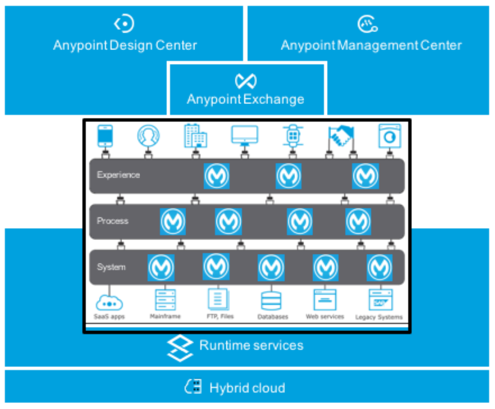
PART 1: Implementing API-Led Connectivity with Anypoint Platform

Objectives

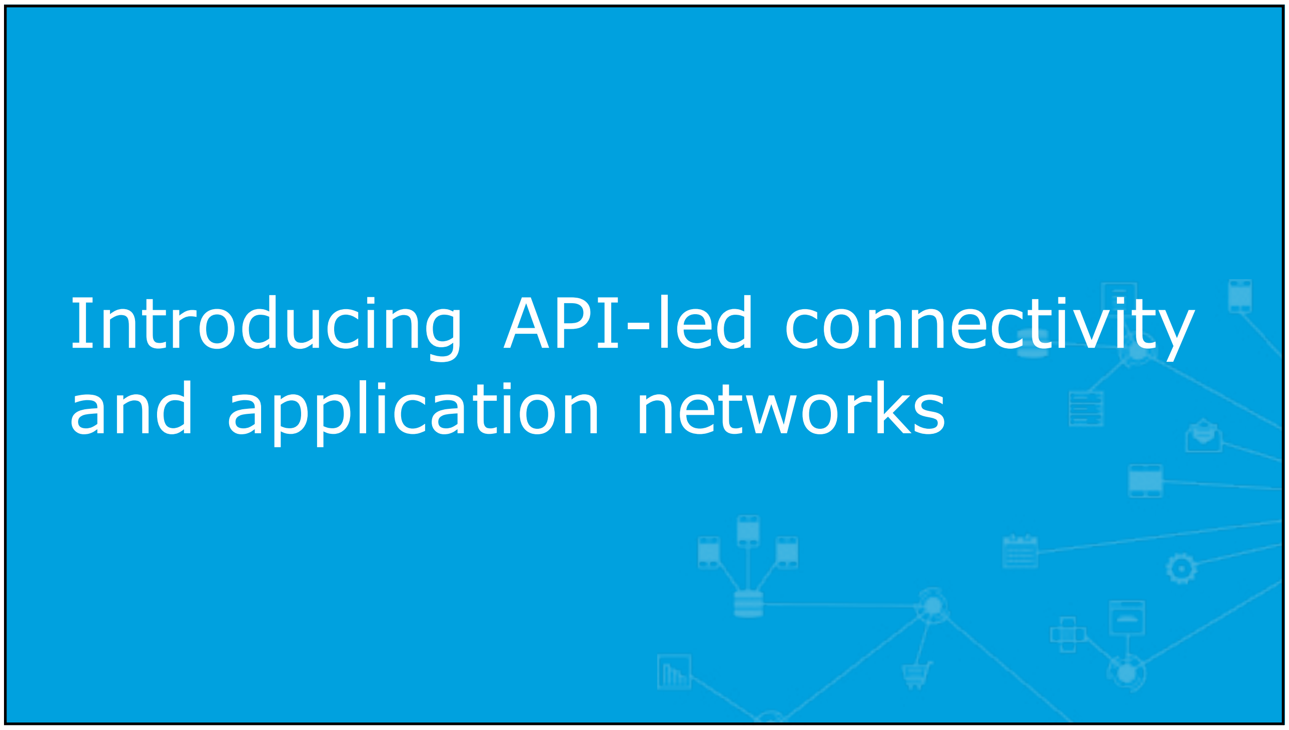
- Describe what API-led connectivity is and its benefits
- Use Anypoint Platform to take an API through its complete lifecycle
- Design, build, deploy, manage, and govern an API



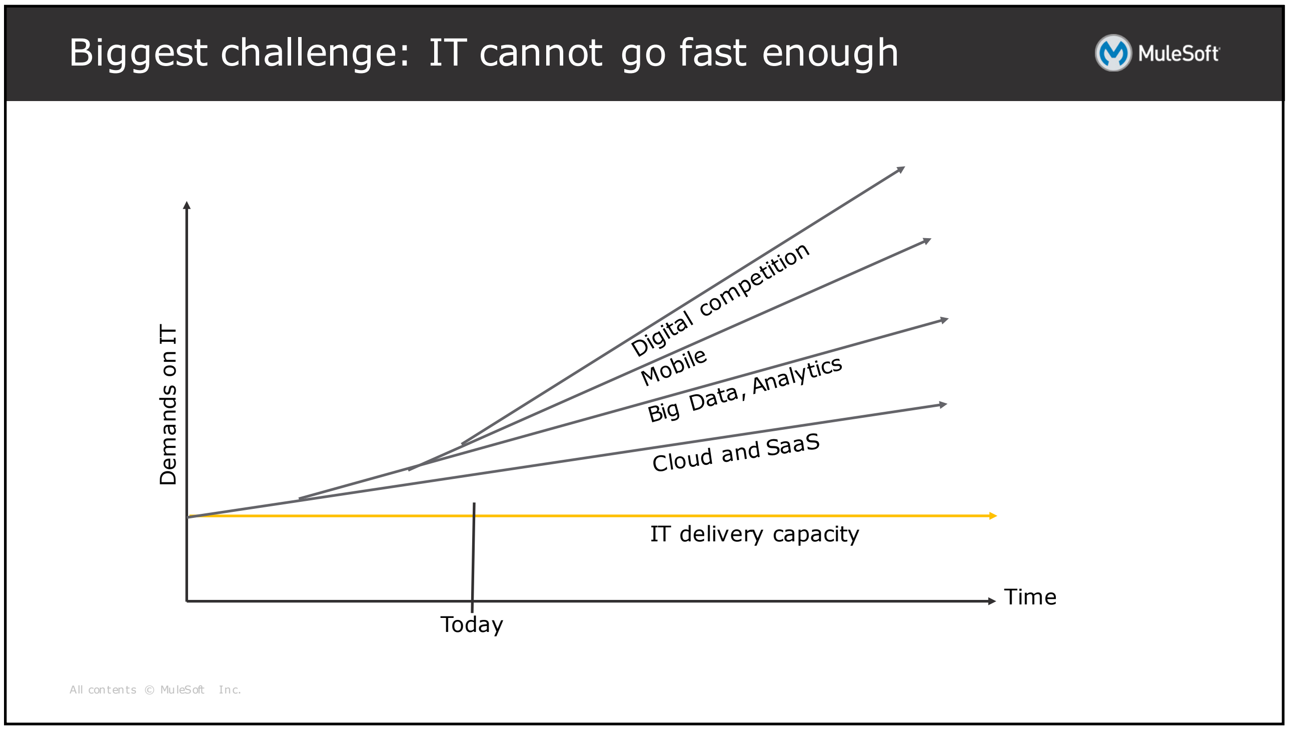
Module 1: Introducing API-Led Connectivity

Objectives

- Identify the problems faced by IT today
- Describe what API-led connectivity is and its benefits
- Explain what web services and APIs are
- Explore API directories and portals
- Make calls to secure and unsecured APIs
- Introduce API-led connectivity with Anypoint Platform
- Explore Anypoint Platform

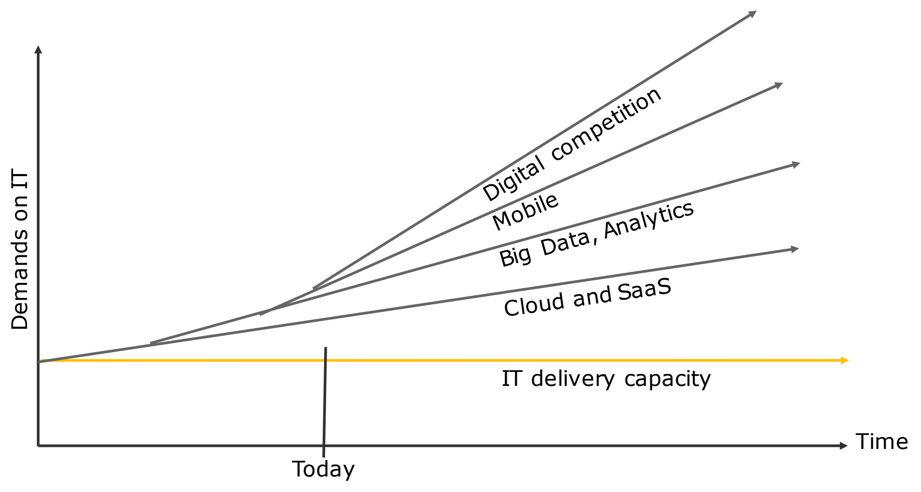


Introducing API-led connectivity and application networks

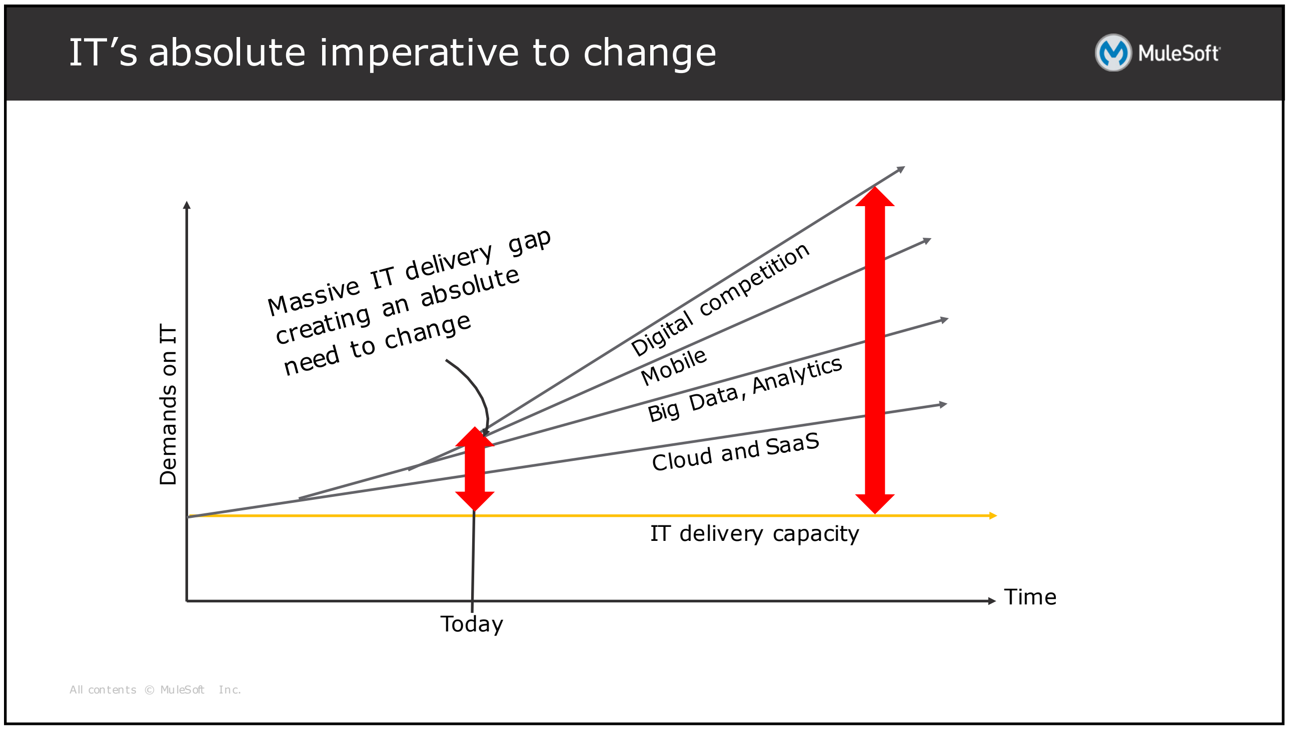


Biggest challenge: IT cannot go fast enough

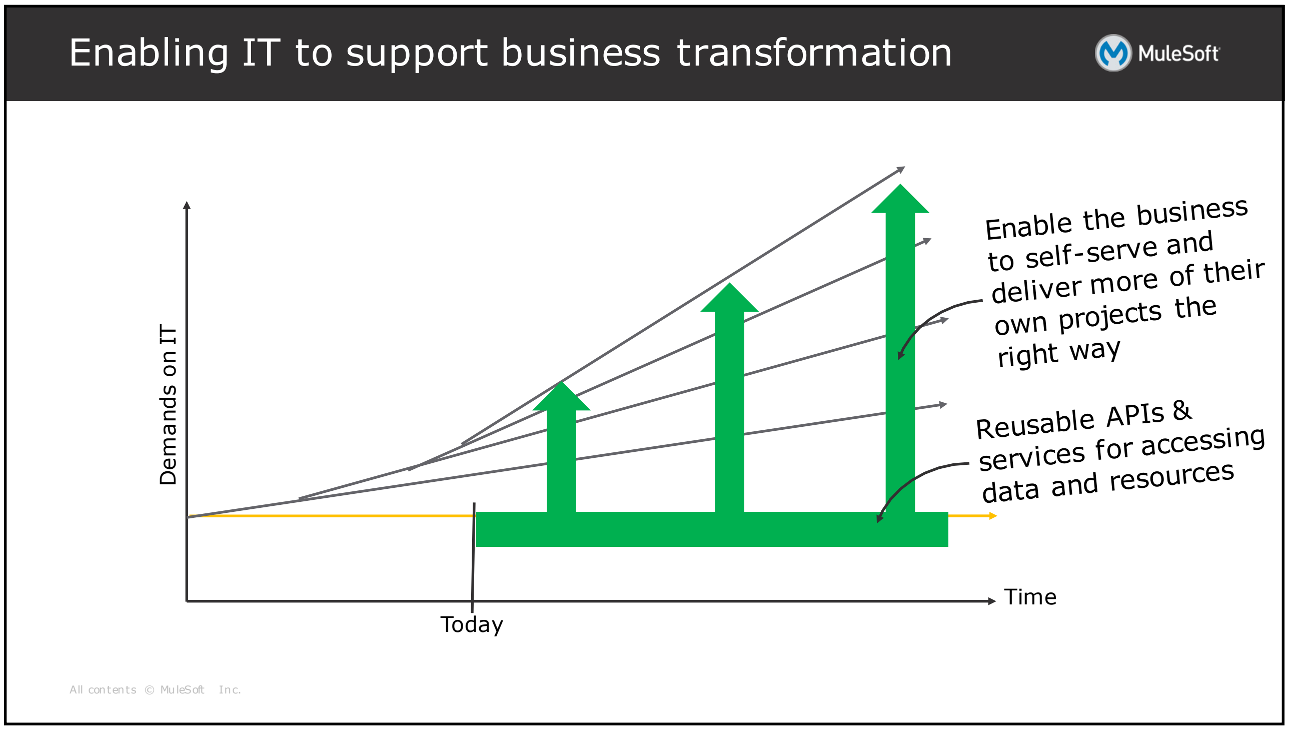
MuleSoft



All contents © MuleSoft Inc.



IT's absolute imperative to change



Enabling IT to support business transformation

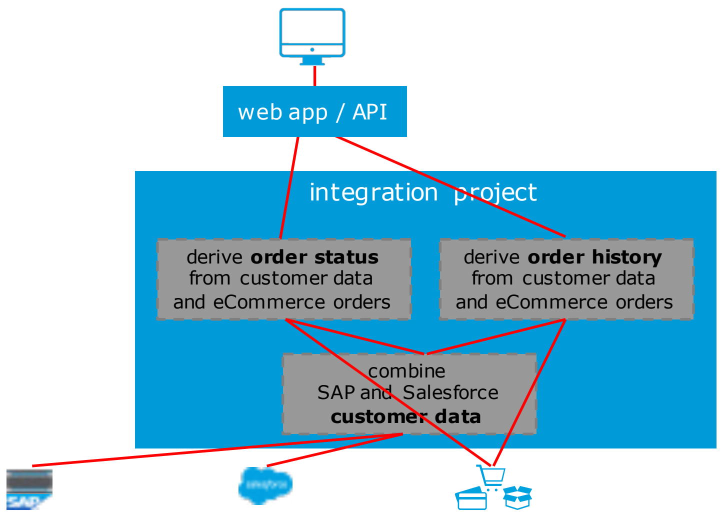
MuleSoft
Enable the business to self-serve and deliver more of their own projects the right way
Reusable APIs & services for accessing data and resources
Demands on IT
Time
Today

A traditional project-based approach

MuleSoft

Project objective: Web app that provides real-time order status and order history for sales team engaging with customers

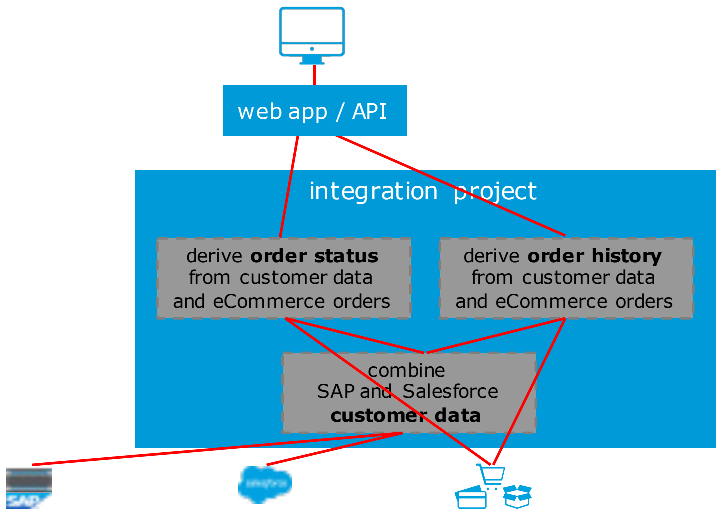
- Order data in eCommerce system
- Inventory data in SAP
- Customer data in SAP, SFDC

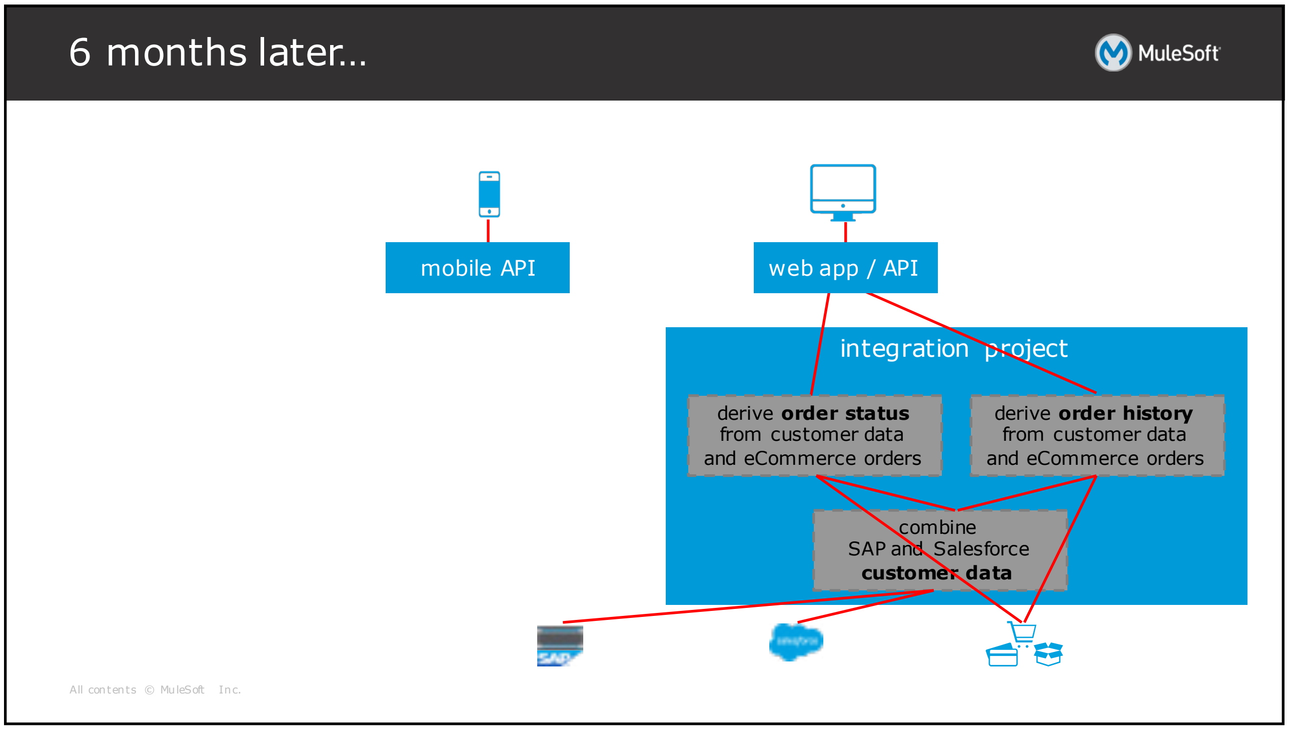

web app / API
integration project
derive **order status** from customer data and eCommerce orders
derive **order history** from customer data and eCommerce orders
combine SAP and Salesforce **customer data**

A traditional project-based approach

MuleSoft

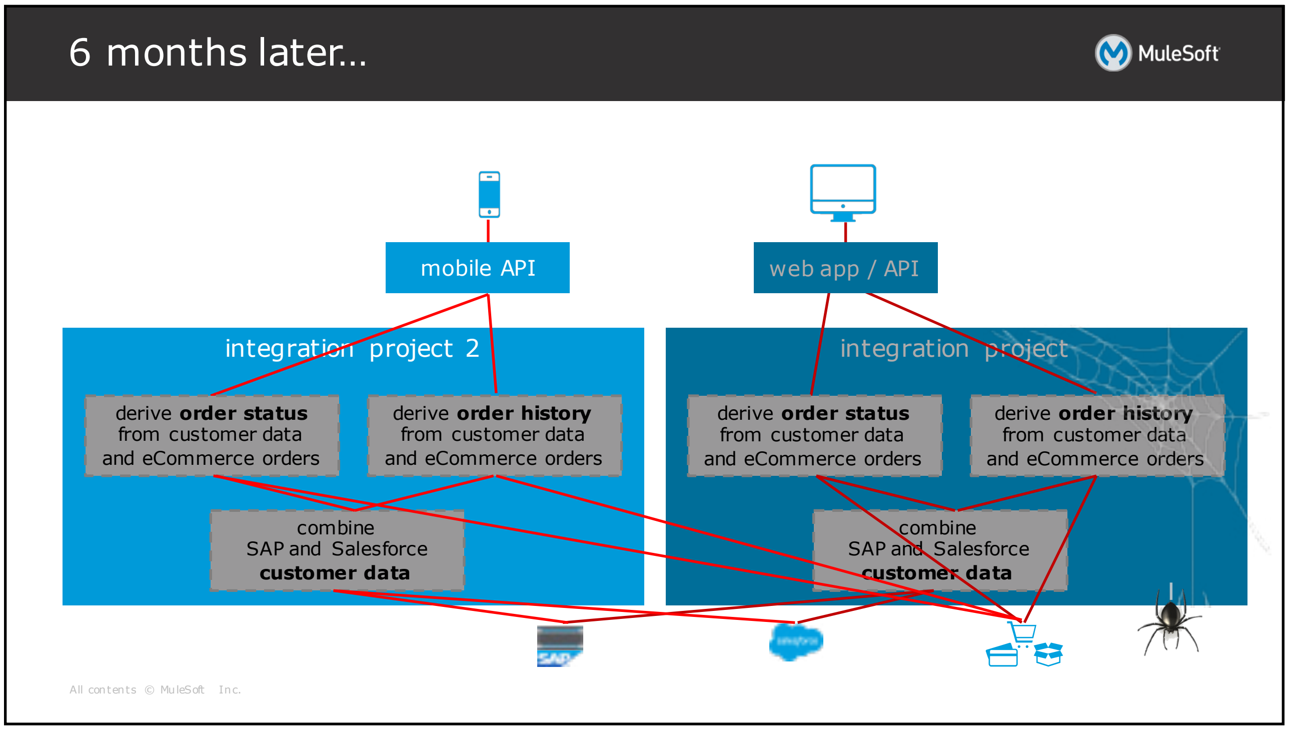
- ✓ On time
- ✓ Within budget
- ✓ Meets requirements
- ✗ No reuse
- ✗ Tightly coupled to apps
- ✗ Lack of governance


web app / API
integration project
derive **order status** from customer data and eCommerce orders
derive **order history** from customer data and eCommerce orders
combine SAP and Salesforce **customer data**



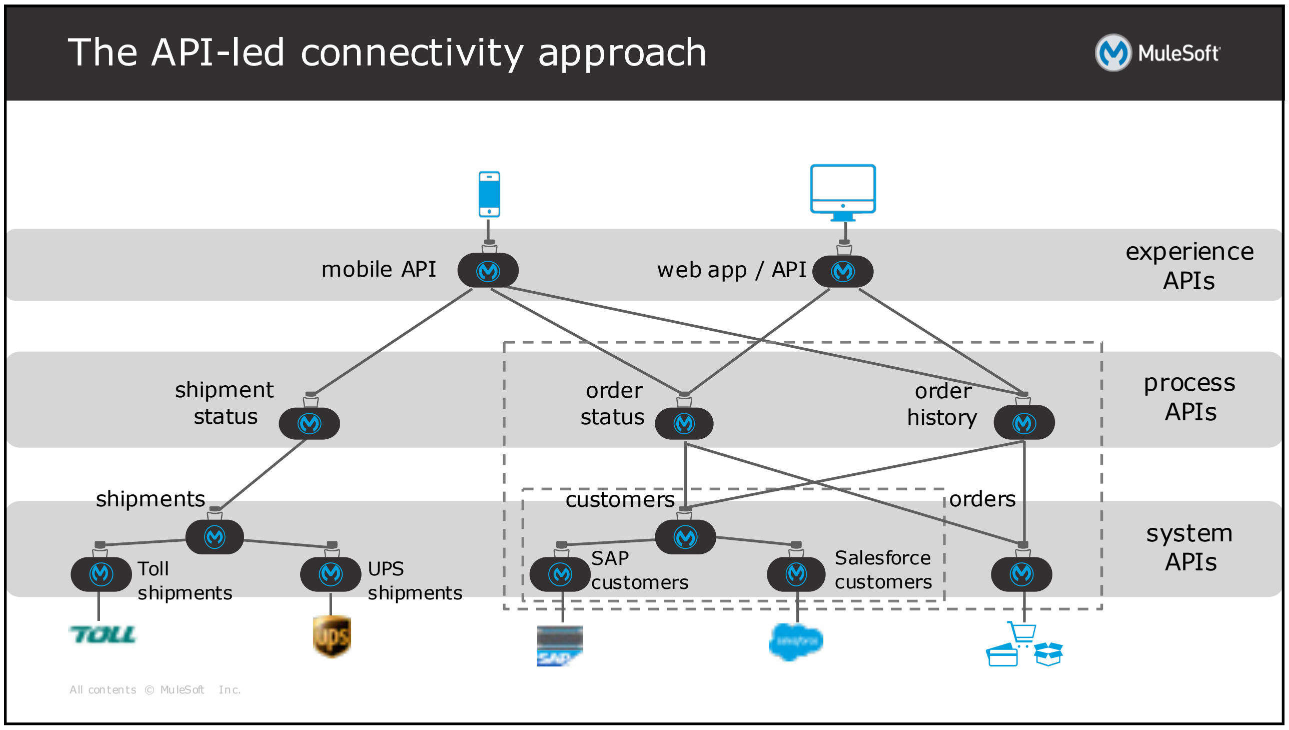
6 months later…

MuleSoft
mobile API
web app / API
integration project
derive **order status** from customer data and eCommerce orders
derive **order history** from customer data and eCommerce orders
combine SAP and Salesforce **customer data**

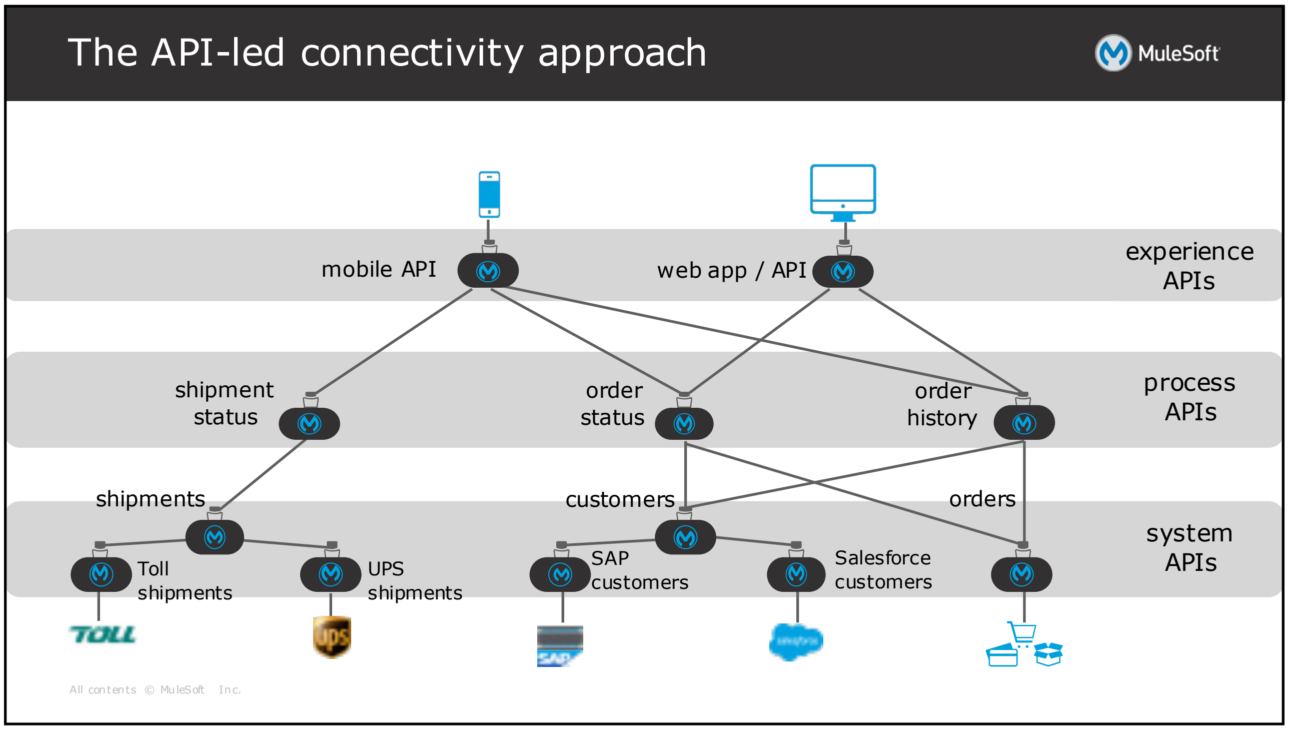


6 months later…

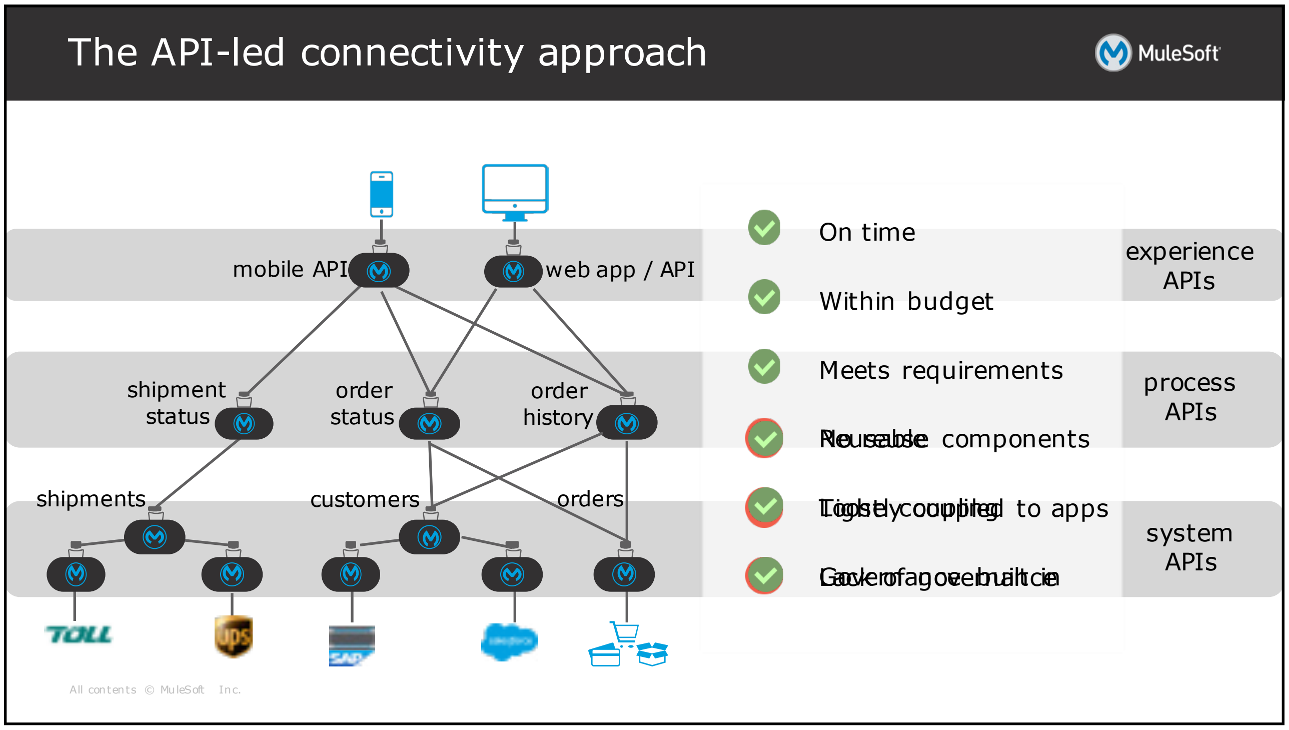
MuleSoft
mobile API
web app / API
integration project 2
derive **order status** from customer data and eCommerce orders
derive **order history** from customer data and eCommerce orders
combine SAP and Salesforce **customer data**
integration project
derive **order status** from customer data and eCommerce orders
derive **order history** from customer data and eCommerce orders
combine SAP and Salesforce **customer data**



The API-led connectivity approach



The API-led connectivity approach



The API-led connectivity approach

✓ On time

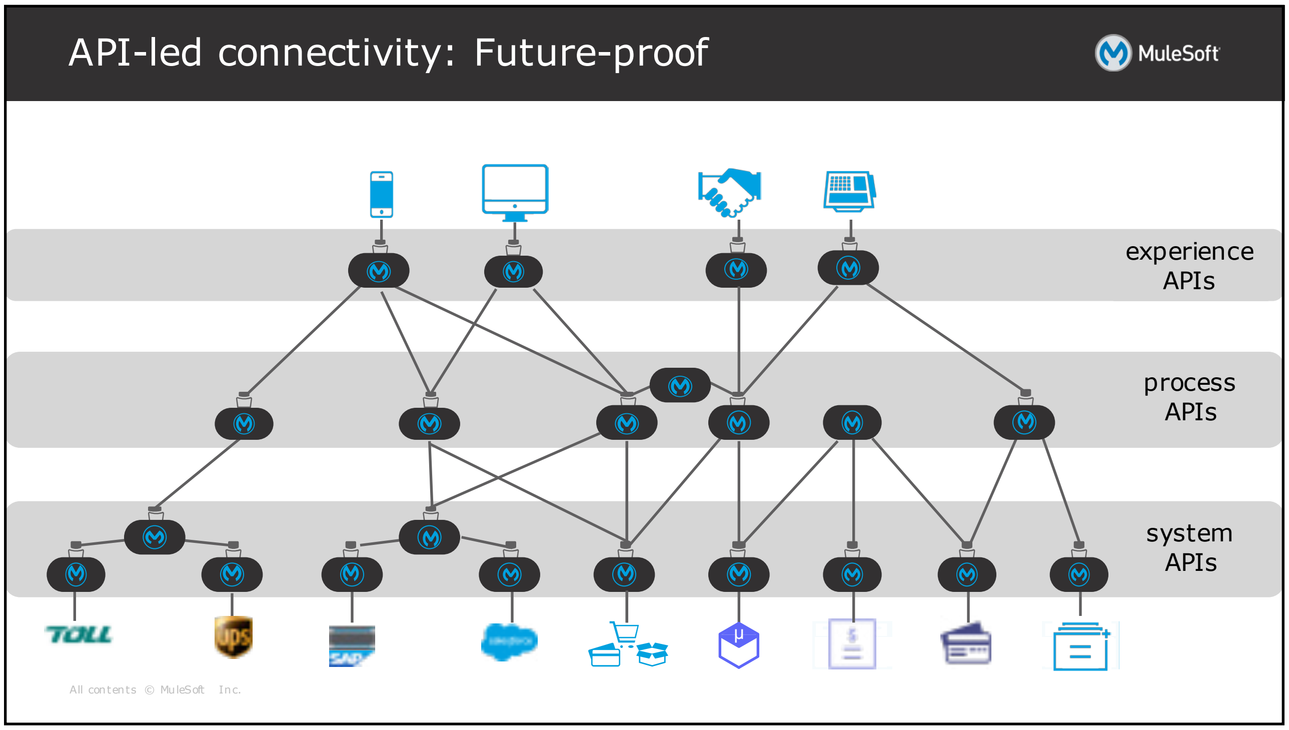
✓ Within budget

✓ Meets requirements

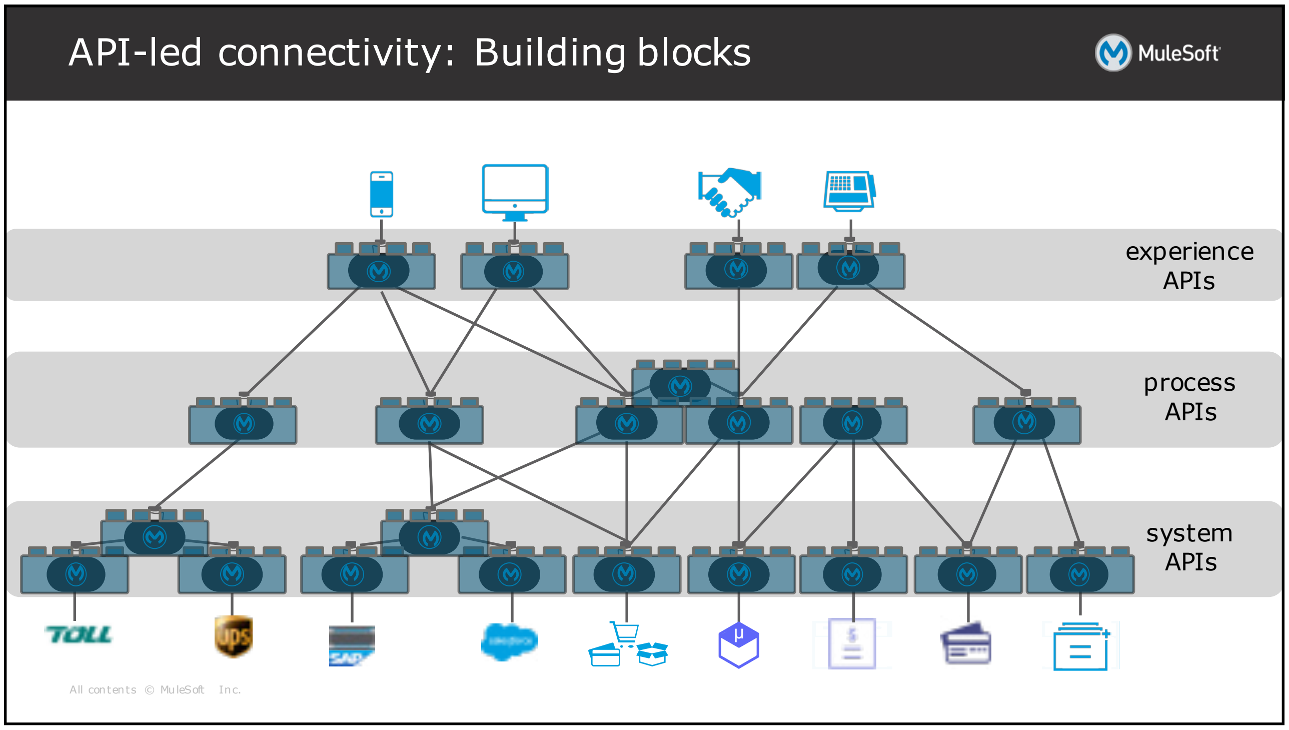
✓ Reusable components / Not reusable

✓ Loosely coupled / Tightly coupled to apps

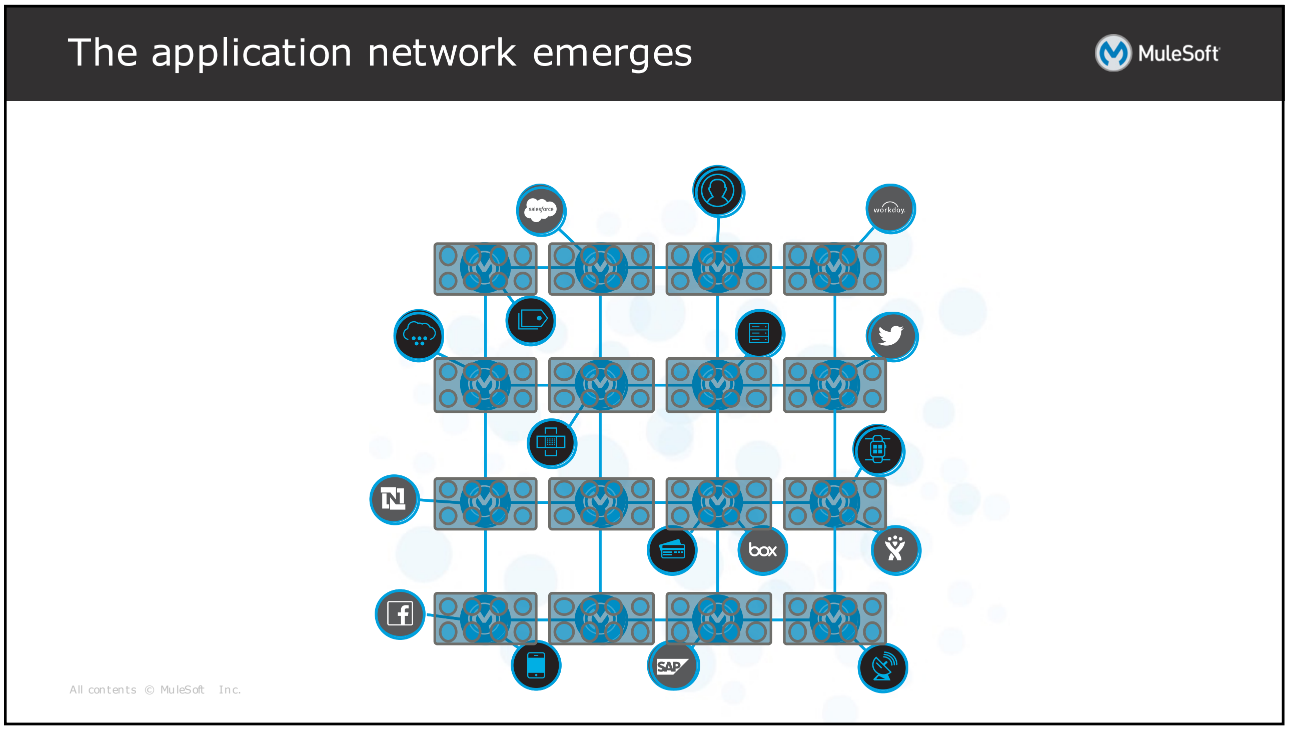
✓ Governance built in / Lack of governance



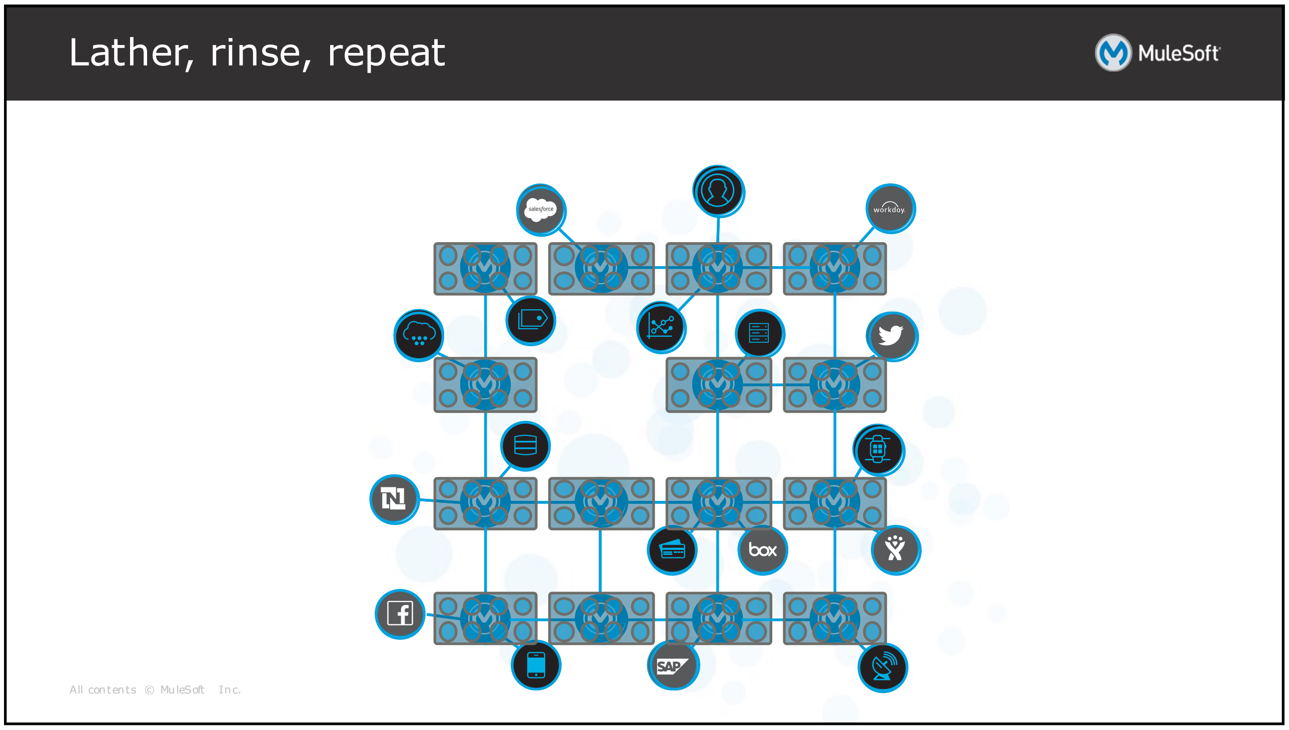
API-led connectivity: Future-proof



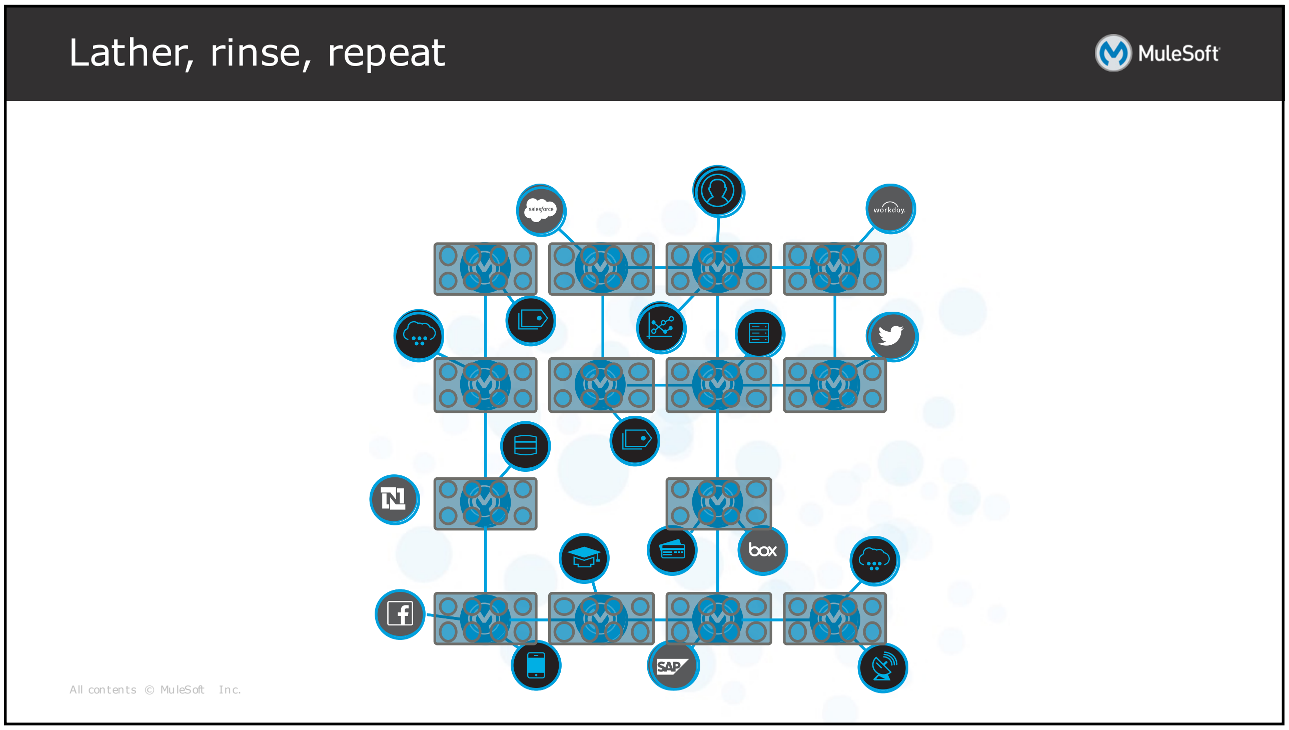
API-led connectivity: Building blocks



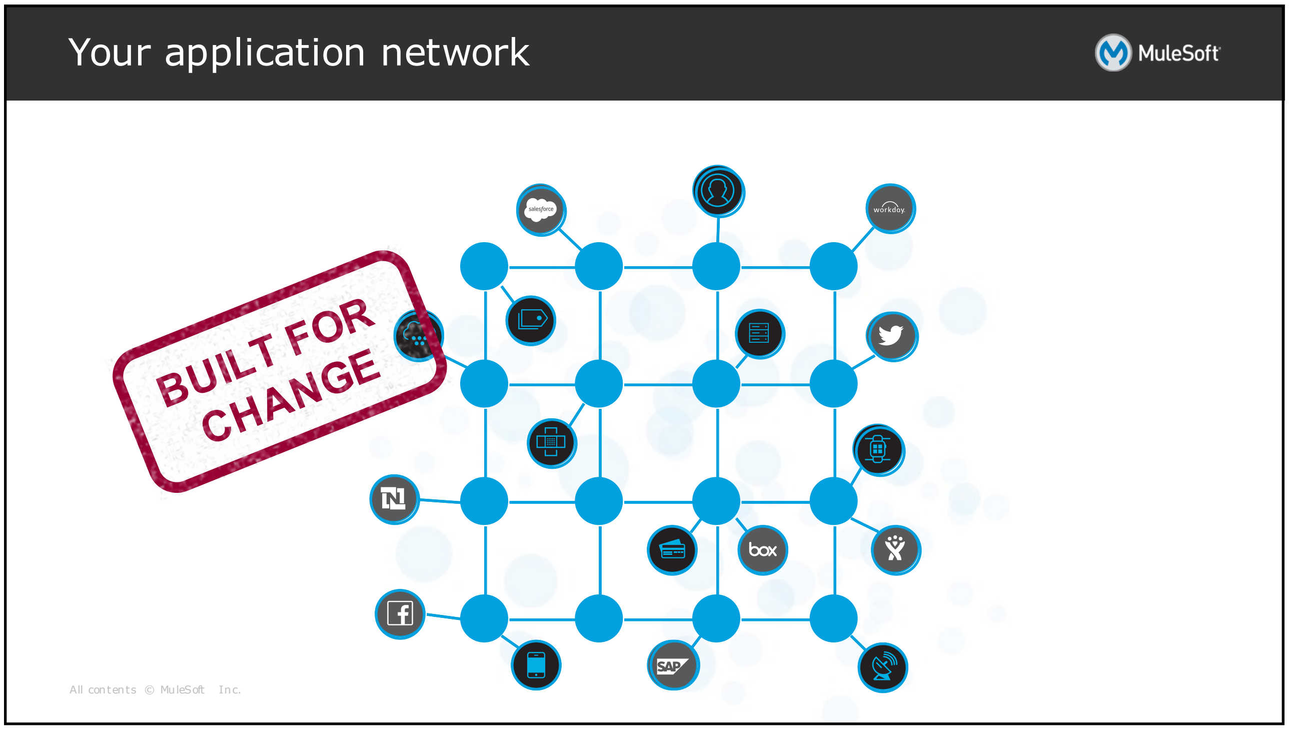
The application network emerges



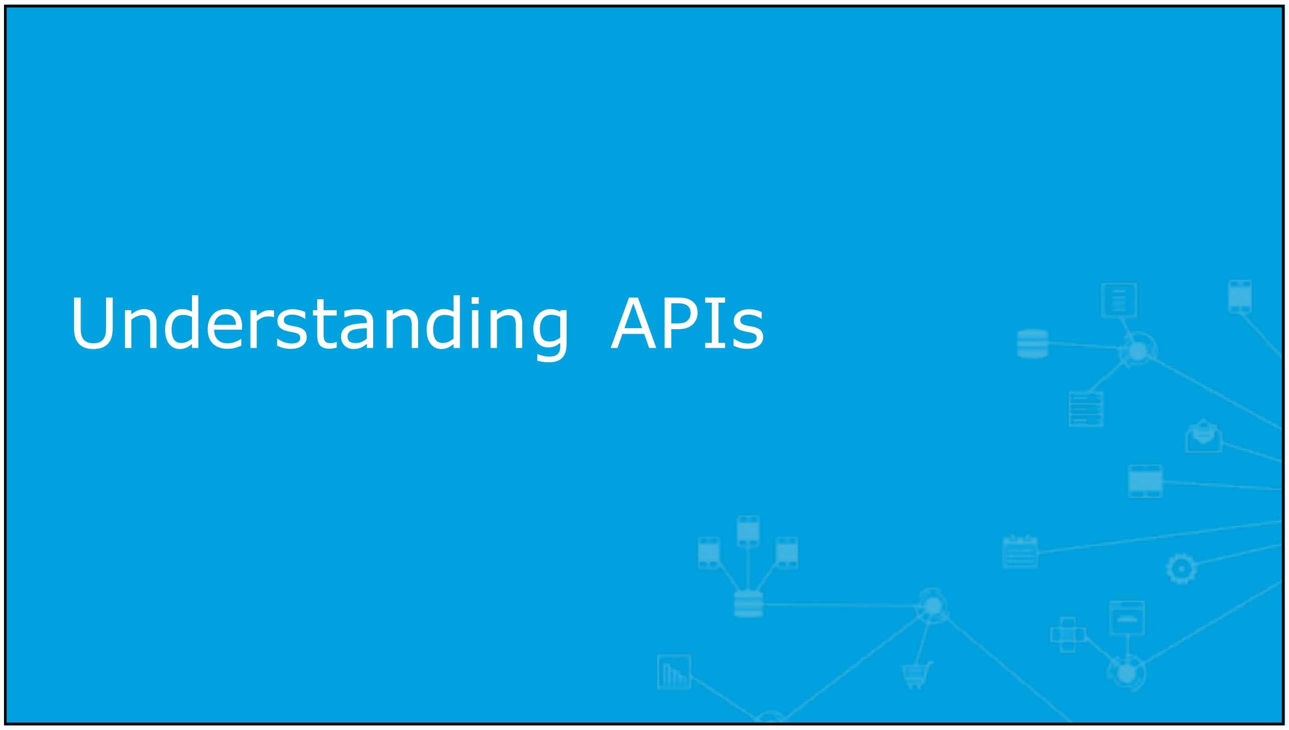
Lather, rinse, repeat



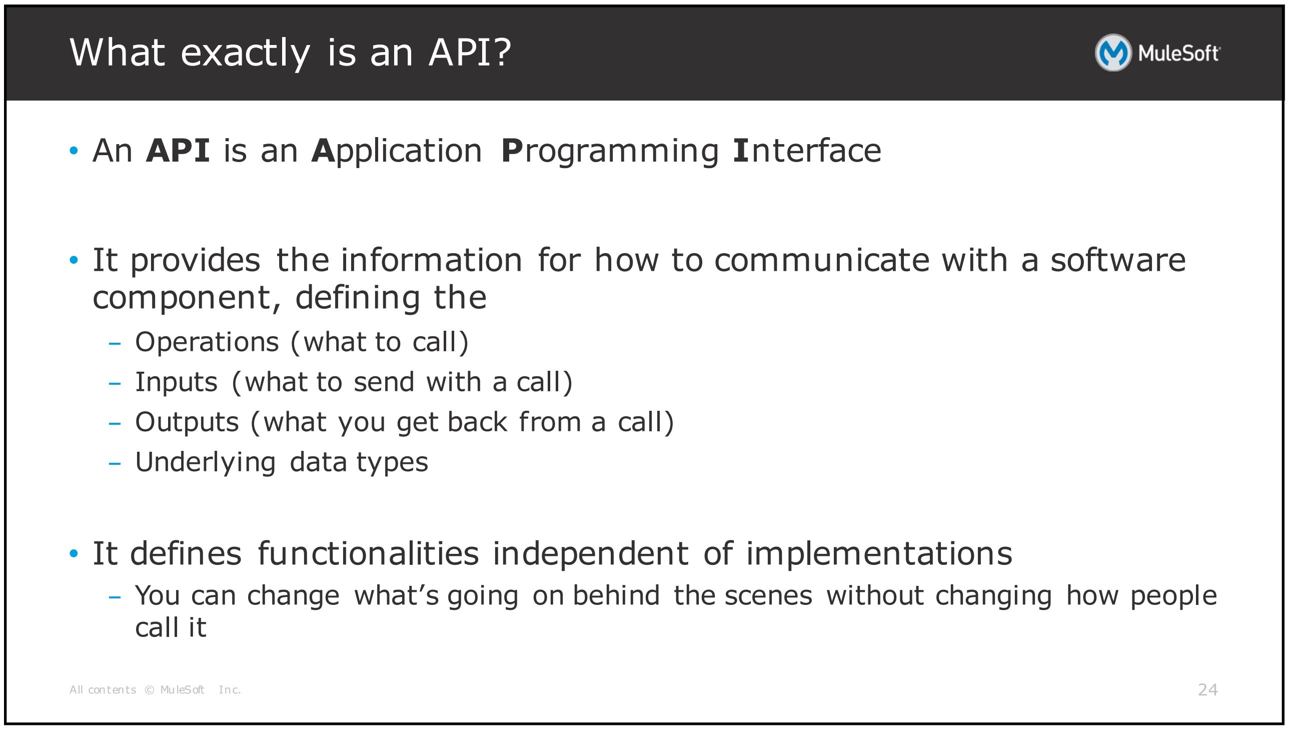
Lather, rinse, repeat



Your application network

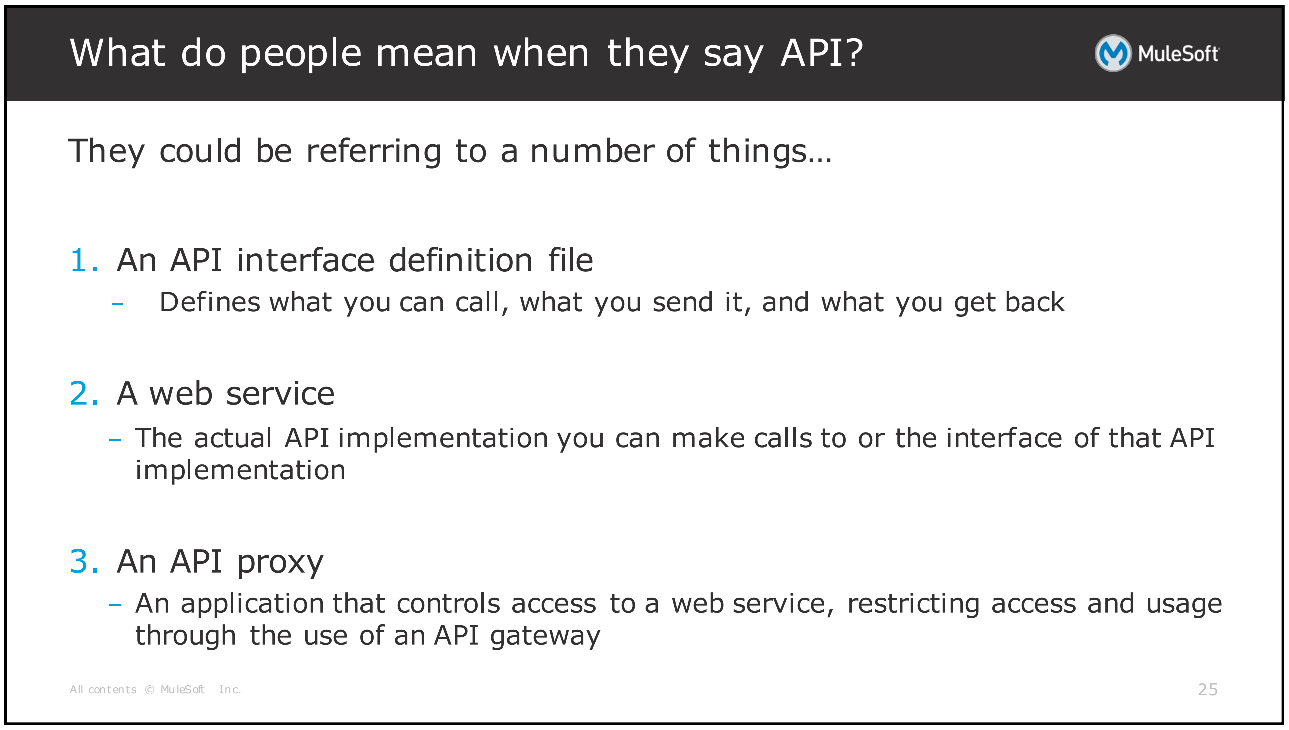


Understanding APIs



What exactly is an API?

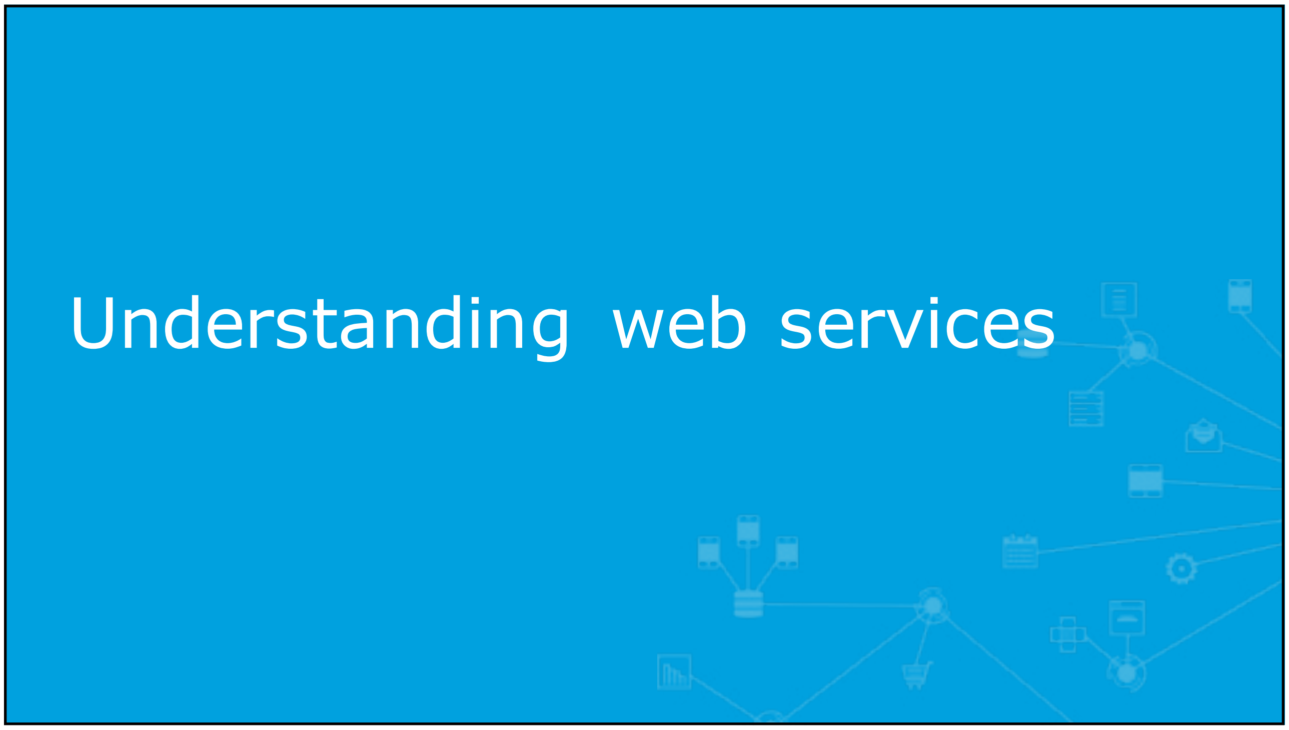
- An **API** is an **A**pplication **P**rogramming **I**nterface
- It provides the information for how to communicate with a software component, defining the
  - Operations (what to call)
  - Inputs (what to send with a call)
  - Outputs (what you get back from a call)
  - Underlying data types
- It defines functionalities independent of implementations
  - You can change what's going on behind the scenes without changing how people call it



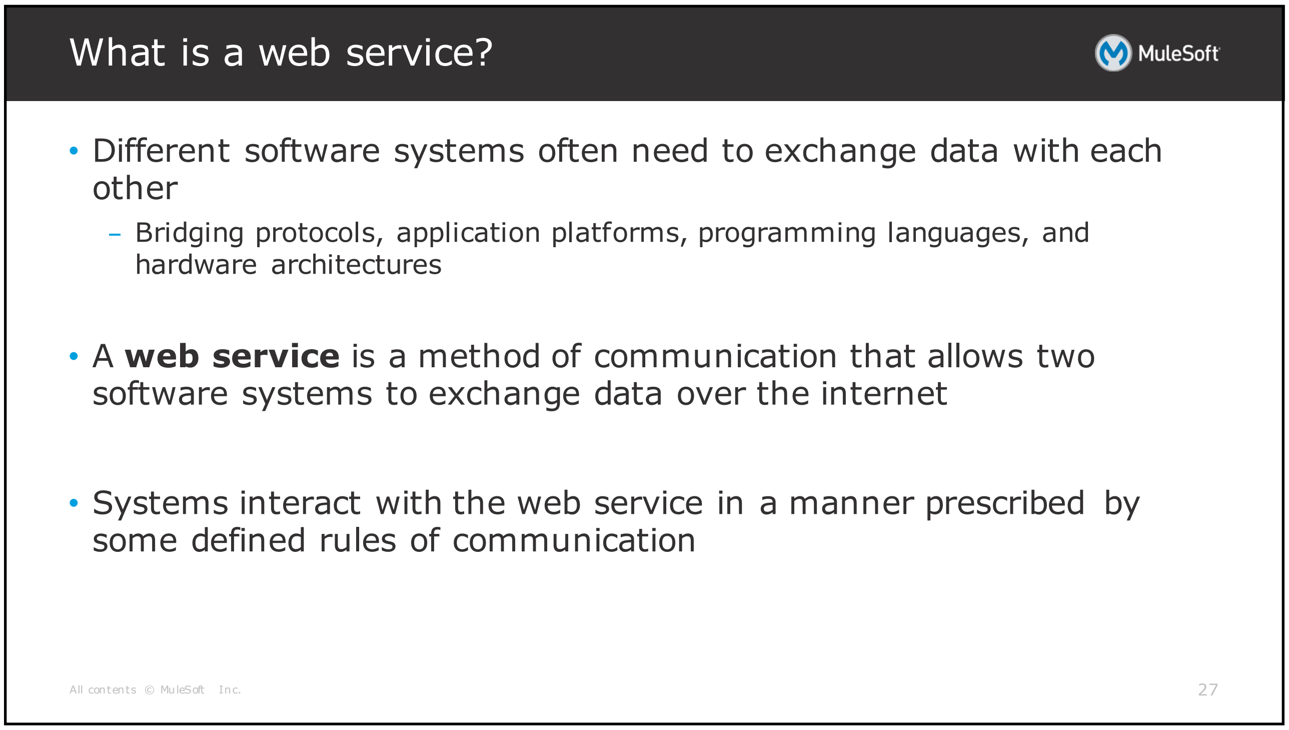
What do people mean when they say API?

They could be referring to a number of things…

1. An API interface definition file
   - 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



Understanding web services



What is a web service?

- Different software systems often need to exchange data with each other
  – Bridging protocols, application 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

Rules for communication

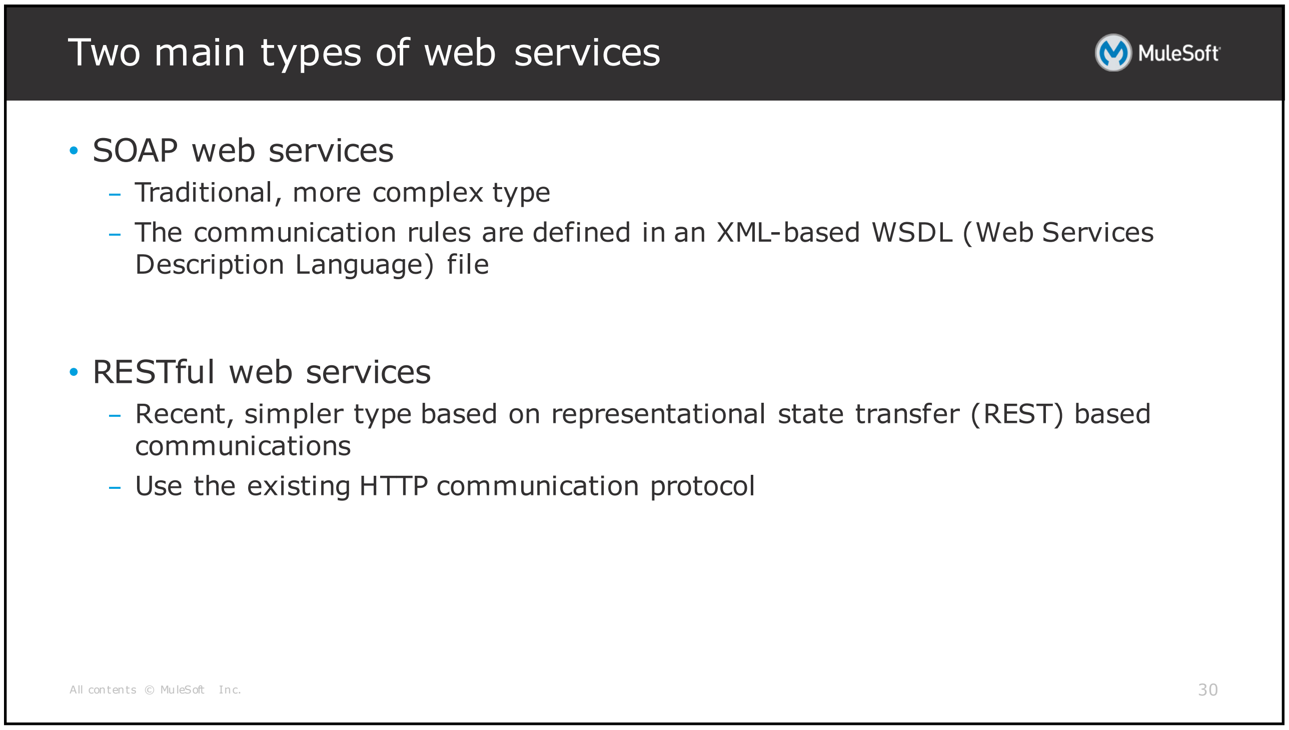
MuleSoft

- The rules must define
  - How one system can request data from another system
  - Which specific parameters are needed in the data request
  - What would be the structure of the data produced
  - What error messages to display when a certain rule for communication is not observed

The parts of a web service

MuleSoft

- **The web service API**
  - Describes how you interact with the web service
  - It may or may not (though it should!) be explicitly defined in a file
  - It could be any sort of text in any type of file but ideally should implement some standard API description language (or specification)
- **The web service interface implementing the API**
  - Is the code providing the structure to the application so it implements the API
  - This may be combined with the actual implementation code
- **The web service implementation itself**
  - Is the actual code and application



Two main types of web services

MuleSoft

- 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 based on representational state transfer (REST) based communications
  - Use the existing HTTP communication protocol

Introducing SOAP web services

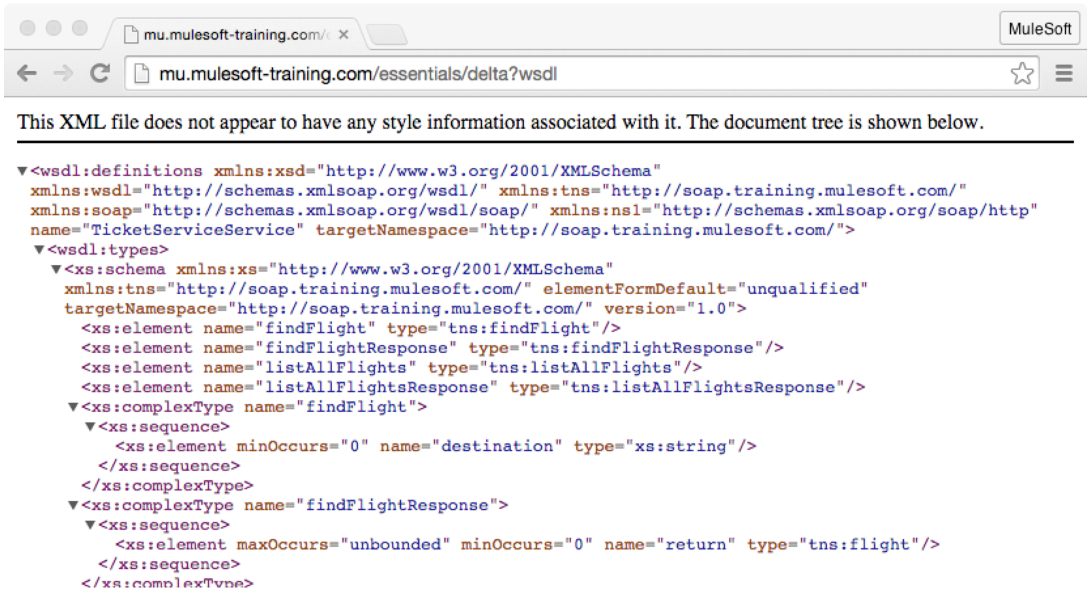
SOAP web services

MuleSoft

- Other systems interact with the web service in a manner prescribed by its WSDL description using SOAP protocol
- WSDL (Web Services Description Language)
  - An interface in XML-based, machine processable format
  - Defines operations, arguments, data types, and more
- SOAP (Simple Object Access Protocol)
  - An XML-based protocol
  - Defines the message architecture and message formats
- Requires tooling to publish and consume them

Example SOAP web service WSDL

MuleSoft


mu.mulesoft-training.com/
MuleSoft
mu.mulesoft-training.com/essentials/delta?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.

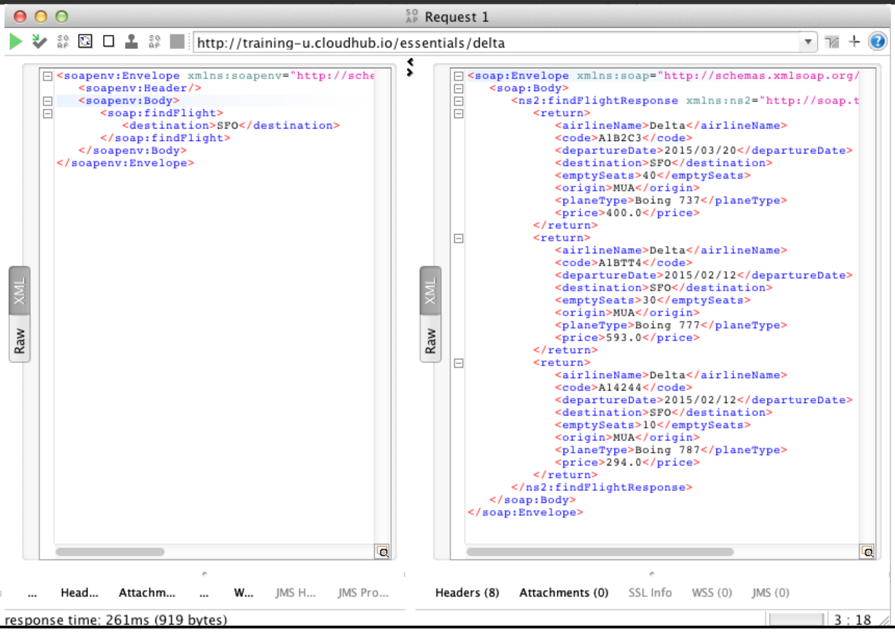
```
▼<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://soap.training.mulesoft.com/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:ns1="http://schemas.xmlsoap.org/soap/http"
 name="TicketServiceService" targetNamespace="http://soap.training.mulesoft.com/">
  ▼<wsdl:types>
    ▼<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:tns="http://soap.training.mulesoft.com/" elementFormDefault="unqualified"
     targetNamespace="http://soap.training.mulesoft.com/" version="1.0">
       <xs:element name="findFlight" type="tns:findFlight"/>
       <xs:element name="findFlightResponse" type="tns:findFlightResponse"/>
       <xs:element name="listAllFlights" type="tns:listAllFlights"/>
       <xs:element name="listAllFlightsResponse" type="tns:listAllFlightsResponse"/>
      ▼<xs:complexType name="findFlight">
        ▼<xs:sequence>
           <xs:element minOccurs="0" name="destination" type="xs:string"/>
         </xs:sequence>
       </xs:complexType>
      ▼<xs:complexType name="findFlightResponse">
        ▼<xs:sequence>
           <xs:element maxOccurs="unbounded" minOccurs="0" name="return" type="tns:flight"/>
         </xs:sequence>
       </xs:complexType>
```

SOAP messages

- SOAP messages are typically sent over HTTP
  - Other protocols can also be used
  - SOAP request is sent as the body of an HTTP POST

– Are based on XML
  - Message request and response are XML files
  - The structure of the XML is validated against an XSD file

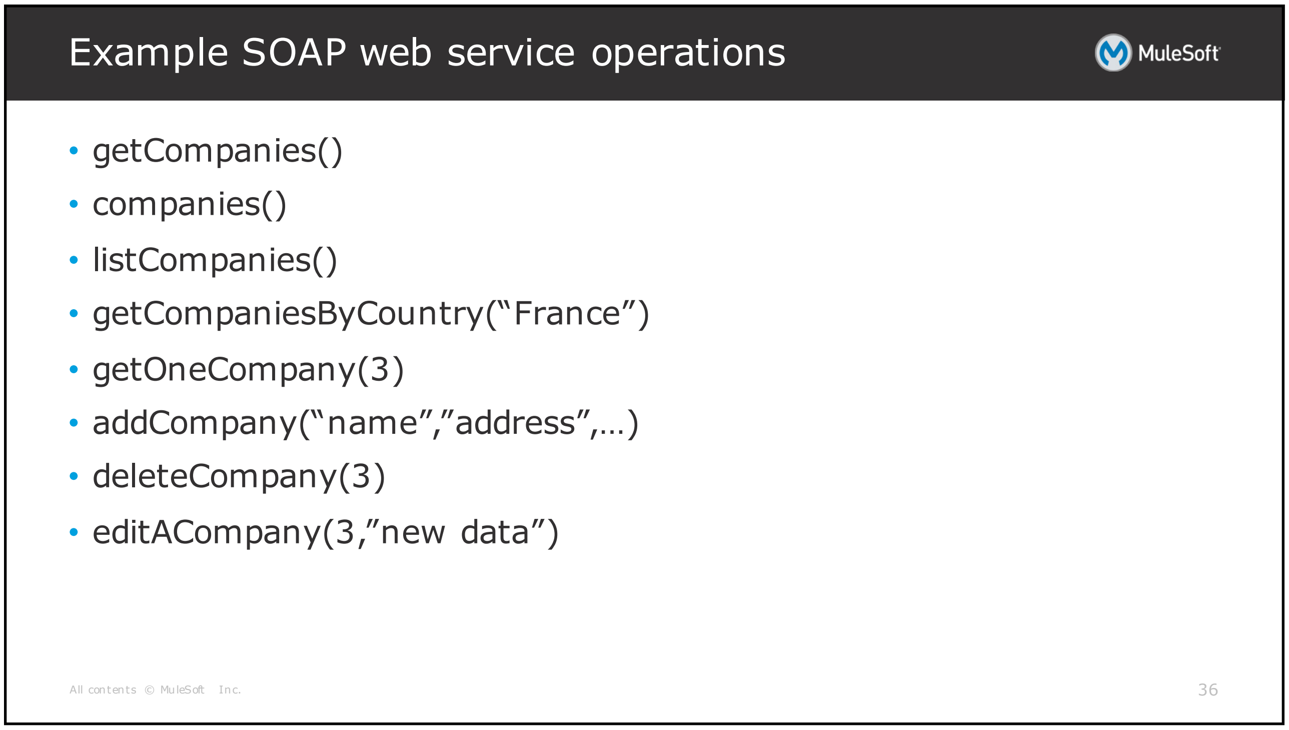
Example SOAP web service request and response


Request 1
http://training-u.cloudhub.io/essentials/delta
XML
Raw
XML
Raw
... Head... Attachm... ... W... JMS H... JMS Pro...
Headers (8) Attachments (0) SSL Info WSS (0) JMS (0)
3 : 18

```
<soapenv:Envelope xmlns:soapenv="http://sche
   <soapenv:Header/>
   <soapenv:Body>
      <soap:findFlight>
         <destination>SFO</destination>
      </soap:findFlight>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/
   <soap:Body>
      <ns2:findFlightResponse xmlns:ns2="http://soap.t
         <return>
            <airlineName>Delta</airlineName>
            <code>A1B2C3</code>
            <departureDate>2015/03/20</departureDate>
            <destination>SFO</destination>
            <emptySeats>40</emptySeats>
            <origin>MUA</origin>
            <planeType>Boing 737</planeType>
            <price>400.0</price>
         </return>
         <return>
            <airlineName>Delta</airlineName>
            <code>A1BTT4</code>
            <departureDate>2015/02/12</departureDate>
            <destination>SFO</destination>
            <emptySeats>30</emptySeats>
            <origin>MUA</origin>
            <planeType>Boing 777</planeType>
            <price>593.0</price>
         </return>
         <return>
            <airlineName>Delta</airlineName>
            <code>A14244</code>
            <departureDate>2015/02/12</departureDate>
            <destination>SFO</destination>
            <emptySeats>10</emptySeats>
            <origin>MUA</origin>
            <planeType>Boing 787</planeType>
            <price>294.0</price>
         </return>
      </ns2:findFlightResponse>
   </soap:Body>
</soap:Envelope>
```

response time: 261ms (919 bytes)

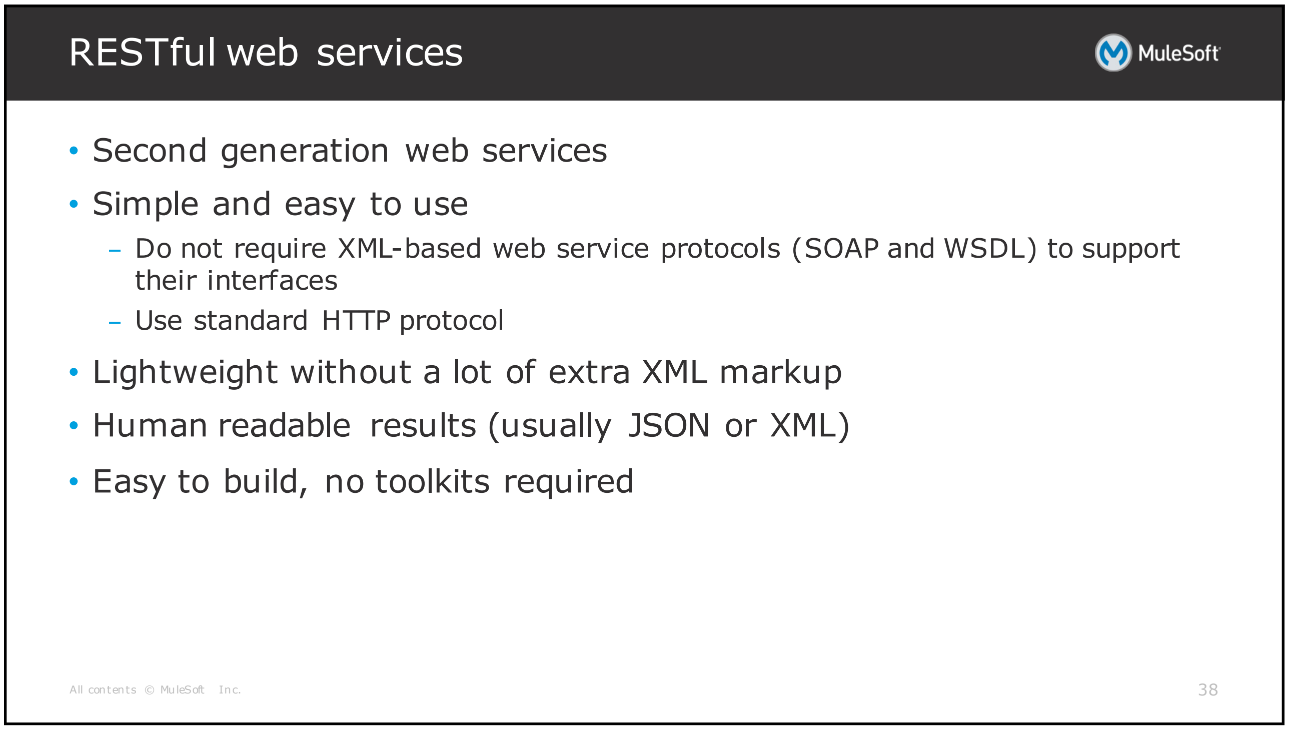


Example SOAP web service operations

MuleSoft

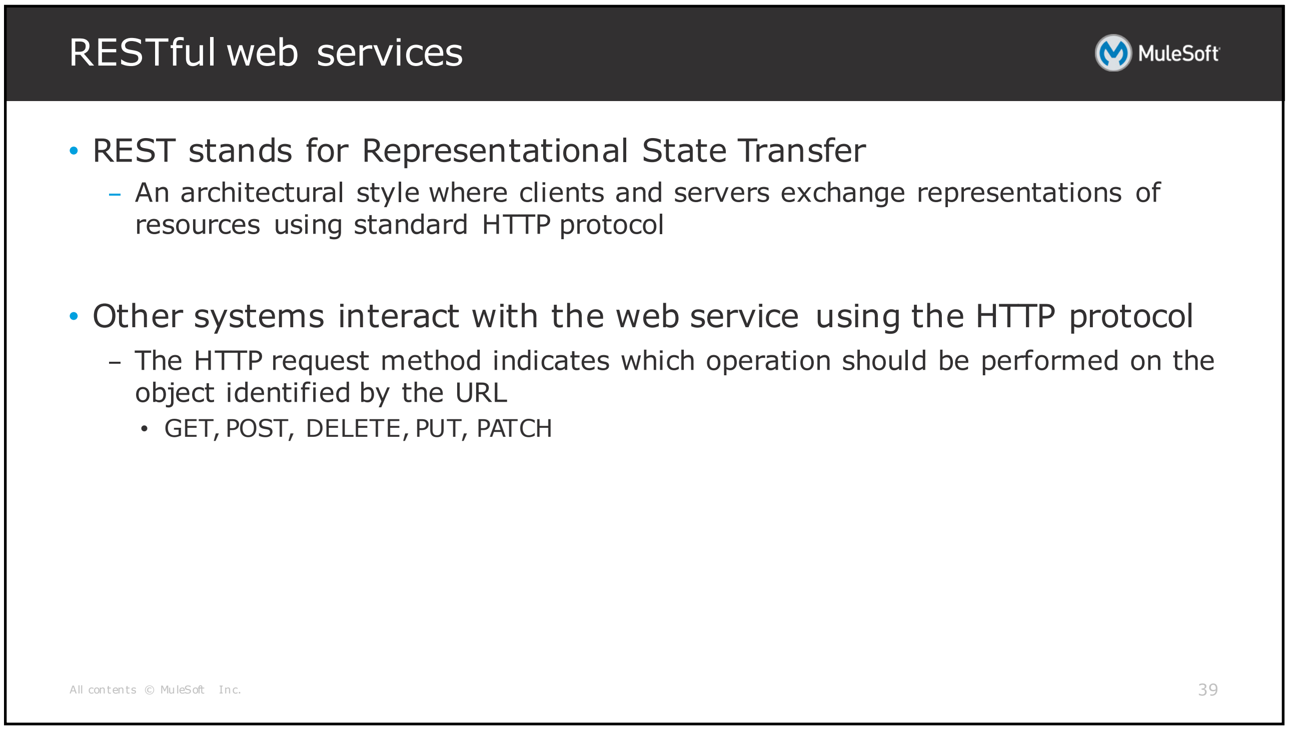
- getCompanies()
- companies()
- listCompanies()
- getCompaniesByCountry("France")
- getOneCompany(3)
- addCompany("name","address",...)
- deleteCompany(3)
- editACompany(3,"new data")

Introducing RESTful web services



RESTful web services

- Second generation web services
- Simple and easy to use
  - Do not require XML-based web service protocols (SOAP and WSDL) to support their interfaces
  - Use standard HTTP protocol
- Lightweight without a lot of extra XML markup
- Human readable results (usually JSON or XML)
- Easy to build, no toolkits required



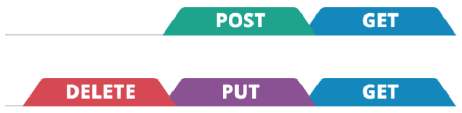
RESTful web services

- REST stands for Representational State Transfer
  - An architectural style where clients and servers exchange representations of resources using standard HTTP protocol
- 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
    - GET, POST, DELETE, PUT, PATCH

RESTful web service requests

MuleSoft

- Data and resources are represented using URIs
- Resources are accessed or changed using a fixed set of operations
  - **GET** retrieves the current state of a resource in some representation (usually JSON or XML)
  - **POST** creates a new resource
  - **DELETE** deletes a resource
  - **PUT** replaces a resource completely
    - If the resource doesn't exist, a new one is created
  - **PATCH** partially updates a resource
    - Just submitted data



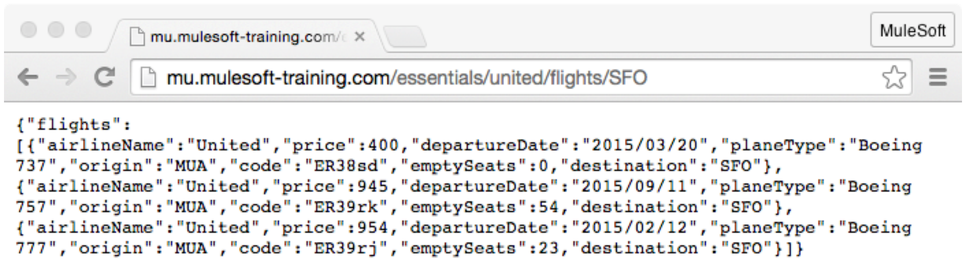
Example RESTful web service calls

MuleSoft

- (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

Example RESTful web service response

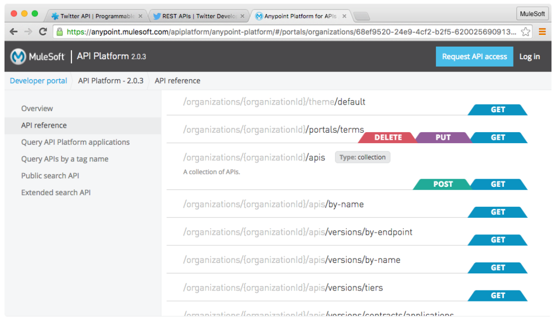
- JSON (JavaScript Object Notation)
  - A lightweight data-interchange format
  - Human-readable syntax
  - Supports collections and maps


mu.mulesoft-training.com/
MuleSoft
mu.mulesoft-training.com/essentials/united/flights/SFO

```
{"flights":
[{"airlineName":"United","price":400,"departureDate":"2015/03/20","planeType":"Boeing
737","origin":"MUA","code":"ER38sd","emptySeats":0,"destination":"SFO"},
{"airlineName":"United","price":945,"departureDate":"2015/09/11","planeType":"Boeing
757","origin":"MUA","code":"ER39rk","emptySeats":54,"destination":"SFO"},
{"airlineName":"United","price":954,"departureDate":"2015/02/12","planeType":"Boeing
777","origin":"MUA","code":"ER39rj","emptySeats":23,"destination":"SFO"}]}
```

Walkthrough 1-1: Explore API directories and portals

- Browse the ProgrammableWeb API directory
- Explore the MuleSoft Developer portal for popular APIs
- View an API definition file
- Explore the MuleSoft Developer portal for Anypoint Platform
- Use the API Console in an Anypoint Platform API Portal to make sample calls to an API

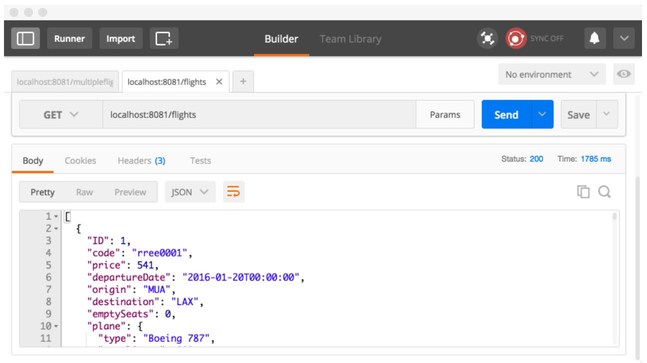

Twitter API | Programmable
REST APIs | Twitter Develo
Anypoint Platform for APIs
MuleSoft
https://anypoint.mulesoft.com/apiplatform/anypoint-platform/#/portals/organizations/68ef9520-24e9-4cf2-b2f5-620025690913...
MuleSoft | API Platform 2.0.3
Request API access
Log in
Developer portal
API Platform - 2.0.3
API reference
Overview
API reference
Query API Platform applications
Query APIs by a tag name
Public search API
Extended search API
/organizations/{organizationId}/theme/default
GET
/organizations/{organizationId}/portals/terms
DELETE
PUT
GET
/organizations/{organizationId}/apis
Type: collection
A collection of APIs.
POST
GET
/organizations/{organizationId}/apis/by-name
GET
/organizations/{organizationId}/apis/versions/by-endpoint
GET
/organizations/{organizationId}/apis/versions/by-name
GET
/organizations/{organizationId}/apis/versions/tiers
GET

Calling RESTful web services

Calling RESTful web services

MuleSoft

- To call web services, you need to write code or have a tool to make the HTTP requests
  - Need to be able to specify the HTTP method, request headers, and request body
- Some tools
  - A cURL command-line utility
  - Postman (for Google Chrome)
  - Advanced Rest Client (for Chrome)
  - More…



```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
```

Making calls to RESTful APIs

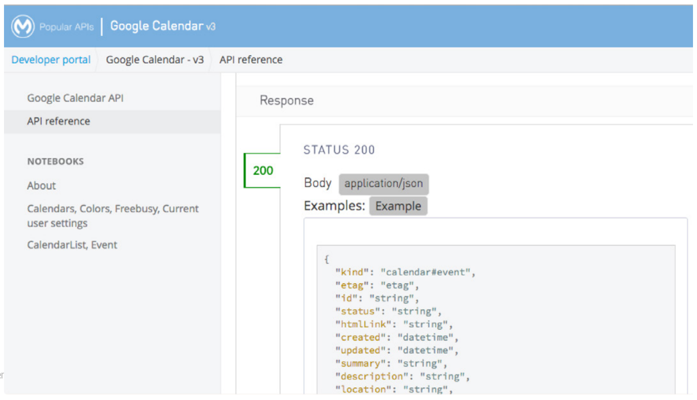
MuleSoft

- 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

Getting responses from web service calls

MuleSoft

- 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



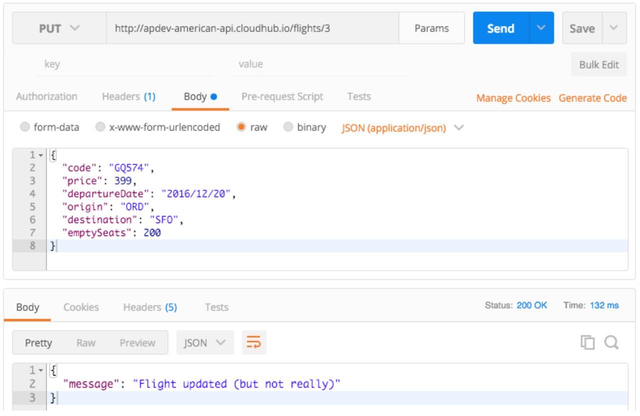
```
{
  "kind": "calendar#event",
  "etag": "etag",
  "id": "string",
  "status": "string",
  "htmlLink": "string",
  "created": "datetime",
  "updated": "datetime",
  "summary": "string",
  "description": "string",
  "location": "string",
```

Common HTTP status codes

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

Walkthrough 1-2: Make calls to an API

- Use Postman to make calls to an unsecured API (an implementation)
- Make GET, DELETE, POST, and PUT calls
- Use Postman to make calls to a secured API (an API proxy)

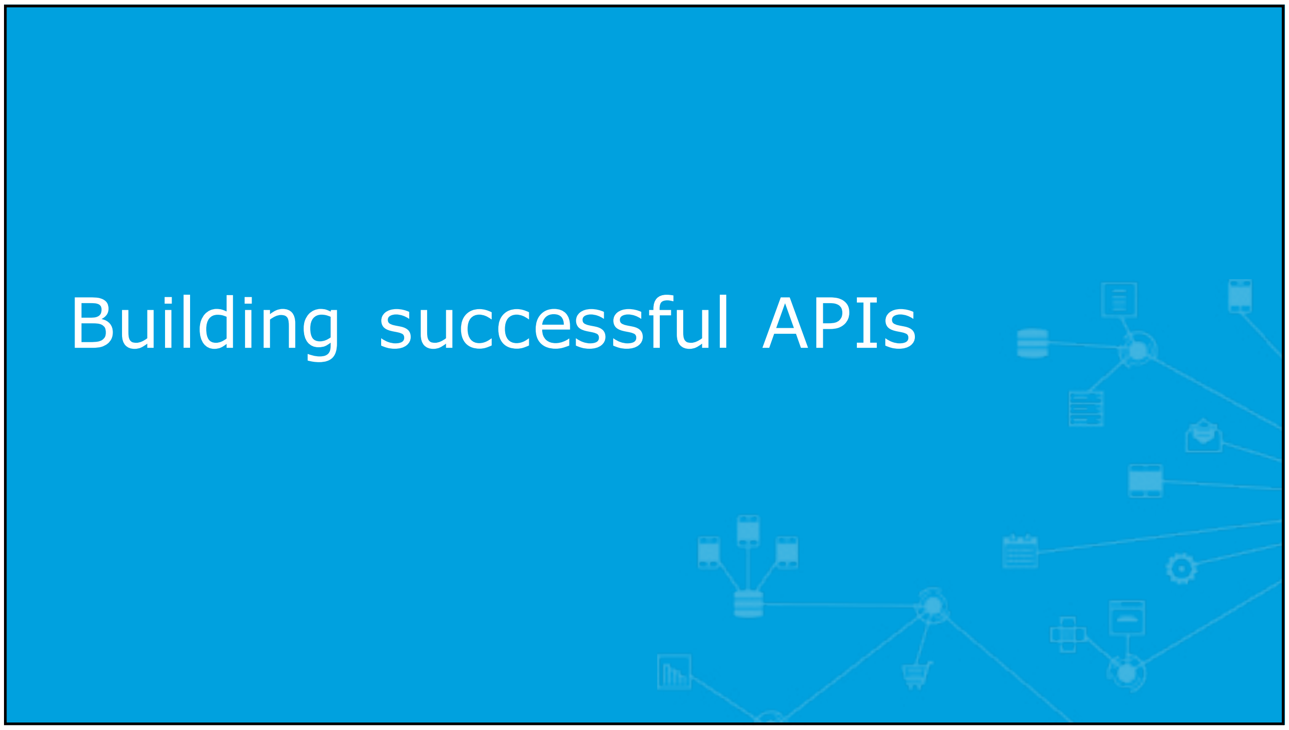


http://apdev-american-api.cloudhub.io/flights/3

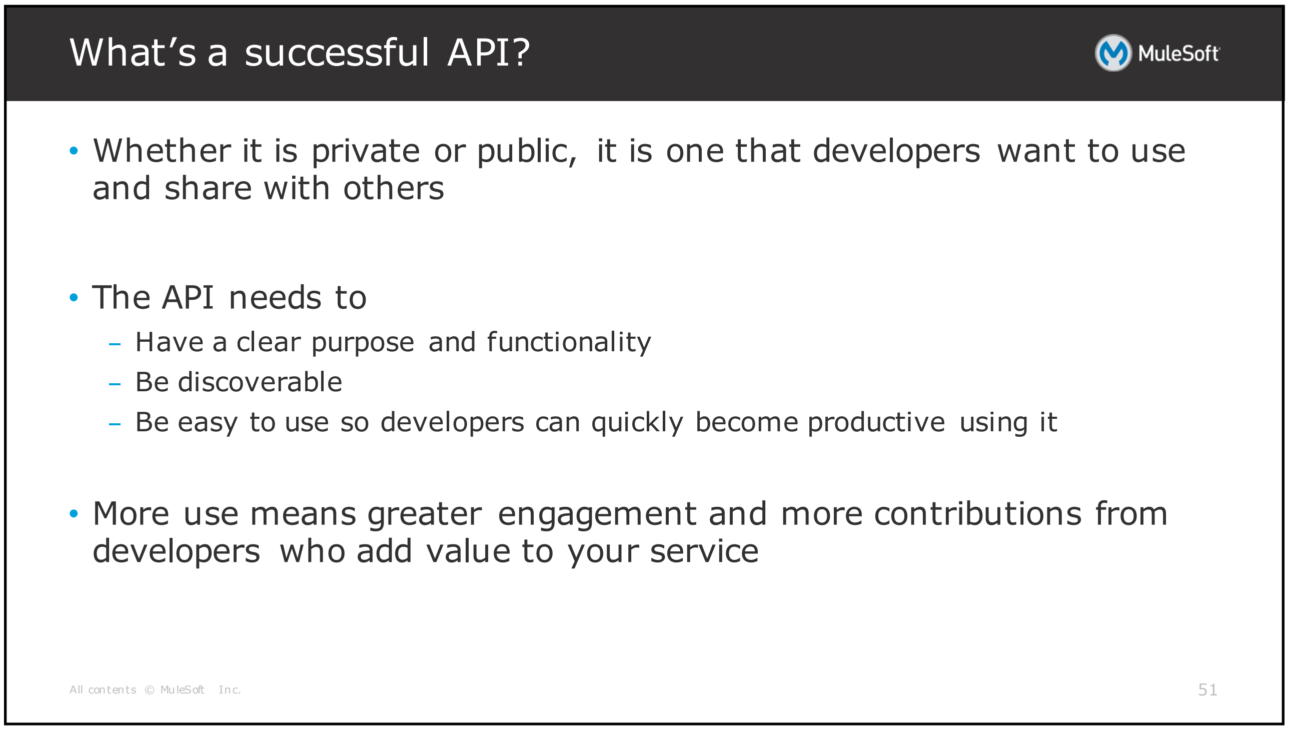
```
{
  "code": "GQ574",
  "price": 399,
  "departureDate": "2016/12/20",
  "origin": "ORD",
  "destination": "SFO",
  "emptySeats": 200
}
```

Status: 200 OK Time: 132 ms

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

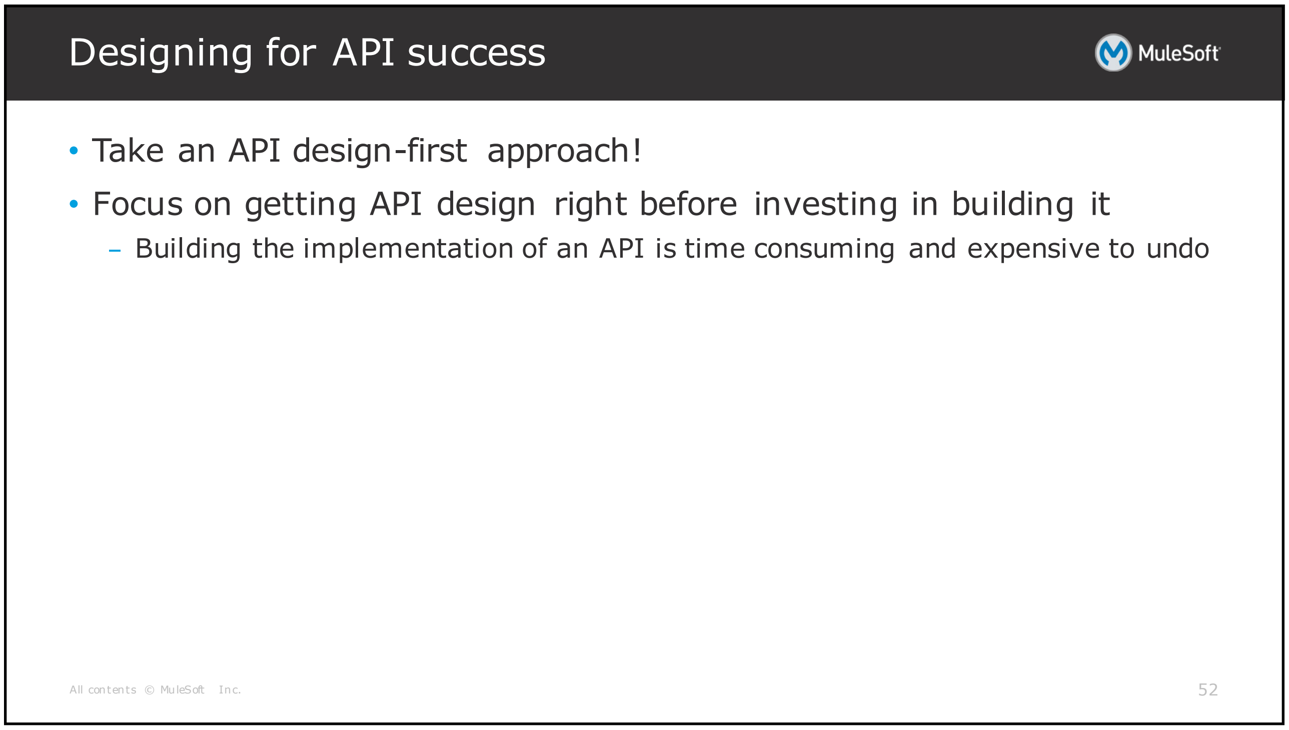


Building successful APIs



What’s a successful API?

- Whether it is private or public, it is one that developers want to use and share with others
- The API needs to
  - Have a clear purpose and functionality
  - Be discoverable
  - Be easy to use so developers can quickly become productive using it
- More use means greater engagement and more contributions from developers who add value to your service

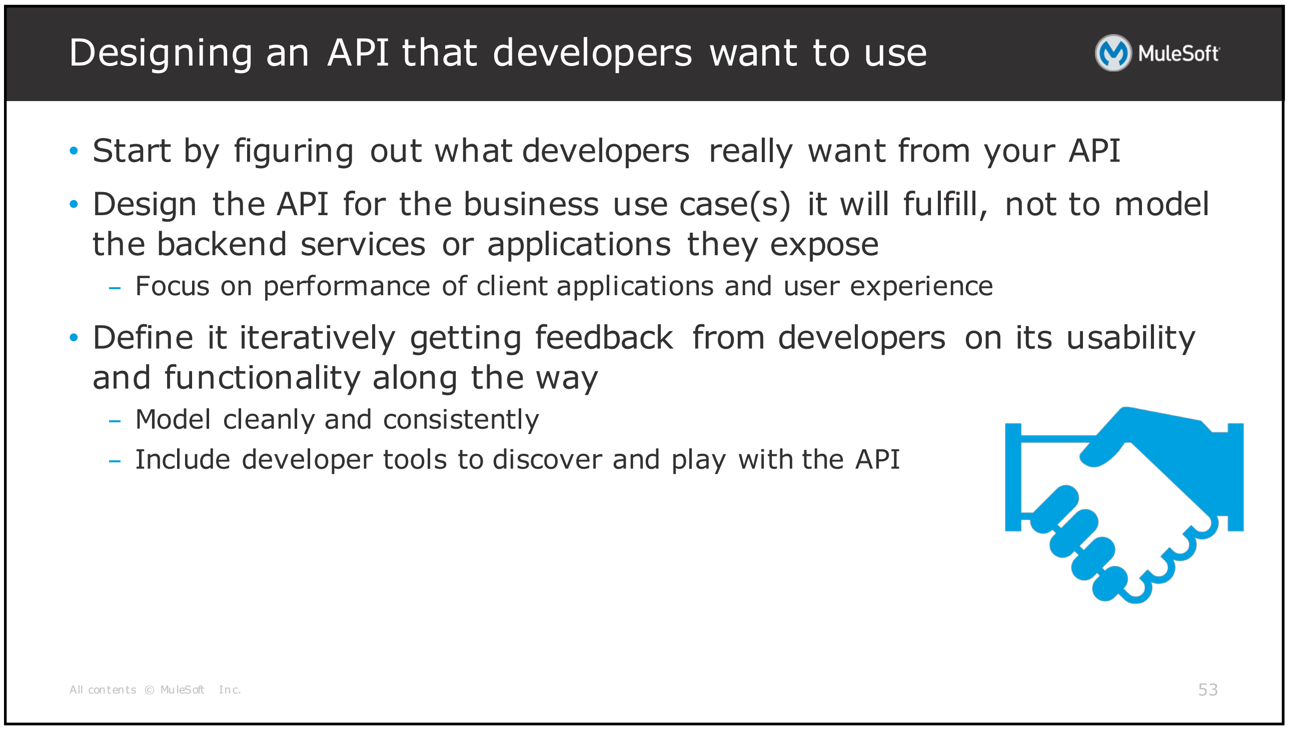


Designing for API success

MuleSoft

- Take an API design-first approach!
- Focus on getting API design right before investing in building it
  - Building the implementation of an API is time consuming and expensive to undo

52



Designing an API that developers want to use

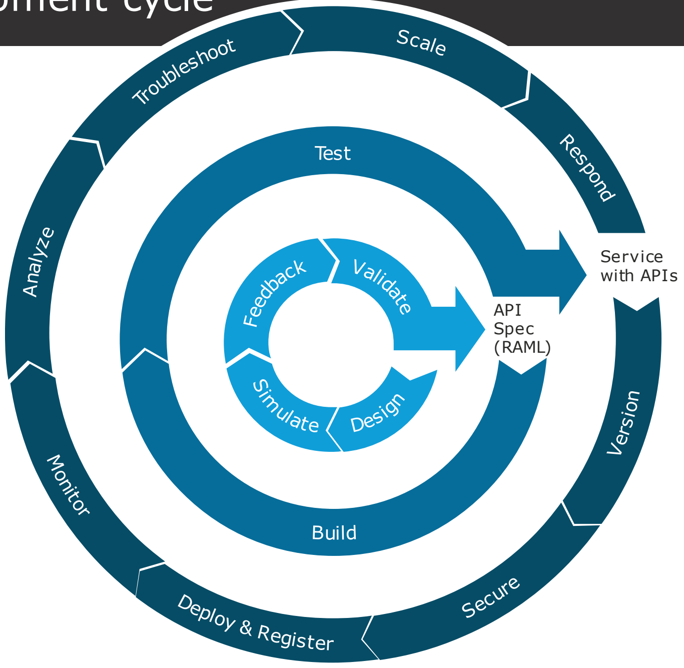
MuleSoft

- Start by figuring out what developers really want from your API
- Design the API for the business use case(s) it will fulfill, not to model the backend services or applications they expose
  - Focus on performance of client applications and user experience
- Define it iteratively getting feedback from developers on its usability and functionality along the way
  - Model cleanly and consistently
  - Include developer tools to discover and play with the API

53

API development cycle

MuleSoft



Introducing API-led connectivity with Anypoint Platform

Anypoint Platform

MuleSoft

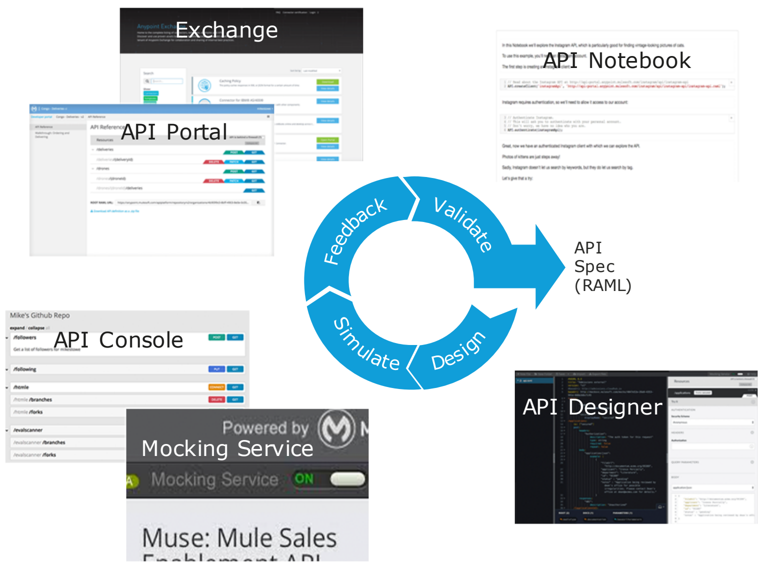
- A unified, highly productive, hybrid integration platform that creates a seamless application network of apps, data and devices with API-led connectivity

- A collection of runtimes, frameworks, tools, and web applications
  - Tools and frameworks for building applications
  - Mule runtime for running applications and applying policies
    - On-prem or in the cloud
  - Web application for
    - Deploying, running, managing, and monitoring applications
    - Defining, managing, and discovering APIs

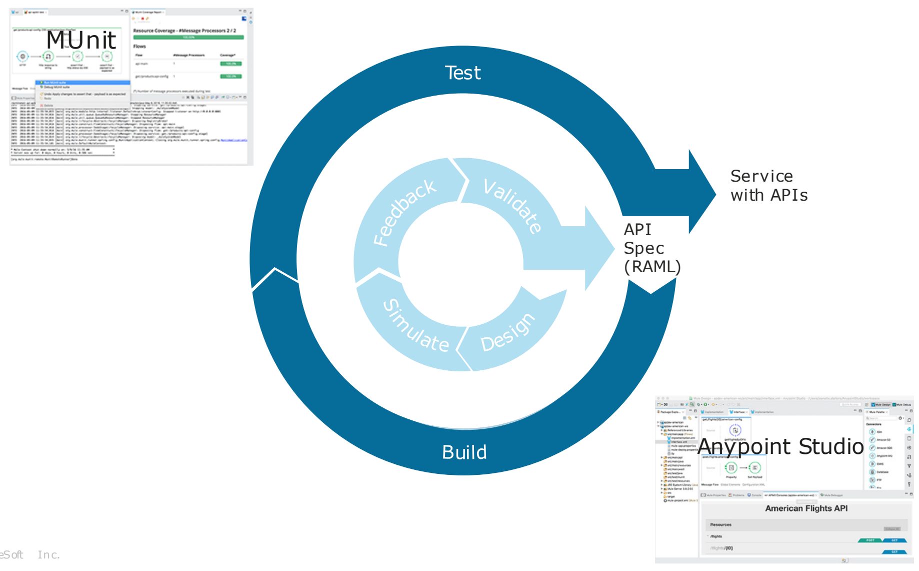
56

API development cycle: API definition

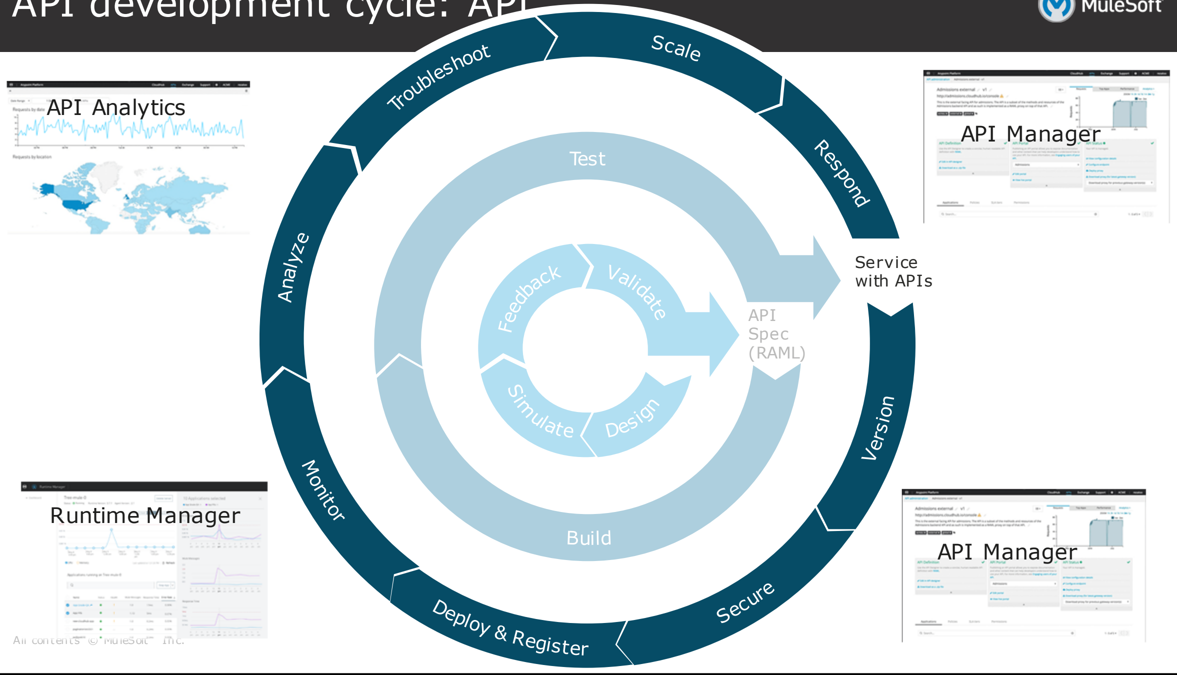
MuleSoft



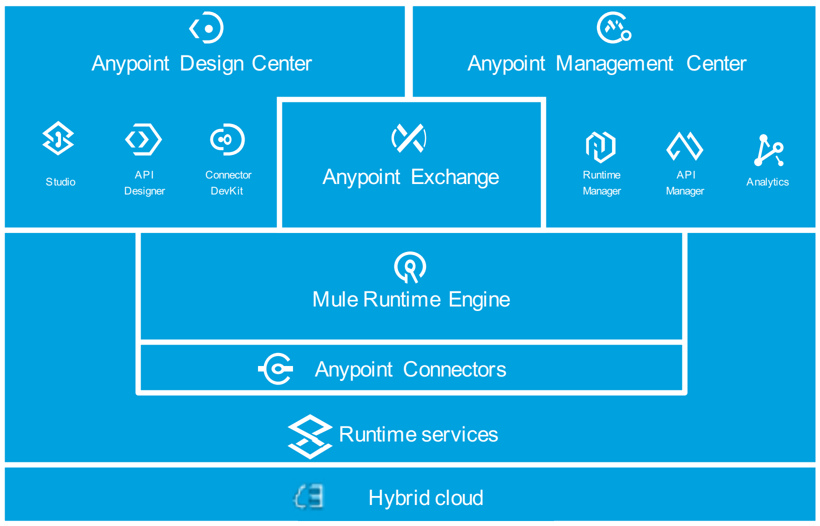
API development cycle: API implementation



API development cycle: API

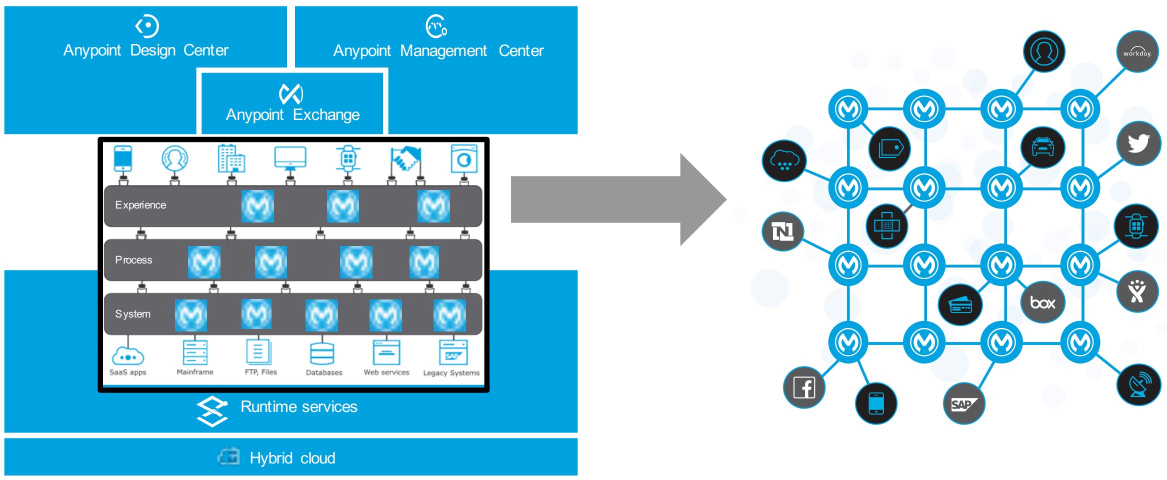


Anypoint Platform: The components



Uniquely enabling an application network

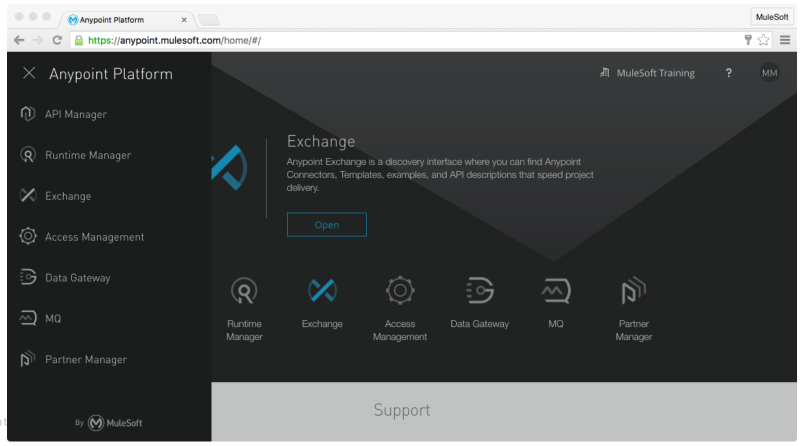
Anypoint Platform **Application Network**



Anypoint Platform: The web application

MuleSoft

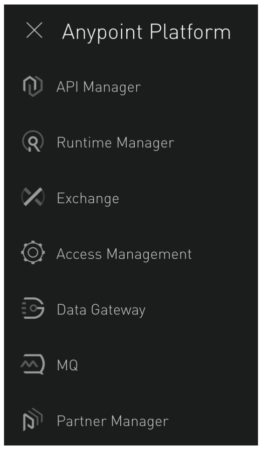
- Availability
  - In the cloud at http://anypoint.mulesoft.com
  - On-prem as part of Anypoint Platform On-Premises Edition



Core functionality of the web application

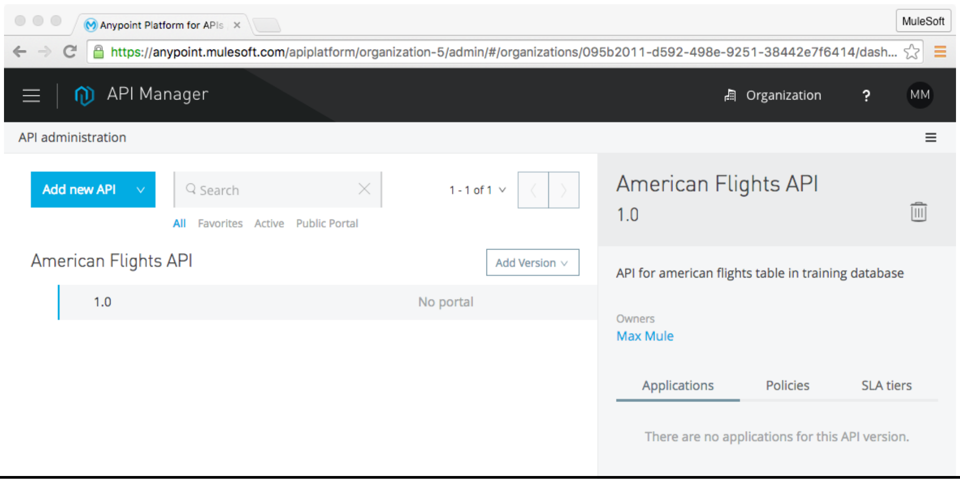
MuleSoft

- Designing APIs (API Designer in API Manager)
- Managing APIs (API Manager)
- Creating API portals (API Portal Designer in API Manager)
- Discovering and exploring APIs (API Portals)
- Testing and simulating APIs (API Console, mocking service, API Notebook)
- Deploying apps to the cloud or on-prem (Runtime Manager)
- Managing and monitoring applications (Runtime Manager)
- Sharing APIs, examples, connectors, and more (Exchange)
- Managing users (Access Management)



Walkthrough 1-3: Explore Anypoint Platform

- Explore Anypoint Platform
- Add an API to Anypoint Platform



American Flights API

1.0 No portal

American Flights API

1.0

API for american flights table in training database

Owners
Max Mule

There are no applications for this API version.

64

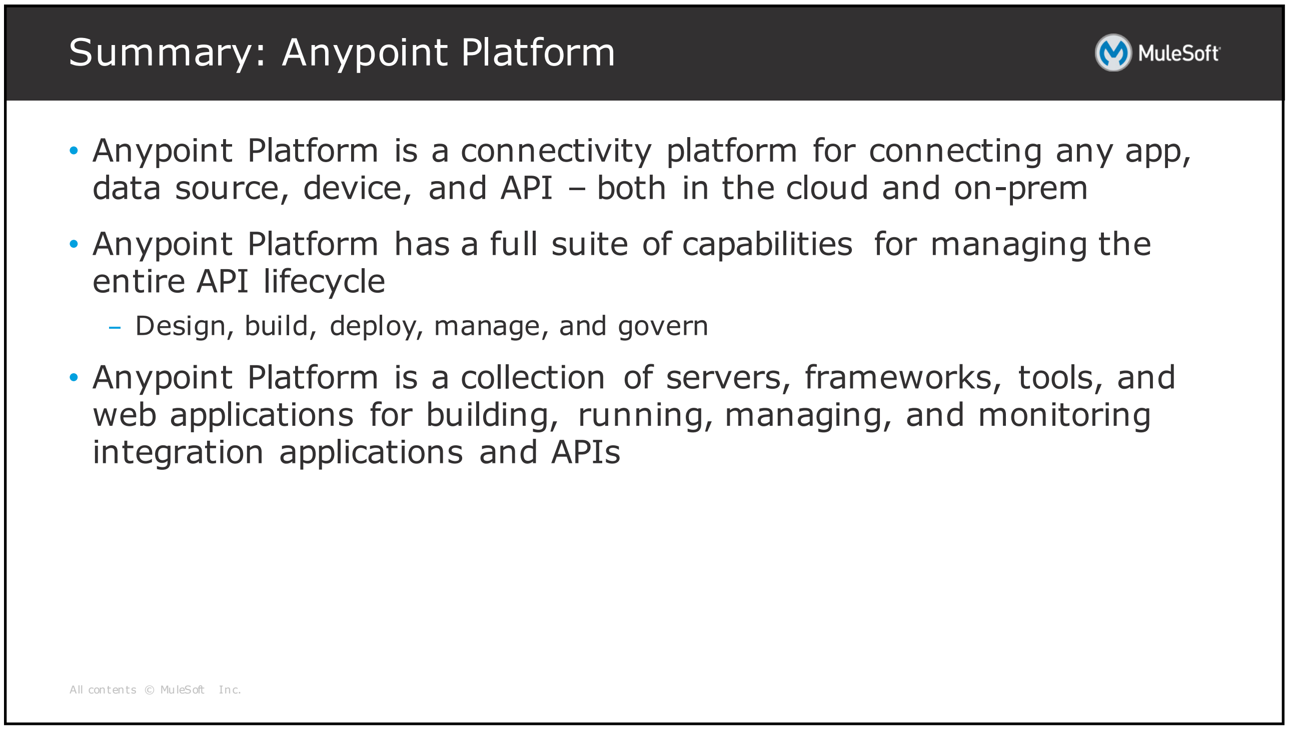
Summary

Summary: API-led connectivity

- Companies today need to rapidly adopt and develop new technologies in order to stay relevant to customers and keep competitive
  - SaaS, mobile, and the Internet of Things (IoT)
- IT needs to be able to rapidly integrate resources and make them available for consumption
- An API-led connectivity approach can help achieve this
  - Package underlying connectivity and orchestration services as easily discoverable and reusable building blocks
  - Expose them with APIs
  - Structure them across distinct systems, process and experience layers, to achieve both greater organizational agility and greater control

Summary: APIs and web services

- A web service is a method of communication that allows two software systems to exchange data over the internet
- An API is an application programming interface that provides info for how to communicate with a software component
- The term API is often used to refer to any part of RESTful web service
  - The web service API (definition file)
  - The web service interface implementing the API
  - The web service implementation itself
  - A proxy for the web service to control access to it
- RESTful web services use standard HTTP protocol and are easy to use
  - The HTTP request method indicates which operation should be performed on the object identified by the URL



Summary: Anypoint Platform

MuleSoft

- Anypoint Platform is a connectivity platform for connecting any app, data source, device, and API – both in the cloud and on-prem
- Anypoint Platform has a full suite of capabilities for managing the entire API lifecycle
  - Design, build, deploy, manage, and govern
- Anypoint Platform is a collection of servers, frameworks, tools, and web applications for building, running, managing, and monitoring integration applications and APIs