



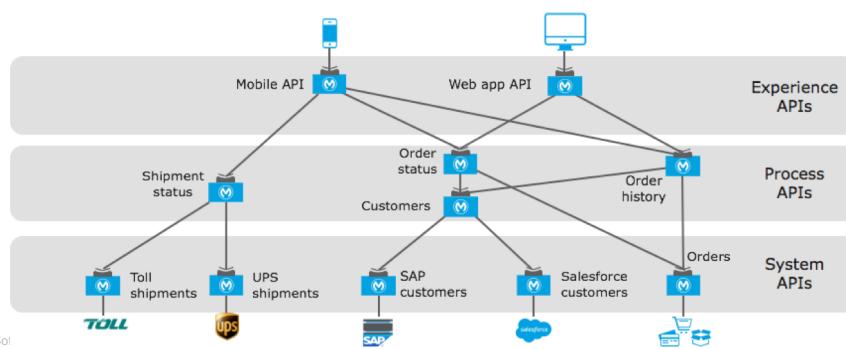
# Module 1: Introducing Application Networks and API-Led Connectivity



At the end of this module, you should be able



- Explain what an application network is and its benefits
- Describe how to build an application network using API-led connectivity
- Explain what web services and APIs are
- Make calls to unsecured and secure APIs

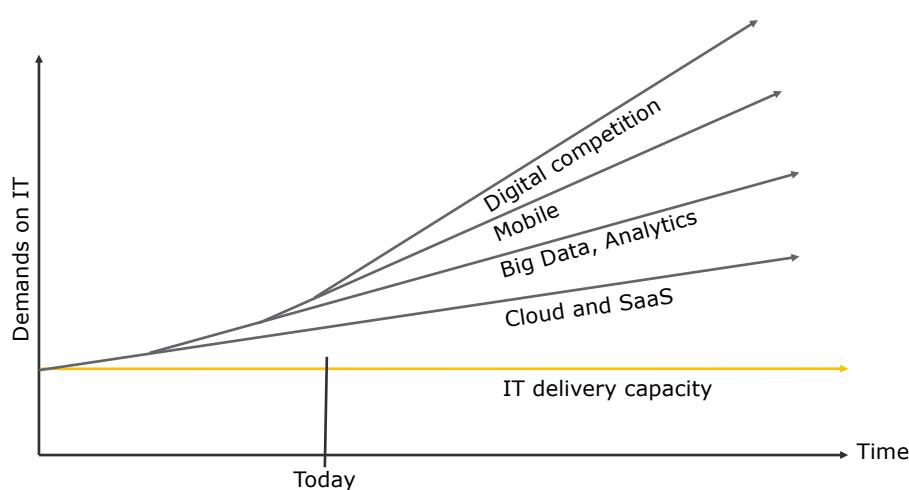


2

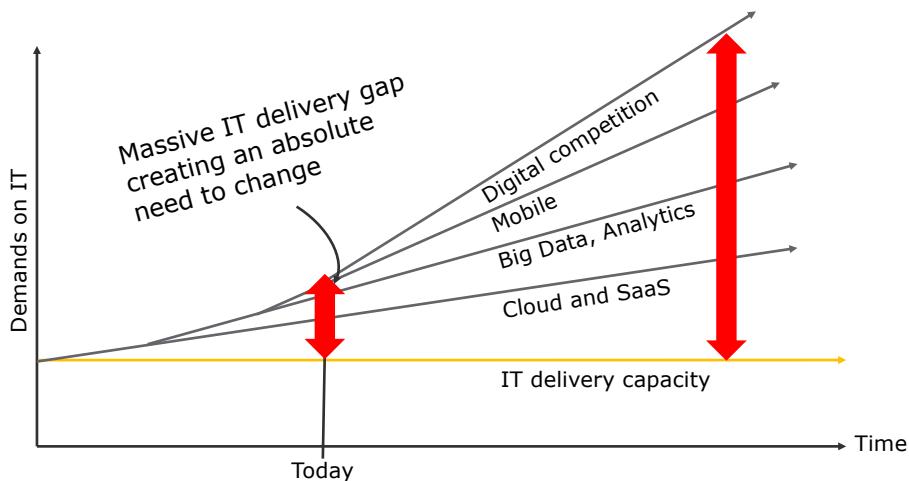
# Identifying the problems faced by IT today



Biggest challenge: IT cannot go fast enough



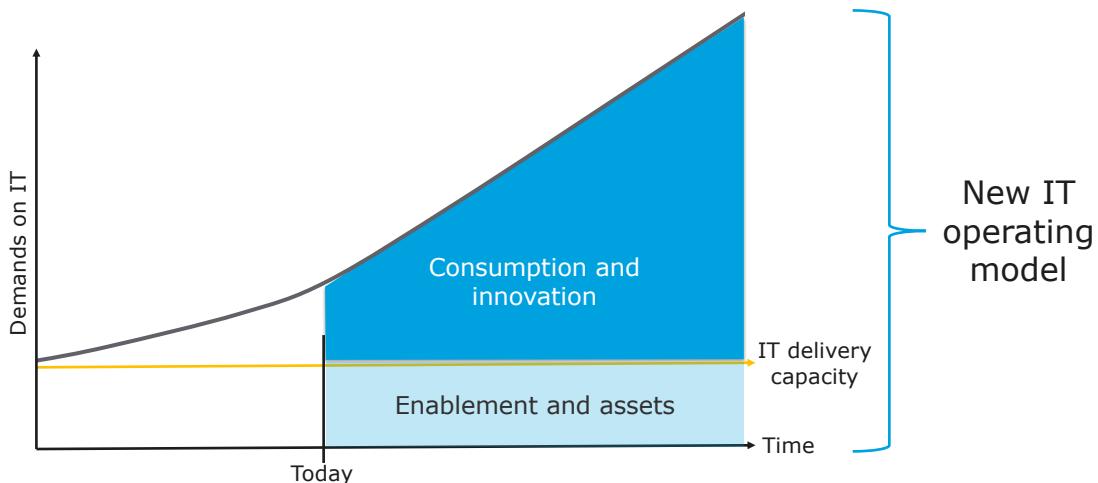
## Digital pressures create a widening IT delivery gap



All contents © MuleSoft Inc.

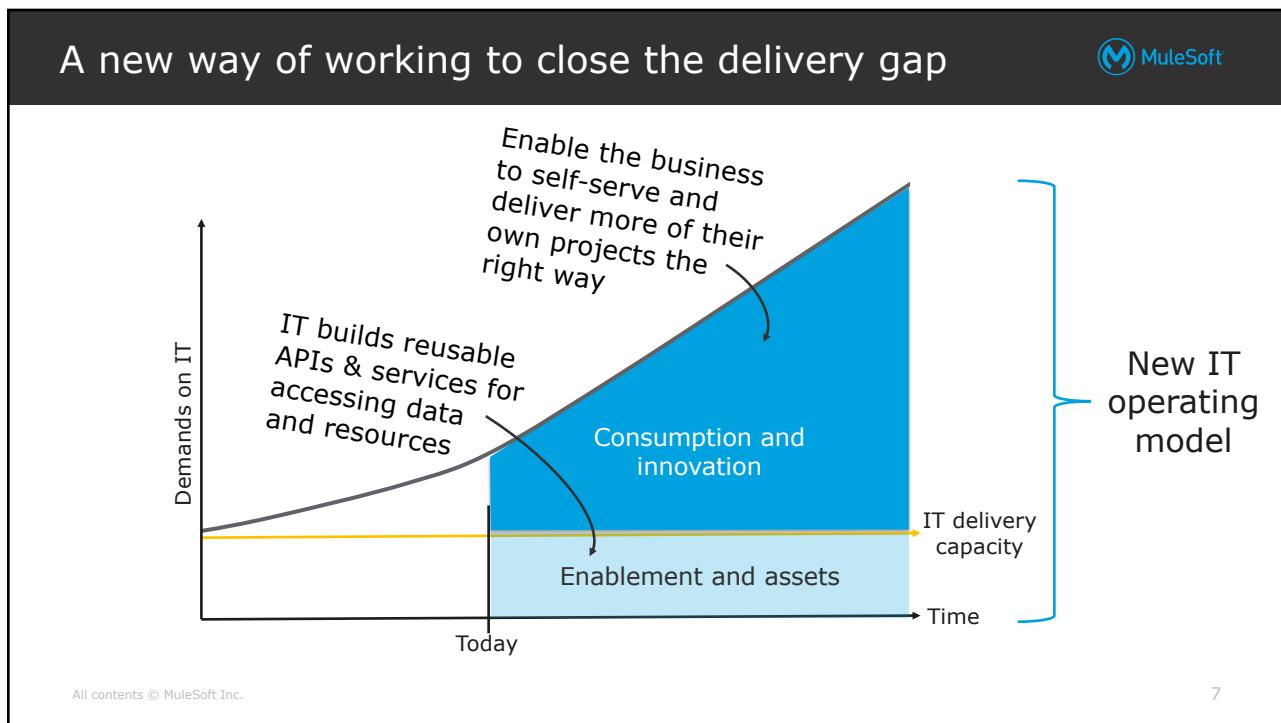
5

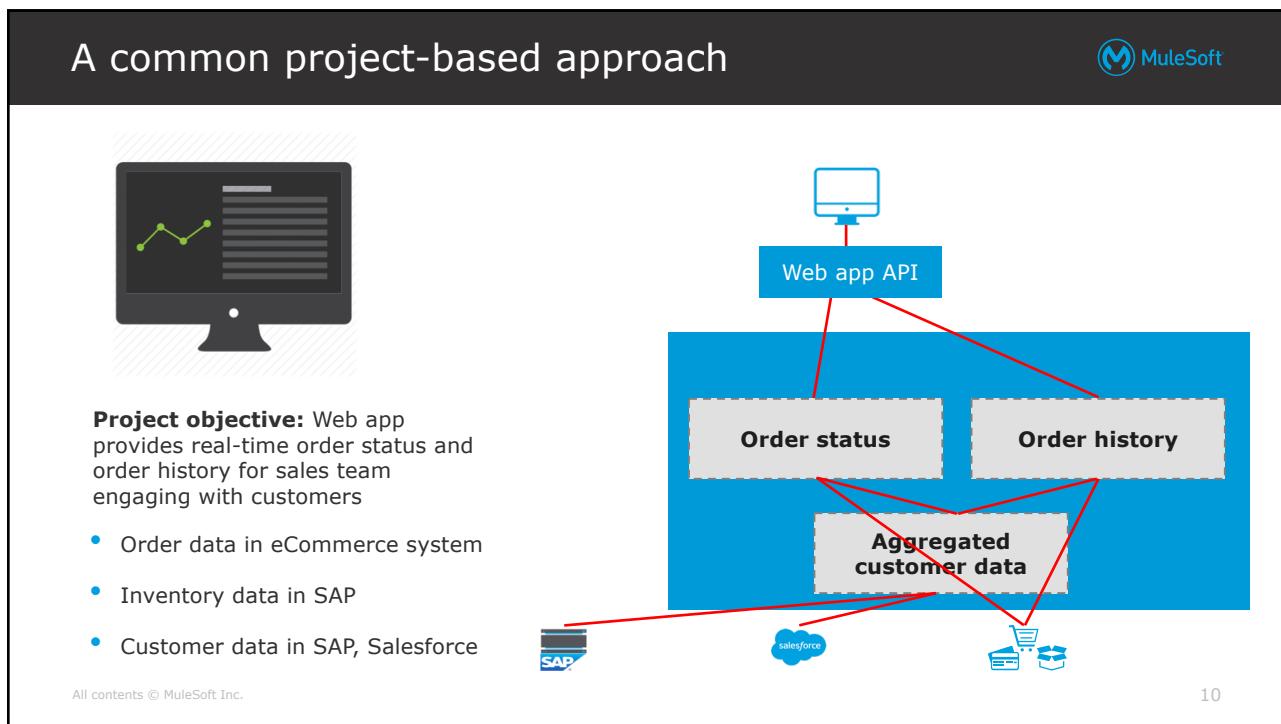
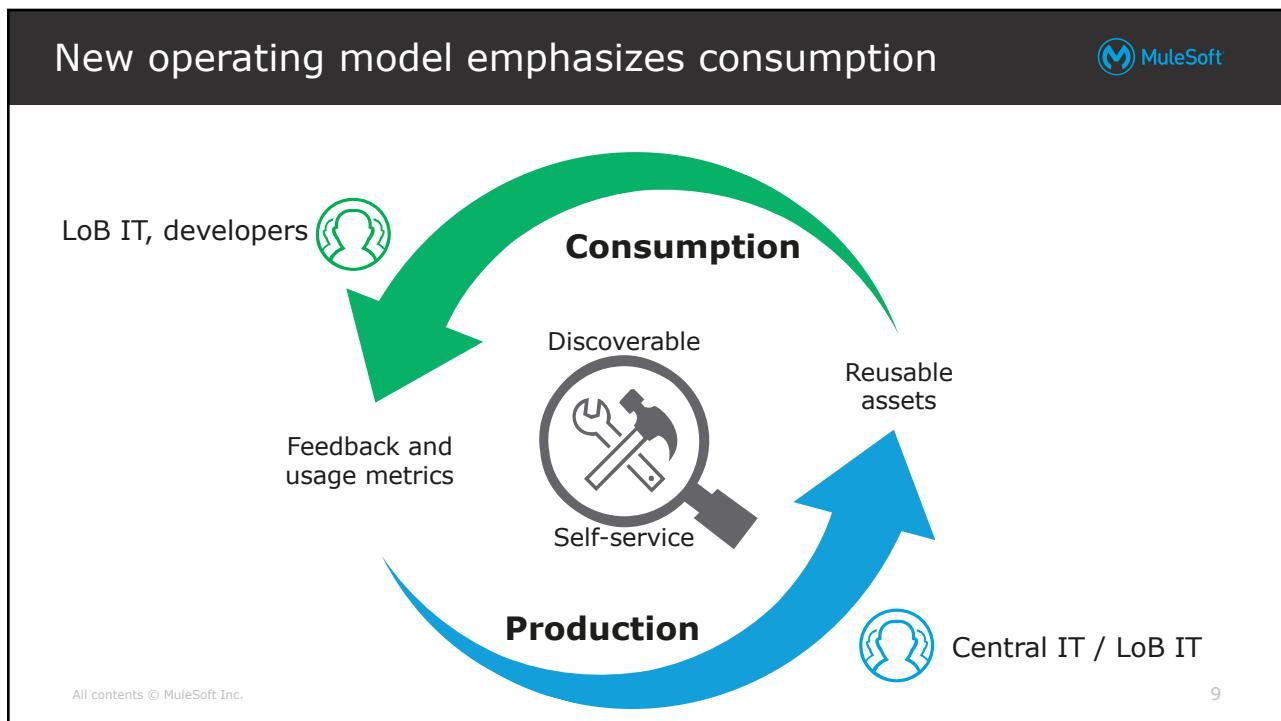
## A new way of working to close the delivery gap



All contents © MuleSoft Inc.

6

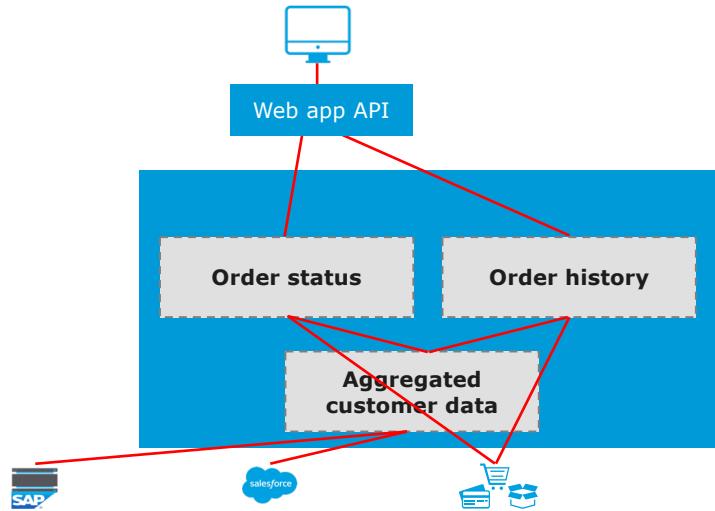




## A common project-based approach



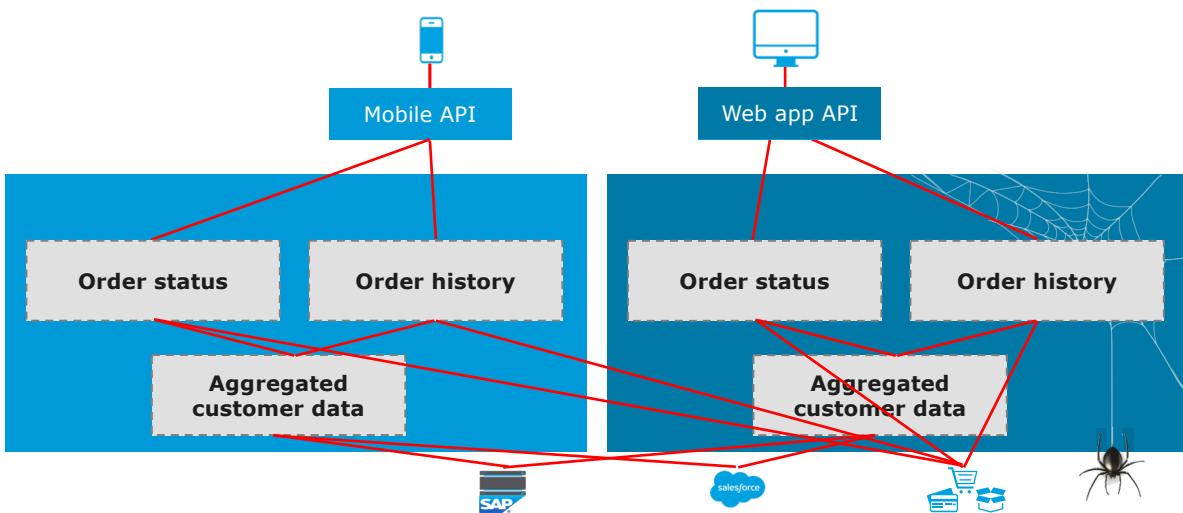
- ✓ On time and within budget
- ✗ Limited opportunity for reuse
- ✗ Tight coupling = brittleness
- ✗ Difficult to govern
- ? Meets business requirements



All contents © MuleSoft Inc.

11

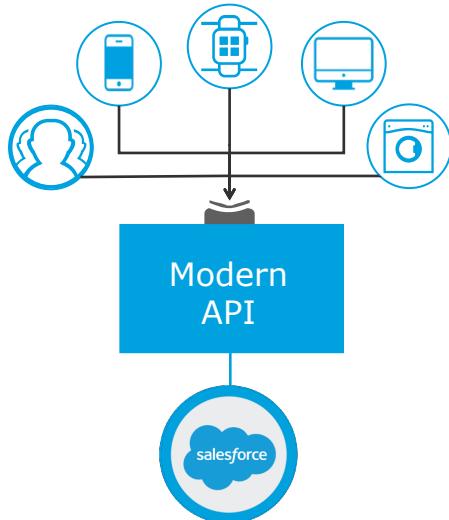
## 6 months later...



All contents © MuleSoft Inc.

12

## Modern API: The core enabler of a new operating model

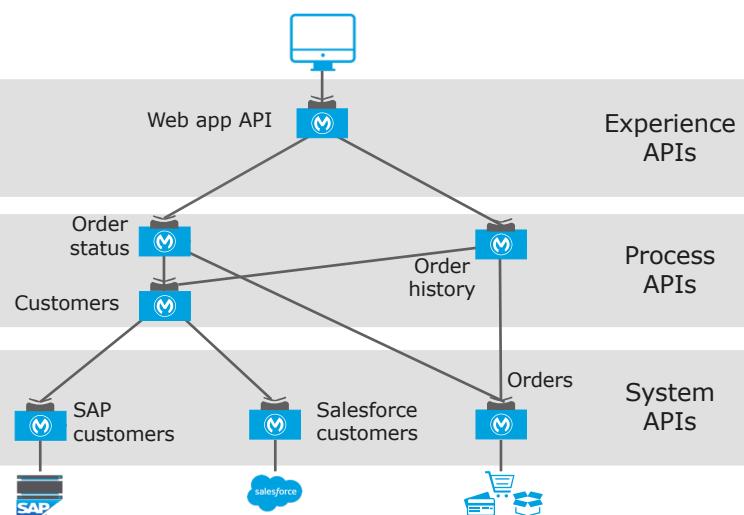


- Discoverable and accessible through self-service
- Productized and designed for ease of consumption
- Easily managed for security, scalability, and performance

All contents © MuleSoft Inc.

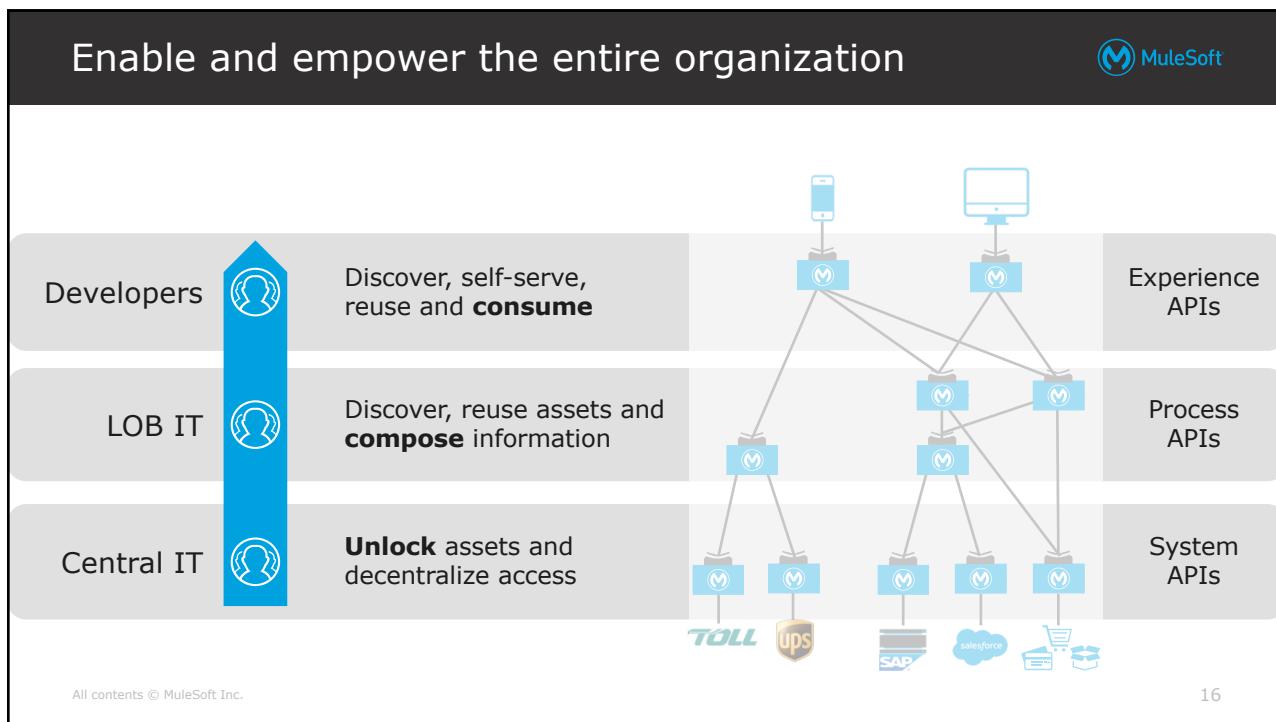
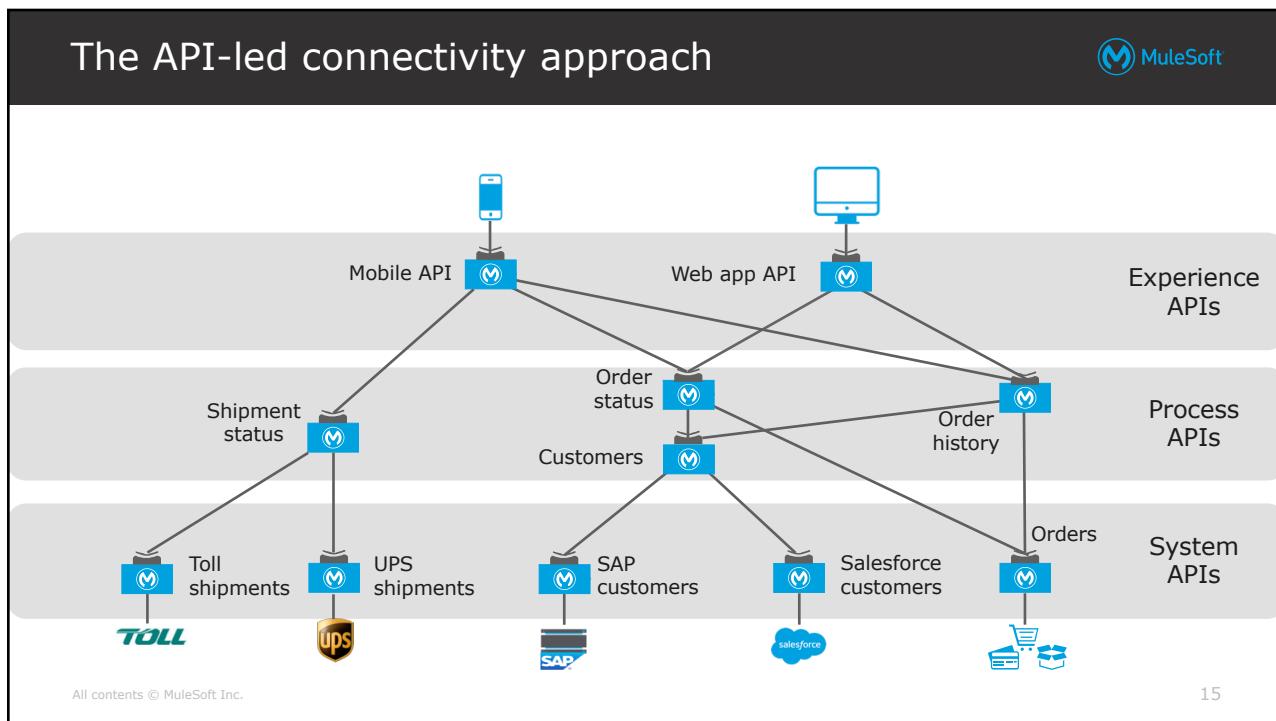
13

## The API-led connectivity approach



All contents © MuleSoft Inc.

14



## Drive outcomes with API-led connectivity

MuleSoft

The diagram shows a central cloud icon with a 'M' logo connected to various devices (mobile phone, laptop, tablet) and other clouds. These are categorized into three levels:

- Experience APIs**: Top level, connected to a mobile phone and a laptop.
- Process APIs**: Middle level, connected to a tablet and two clouds.
- System APIs**: Bottom level, connected to a mobile phone, a cloud, and several business logos: TOLL, UPS, SAP, salesforce, and a shopping cart icon.

**Outcomes:**

- On time and within budget
- Drives reuse vs build
- Designs in readiness for change
- Builds in governance, compliance, security, and scalability
- Meets the needs of your business

All contents © MuleSoft Inc.

17

## C4E: Organizing differently to drive API-led connectivity

MuleSoft

The diagram illustrates the Center for Enablement (C4E) as a cross-functional team. It is represented by a central triangle connecting three main components:

- Central IT**: Represented by a building icon.
- Innovation teams**: Represented by a lightbulb icon.
- LoB** (Line of Business): Represented by a bar chart icon.

**Characteristics of C4E:**

- 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

All contents © MuleSoft Inc.

18

# Achieving an application network



Application landscape



Every project adds value to the application network

**MuleSoft**

**Project 1 – API-led**

All contents © MuleSoft Inc.

21

Every project adds value to the application network

**MuleSoft**

**Project 1 – API-led**

**C4E**

Self-serve assets on the application network

- Order status
- Shipments
- Orders
- Shipment status
- Customers

All contents © MuleSoft Inc.

22

Every project adds value to the application network

MuleSoft

**Project 1 – API-led**

All contents © MuleSoft Inc.

**C4E**

Self-serve assets on the application network

Order status	Shipments	Orders
Shipment status	Customers	

23

Speed. Agility. Innovation.

MuleSoft

**An application network**

- Emerges bottoms-up via self-service
- Provides visibility, security and governability at every API node
- Is recomposable: it bends, not breaks – **built for change**

All contents © MuleSoft Inc.

24

# Deconstructing APIs



## What exactly is an API?



- An **API** is an **Application Programming Interface**
- 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

## What do people mean when they say API?



They could be referring to a number of things...

### 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

All contents © MuleSoft Inc.

27

# Reviewing web services



## What is a web service?



- 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

## The parts of a web service



- **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



- 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

All contents © MuleSoft Inc.

31

## Reviewing RESTful web services



## 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



All contents © MuleSoft Inc.

33

## Example RESTful web service calls



- 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

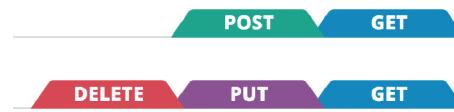
All contents © MuleSoft Inc.

34

## RESTful web service request methods



- **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



All contents © MuleSoft Inc.

35

## Example RESTful web service response



- **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

The screenshot shows a browser window with the URL [mu.learn.mulesoft.com/united/flights/SFO](http://mu.learn.mulesoft.com/united/flights/SFO). The page displays a JSON object representing flight information:

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

## Learning about APIs



- API documentation
  - Should include the list of all possible resources, how to get access to the API, and more
- API portals
  - Accelerate onboarding by providing developers a centralized place for discovering all the tools they need to successfully use the API, which could include
    - Documentation, tutorials, code snippets, and examples
    - A way to register applications to get access to the API
    - A way to provide feedback and make requests
    - A way to test the API by making calls to it
- Discover APIs in API directories and marketplaces
  - For example, **ProgrammableWeb**, which has over 19,000 APIs

All contents © MuleSoft Inc.

33

## Walkthrough 1-1: Explore an API directory and an API portal



- Browse the ProgrammableWeb API directory
- Explore the API reference for an API (like Twitter)
- Explore the API portal for an API to be used in the course

The image shows two web pages side-by-side. On the left is the ProgrammableWeb API directory, featuring a search bar for 'Search the Largest API Directory on the Web' and a table of APIs like Google Maps and Twitter. On the right is the MuleSoft API portal, showing the 'American Flights API | v2' and its endpoint details: `/[ID] : get`. It includes sections for 'Request' (with a 'GET /flights/[ID]' button), 'Parameters' (with a table for 'Parameter', 'Type', and 'Description'), and 'API instances' (with a table for 'ID', 'Type', and 'Description').

All contents © MuleSoft Inc.

38

# Calling RESTful web services



## Calling RESTful web services



- 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
- Example tools include
  - An API portal with an API console
  - Advanced Rest Client
  - Postman
  - A cURL command-line utility

The screenshot shows the Advanced REST client interface. The request URL is `http://training4-american-api.cloudhub.io/flights`. The response is a 200 OK status with a response time of 1290.08 ms. The JSON response data is:

```

{
  "ID": 1,
  "code": "rrree0001",
  "price": 541,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MIA",
  "destination": "LAX"
}

```

All contents © MuleSoft Inc.

## 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

All contents © MuleSoft Inc.

41

## 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

Response

200	Type application/atom+xml	Schema	Examples
401			<input checked="" type="checkbox"/> <pre>&lt;?xml version='1.0' encoding='UTF-8'?&gt; &lt;feed xmlns='http://www.w3.org/2005/Atom'       xmlns:openSearch='http://a9.com/-/spec/opensearch/1.1/'       xmlns:gContact='http://schemas.google.com/contact/2008'       xmlns:batch='http://schemas.google.com/gdata/batch'       xmlns:gd='http://schemas.google.com/g/2005'       gd:etag='feedEtag'&gt;</pre>

All contents © MuleSoft Inc.

42

## 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

All contents © MuleSoft Inc.

43

## Walkthrough 1-2: Make calls to an API



- Use ARC to make calls to an unsecured API (an implementation)
- Make GET, DELETE, POST, and PUT calls
- Use ARC to make calls to a secured API (an API proxy)
- Use the API console in an API portal to make calls to a managed API using a mocking service
- Use the API console to make calls to an API proxy endpoint

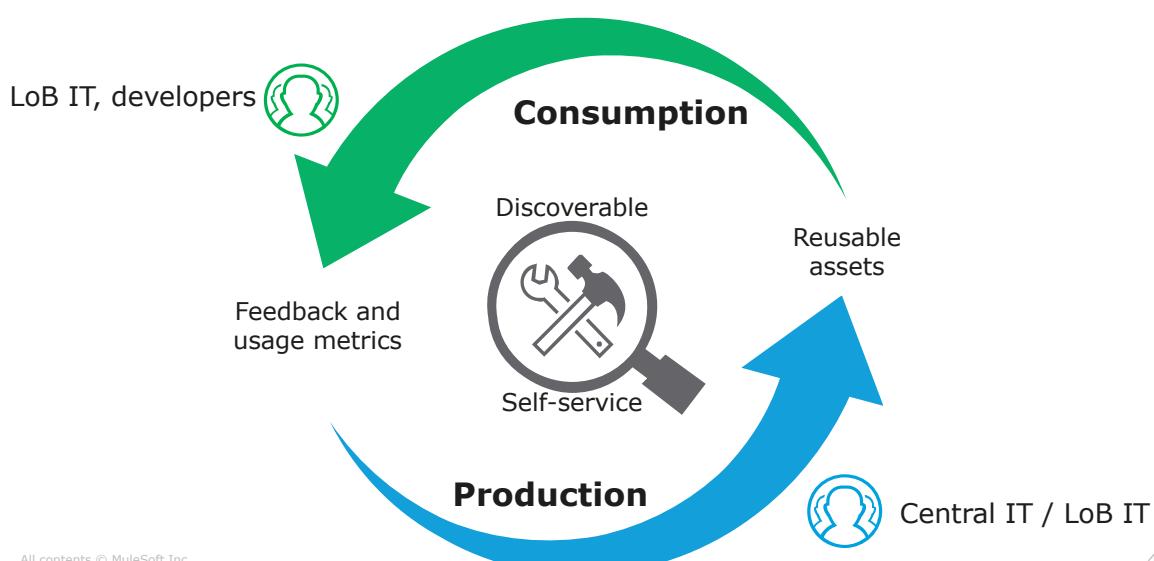
The image displays two side-by-side screenshots illustrating API interaction. On the left, the 'Advanced REST client' interface is shown with a successful '200 OK' response for a GET request to the endpoint 'http://training4-american-api.cloudhub.io/flights'. On the right, the 'American Flights API | v2' page from the MuleSoft API portal is shown, featuring a 'GET /flights : get' operation with its details and parameters.

44

# Successfully creating application networks using API-led connectivity



Producing discoverable and consumable assets is key



## Designing for API success



- Create APIs that developers **can find** and **want to use** and share with others
  - Design the API for the business use cases it will fulfill, not to model the backend services or applications they expose
  - Focus on performance of client applications and user experience
- Take an **API design-first approach!**
- **Get API design right** before investing in building it
  - Define it iteratively getting feedback from developers on its usability and functionality along the way
  - Building the implementation of an API is time consuming and expensive to undo

All contents © MuleSoft Inc.

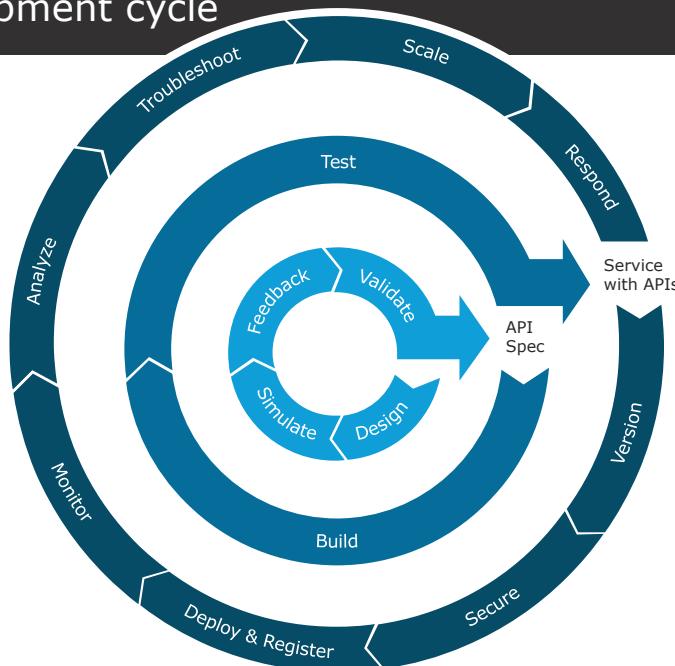
47

## API development cycle

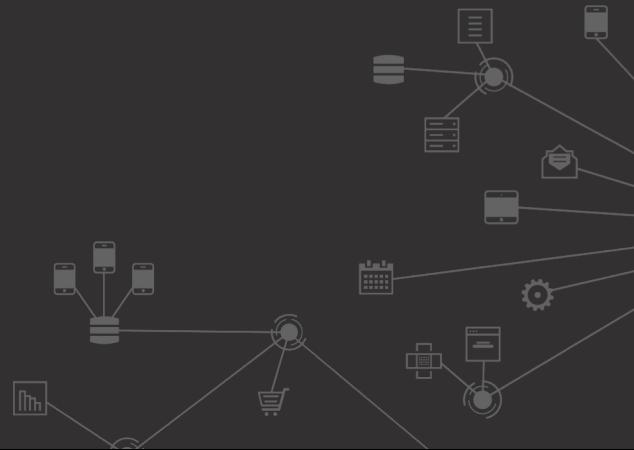


All contents © MuleSoft Inc.

48



# Summary



## Summary



- Companies today need to **rapidly adopt and develop** new technologies in order to stay relevant to customers & keep competitive
- IT needs to be able to rapidly integrate resources and make them **available for consumption**
  - An **API-led connectivity** approach can help achieve this
- To drive API-led connectivity, create a **C4E** (Center for Enablement)
  - A cross-functional team to ensure assets across the organization are productized, published, and widely consumed
- **An application network** is a network of applications, data, and devices connected with APIs to make them pluggable and to create reusable services

## Summary



- 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 or specification 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