploy, and operate an application network.
Integration Trailblazer	A person within the ranks of the company who champions the idea of API-led connectivity as an opportunity to revolutionize how business is done (and lead in the new, digital economy).
Citizen Integrators	Typically a line of business manager, Salesforce administrator, or other non-technical user enabled by user friendly citizen integration tools, like MuleSoft Composer, to build new innovative integrations and customer experiences with clicks instead of code.

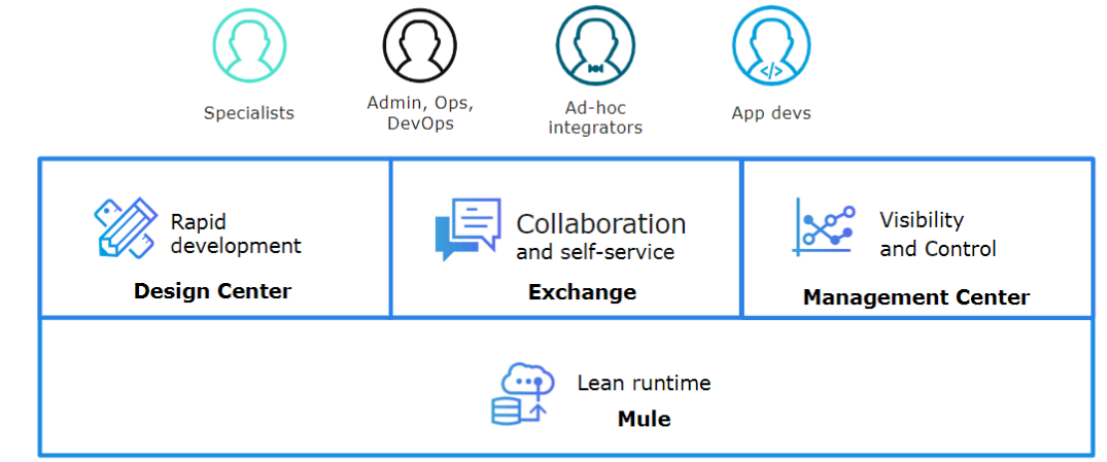
5.1.2. Integrate Your Systems

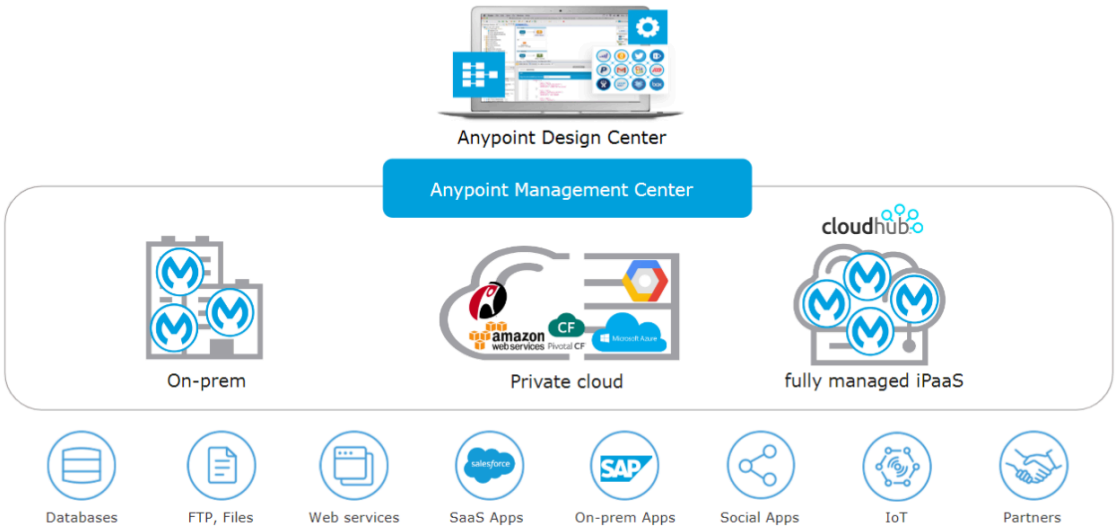
Tightly Coupled Integration	**API-Led Connectivity**
Design for short-term needs	Design for future flexibility
Point-to-point integrations	3-layered API architecture
Scale by repetitive effort	Scale by reuse
Spaghetti code	Application network

5.2. Getting Started with Anypoint Platform - Module 2

5.2.1. Introducing Anypoint Platform

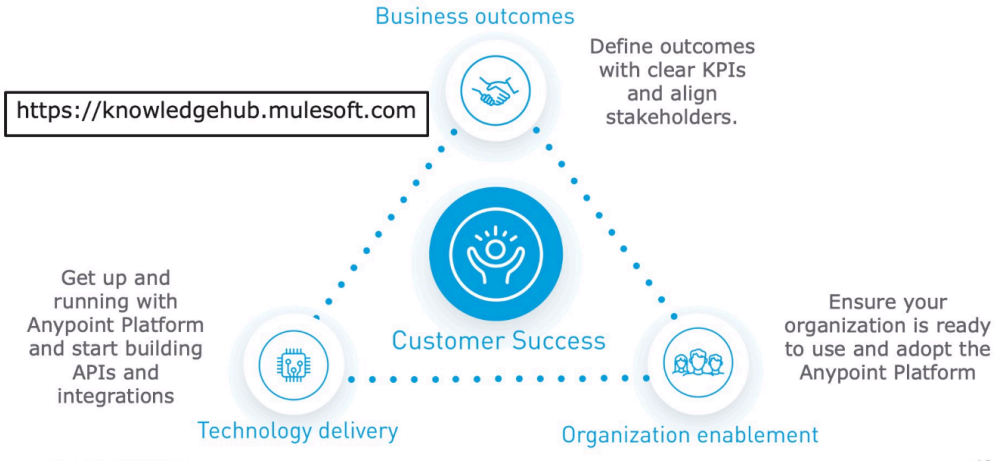




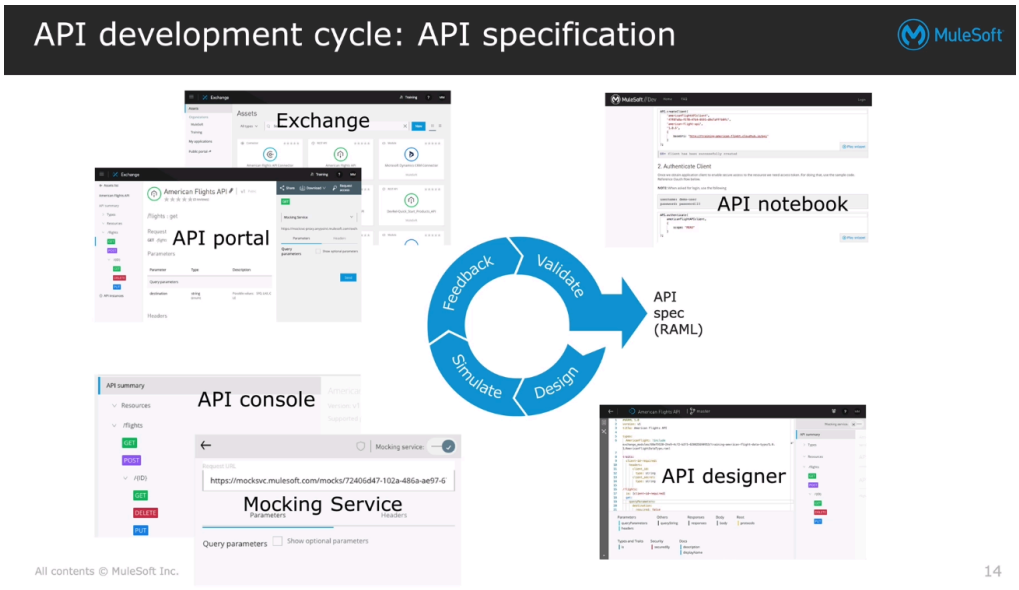


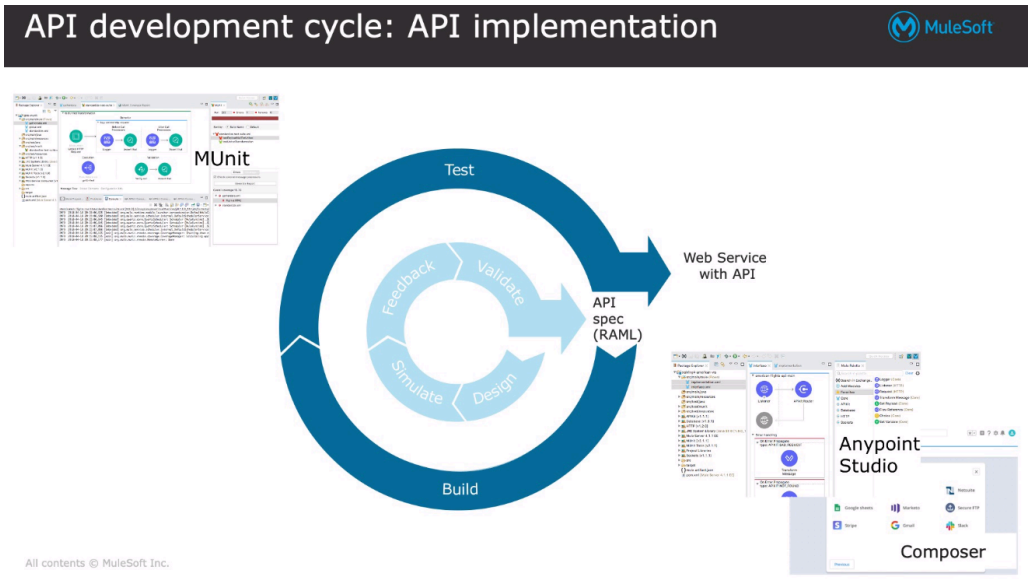
5.2.2. Achieving success with Anypoint Platform

MuelSoft Catalyst

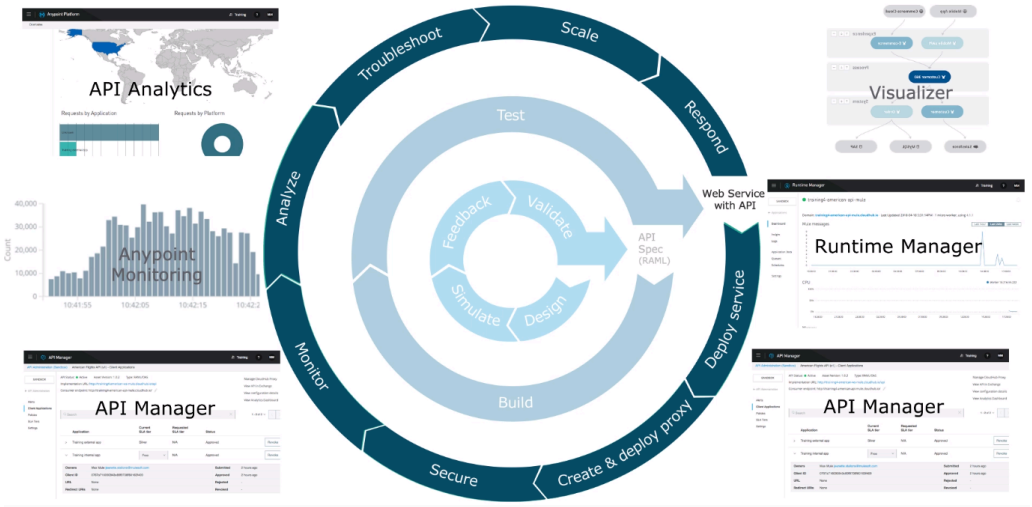


5.2.3.Introducing the components of Anypoint Platform

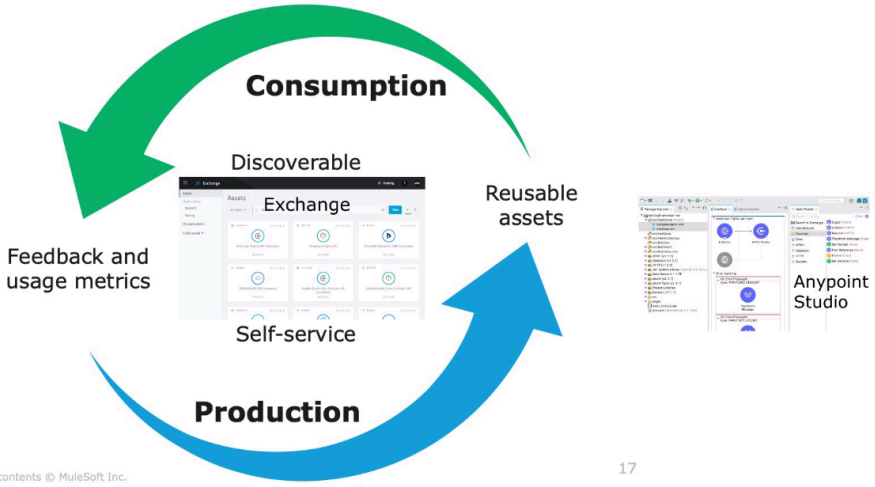




API development cycle: API management

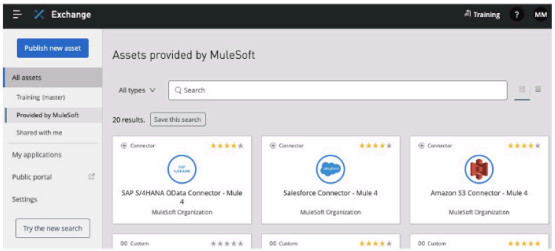


API lifecycle: Discovery and consumption



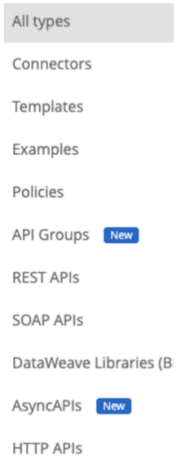
Anypoint Exchange

- A library of assets
- The central repository that is critical to the success of building an application network
- Provides a place where assets can be published for discovery and reuse
- Discovery is supported by strong search capabilities



What does (and should) Exchange contain?

- MuleSoft-provided **public** assets available in all accounts to all users
  - You can work with MuleSoft to get APIs and connectors certified and added
- **Private** content only available to people in your org
  - Assets added by anyone in your org are added to your private Exchange
- Your organization should populate it to contain everything you need to build your integration projects
  - Including APIs, connectors, diagrams, videos, links, and more

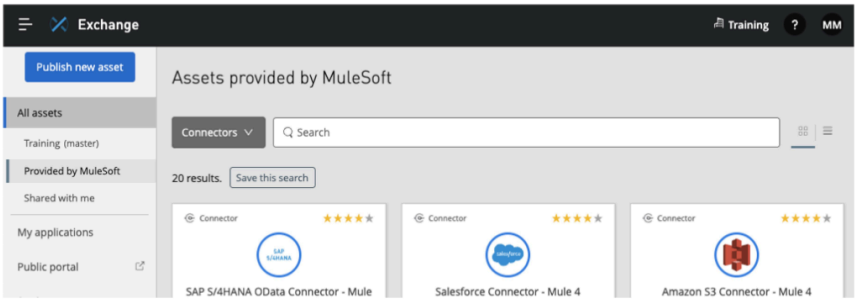


REST APIs and API portals in Anypoint Exchange

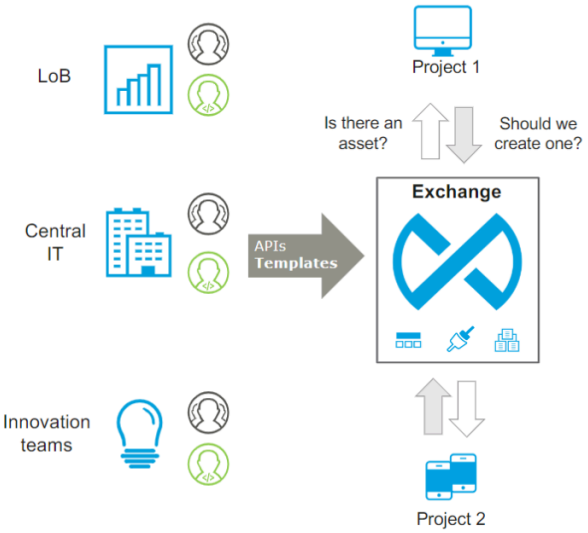
- When a REST API is added to Exchange, an **API portal** is automatically created for it
- An API portal has
  - Auto-generated **API documentation**
  - An **API console** for consuming and testing APIs
  - An **automatically generated API endpoint** that uses a **mocking service** to allow the API to be tested without having to implement it
- API portals can be shared with both internal and external users
- In the last module, you used a public API portal created from Anypoint Exchange for a private organization (Muletraining)

REST connectors in Anypoint Exchange

- When a RAML 1.0 API specification is added to Exchange, a **connector** is automatically created for it
  - The connector can be used in Mule applications to make calls to that API
  - REST Connect is the name of the technology that performs this conversion



Using Exchange: Success of C4E in action

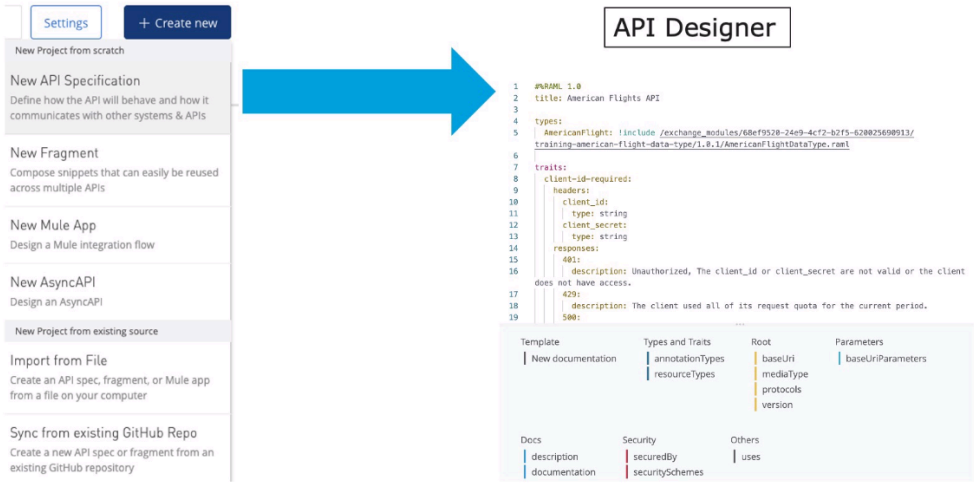


5.2.4. Building integration applications and APIs with Design Center

Design Center applications

Application	Purpose	In this course	Additional courses
API Designer	Web app for designing, documenting, and mocking APIs	Module 3	• Anypoint Platform: API Design
Anypoint Studio	Desktop IDE for implementing APIs and building integration applications	Module 4	• Anypoint Platform Development: Fundamentals • Production-Ready Development Practices • Production-Ready Integrations

Creating API specifications with API Designer



Creating Mule applications with Anypoint Studio

- **Mule applications** can be created using Anypoint Studio
  - Studio creates XML code visually by adding components and processors to flows
    - The XML can also be manually edited or created
  - You can use connectors and other assets from Exchange
  - Other tools can be used to write code (primarily XML) to create applications

- Under the hood, Mule applications are Java applications using Spring

Mule is the runtime engine of Anypoint Platform

MuleSoft

- **A lightweight integration and automation platform** that allows developers to connect apps together quickly and easily, enabling them to exchange data
  - Acts as a transit system for carrying data between apps (the Mule)
  - Can connect all systems including web services, JMS, JDBC, HTTP, & more
- **Decouples point-to-point integrations** by having all (non-Mule) apps talk to a Mule runtime instead of directly to each other
- **Enforces policies for API governance**
- **Can be deployed anywhere**, can integrate and orchestrate events in real time or in batch, and has universal connectivity


On-premises & Private Cloud


Hybrid


Hosted by MuleSoft (CloudHub)

Mule applications are designed to run on Mule

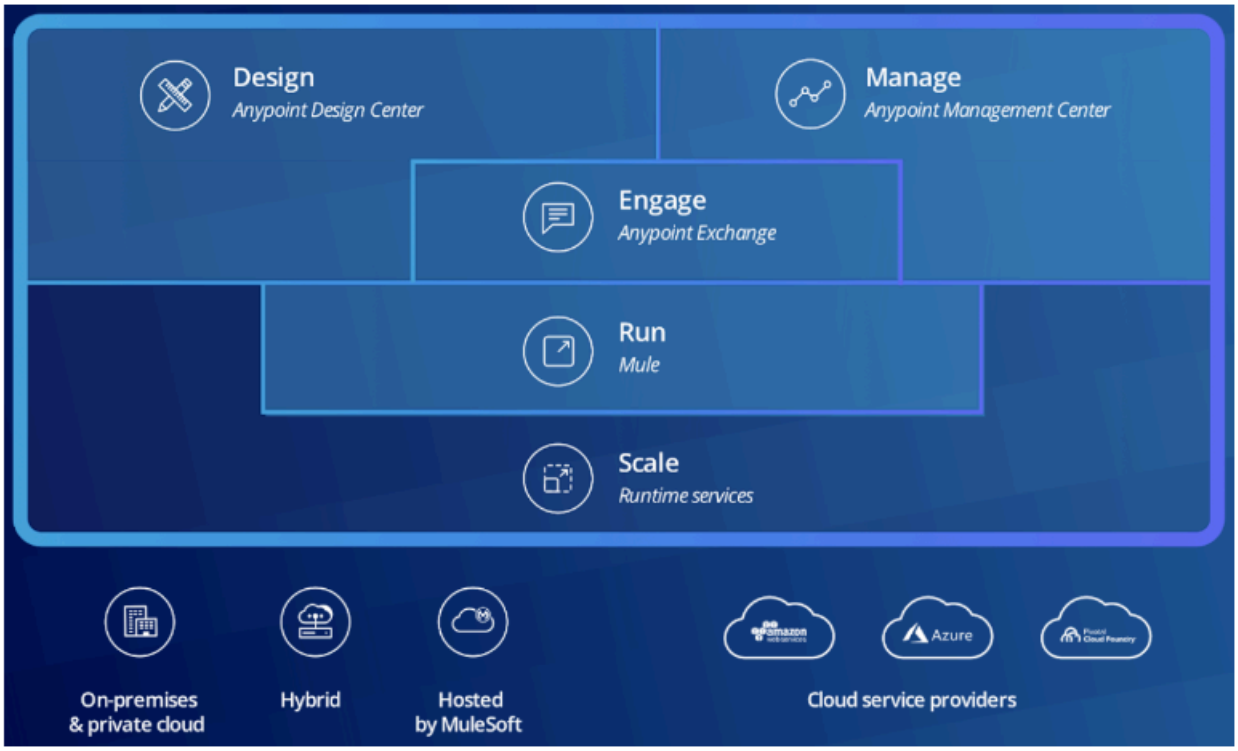
MuleSoft

- **Mule applications** are created by integration developers to tie together various subsystems
- These applications are deployed to a **Mule Runtime** which enables them to consume inbound data in a predefined Mule message format
- The applications transform and route Mule messages in paths called flows and in stages called components or processors
- Mule finally delivers the transformed data to a recipient or destination
- Mule applications can be deployed to anywhere a Mule Runtime is hosted
  - Mule runtimes can be MuleSoft-hosted in the cloud (CloudHub), private-hosted in the cloud, on-premise, or a hybrid

5.3. Introduction to Anypoint Connectors

- Inbound endpoint, Message processor, Outbound endpoint
- Software Applications: AP, ERP, CRM, etc
- Databases: HDFS, MongoDB, etc
- Protocols: LDAP, WebSocket, etc
- Support Categories: Select connectors, Premium connectors, Mulesoft certified connectors

5.4. Deployment options of Anypoint Connectors



Plane	CloudHub	Hybrid	Runtime Fabric (RTF)	Private Cloud Edition (PCE)	Pivotal Cloud Foundry (PCF)
Control	MuleSoft Cloud	MuleSoft Cloud	MuleSoft Cloud	On-premise or Private Cloud	On-premise or Private Cloud
Runtime	MuleSoft Cloud	On-premise or Private Cloud	On-premise or Private Cloud (K8s or Bare metal)	On-premise or Private Cloud	On-premise or Private Cloud

- On-premise or Private Cloud
  - Single-tenant
  - PCE - On-premise or private cloud (AWS)
- MuleSoft cloud:
  - Multi-tenant
  - US or EU region cloud
  - Government cloud

5.5. APIs and DevOps

Lifecycle stage	Stage activities	Anypoint Platform components
Design	• Create an API specification • Define data types • Add security patterns	• API designer • Anypoint Exchange
Simulate	• Describe examples Author error messages	• Mocking Service NOTE: With a mock service, frontend developers can begin crafting the user experience before the API is implemented, allowing parallel development and reducing time to market
Collect feedback	• Create on-ramp • Publish portal	• Developer Portal • Anypoint Exchange
Validate	• Create runnable tests • Orchestrate API calls	• Anypoint Studio • Anypoint Exchange
Build	• Import API spec • Manage dependencies • Compose with connectors and templates • Static config analysis • Link to version control system	• Anypoint Studio • Anypoint Exchange
Test	Author unit and functional tests Define pass/fail criteria	• MUnit • Anypoint Exchange
• **Deploy** • **Secure** • **Promote**	• Deploy artifacts • Promote across environments • Apply policies • Control access	• Anypoint CLI • Anypoint Platform • Runtime manager • API manager
• **Operate** • **Secure** • **Monitor** • **Manage** • **Analyze**	• Measure SLAs, monitor utilization, adjust resources • Track performance against KPIs	• Runtime manager • Management agent • Anypoint Analytics

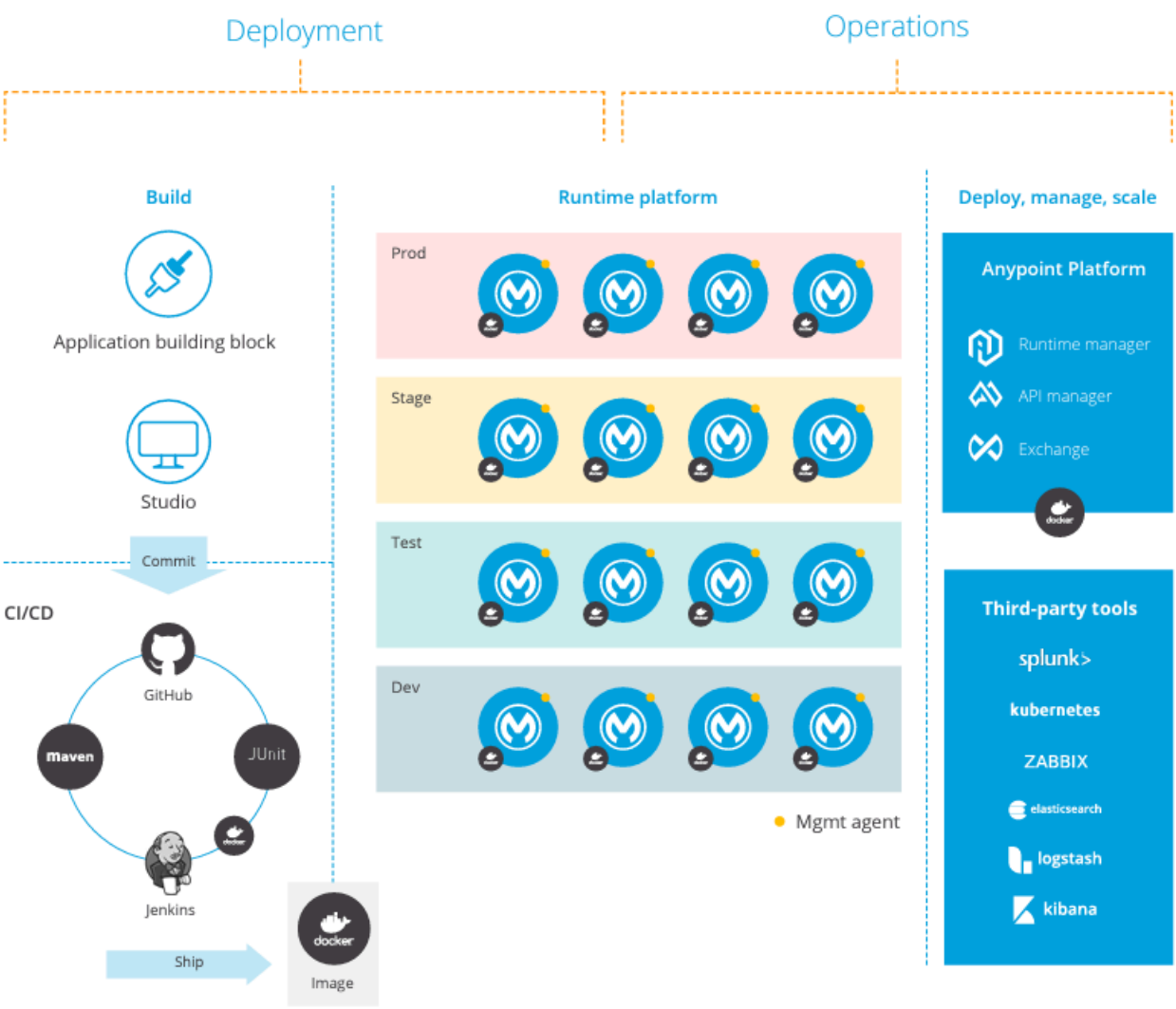


Figure 3: Operating Anypoint Platform

5.6. Anypoint Exchange Asset Types

API Groups	A set of APIs bundled into a single asset.
API Spec Fragments	A part of an API specification that is reused to build a complete API specification.
AsyncAPI Specifications	An AsyncAPI specification file that specifies an event-driven API.
Connectors	Packaged connectivity to an Anypoint Platform endpoint using third-party APIs and standard integration protocols. Use connectors within your application's flows to send and receive data using a protocol or specific API. Anypoint Studio comes with many bundled connectors, and Exchange has many more.
Custom	A description and an optional file to explain aspects of your system, to provide instructional videos, or to describe product or organizational documentation.
DataWeave Libraries	Packaged modules and mappings to share and reuse across applications.
Examples	Applications that are ready to run in Anypoint Studio and demonstrate a use case or solution.
GraphQL API	A schema definition that contains the object types and definitions that are used to interact with the API through the GraphQL specification. GraphQL enables you to query an API that supports this language in a much more flexible way than REST.
HTTP APIs	A placeholder for an endpoint for use by private Exchange users who want to manage the endpoint with API Manager. Instead of requesting access to multiple APIs to satisfy a use case, a developer can access the group in one step.
Policies	Configuration modules to extend the functionality of an API and enforce capabilities such as security.
REST APIs	RAML or OAS files that specify APIs referenced by an HTTP Request connector to expose metadata to Anypoint Studio.
RPA Templates	RPA Activity Templates Activity templates are reusable code within RPA to standardize the most used activity steps and make them available to other developers in the organization. File type is `.calw`. RPA Process templates Templates for RPA processes that can be imported into RPA Builder as projects and are customizable before deploying to RPA bots. Process templates are built on common patterns for RPA to build automation faster. File type is `.crpa`.
Ruleset	YAML files that describe a ruleset.
SOAP APIs	A WSDL file that specifies an API.
Templates	Packaged integration patterns built on best practices to address common use cases. Complete the template's use case or solution by supplying your own information, customizing or extending the templates as needed.

Describe the components and benefits of Anypoint Platform for API management
• Identify the primary components of Anypoint Platform and their benefits for API management • Identify how MuleSoft products realize the goals of full lifecycle API development and Universal API Management (UAPIM) • Explain the advantages of API-led connectivity with Anypoint Platform over other integration and API management approaches

6. Describe the components and benefits of Anypoint Platform for API management

6.1. Introducing universal API management on Anypoint Platform

- **API Designer**: provides a visual or code-based guided experience for designing, documenting, and testing APIs in any language (RAML, OAS, AsyncAPI)
- **Anypoint CLI**: Command line interface
- **Flex Gateway**: ultrafast, designed to manage and secure APIs running anywhere
- **API Manager**: manage, govern, and secure APIs
- **API Governance**: maintain standard quality and security while developers want to avoid overhead caused by conformance review cycles
- **Anypoint Exchange**: marketplace of reusable, pre-built assets
- **API Community Manager**: create and nurture a community of developers and partners to foster adoption of API products
- **API Experience Hub**: create a consolidated source of truth and a digital storefront for all your enterprise APIs fast with out of the box templates
- **Anypoint DataGraph**: reuse multiple APIs in a single request and serve data from all your APIs to developers instantly (GraphQL)

7. Ancillary

7.1. Designing API

Approaches to API design

Hand coding

API Blueprint

apiary

OpenAPI Spec

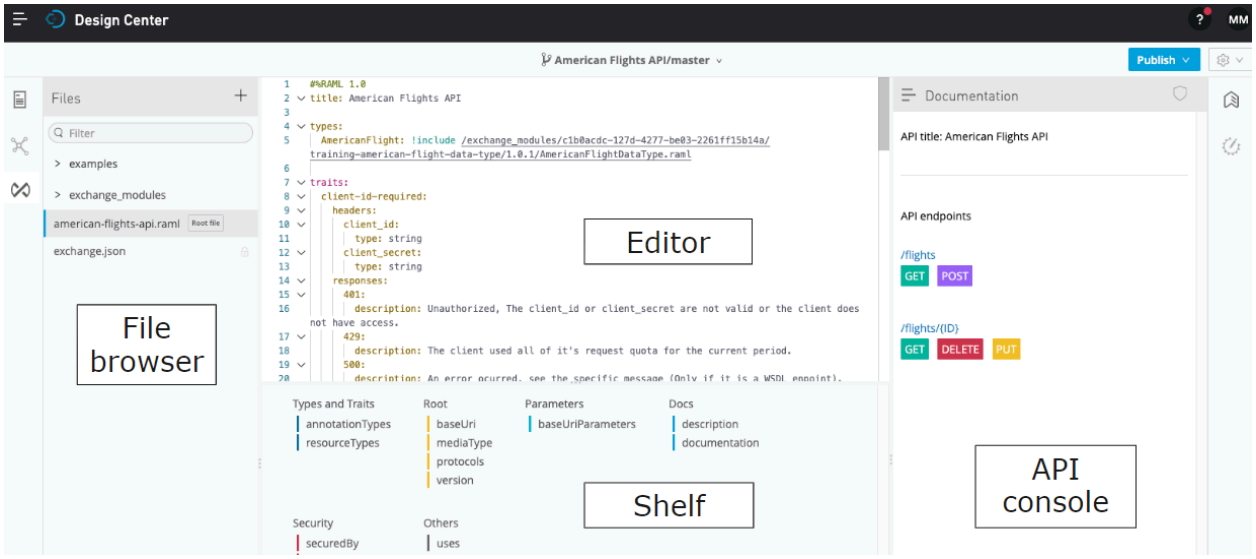
OPEN {API} INITIATIVE

AsyncAPI Spec

AsyncAPI

RAML

RAML



Modularizing APIs

MuleSoft

- Instead of including all code in one RAML file, you can modularize it and compose it of reusable fragments
  - **Data types**, **examples**, traits, resource types, overlays, extensions, security schemes, documentation, annotations, and libraries
- Fragments can be stored
  - In different files and folders within a project
  - In a separate API fragment project in Design Center
  - In a separate RAML fragment in Exchange

API Exchange:

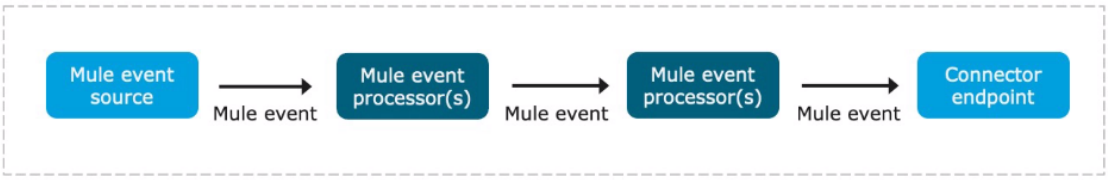
- Public: External developers
- Private: Internal developers

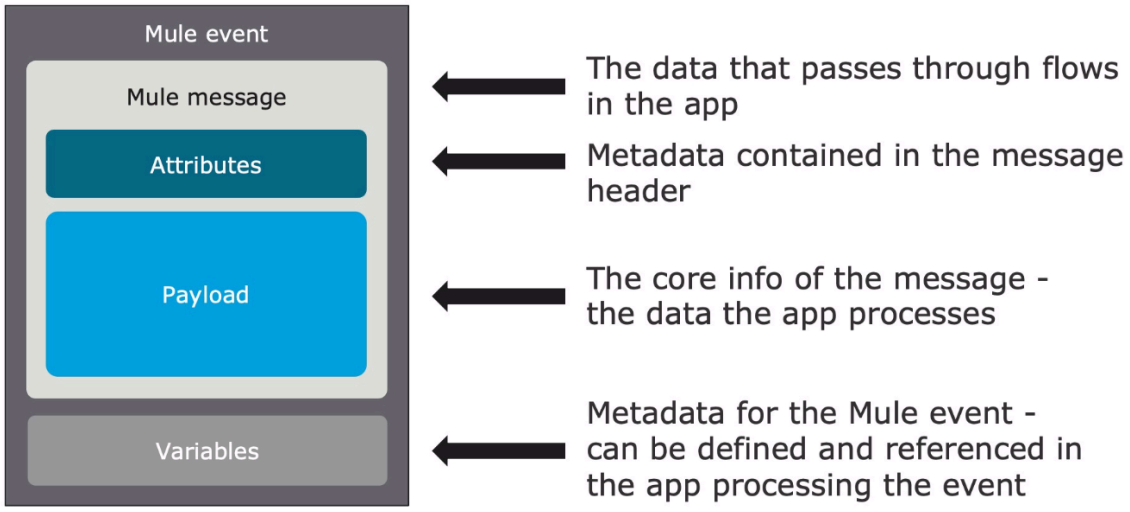
Note: API Designer API Exchange (Versions: Stable, Development, Deprecated)

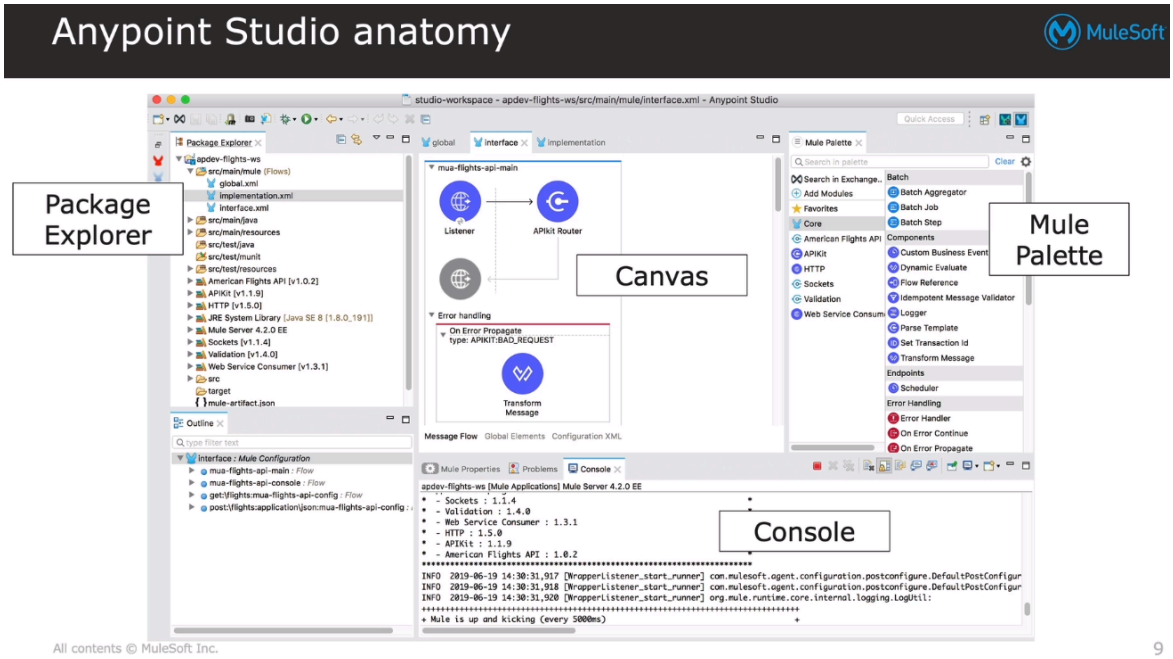
API Community/Developer/Public Portal

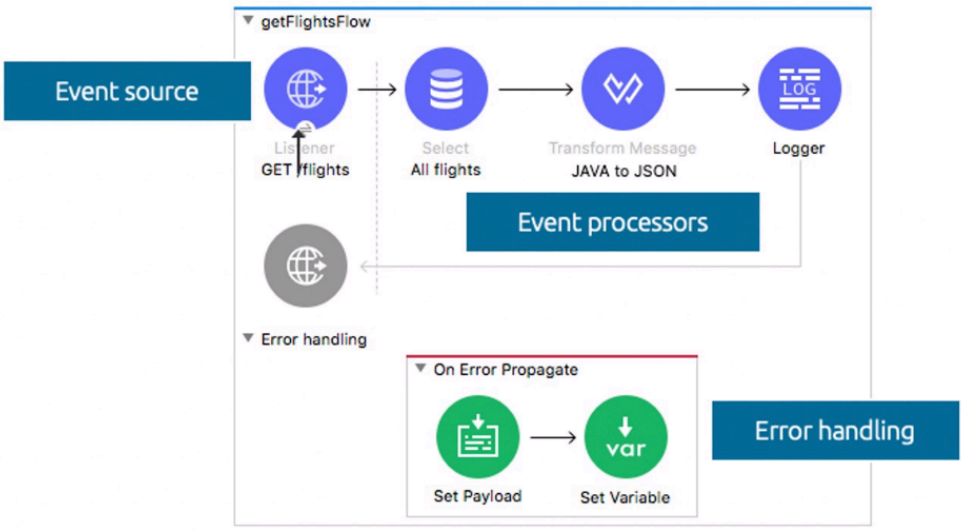
7.2. Building API

Flow





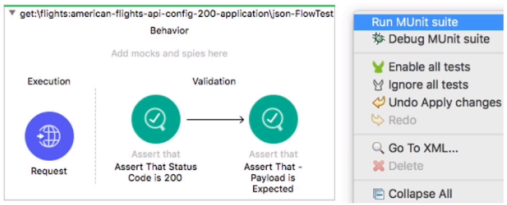




Automating testing of applications

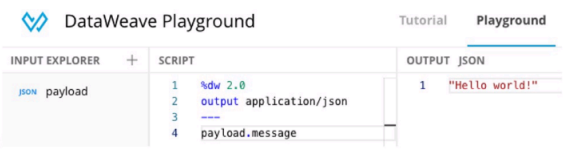
MuleSoft

- You can automate testing of Mule applications using MUnit
- MUnit is a Mule app testing framework for building automated tests
- MUnit is fully integrated with Anypoint Studio
  - You can create, design, and run MUnit tests and suites of tests just like you do Mule applications



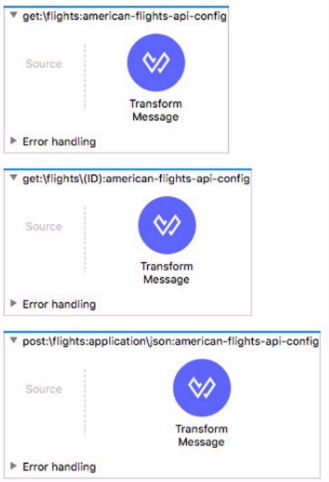
- MUnit is covered in *Anypoint Platform Development: Production-Ready Development Practices*

- **DataWeave 2.0** is the expression language for Mule to access, query, and transform **Mule 4** event data
- DataWeave is fully integrated with Studio
  - Graphical interface with payload-aware development
  - In Studio, the **Transform Message** component is used for transformations
- **DataWeave Playground**
  - Interactive browser environment for mocking
  - developer.mulesoft.com/learn/dataweave



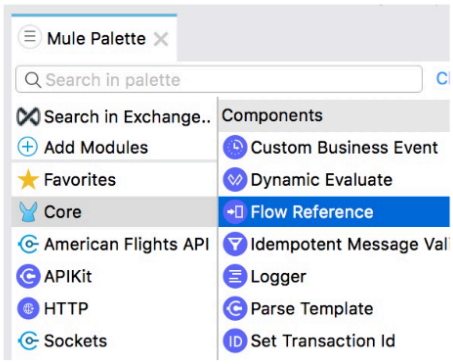
Creating RESTful interfaces automatically using APIkit

- **APIkit** is an open-source toolkit that includes an Anypoint Studio plugin
- The Anypoint Studio **APIkit plugin** can generate an interface automatically from a RAML API definition
  - For new or existing projects
  - Can also work with OAS
- It generates a main routing flow and flows for each of the API resource / method pairs
- You add processors to the resource flows to hook up to your backend logic



Passing messages to other flows

- Flows can be broken into multiple flows
  - Makes the graphical view more intuitive and the XML code easier to read
  - Promotes code reuse
  - Easier to test with MUnit
- All flows are identified by name and can be called via **Flow Reference** components in other flows
- Studio can list all flow references to a flow or subflow
  - Also provides navigation

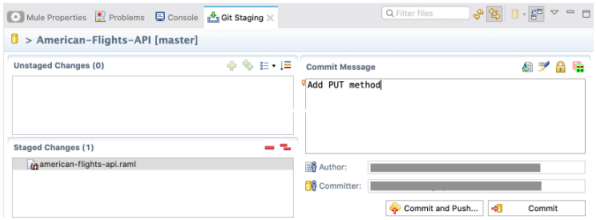


Synchronizing API specifications

- **API Sync** feature of Anypoint Studio enables you to
  - Pull specifications from Design Center into Studio
    - You already did this!
    - You can also initiate the creation of API specifications from scratch in Studio
  - Edit the specification offline in Anypoint Studio
  - Push the updates back to Design Center
  - Publish the new API asset version to Exchange
- This lets you develop Mule applications while following API lifecycle development practices from within Anypoint Studio
- If an API specification changes in Exchange, the generated API interface in Anypoint Studio can be updated
  - Flows that have already been modified are not overwritten

Walkthrough 4-7: Synchronize changes to an API specification between Studio and Anypoint Platform

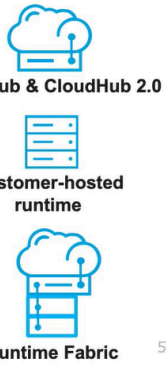
- Create an editable version of an API specification in Anypoint Studio
- Make changes to an API specification in Anypoint Studio
- Push the changes from Anypoint Studio to Design Center
- Publish the modified API specification from Studio to Exchange
- Update the version of an API specification used in a Mule project
- Rescaffold an API interface from an updated API specification



Deploying and Managing API:

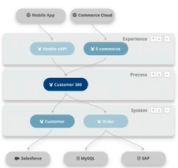
Deploying applications

- During development, applications are deployed to an embedded Mule runtime in Anypoint Studio
- For everything else (testing, Q&A, and production), applications can be deployed to
  - **CloudHub & CloudHub 2.0**
    - Platform as a Service (PaaS) component of Anypoint Platform
    - MuleSoft-hosted Mule runtimes on AWS
    - A fully-managed, multi-tenanted, globally available, secure and highly available cloud platform for integrations and APIs
  - **Customer-hosted Mule runtimes**
    - On bare metal or cloud service providers: AWS, Azure, and Pivotal Cloud Foundry
  - **Anypoint Runtime Fabric**
    - Customer-hosted container service of the runtime plane

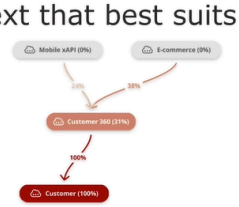


Viewing Deployed Applications with Visualizer

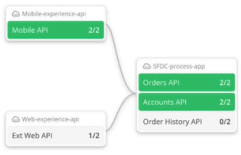
- Visualizer provides a real-time view into your application architecture in a context that best suits your role



Architecture



Troubleshooting

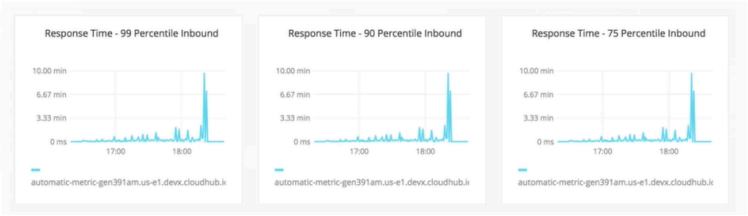


Policies

- Organizes APIs and applications into relational diagrams
  - Promotes best practices and layer-based architectural consistency
  - Pinpoints issues rapidly through root cause analysis
  - Enables visibility into the policy compliance of APIs
- Diagram data is secure and automatically & dynamically updated

Understanding the State of Your Infrastructure with Anypoint Monitoring

- Anypoint Monitoring provides visibility into integrations across your app network



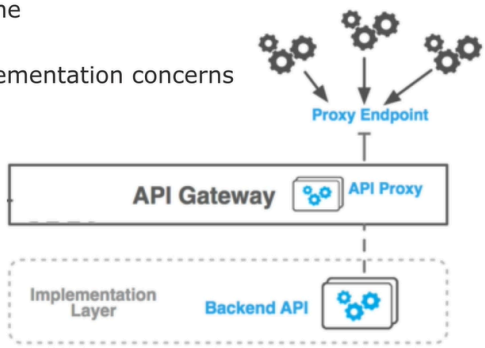
- Its monitoring tools are designed to reduce the time to identify and resolve issues by providing ways to
  - Aggregate and map metrics across dependent systems in real-time
  - Configure dashboards and alerts to reduce issue identification time
  - Store and search log data at scale

Region > Availability Zones > Workers

Scaling: Horizontal vs Vertical

Restricting access to APIs

- An **API proxy** is an application that controls access to a web service, restricting access and usage through the use of an API gateway
- The **API Gateway** is a runtime designed and optimized to host an API or to open a connection to an API deployed to another runtime
  - Included as part of the Mule runtime
    - Separate licenses required
  - Separates orchestration from implementation concerns

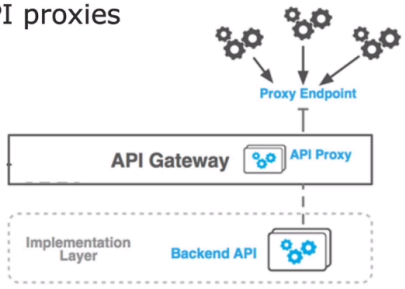


The API Gateway is the point of control

- **Determines which traffic** is authorized to pass through the API to backend services
- **Meters the traffic** flowing through
- **Logs** all transactions, collecting and tracking analytics data
- Applies runtime policies to **enforce governance** like rate limiting, throttling, and caching

Restricting access to APIs

- Use **API Manager** to manage access to API proxies
  - Define SLA tiers
  - Apply runtime policies
- The **API Gateway** enforces the policies
- **API Autodiscovery** is a mechanism that enables a deployed Mule application to
  - Download policies from API Manager
  - Act as its own proxy



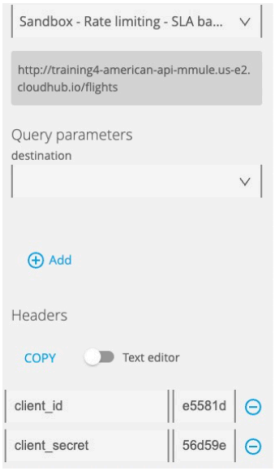
Applying policies to restrict access

- There are **out-of-the box** policies for many common use cases
  - Rate limiting
  - Spike control
  - Security
- You can define **custom** policies (using XML and YAML)
- You can apply **multiple** policies and set the order
- You can define **automated** policies to comply with common requirements
  - Requires a MuleSoft-hosted control plane

Client ID enforcement	JSON threat protection
Cross-Origin resource sharing	Basic Authentication - LDAP
OAuth 2.0 access token enforcement	Message Logging
Header Injection	Rate limiting
Header Removal	Rate limiting - SLA based
Basic authentication - Simple	Spike Control
IP blacklist	XML threat protection
IP whitelist	

Enforcing access to APIs using SLA tiers

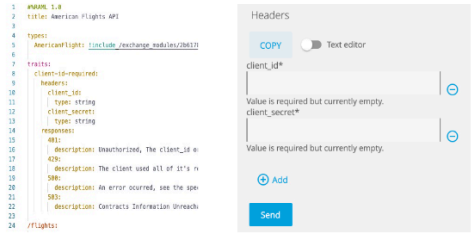
- To enforce, apply an **SLA based** rate limiting policy
- SLA based policies require all applications that consume the API to
  - **Register** for access to a specific tier
    - From an API portal in private or public Exchange
  - **Pass their client credentials** in calls made to the API



Walkthrough 5-5: (Optional) Add client ID enforcement to an API specification

- Modify an API specification to require client id and client secret headers with requests
- Update a managed API to use a new version of an API specification
- Call a governed API with client credentials from API portals

Note: If you do not complete this exercise for Fundamentals, the REST connector that is created for the API and that you use later in the course will not have client_id authentication



7.3. iPass

iPaaS is a platform for building and deploying integrations within the cloud and between the cloud and the enterprise. Enterprise iPaaS solutions are the next generation of cloud applications that enable connectivity with other cloud-based software, SaaS, on-premises, and legacy applications. With iPaaS, users can develop and deploy integration flows between these systems without installing or managing any hardware or middleware.

7.4. API Management

Designing, publishing, documenting, analyzing, governing, and securing.

7.5. Spec-Driven Development

RAML

- Create expectations: When working with developers, it is critical to inform them of what is expected, when it should be delivered, and what pain points are to be solved by the API functionality.
- Service messaging: With the goal of creating new products and services, or transforming existing products and services, it is paramount to make sure those services and the APIs that provide access to them align with business goals and lead to services that deliver value.
- Case studies: It's key to back up assumptions with viable case studies that illustrate the improvements that API adoption brings to the table.
- Documentation and support: Make sure the proper tools are in place for the dev team to document their progress, as well as address change management, along with exposing the capabilities of an API. Ensure that support for the both the development and implementation team is readily available.
- SDKs and libraries: Provide the necessary resources to the development team to speed services development and implementation by offering resources that contain reusable code and processes.

7.6. Youtube Material

Cloud Computing and Types of clouds

- **Private Cloud** : Dedicated to a single organization and runs on resources that are solely managed and operated and by that organization
- **Public Cloud** : Cloud computing environment that is owned and operated by third party service provider and made available to public over internet. Popular cloud providers include AWS, Microsoft Azure and Google Cloud Platform
- **Hybrid Cloud** : Combines both private and public clouds. Organizations that adopt hybrid cloud strategy, use public cloud for less sensitive workloads and private cloud for more sensitive workloads.
- **Multi Cloud** : Here, organization uses multiple cloud services from different cloud providers. It enables organizations to take advantage of best-of-breed services offered by multiple providers to reduce the dependence on single cloud provider.

Cloud computing models

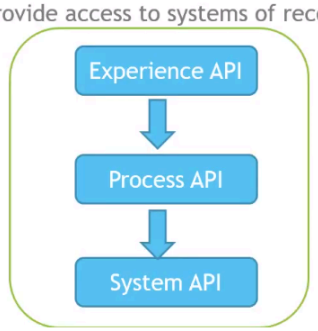
- **IaaS** : Infrastructure as a service is a cloud computing model where the cloud provider offers virtualized computing resources such as servers, storage and networking over internet. Examples of IaaS providers include AWS, Microsoft Azure, GCP.
- **PaaS** : Platform as a service is a cloud computing model where the cloud provider offers a platform for customers to develop, run and manage their own applications without having to worry about the underlying infrastructure. Examples of PaaS providers include Heroku, Google app engine.
- **SaaS** : Software as a service is a cloud computing model where the cloud provider offers a complete software application that customers can access and use over internet. Examples of SaaS providers include Salesforce, Microsoft Office 365.

Introduction to MuleSoft

- **Complete Integration Solution** : MuleSoft is a complete integration platform as a service (iPaaS), providing full range of tools and services for building, deploying and managing integrations between various systems and applications
- **API-First Approach** : MuleSoft uses an API-first approach, which means the creation and management of APIs is at the center of the platform.
- **Wide Range of Connectors** : MuleSoft provides a rich set of pre-built connectors, called Anypoint connectors.
- **Scalable and reliable execution environment** : MuleSoft runtime provides a scalable and reliable execution environment for running integrations and APIs.
- **Comprehensive management and admin tools** : MuleSoft provides a range of management and admin tools for monitoring and managing integrations, as well as features for managing API access and security

API-led Connectivity

- API-led connectivity is the methodology for designing and developing APIs that focus on creating reusable and modular pieces of functionality which are combined to form end to end business capabilities. Below are the key components of API-led connectivity :
- Experience APIs : These are designed to enable the delivery of user-facing experiences. Provide a way for applications to access data and functionality through a simple and consistent interface, abstracting away the underlying systems of record.
- Process APIs : These are used to manage and orchestrate business processes. Provide a way for different systems to communicate and collaborate to accomplish a specific business outcome.
- System APIs : System APIs provide access to systems of record , such as databases and legacy systems.



REST APIs

Set of architectural principles for building web-based APIs. Following are some of the key characteristics of REST APIs:

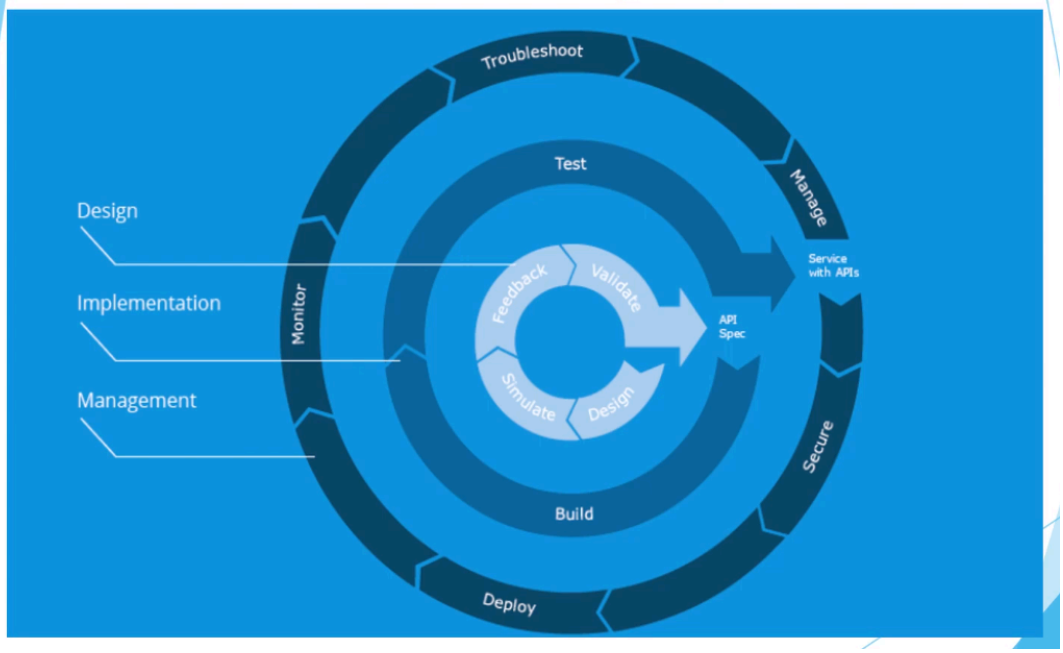
- **Client Server Architecture** : REST APIs follow client-server architecture where client makes requests to server and server returns responses.
- **Statelessness** : REST APIs are stateless, meaning request from client to server contains all information necessary to complete the request.
- **Cacheability** : REST APIs should be designed to be cacheable, meaning that responses from APIs can be stored by intermediate caches to improver performance.
- **Layered System**: REST APIs follow a layered system architecture where different layers of system can be isolated and updated independently.
- **Uniform Interface** : REST APIs have a uniform interface meaning that they use a consistent set of operations (such as GET, POST, PUT and DELETE) to interact with resources.

Common HTTP codes and operations

HTTP Code	Meaning
1xx	Information
2xx	Success/OK
3xx	Redirection
4xx	Client Side Error
5xx	Server Side Error

HTTP Operation	Meaning
GET	Retrieve resource information
POST	Create a new resource
DELETE	Delete the resource
PUT	Update the resource
PATCH	Partial update of resource

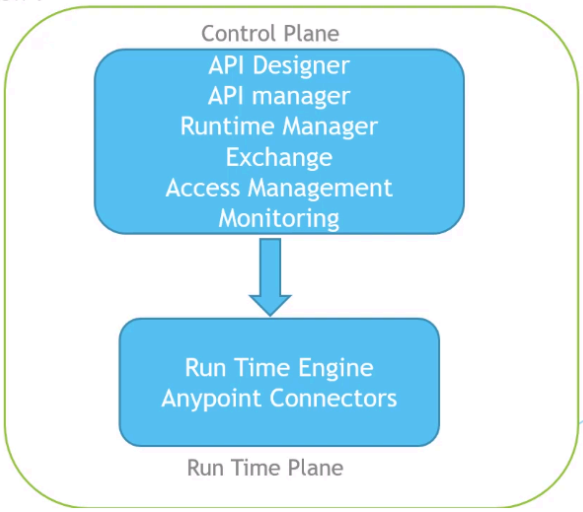
API Life Cycle



MuleSoft Anypoint Platform

MuleSoft's Anypoint platform is a unified platform for API-led connectivity that provides complete solution for designing, building, managing and deploying APIs and integrations. It consists of 2 main components as depicted below :

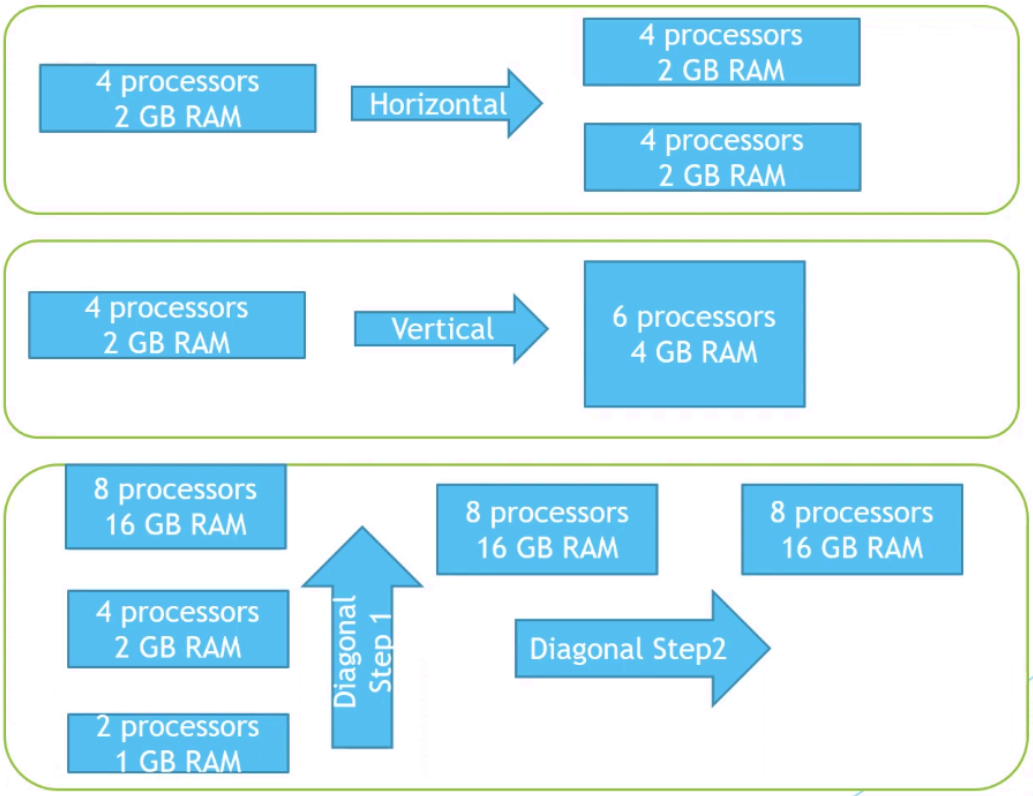
:



MuleSoft Deployment Models

Deployment Model	Cloud Hub	Standalone Mule Runtime	RTF on VMs/Bare Metal	RTF on self-managed Kubernetes	Anypoint Platform Private Cloud Edition
Control Plane Hosted by	MuleSoft	MuleSoft	MuleSoft	MuleSoft	Customer
Run Time Plane Hosted By	MuleSoft	Customer	Customer	Customer	Customer

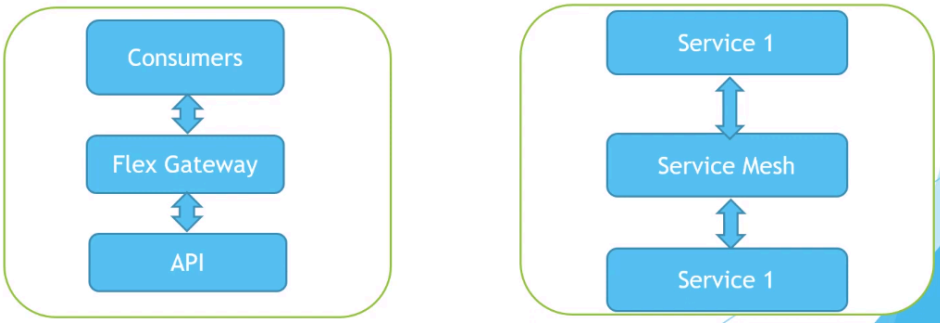
Scaling

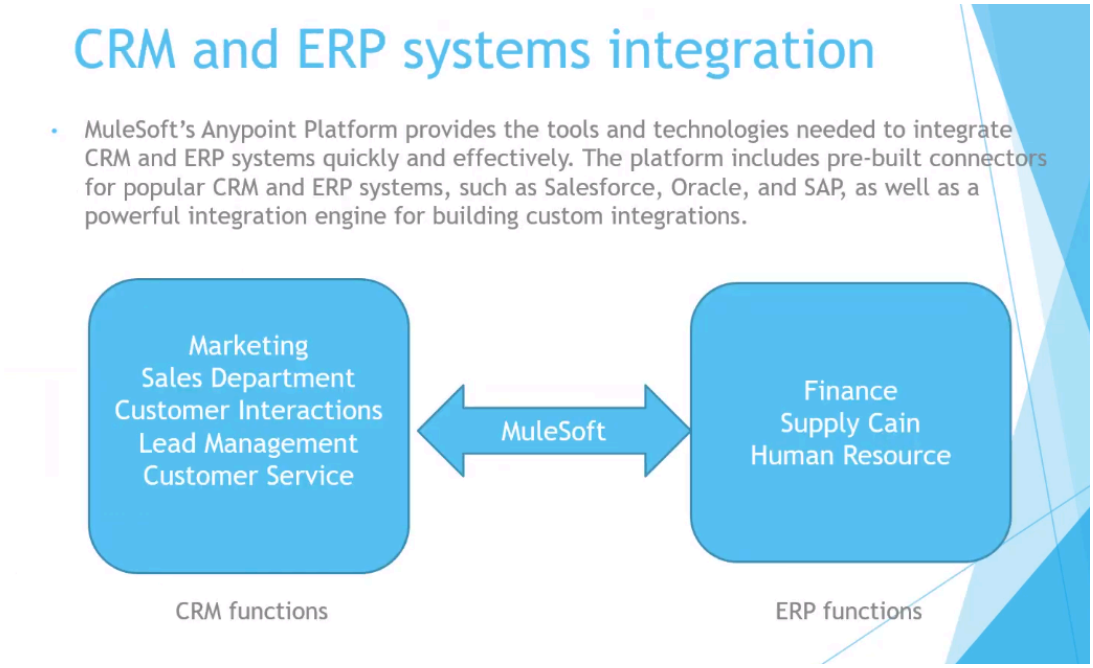


Flex Gateway & Service Mesh

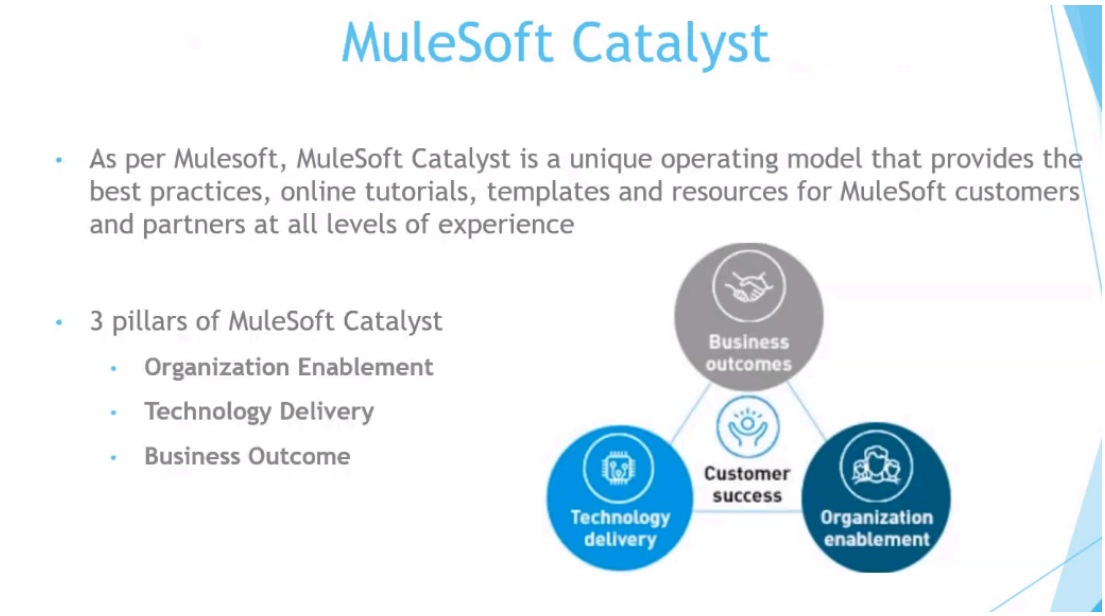
- **Flex Gateway :** It's a component of MuleSoft's Anypoint Platform that provides API management capabilities for managing, securing, and analyzing API traffic. It allows you to manage and secure APIs through features like rate limiting, security policies and analytics.

- **Service Mesh :** It's a dedicated infrastructure layer for managing network traffic between microservices in a distributed system. Service mesh provides features like load balancing, traffic management, service discovery and security.





Enterprise Resource Planning (ERP): Corporate management.
Customer Relationship Management (CRM): Sales and Marketing. Relation between Business and Clients.
Human Capital Management (HCM): Businesses manage their employees, from hire to retire
Supply Chain Management (SCM): Flow of goods, data, and finances related to a product or service, from the procurement of raw materials to the delivery of the product at its final destination.



7.7. Quizlet

What is the IT delivery gap?	The growing gap between IT delivery capacity and the ability of IT to meet those demands
What is the MuleSoft Catalyst?	MuleSoft Catalyst is centered on three core pillars: Business Outcomes, Organizational Enablement, and Technology Delivery. BOT
What are some common factors for IT delivery gaps?	1. Lack of alignment around Business Outcomes 2. Organization wasn't Enabled to deliver success 3. Project took too long to realize value
What is MuleSoft's approach for closing IT gaps?	A new approach enables central IT to focus on operating, connecting, and abstracting systems of record (System APIs). Subsequently, the line of business IT benefits by consuming previously built assets and then extending those resources into new solutions, ultimately delivering customer-facing innovation. This enables a model for PRODUCTION, CONSUMPTION, and FEEDBACK... Also known as an Application Network.
What is API-Led Connectivity?	A production + consumption model (i.e. core capabilities must be packaged up for consumption) or a way to connect data to applications through a series of reusable and purposeful modern APIs (System, Process, and Experience). SPE
What is an Application Network?	An application network emerges from our approach to enterprise integration, called API-LED CONNECTIVITY, and our equally unique approach to organizational structure and application delivery, the CENTER FOR ENABLEMENT.
What is an Application Building Block?	An Application Network is composed of application building blocks. These have multiple elements. The API interface, the API implementation, and the API management can be considered as building blocks. All building blocks have their own specific, unique lifecycles to follow. IIM
What are System API?	System APIs provide a means of insulating the user from the complexities or any changes to the underlying systems we are trying to integrate (authenticate) with. It is focused on Connectivity.
What are Process APIs?	Process APIs interact with and shape data within a single system or across systems (breaking down data silos) and are created here without a dependence on the source systems from which that data originates. This is typically where the business logic is implemented for further processing data.
What are Experience APIs	Experience APIs are the means by which data can be reconfigured so that it is most easily consumed by its intended audience, all from a common data source. Experience APIs are concerned with how data is presented securely.
What is the Center for Enablement (C4E)?	With a major culture shift that the new API development model requires, there needs to be a people and business process component as well. We suggest doing this by establishing a cross-functional team called a Center for Enablement, or C4E.
What are the different paths/playbooks for the MuleSoft Catalyst methodology?	● Business Outcome ● Organizational Enablement o Center for Enablement (C4E) o Internal Support o Training

	● Technology Delivery o Anypoint Platform o Projects
What is agile development?	An iterative approach to project delivery, in which software is built in an incremental fashion from the beginning of the project by creating user stories, prioritizing, and then implementing code.
What are the three main building blocks for the complete MuleSoft API lifecycle?	Design, Implement, and Management.
What are the stages of design an API in MuleSoft?	Start from an outside-in perspective and then Design, Simulate, Feedback, and Validate. DSFV
What are the stages of implementation in MuleSoft?	Accomplished in a systematic manner, we Build and Test in the implementation phase (Roles involved in this stage include: Integration Dev, Architects, Ops, API Manager). BT
How do we manage an API with MuleSoft?	Embracing modern DevOps-centric processes and tooling is critical to reduce mean time-to-production then Secure, Deploy, Monitor, Troubleshoot, and Manage. SDMTM
Describe the different Roles within a MuleSoft team	Platform Architect, Integration Architect, Developer, and Operations.
What is DevOps?	The word DevOps is a combination of Development and Operations, symbolizing that these functions must merge and cooperate to meet business requirements. CI/DC, automated testing, and active monitoring are the most common principles.
What does CI/CD stand for?	Continuous Integration/Continuous Delivery
What in an API?	An API is equivalent to a user interface, except it's designed for software instead of humans. This is why APIs are often described in the media as technology that allows applications to talk to one another.
What is Cloud Computing?	A collection of virtualized, software-defined information technology (IT) functions that have been abstracted from the hardware. Think of the cloud as a virtual data center.
What is Software as a Service (SaaS)?	Application software. This provides the user-facing applications that enable business.
What is Platform as a Service (PaaS)?	An infrastructure that supports application development. PaaS offers developers access to managed programming language and database ecosystems in which they can automatically deploy their application code. The middleware that enables the development of advanced applications (I.e. Operating systems, Middleware, Software, Runtime).
What is Infrastructure as a Service (IaaS)?	A virtualized environment on which systems can be deployed. IaaS is the bottom base layer and provides the underlying infrastructure that enables and supports PaaS and SaaS. (i.e. Servers, Storage, Networks, and Security).
What is software scalability?	From a business point-of-view, scalability is the ability to serve customers seamlessly even when a sudden change in demand occurs. From an IT point-of-view, scalability is the ability to add/remove infrastructure resources needed by business applications to manage the increased/decreased demand in the number of business transactions.

What is a Vertical Scale?	Increase the compute and/or memory capacity of the server
What is a Horizontal Scale?	Add more servers to the server pool
What is a Diagonal scale?	Increase the compute memory capacity and then add more servers horizontally once you've maximized your vertical.
What does SOA mean?	Service Oriented Architecture
What are Microservices?	A variant of the service-oriented architecture structural style - it is an architectural pattern that arranges an application as a collection of loosely coupled, fine-grained services, communicating through lightweight protocols. Its approach is consistent with microservices and vice versa.
What are the 3 main components for API-Led Connectivity?	Interface, Orchestration, and Connectivity. Use a top-down approach for design, starting with how data is presented to the interface, how data is translated/manipulated, and how data is connected. IOC
What is the Interface?	Presentation of data in a governed and secure form eAPI
What is the Orchestration?	Application of logic to that data, such as transformation or enrichment pAPI
What is Connectivity concerned with in MuleSoft?	Access to source data, whether from physical or external services (i.e. System APIs) sAPI
What is the aggregation of data?	Data from multiple applications are copied into a central location for further analysis and processing, for example, data warehouses, data lakes, or dashboards tools (i.e. APIs, iPaas, Virtualization, B2b transfers, File Transfers, and ETL).
What does streaming data mean?	Typically, such devices can generate data either at a slow pace (every minute or hour) or at a fast pace (sub-milli-second) to provide updates to a subscriber/listener (i.e. Messaging Bus, Streaming Platforms).
What is bulk or batch data movement?	Scenarios where delays in presenting information is acceptable or presenting information in real-time is not required. Such scenarios optimize and provide better utilization of the resources.
Synchronous and Asynchronous triggers?	Exchange data in either a sequential pre-defined order or return a promise and start other processes while waiting for a response (i.e. APIs, Robotic Process Automation, iPaaS)
What is orchestration and data processing?	A layer of abstraction is developed between the source and target where deeper business logic is embedded (Process APIs). Often the orchestration use cases are heavily tied and specific to the underlying business domains (i.e. iPaas, Robotic Process Automation)
What does ETL stand for? When is it used?	Extract, Transform, Load. ETL can be used to store legacy data or aggregate data to analyze and drive business decisions.
What are the 4 integration primitives? Describe each primitive.	Components, Interactions, Messages, and Interfaces. Components relate to each other through Interactions. Interactions consist of one or more Messages, and are communicated through Interfaces.
What are the 3 Interaction type primitives? Describe each primitive	Queries, Commands, and Events. Queries do not change the system state, Commands collaborate to change the system state, and Events that communicate the changed state.
What are the 5 interaction pattern primitives? Describe each primitive.	Request-reply, One-way, Multicast, Batch, and Stream. Request-reply is when a component sends a message to another and expense a response. One-way is when a component sends message to another without expecting a response. Multicast is when a component sends a single message to multiple components. Batch is when a component sends a bounded collection of messages to another

	component. Stream is when an unbounded serious of messages is sent from one component to another.
What are the 3 interaction composition primitives? Describe each primitive.	Aggregation, Orchestration, and Choreography compositions. Aggregation is a form of stateless composition where a component receives a single query request, then executes multiple queries to other components and assembles their responses to formulate a reply. Orchestration is the stateful coordination of interactions, where the flow may change depending on intermediate results. The third composition pattern, choreography, is the reactive coordination of component actions based on interaction triggers and resulting events. Choreography is where microservices work independently but coordinate with each other using cues or events.
What are MuleSofts 3 pillars of Observability?	Logs, Metrics, and Traces.
What are Logs in MuleSoft?	Messages logged by an application, system or OS when an event occurs.
What are Metrics in MuleSoft?	Metrics are measurements, for example a system sending the current CPU or memory state to a collector.
What are Traces in MuleSoft?	Clues left behind by a transaction as it flows through different applications. Traces weave a complete transaction from beginning to end.
What is Cloud deployment in MuleSoft?	A cloud-based application is fully deployed in the cloud. Cloud deployment options include Commercial Anypoint and Government Anypoint.
What are MuleSofts 3 deployment options?	Cloud, Hybrid, Private Cloud (i.e. On-Premise).
What is Hybrid deployment in MuleSoft?	A hybrid deployment is a way to connect infrastructure and applications between cloud-based resources and on-premises systems. Hybrid options include classic mule runtimes and container based runtime fabrics.
What is a Private Cloud in MuleSoft?	Deploying resources on-premises, by using virtualization software and resource management tools, is called private cloud. Customers own Anypoint private cloud deployed in their datacenter.
What are the limitations or challenges created by using Microservices?	Governance, security, and discoverability.
What is an API Gateway?	API Gateway is one of three components in API Management; the others being developer portals and API security. The API Gateway controls North-South traffic out of the Kubernetes cluster, it is there for security purposes.
What is an Anypoint Connector?	Reusable extensions to Mule runtime engine (Mule) that enable you to integrate a Mule app with third-party APIs, databases, and standard integration protocols. Connectors abstract the technical details involved with connecting to a target system.
What is the typical DevOps lifecycle?	The lifecycle starts with establishing a continuous integration process, then transitions to producing and deploying software efficiently through a pipeline.
What is Maven?	A build automation tool and dependency manager.
What is MUnit?	The MuleSoft integration testing framework, to help our developers and our customers create tests with mock data.
What is Jenkins?	Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

What is a web service?	A method of communication that allows two software systems to exchange data over the internet. It is the actual API implementation you make callouts to.
What is an API proxy?	It controls access to a web service, restricting access and usage through the use of an API gateway
What is an API Notebook?	An API Notebook is the artifact to convey the inspiration for what is possible with an API, for example a MuleSoft portal.
What is Virtualization?	Virtualization enables a single server to run the operating systems and applications from multiple servers simultaneously.
What does data sync mean?	When data is required in other systems to complete the transaction, for example sending order information to an ERP and returning an external order number (i.e. APIs, Messaging Bus, iPaas, Virtualization, File Transfer).
What is KafKa?	Kafka is a distributed data store optimized for ingesting and processing streaming data in real-time. Kafka is primarily used to build real-time streaming data pipelines and applications that adapt to the data streams.
What is the purpose of an API specification? What are some benefits to having an API spec?	An API spec consists of a plan of how your API should look structurally - like a blueprint of a house. It's a key part of API development because it can help you isolate design flaws or problems before you write a line of code saving you time by future-proofing your design/solution.
What is the difference between monolithic and microservices?	A monolithic application is built as a single unified unit while a microservices architecture is a collection of smaller, independently deployable services. Monolithic architectures are hard to develop and maintain, a microservices architecture allows for greater agility with its smaller, more targeted services.
What is a Service Mesh?	A service mesh is an architectural pattern for microservice deployments. It is just an infrastructure layer that sits on the top of the micro services and handles all the communications between services. It also addresses the challenges created by microservices by drawing out common capabilities of security, fault tolerance, and management out of the service code.
What is the difference between a service mesh and an API Gateway?	A service mesh aims to manage internal service-to-service communication, while an API Gateway is primarily focused to manage traffic from client-to-service.
What is the Control plane in Anypoint Platform?	The Anypoint Platform control plane provides a set of cloud services that simplify the design, reuse, and management of integrations and APIs (i.e. US Cloud, EU Cloud, Government Cloud, and Private Cloud).
What is the Runtime plane?	The Anypoint Platform runtime plane is where applications are deployed, and also where the Mule runtime engine and other application-related services, such as Anypoint Connectors run (i.e. CloudHub, Anypoint Runtime Fabric, On Premise Mule Runtime).
When is the Runtime Fabric a good deployment option?	Use Anypoint Runtime Fabric to deploy Mule runtimes within your own data centers, whether it's in a private IaaS (Microsoft Azure or AWS) or on-premises infrastructure.