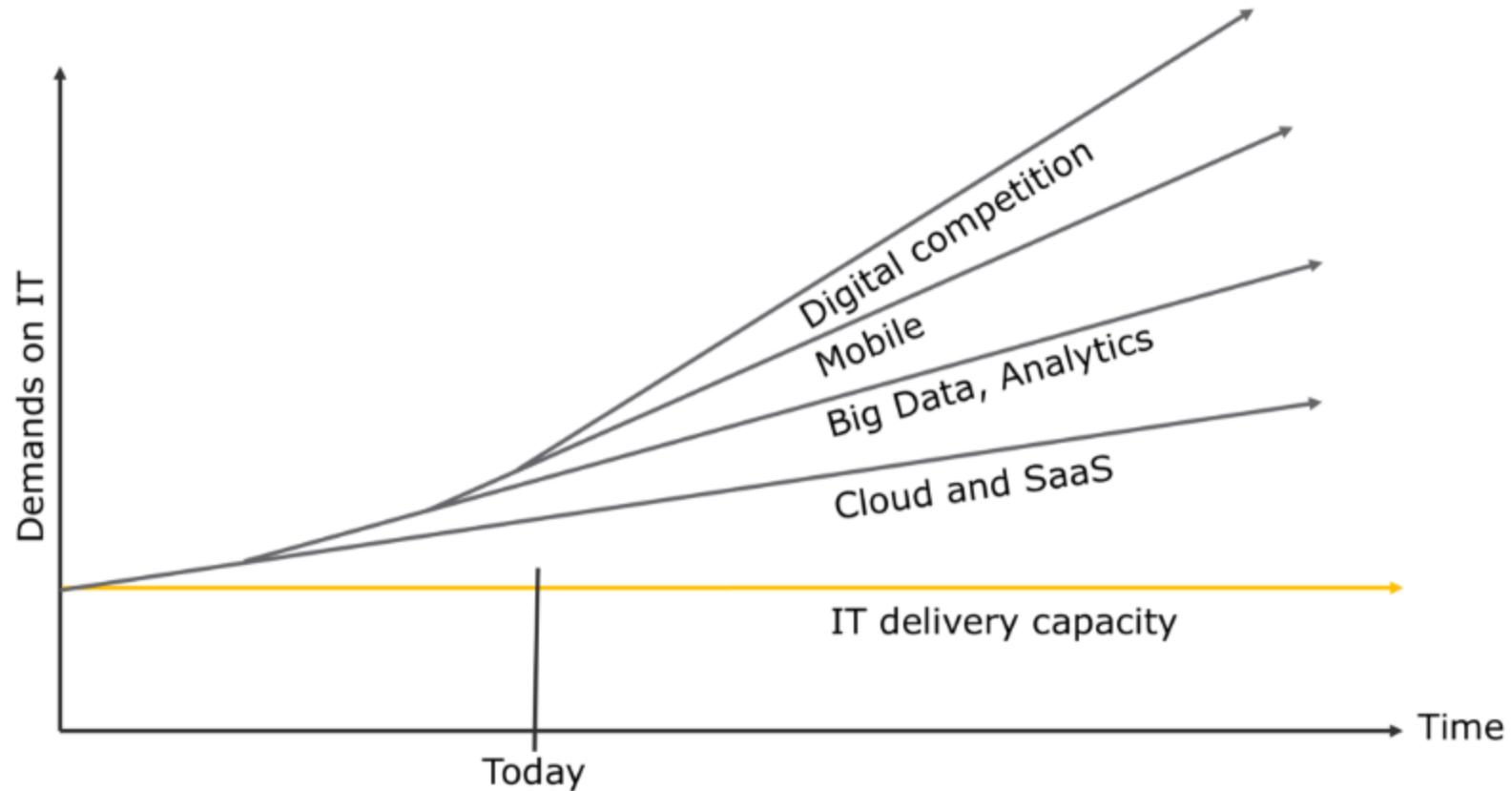


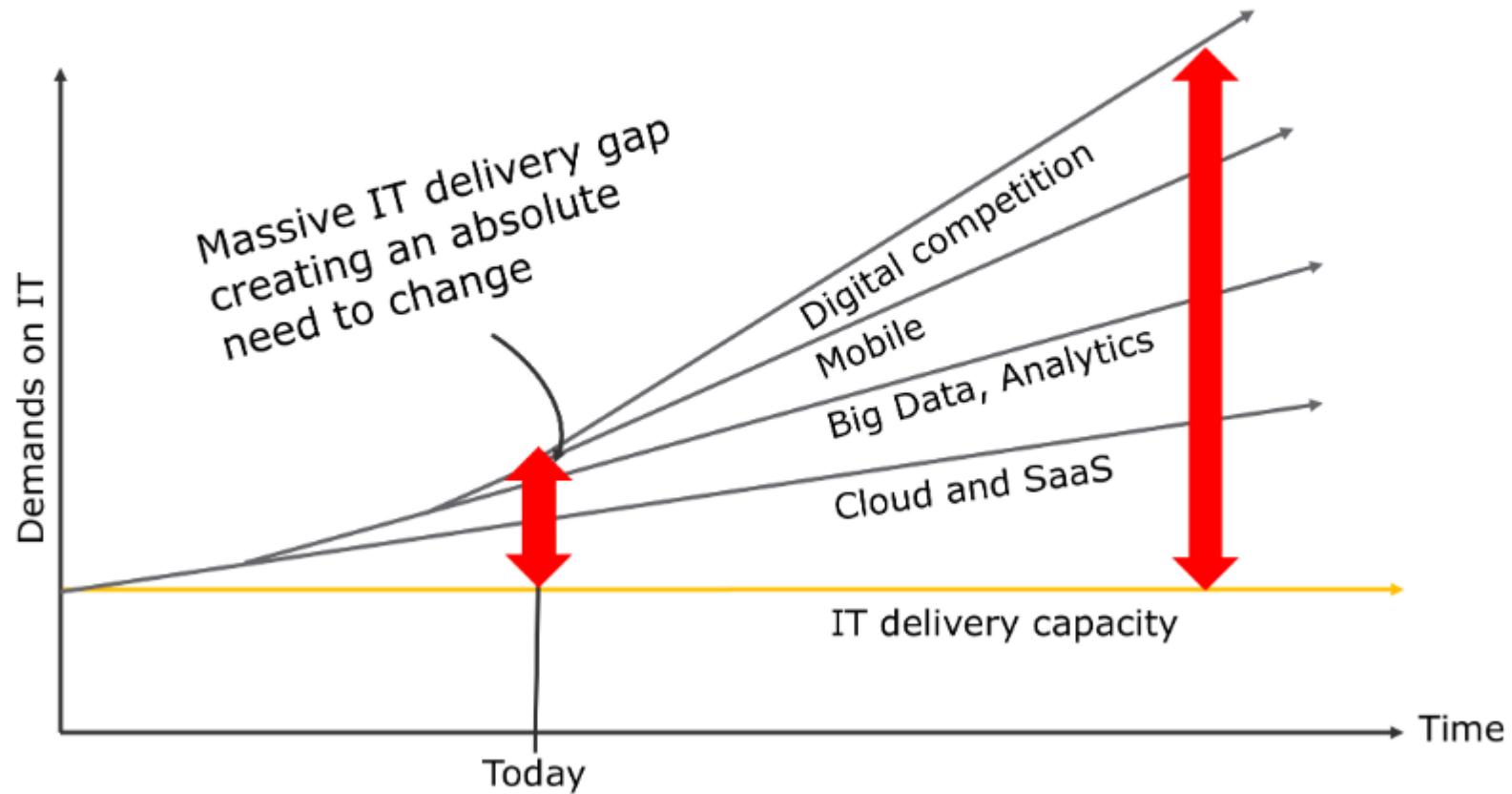
Identifying the problems faced by IT today



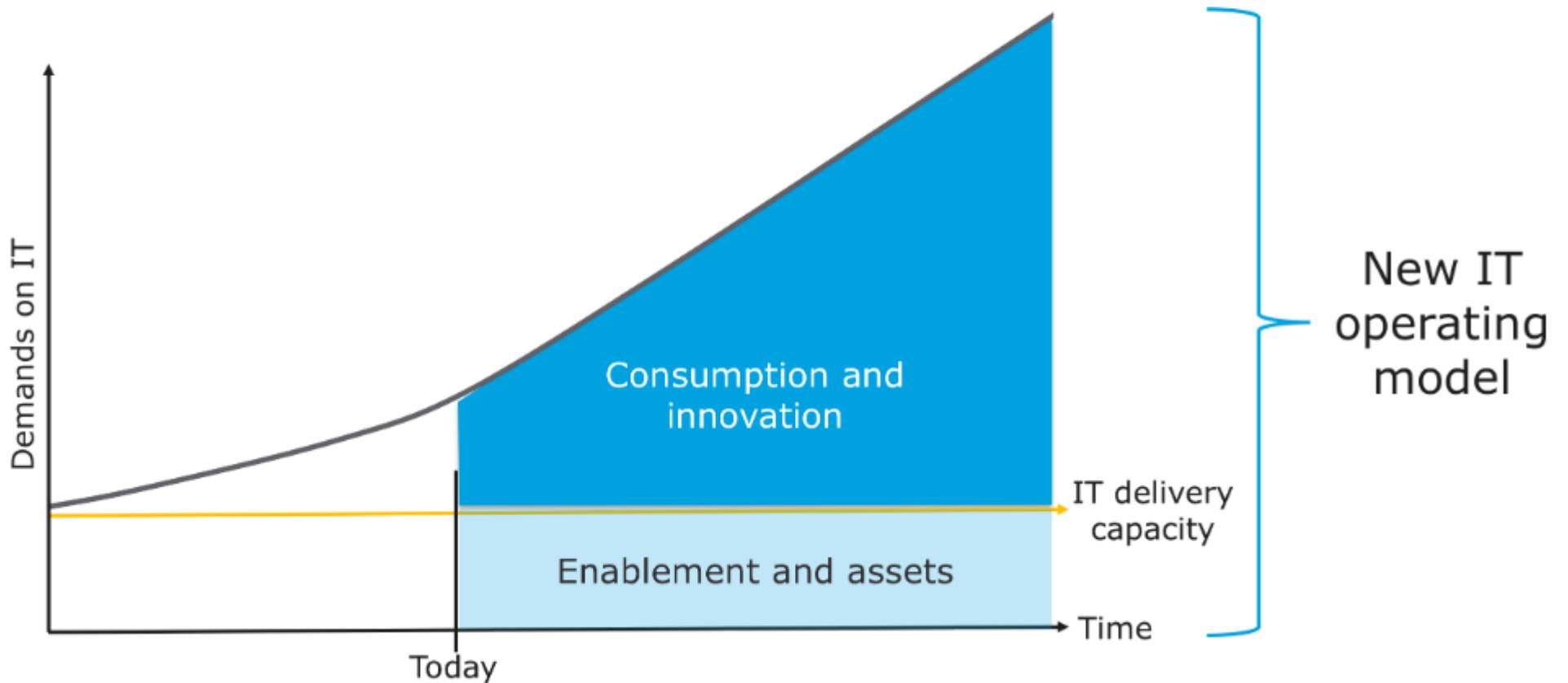
Biggest challenge: IT cannot go fast enough



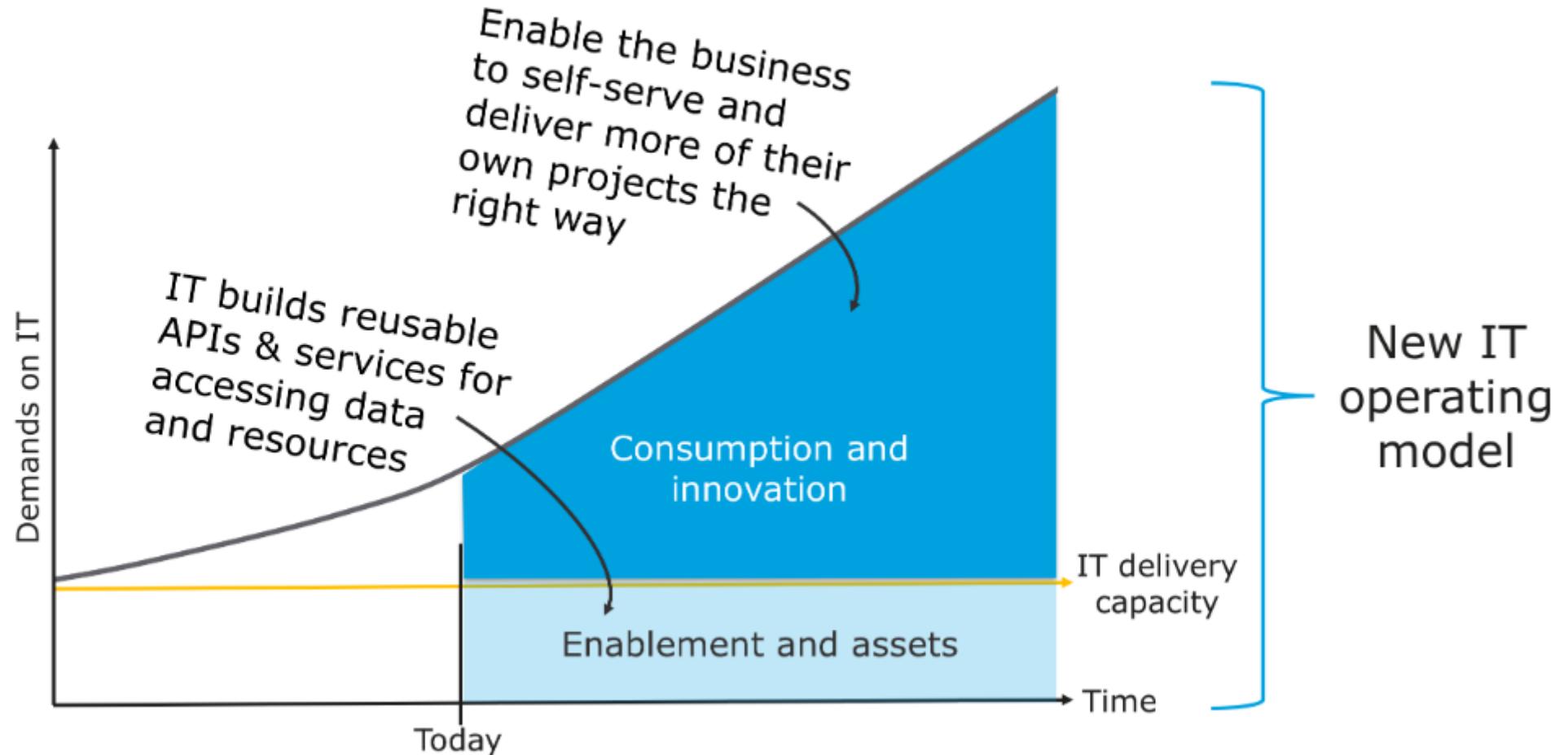
Digital pressures create a widening IT delivery gap



A new way of working to close the delivery gap



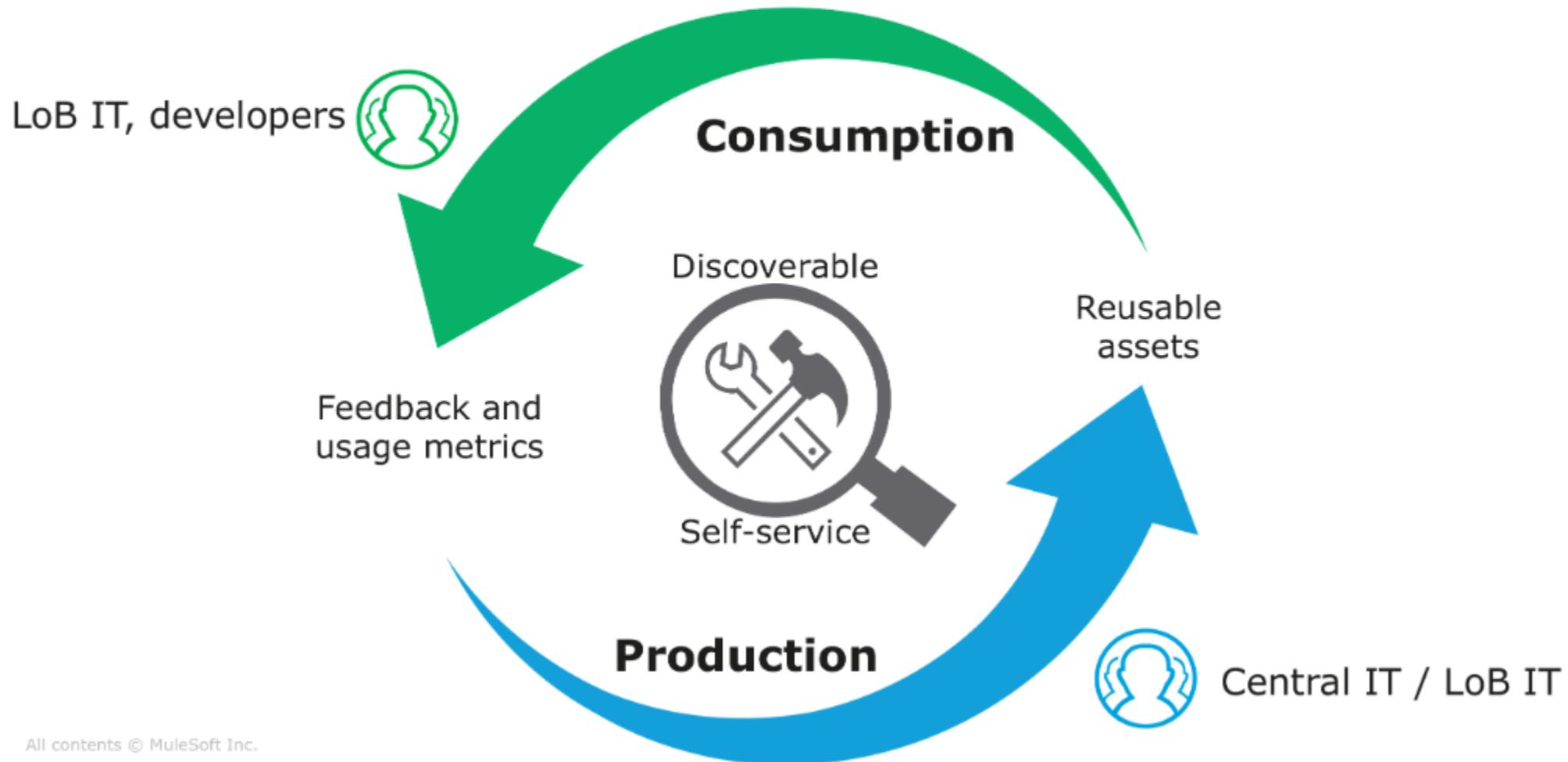
A new way of working to close the delivery gap



Introducing a new IT operating model



New operating model emphasizes consumption

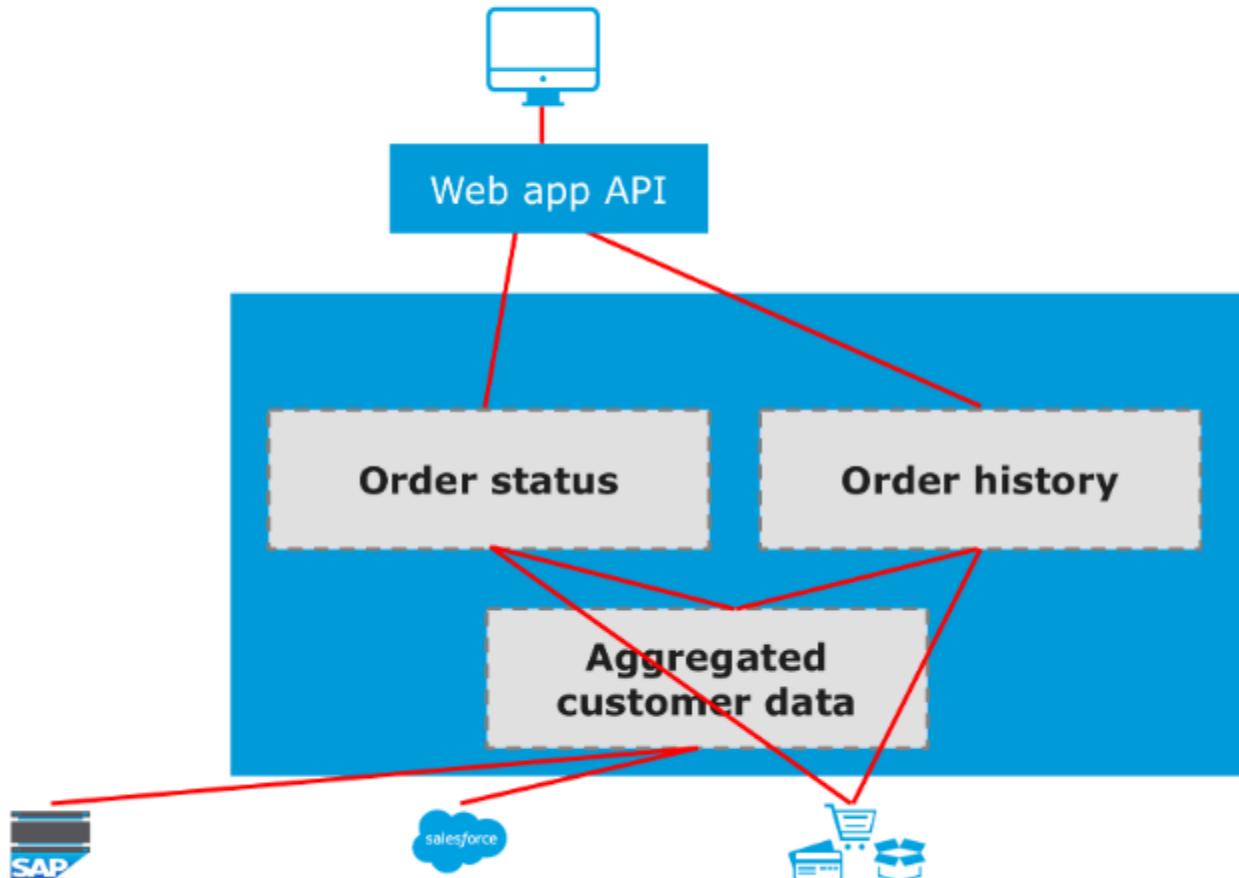


A common project-based approach



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

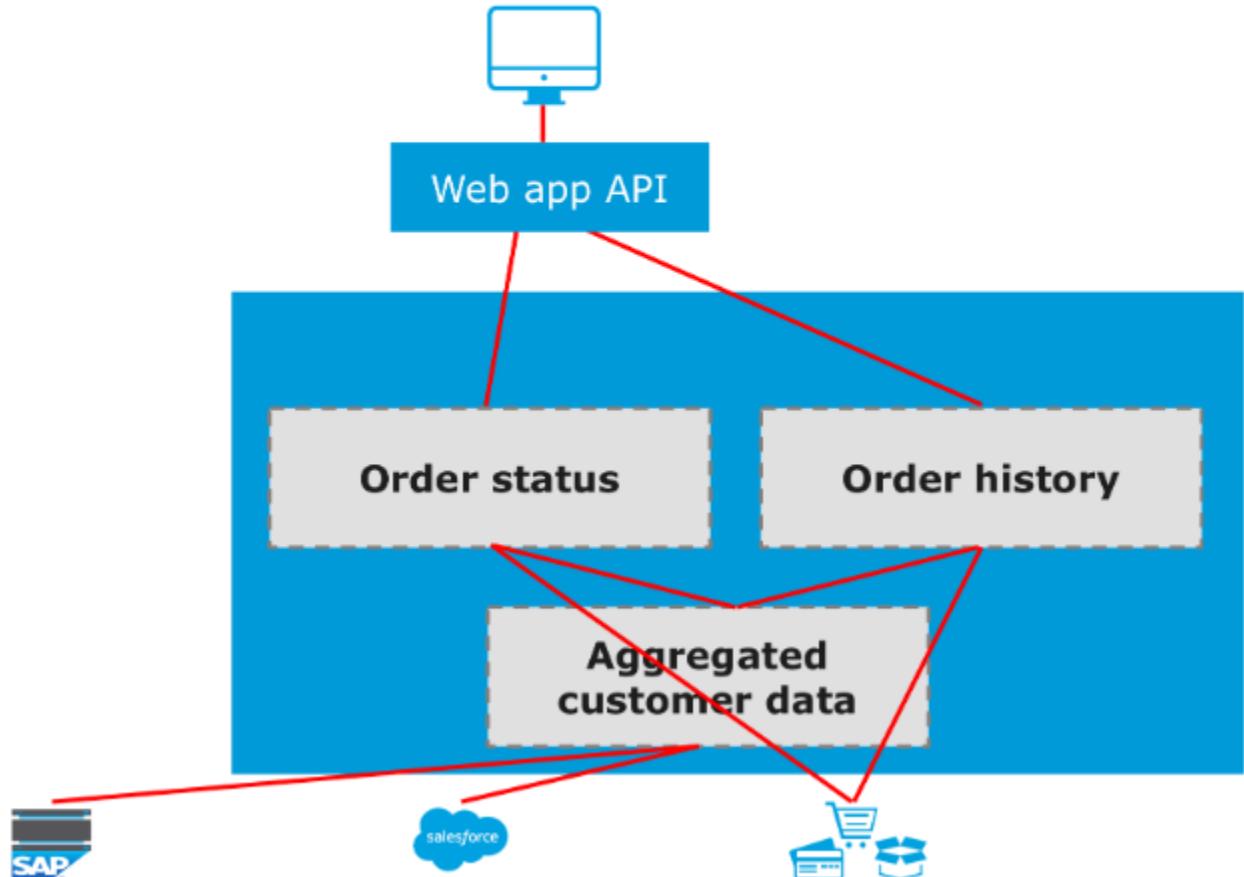
- Order data in eCommerce system
- Inventory data in SAP
- Customer data in SAP, Salesforce



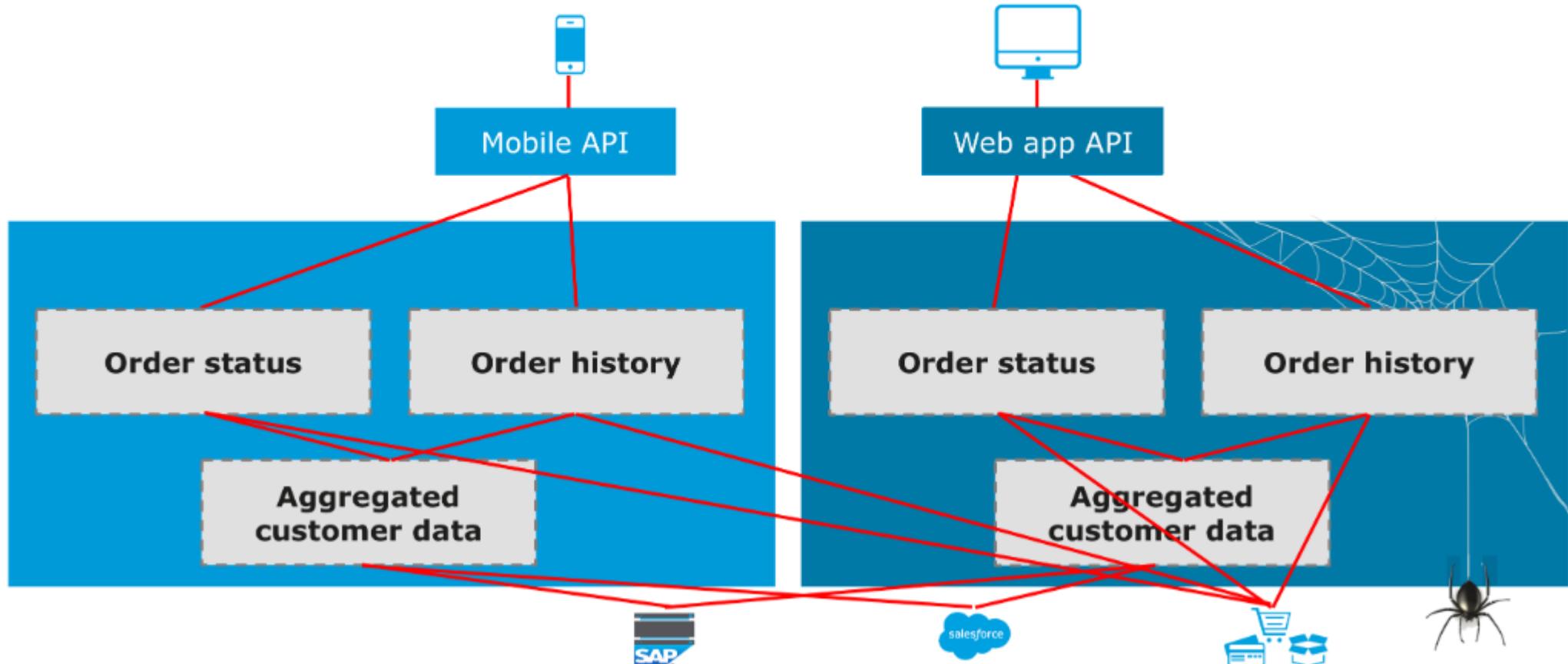
A common project-based approach



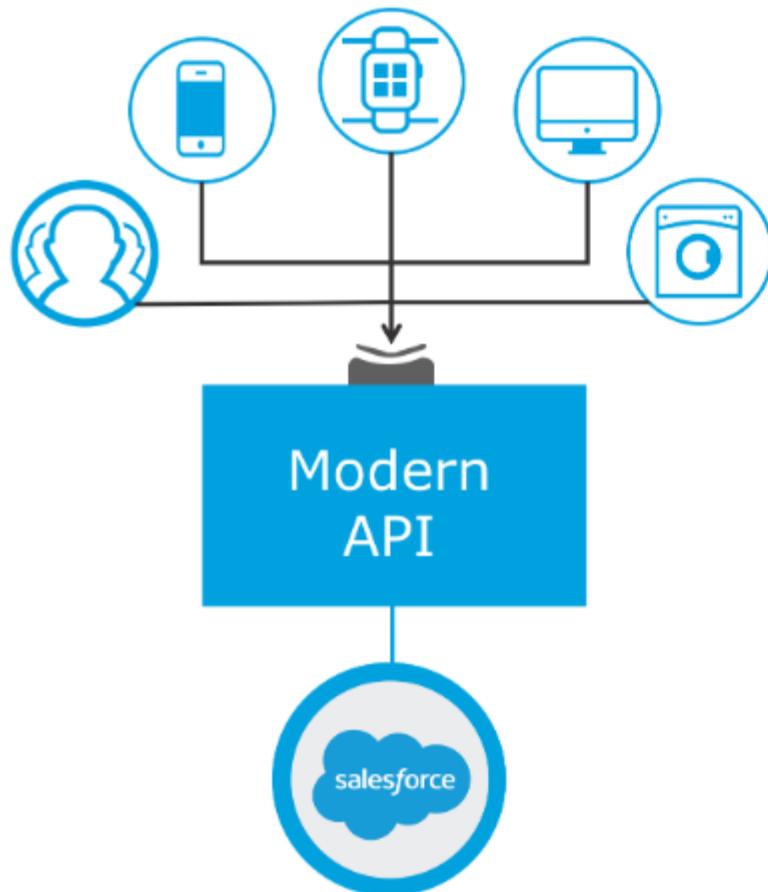
- ✓ On time and within budget
- ✗ Limited opportunity for reuse
- ✗ Tight coupling = brittleness
- ✗ Difficult to govern
- ⚠ Meets business requirements



6 months later...

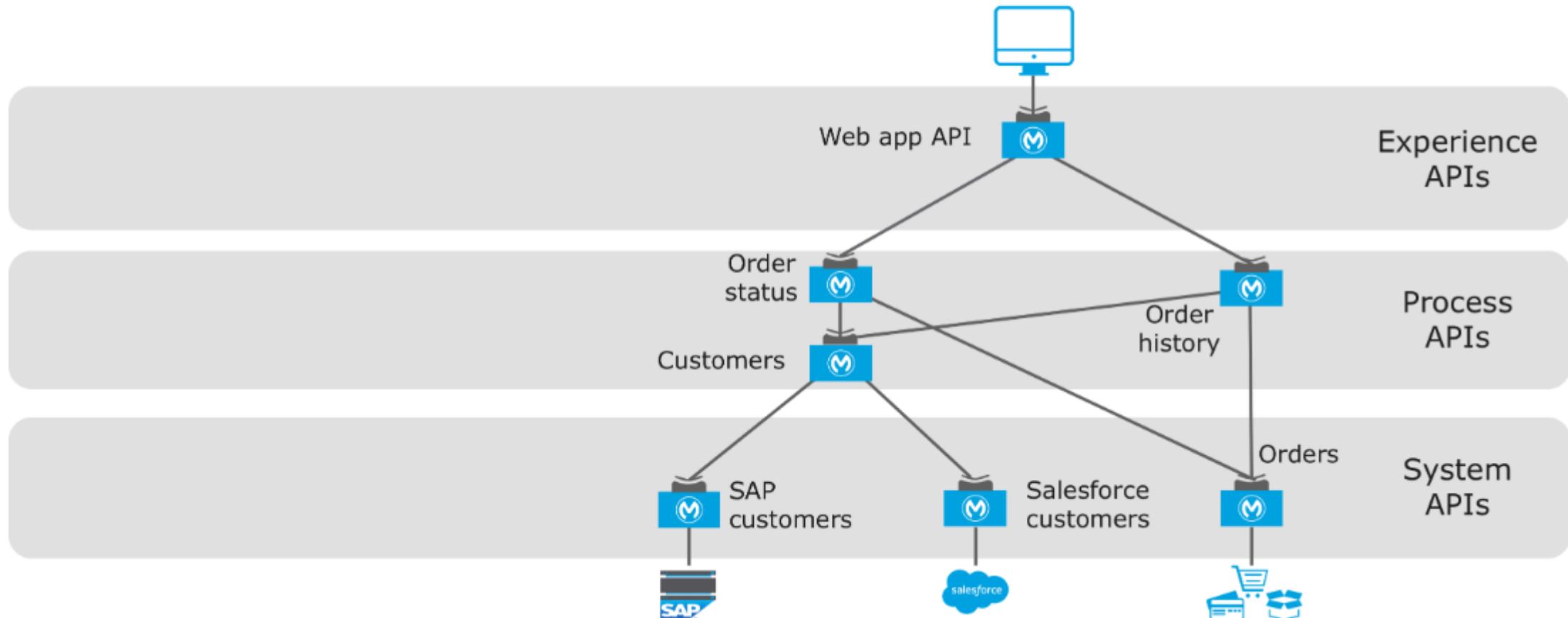


Modern API: The core enabler of a new operating model

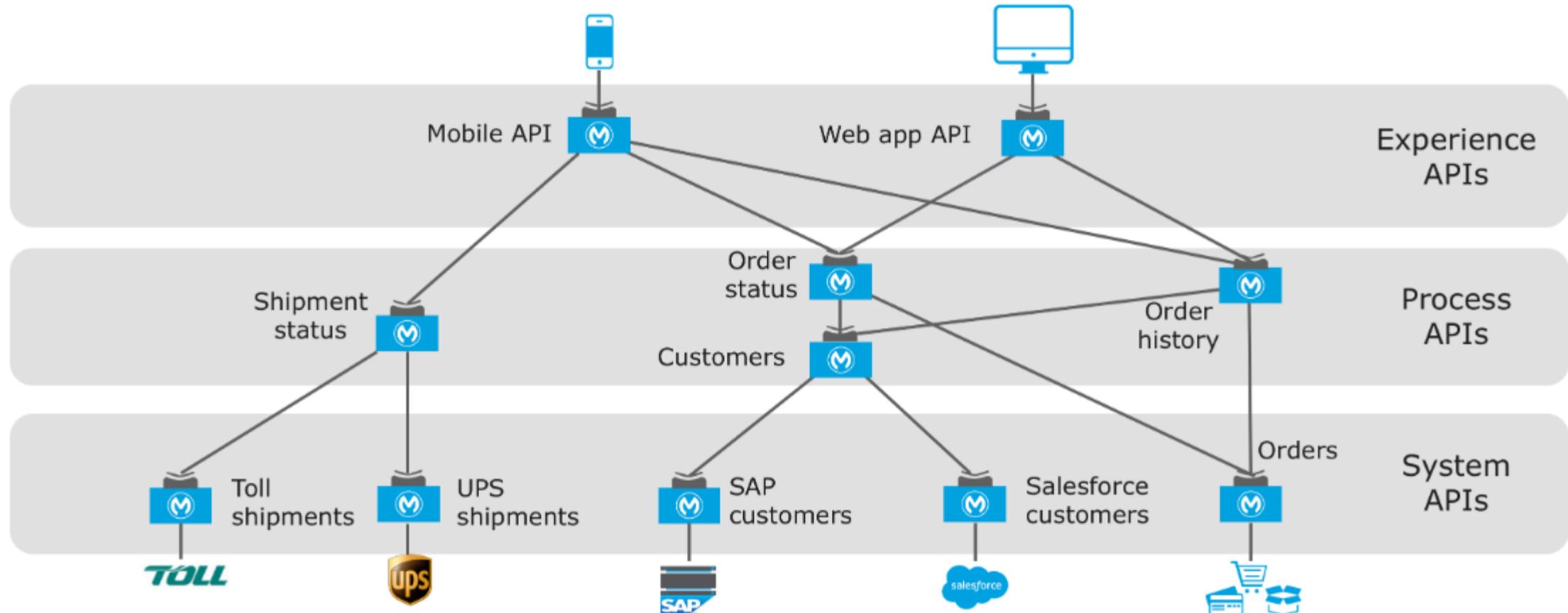


- Discoverable and accessible through self-service
- Productized and designed for ease of consumption
- Easily managed for security, scalability, and performance

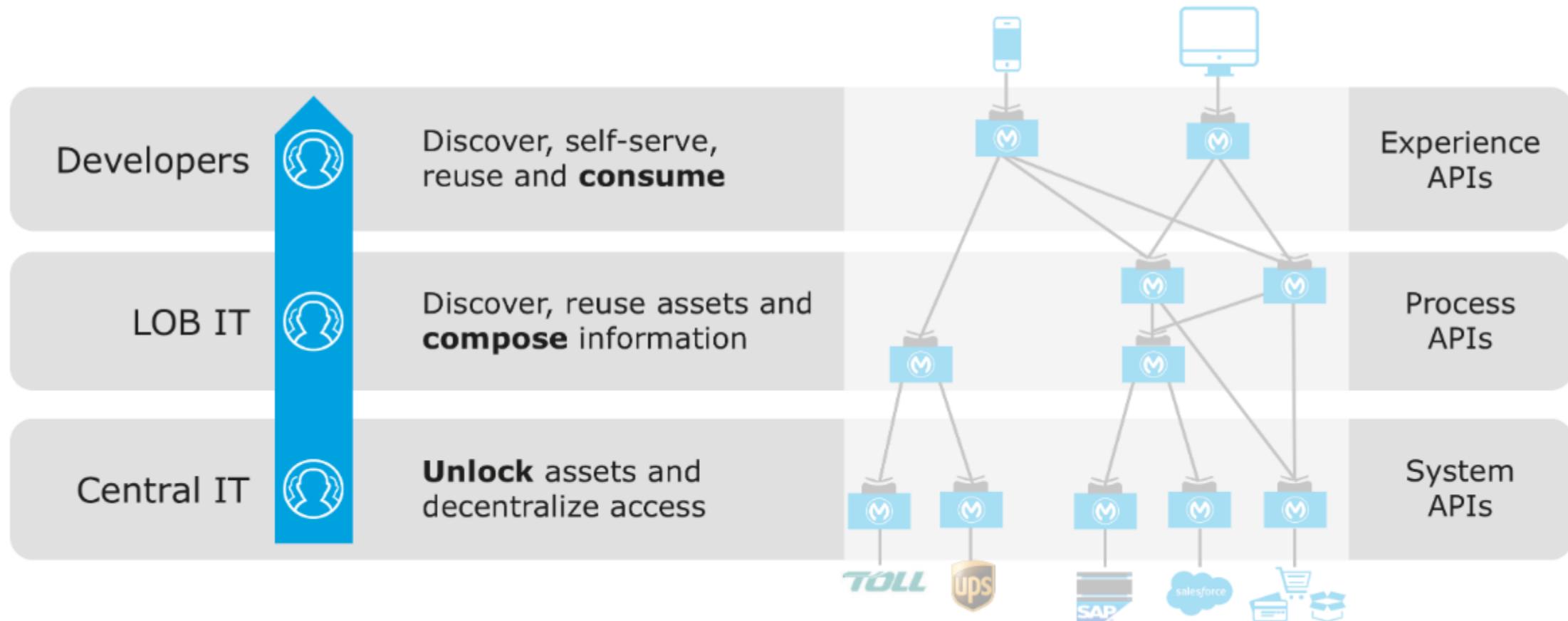
The API-led connectivity approach



The API-led connectivity approach



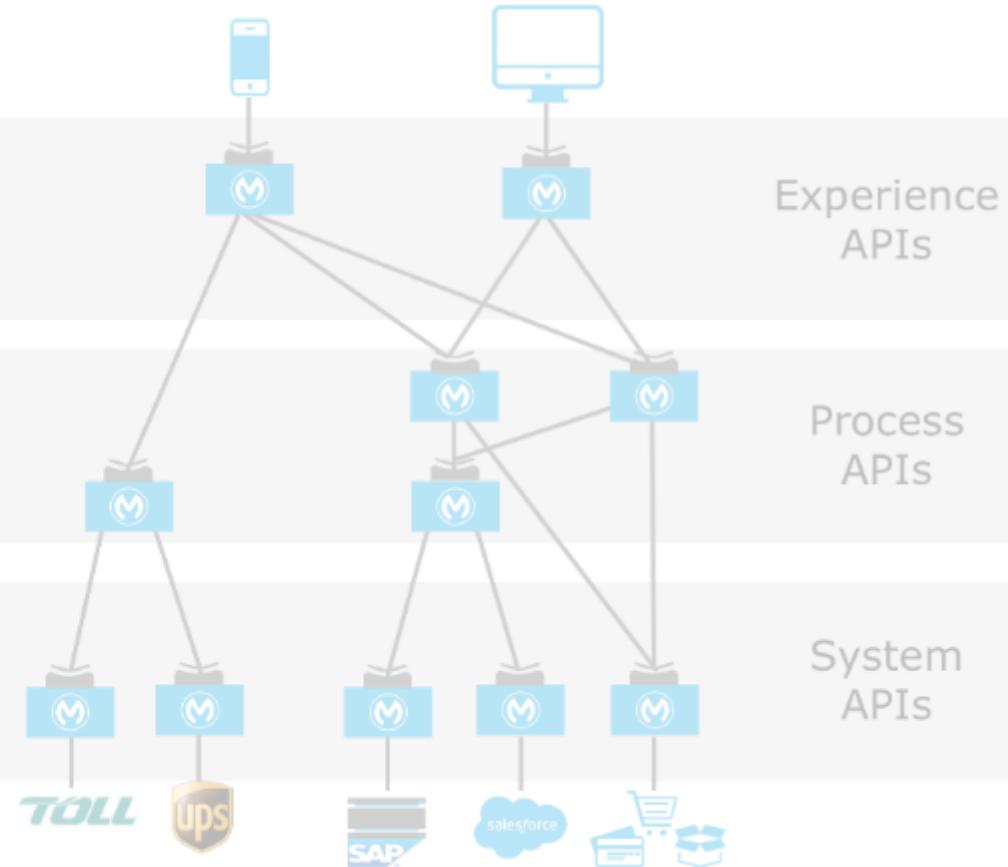
Enable and empower the entire organization



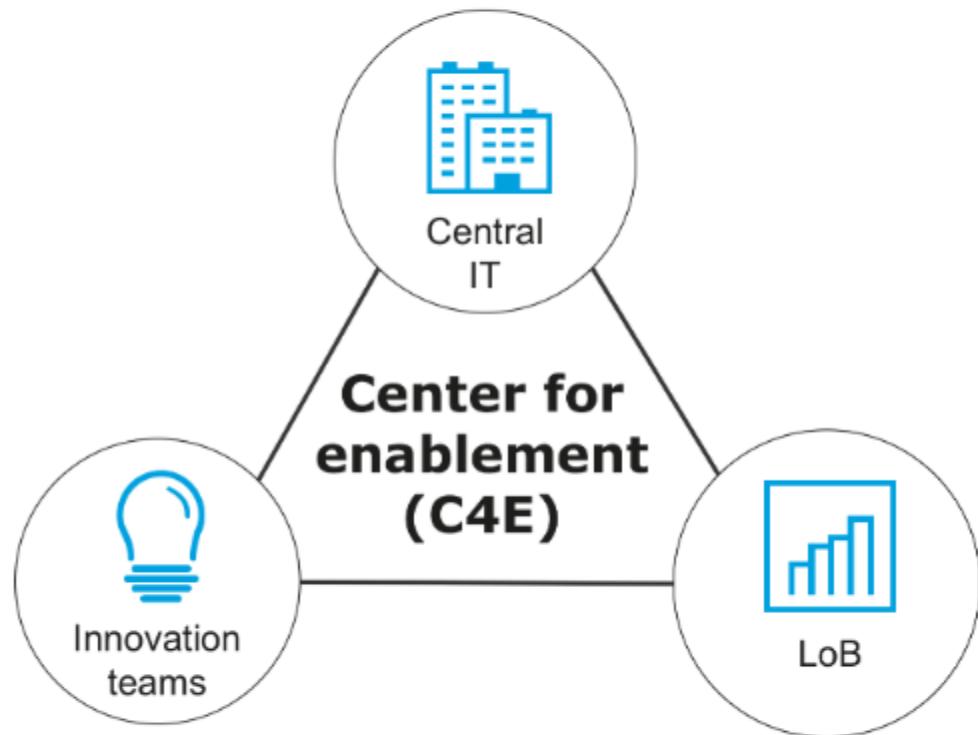
Drive outcomes with API-led connectivity



- ✓ 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



C4E: Organizing differently to drive API-led connectivity



- 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

Achieving an application network



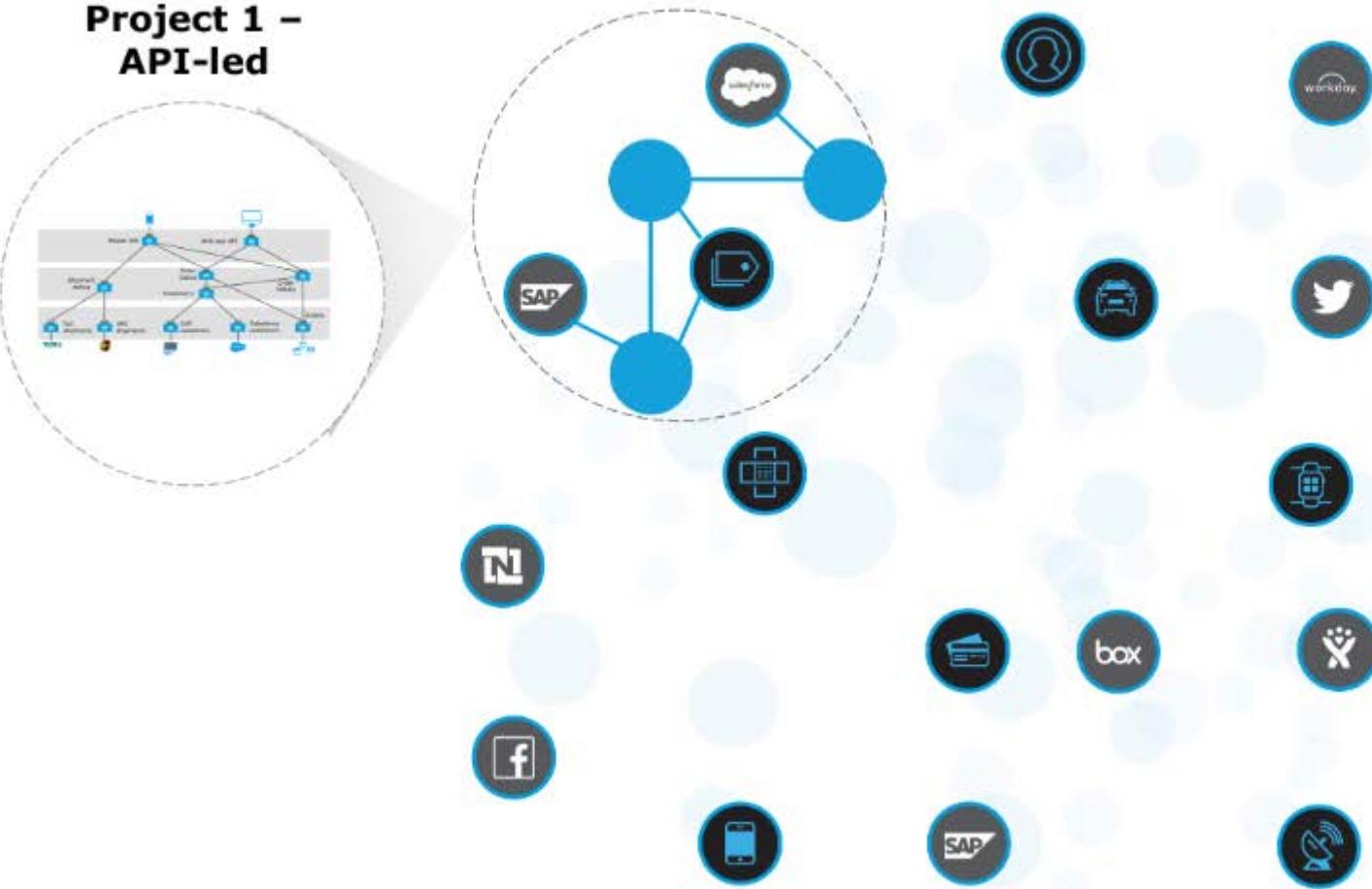
Application landscape



Every project adds value to the application network



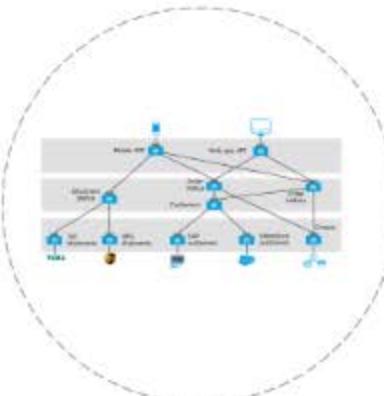
Project 1 – API-led



Every project adds value to the application network

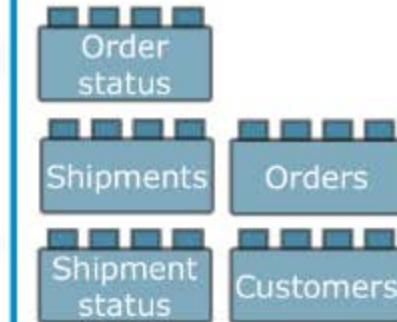


Project 1 – API-led



C4E

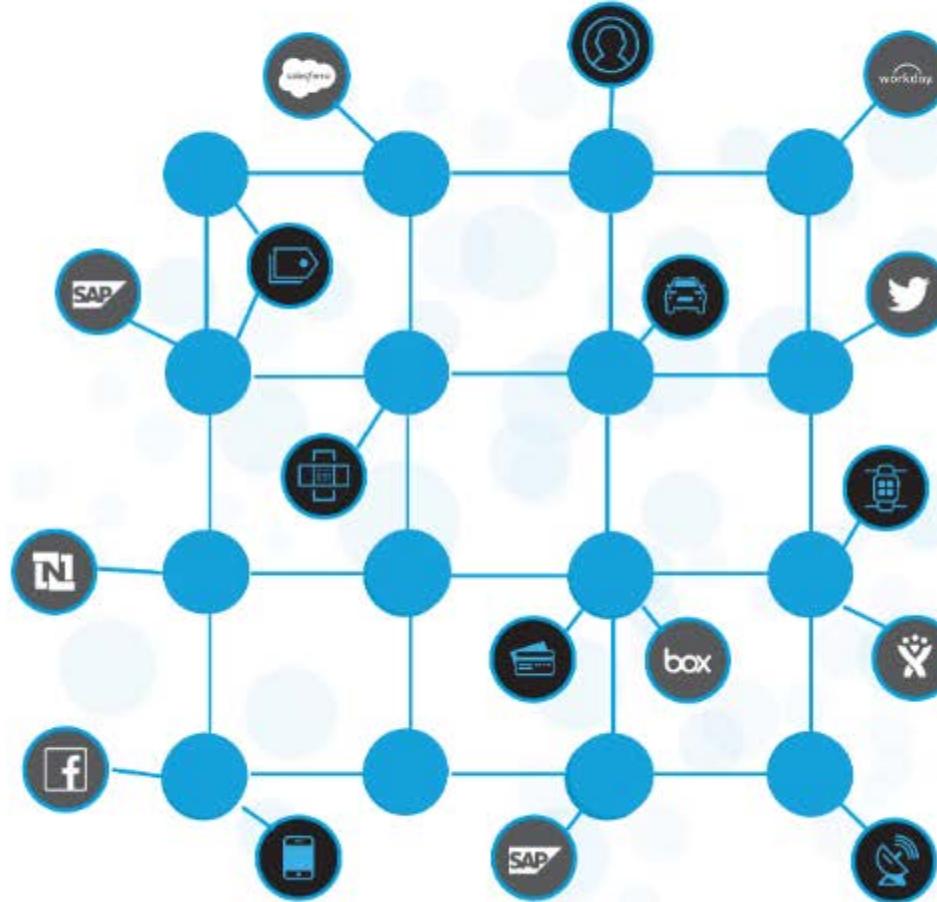
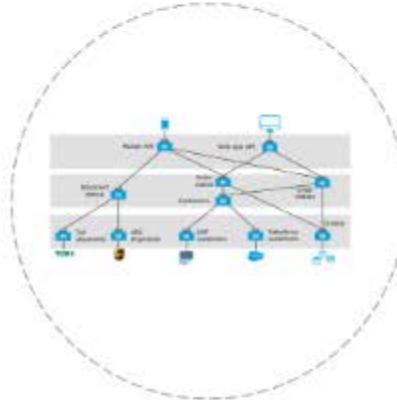
Self-serve assets
on the
application network



Every project adds value to the application network

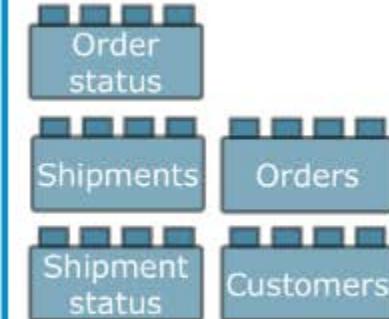


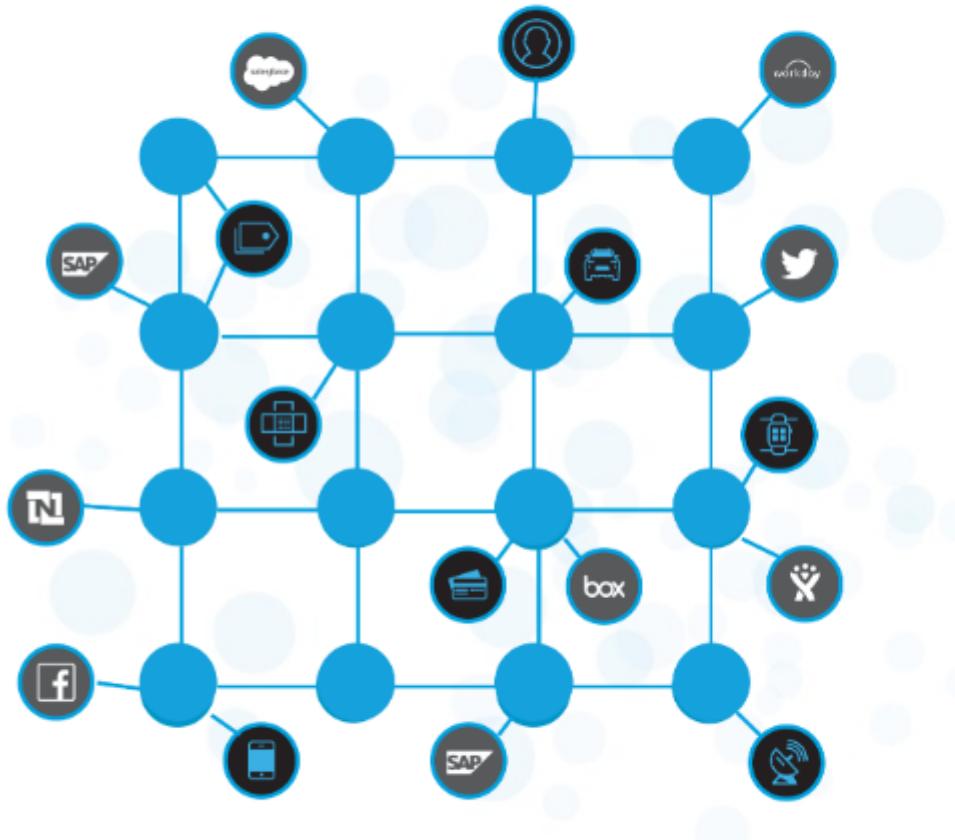
Project 1 – API-led



C4E

Self-serve assets
on the
application network





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**

Deconstructing APIs



- 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

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

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 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

Reviewing 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



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

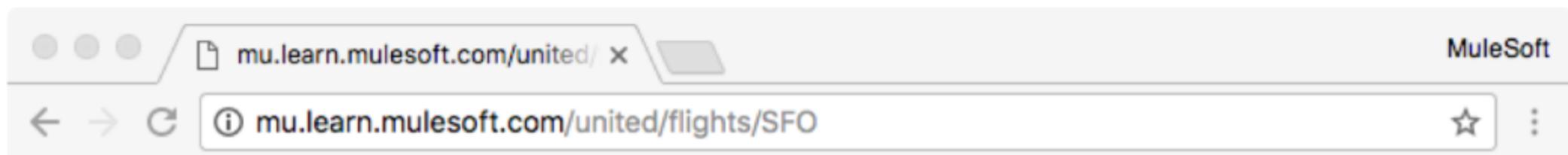
- **GET** retrieves the current state of a resource in some representation (usually JSON or XML)
- **POST** creates a new resource
- **DELETE** deletes a resource
- **PUT** replaces a resource completely
 - If the resource doesn't exist, a new one is created
- **PATCH** partially updates a resource
 - Just submitted data



Example RESTful web service 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



```
{"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}]} 
```

- 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

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 side-by-side screenshots of API documentation.

ProgrammableWeb API DIRECTORY:

- Header: ProgrammableWeb, API DIRECTORY, API NEWS.
- Sub-header: LEARN ABOUT APIs, WHAT IS AN API?, TUTORIALS, API CHARTS & RESEARCH.
- Middle section: A banner for "CASE STUDY: Building digital platforms" featuring the MuleSoft logo.
- Section: Search the Largest API Directory on the Web. Includes a search bar ("Search Over 19,516 APIs") and a "SEARCH APIs" button.
- Section: Filter APIs. Includes a dropdown menu ("By Category") and a checkbox for "Include Deprecated APIs".
- Table: Shows a list of APIs with columns: API Name, Description, Category, and Submitted. Entries include "Google Maps" and "Twitter".

MuleSoft // Training API portal:

- Header: MuleSoft // Training, Home.
- Left sidebar: Assets list, American Flights API, API summary, Types, Resources, /flights, /{ID}.
- Right sidebar: Methods: GET, POST, GET, DELETE, PUT.
- Section: American Flights API | v2. Shows a house icon and 5 stars (0 reviews).
- Section: /{ID} : get. Shows Request: GET /flights/{ID}.
- Section: Parameters. Shows a table with columns: Parameter, Type, Description. It lists "URI parameters" and "ID (required)" as a string.

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

In this walkthrough, you locate and explore documentation about APIs. You will:

- 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 screenshot shows two side-by-side API documentation pages. On the left is the ProgrammableWeb API directory, featuring a search bar and a list of APIs. One entry is highlighted: "Google Maps" (Category: Mapping, Submitted: 12/05/2005). On the right is the MuleSoft API portal for the "American Flights API | v2". It displays the API summary, including its rating (4 stars) and number of reviews (0). Below this is the API endpoint definition: "/{ID} : get", with a "Request" section showing the GET method and URL template "/flights/{ID}", and a "Parameters" section showing a required parameter "ID" of type string.

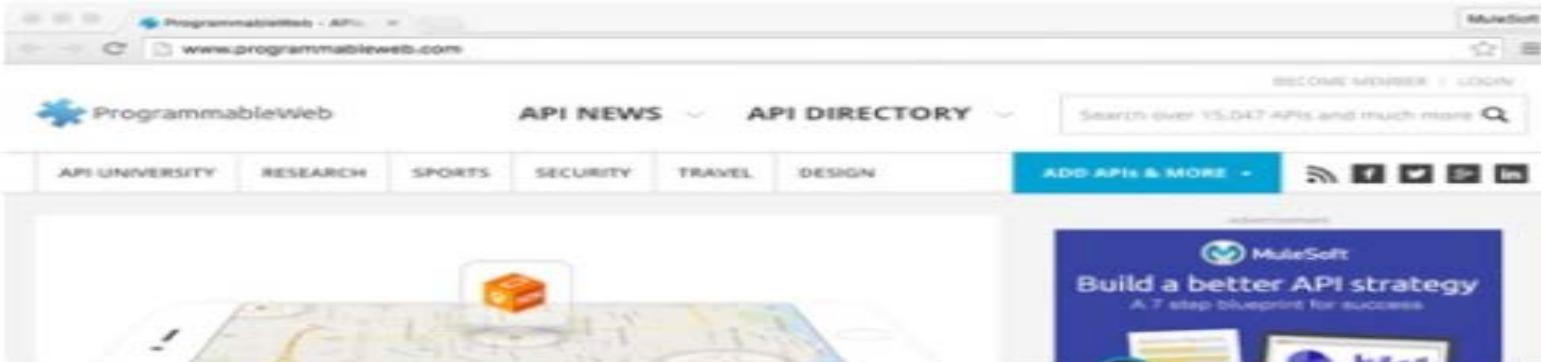
Browse the ProgrammableWeb API directory

1. In a web browser, navigate to <http://www.programmableweb.com/>.
2. Click the API directory link.



Browse the ProgrammableWeb API directory

1. In a web browser, navigate to <http://www.programmableweb.com/>.
2. Click the API directory link.

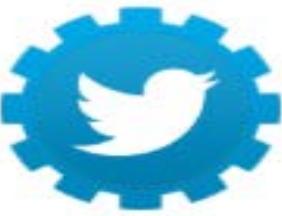


3. Browse the list of popular APIs.

API Name	Description	Category	Submitted
Google Maps	[This API is no longer available. Google Maps' services have been split into multiple APIs, including the Static Maps API , ...]	Mapping	12.05.2005
Twitter	[This API is no longer available. It has been split into multiple APIs, including the Twitter Ads API , Twitter Search Tweets...]	Social	12.08.2006
YouTube	The Data API allows users to integrate their program with YouTube and allow it to perform many of the operations available on the website. It provides the capability to search for videos, retrieve...	Video	02.08.2006
Flickr	The Flickr API can be used to retrieve photos from the Flickr photo sharing service using a variety of feeds - public photos and videos, favorites, friends, group pools, discussions, and more. The...	Photos	09.04.2005

Explore the API reference for the Twitter API

4. Click the link for the Twitter API (or some other) API.
5. In the Specs section, click the API Portal / Home Page link.



Twitter API

Social Blogging

[This API is no longer available. It has been split into multiple APIs, including the [Twitter Ads API](#), [Twitter Search Tweets API](#), and [Twitter Direct Message API](#).]

This profile is maintained for historical, research, and reference purposes only.]

The Twitter micro-blogging service includes two RESTful APIs. The Twitter REST API methods allow developers to access core Twitter data. This includes update timelines, status data, and user information. The Search API methods give developers methods to interact with Twitter Search and trends data. The API presently supports the following data formats: XML, JSON, and the RSS and Atom syndication formats, with some methods only accepting a subset of these formats.

Summary	SDks (72)	Articles (267)	How To (11)	Sample Source Code (25)	Libraries (38)	Developers (820)	Followers (1919)	Comments (4)
---------	--------------	-------------------	----------------	----------------------------	-------------------	---------------------	---------------------	-----------------

SPECS

API Endpoint	http://twitter.com/statuses/
API Portal / Home Page	https://dev.twitter.com/rest/public
Primary Category	Social

- In the new browser tab that opens, click Tweets in the left-side navigation.
- In the left-side navigation, click Post, retrieve and engage with Tweets.



Search all documentation...

Post, retrieve and engage with Tweets

Basics

Accounts and users

Tweets

[Post, retrieve and engage with Tweets](#)

[Get Tweet timelines](#)

[Curate a collection of Tweets](#)

[Optimize Tweets with Cards](#)

[Search Tweets](#)

[Filter realtime Tweets](#)

[Sample realtime Tweets](#)

[Get batch historical Tweets](#)

[Overview](#) [Guides](#) [API Reference](#)

Tweets

- POST statuses/update
- POST statuses/destroy/:id
- GET statuses/show/:id

Retweets

- POST statuses/retweet/:id
- POST statuses/unretweet/:id
- GET statuses/retweets/:id
- GET statuses/retweets_of_me
- GET statuses/retweeters/:ids

Likes (formerly favorites)

- POST favorites/create/:id
- POST favorites/destroy/:id
- GET favorites/list

- Browse the list of requests you can make to the API.
- Select the API Reference tab beneath the title of the page.

10. Review the information for POST statuses/update including parameters, example request, and example response.

The screenshot shows the Twitter developer documentation page. At the top, there's a navigation bar with links for Developer, Use cases, Products, Docs, More, Apply, a search icon, and a Sign In button. Below the navigation, there's a table with one row. The table has five columns: a parameter name, its type, a detailed description, and two sample values (true and false). The parameter is 'fail_dm_commands', the type is 'optional', the description is 'When set to true, causes any status text that starts with shortcode commands to return an API error. When set to false, allows shortcode commands to be sent in the status text and acted on by the API.', and the sample values are 'true' and 'false'.

fail_dm_commands	optional	When set to true, causes any status text that starts with shortcode commands to return an API error. When set to false, allows shortcode commands to be sent in the status text and acted on by the API.	true	false
------------------	----------	--	------	-------

Example Request

```
POST https://api.twitter.com/1.1/statuses/update.json?
status=Maybe%20he%27ll%20finally%20find%20his%20keys.%20%23peterfalk
```

Example Response

```
{
  "coordinates": null,
  "favorited": false,
  "created_at": "Wed Sep 05 00:37:15 +0000 2012",
  "truncated": false,
  "id_str": "243145735212777472",
  "entities": {
    "urls": [
```

11. Close the browser tab.

Explore an API portal for an API to be used in the course

12. Return to the course snippets.txt file.
13. Copy the URL for the MuleSoft Training API portal.
14. Return to a browser window and navigate to that URL: <https://anypoint.mulesoft.com/exchange/portals/muletraining/>.

The screenshot shows the MuleSoft Training API portal interface. At the top, there is a navigation bar with the MuleSoft logo, the text "MuleSoft // Training", a "Home" link, and a "Login" button. Below the navigation bar, a large banner displays the text "Welcome to your MuleSoft Training portal!" and "Build your application network faster! Get started with powerful tools, intuitive interface, and best in class documentation experience." In the main content area, the word "Assets" is displayed. Below it, there is a search bar with the placeholder "Search" and a dropdown menu labeled "All types". A single API asset card is visible, titled "American Flights API". The card features a circular icon with a house symbol, a "REST API" label, and a five-star rating icon.

15. Click the American Flights API.
16. Browse the API Summary on the left-side and discover the resources that are available for the API.

The screenshot shows the MuleSoft API Training interface. At the top, there is a navigation bar with the MuleSoft logo, "MuleSoft // Training", "Home", and "Login". Below the navigation bar, on the left side, is a sidebar with the following menu items:

- [← Assets list](#)
- [American Flights API](#) (highlighted with a blue background)
- [API summary](#)
- [Types](#)
- [Resources](#)
- [/flights](#)
 - [GET](#)
 - [POST](#)
 - [/\({ID}\)](#)
 - [GET](#)
 - [DELETE](#)
 - [PUT](#)
- [API instances](#)

The main content area displays the "American Flights API" summary. It features a green circular icon with a white house symbol, the API name "American Flights API", its version "v2", and a rating of "★ ★ ★ ★ ★ (0 reviews)". There is also a "Request access" button with a magnifying glass icon.

Below the summary, a note states: "NOTE: Version v2 of this API is for use with Mule 4.1 courses. It has a 4.XX proxy that uses **HEADERS** for client_id authentication."

The description of the API notes: "The American Flights API is a system API for operations on the `american` table in the `training` database."

The "Supported operations" section lists the following methods for the /flights endpoint:

- Get all flights
- Get a flight with a specific ID
- Add a flight
- Delete a flight
- Update a flight

17. Select the GET method.

18. Review the information about the GET method; you should see there is an optional query parameter called destination.

MuleSoft // Training Home Login

← Assets list American Flights API | v2 Download Request access

GET

Mocking Service

https://mocksvc-proxy.anypoint.mulesoft.com/exc

Parameters Headers

Query parameters Show optional parameters

Send

/flights : get

Request

GET /flights

Parameters

Parameter	Type	Description
destination	string (enum)	Possible values: SFO, LAX, CLE

GET

POST

PUT

DELETE

(ID)

API instances

The screenshot shows the MuleSoft Anypoint Platform interface. On the left sidebar, under 'American Flights API', the '/flights' endpoint is selected, indicated by a green circle around the 'GET' button. The main content area displays the '/flights : get' endpoint, its 'Request' method (GET /flights), and its 'Parameters' section. The 'Parameters' table lists a single parameter: 'destination' of type 'string (enum)' with possible values 'SFO', 'LAX', and 'CLE'. A green circle highlights the 'destination' parameter in the table. The top right of the screen shows download and request access buttons, and the bottom right has a 'Send' button.

19. Scroll down and review the Headers and Response sections.

Headers

Parameter	Type	Description
client_id (required)	string	
client_secret (required)	string	

Response

200

Type application/json

Type

Examples

```
{
  "ID": "integer",
  "code": "string",
  "price": "number",
  "departureDate": "string",
  "origin": "string",
  ...}
```

20. In the left-side navigation, select the DELETE method.
21. Locate information about the required ID URI parameter.

The screenshot shows a detailed view of an API endpoint for deleting a flight. On the left, a sidebar navigation lists 'API summary', 'Types', 'Resources' (expanded), and '/flights' (expanded). Under '/flights', there are four methods: 'GET' (green button), 'POST' (purple button), 'DELETE' (red button), and 'PUT' (blue button). The 'DELETE' method is highlighted. Below the methods, 'API instances' is listed. To the right of the sidebar, the endpoint path is shown as '/{ID} : delete'. The main content area is titled 'Request' and contains the DELETE method signature: 'DELETE /flights/{ID}'. Below this, a table titled 'Parameters' lists the required URI parameter 'ID' (string type, required) under the 'URI parameters' column. The table has columns: Parameter, Type, and Description.

Parameter	Type	Description
ID (required)	string	

22. Review the information for the other resources.

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

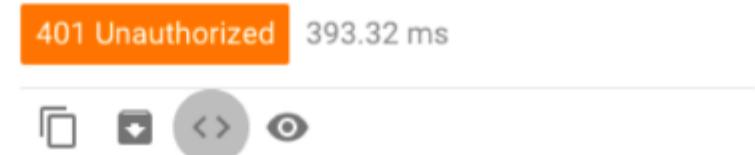
The screenshot shows the Advanced REST client interface. The top bar displays the title "Advanced REST client". Below it, the main header says "Request" and shows the URL "http://training4-american...". The method is set to "GET". A "SEND" button is visible. The "Parameters" dropdown is open, showing a single parameter: "origin": "HUA", "destination": "LAX". The response status is "200 OK" with a time of "1290.08 ms". The response body is displayed as JSON:

```
{  
  "ID": 1,  
  "code": "FREE0001",  
  "price": 541,  
  "departureDate": "2016-01-28T00:00:00",  
  "origin": "HUA",  
  "destination": "LAX",  
}
```

On the right side, there's a preview of the API endpoint "American Flights API" with a green "GET" button. The API summary shows "flights : get". The "Request" section includes "Parameters" and "Headers" tabs, with "Parameters" currently selected. It lists "origin" and "destination" with their respective values. The "Headers" tab shows "Content-Type: application/json" and "Accept: application/json".

- **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

Response

200	Type application/atom+xml	Schema	Examples
401		<input checked="" type="checkbox"/>	

```
<?xml version='1.0' encoding='UTF-8'?>
<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'>
```

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 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 screenshot shows the Advanced REST client interface. The 'Request' tab is active, displaying a successful GET request to `http://training4-american-api.cloudhub.io/flights`. The response status is 200 OK with a duration of 1290.08 ms. The response body is a JSON object:

```
{ "ID": 1, "code": "rree0001", "price": 541, "departureDate": "2016-01-20T00:00:00", "origin": "MIA", "destination": "LAX", }
```

On the right, the MuleSoft // Training API portal is shown for the 'American Flights API'. It lists the `/flights : get` endpoint under the 'GET /flights' section. The 'Parameters' table includes a 'destination' parameter of type string with possible values SFO, LAX, and NY.

Walkthrough 1-2: Make calls to an API

In this walkthrough, you make calls to a RESTful API. You will:

- Use Advanced REST Client to make calls to an unsecured API (an implementation).
- Make GET, DELETE, POST, and PUT calls.
- Use Advanced REST Client 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 screenshots of API development tools. The left screenshot shows the 'Advanced REST client' interface with a successful GET request to 'http://training4-american-api.cloudhub.io/flights' resulting in a 200 OK response. The right screenshot shows the 'MuleSoft // Training' API console for the 'American Flights API v2', specifically the '/flights : get' endpoint, also showing a successful response. Both screenshots include detailed response bodies and header information.

Use Advanced REST Client to make GET requests to retrieve data

1. Return to or open Advanced REST Client.
2. Make sure the method is set to GET.
3. Return to the course snippets.txt file.
4. Copy the URL for the American Flights web service:
<http://training4-american-ws.cloudhub.io/api/flights>.

Note: This is the URL for the API implementation, not the managed API proxy. The -ws stands for web service.

5. Return to Advanced REST Client and paste the URL in the text box that says Request URL.

Method	Request URL	SEND	⋮
GET	http://training4-american-ws.cloudhub.io/api/flights	▼	

Parameters ▾

6. Click the Send button; you should get a response.
7. Locate the return HTTP status code of 200.

8. Review the response body containing flights to SFO, LAX, and CLE.

200 OK 1283.27 ms

DETAILS ▾



```
[Array[11]
-0: {
  "ID": 1,
  "code": "rree0001",
  "price": 541,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX",
  "emptySeats": 0,
  -"plane": {
    "type": "Boeing 787",
    "totalSeats": 200
  }
},
-1: {
  "ID": 2,
  "code": "eefd0123",
}
```

9. Click the Toggle raw response view button.

200 OK 1283.27 ms DETAILS ▾

[
 {
 "ID": 1,
 "code": "rree0001",
 "price": 541,
 "departureDate": "2016-01-20T00:00:00",
 "origin": "MUA",
 "destination": "LAX",
 "emptySeats": 0,
 "plane": {
 "type": "Boeing 787",
 "totalSeats": 200
 }
 },

10. Click the arrow to the right of the URL.

10. Click the arrow to the right of the URL.

11. In the area that appears, set the Param name to destination and the value to CLE.

The screenshot shows a user interface for making a REST API request. At the top, there are fields for 'Method' (set to 'GET') and 'Host' (set to 'http://training-american-ws.cloudhub.io'). Below these is a 'Path' field containing '/api/flights'. To the right of the path are three buttons: 'SEND', a blue square button, and a vertical ellipsis '...'. Underneath the path is a 'Query parameters' section. It contains a row with a toggle switch (set to 'destination'), a text input field with the value 'CLE', and a close button 'X'. Below this row is a 'ADD' button enclosed in a rounded rectangle.

Method	Host	Path	Query parameters
GET	http://training-american-ws.cloudhub.io	/api/flights	destination: CLE

12. Click the Send button; you should get just flights to CLE returned.

13. Click the X next to the parameter to delete it.

14. Click the arrow to the right of the URL to collapse the parameters section.

15. Change the request URL to use a uri parameter to retrieve the flight with an ID of 3: <http://training4-american-ws.cloudhub.io/api/flights/3>.

16. Click the Send button; you should see only the flight with that ID returned.

200 OK 1235.94 ms



```
[Array[1]
-0: {
  "ID": 3,
  "code": "ffee0192",
  "price": 300,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX",
  "emptySeats": 0,
  -"plane": {
    "type": "Boeing 777",
    "totalSeats": 300
  }
}
```

Make DELETE requests to delete data

17. Change the method to DELETE.
18. Click the Send button; you should see a 200 response with a message that the Flight was deleted (but not really).

Note: The database is not actually modified so that its data integrity can be retained for class.

Method Request URL
DELETE

Parameters

200 OK 253.51 ms



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

19. Remove the URI parameter from the request: <http://training4-american-ws.cloudhub.io/api/flights>.

20. Click the Send button; you should get a 405 response with a message of method not allowed.

405 Method Not Allowed 200.86 ms



```
{  
    "message": "Method not allowed"  
}
```

Make a POST request to add data

21. Change the method to POST.
22. Click the Send button; you should get a 415 response with a message of unsupported media type.

Method Request URL
POST ▾ http://training-american-ws.cloudhub.io/api/flights ▾

SEND :

Parameters ▾

415 Unsupported Media Type 261.25 ms

DETAILS ▾



```
{  
    "message": "Unsupported media type"  
}
```

23. Click the arrow next to Parameters beneath the request URL.
24. Click in the Header name field, type C, and then select Content-Type.
25. In the Header value field, select application/json.

Parameters ^

Headers	Authorization	Body	Variables	Actions
<input type="checkbox"/> Toggle source mode	+	Insert headers set		
Header name Content-Type	Header value application/json			

26. Select the Body tab.
27. Return to the course snippets.txt file and copy the value for American Flights API post body.

28. Return to Advanced REST Client and paste the code in the body text area.

Parameters ^

Headers	Authorization	Body	Variables	Actions
Body content type application/json				

FORMAT JSON MINIFY JSON

```
{
  "code": "GQ574",
  "price": 399,
  "departureDate": "2016/12/20",
  "origin": "ORD",
  "destination": "SFO",
  "emptySeats": 200,
  "plane": {
    "type": "Boeing 747",
    "totalSeats": 400
  }
}
```

29. Click the Send button; you should see a 201 Created response with the message Flight added (but not really).

201 Created 274.80 ms DETAILS ▾

□ ↻ ⌂ ⌃

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

30. Return to the request body and remove the plane field and value from the request body.
31. Remove the comma after the emptySeats key/value pair.

Headers	Authorization	Body	Variables	Actions
Body content type application/json				
		<p>FORMAT JSON MINIFY JSON</p> <pre>{ "code": "GQ574", "price": 399, "departureDate": "2016/12/20", "origin": "ORD", "destination": "SFO", "emptySeats": 200 }</pre>		

32. Send the request; the message should still post successfully.
33. In the request body, remove the emptySeats key/value pair.

34. Delete the comma after the destination key/value pair.

Headers	Authorization	Body	Variables	Actions
Body content type application/json				
		<p>FORMAT JSON MINIFY JSON</p> <pre>{ "code": "GQ574", "price": 399, "departureDate": "2016/12/20", "origin": "ORD", "destination": "SFO" }</pre>		

35. Send the request; you should see a 400 Bad Request response with the message Bad request.

400 Bad Request 291.33 ms DETAILS ▾

✖️ ↻ ⌂ ⌂

```
{
  "message": "Bad request"
}
```

Make a PUT request to update data

36. Change the method to PUT.
37. Add a flight ID of 3 to the URL.
38. Click the Send button; you should get a 400 Bad Request.

400 Bad Request 190.50 ms DETAILS ▾

□ ▶ <> ⏷

```
{  
    "message": "Bad request"  
}
```

39. In the request body field, press Cmd+Z or Ctrl+Z so the emptySeats field is added back.
40. Send the request; you should get a 200 OK response with the message Flight updated (but not really).

200 OK 225.24 ms DETAILS ▾

□ ▶ <> ⏷

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

Make a request to a secured API

41. Change the method to GET.
42. Change the request URL to <http://training4-american-api.cloudhub.io/flights/3>.

Note: The -ws in the URL has been changed to -api and the /api removed.

43. Click the Send button; you should get a 401 Unauthorized response with a message about an invalid client_id or secret.

The screenshot shows the Advanced REST Client interface. At the top, there's an orange bar with the status "401 Unauthorized" and a duration of "393.32 ms". To the right is a "DETAILS" dropdown. Below the status bar are several icons: a copy icon, a refresh icon, a comparison icon, and an eye icon. The main response body is displayed below these icons, showing the error message "Unable to retrieve client_id from message".

Below the response body, there are sections for "Parameters", "Headers", "Authorization", "Variables", and "Actions". The "Headers" section is currently active, indicated by a blue underline. It contains a "Toggle source mode" button, an "Insert headers set" button, and a table with three rows:

Header name	Header value	Actions
Content-Type	application/json	X
client_id	d1374b15c6864c3682ddbed2a247a826	X
client_secret	4a87fe7e2e43488c927372AEF981F066	X

50. Click the Send button; you should get data for flight 3 again.

Note: The API service level agreement (SLA) for the application with this client ID and secret has been set to allow three API calls per minute.

200 OK 1399.56 ms DETAILS ▾

Array(1)
-0: {
 "ID": 3,
 "code": "ffee0192",
 "price": 300,
 "departureDate": "2016-01-20T00:00:00",
 "origin": "MUA",
 "destination": "LAX",
 "emptySeats": 0,
 "plane": {
 "type": "Boeing 777",
 "totalSeats": 300
 }
}

51. Click the Send button three more times; you should get a 429 Too Many Requests response with a Quota has been exceeded message.

Note: The API service level agreement (SLA) for the application with this client ID and secret has been set to allow three API calls per minute.

429 Too Many Requests 209.24 ms DETAILS ▾

{
 "error": "Quota has been exceeded"
}

Use the API console in the API portal to make requests to the API using a mocking service

52. Return to the browser window with the American Flights API portal at <https://anypoint.mulesoft.com/exchange/portals/muletraining>.
53. In the left-side navigation click the GET method for /flights.
54. Review the Headers and Response sections.
55. In the Response section, select Examples.

Response

Type application/json

200 Type Examples

```
[{"ID": 1, "code": "ER38sd", "price": 400, "departureDate": "2017/07/26", "origin": "CLE", "destination": "SFO", "emptySeats": 0, "plane": {"type": "Boeing 737", "totalSeats": 150}}, {"ID": 2, "code": "ER451f", "price": 540.99, "departureDate": "2017/07/27", "origin": "SFO", "destination": "ORD", "emptySeats": 54, "plane": {"type": "Boeing 777", "totalSeats": 300}}]
```

56. In the API console located on the right side of the page, make sure the Mocking Service endpoint is selected.
57. Look at the endpoint URL that is displayed.

The screenshot shows a portion of an API console interface. At the top, there are two buttons: "Download" with a downward arrow icon and "Request access" with a key icon. Below these, a green "GET" button is visible. Underneath, the text "Mocking Service" is followed by a dropdown arrow. The URL "https://mocksvc-proxy.anypoint.mulesoft.com/exc" is partially visible, suggesting it's part of a larger endpoint path.

58. Click the Show optional parameters checkbox.
59. Select LAX in the destination drop-down menu.
60. Click Send.

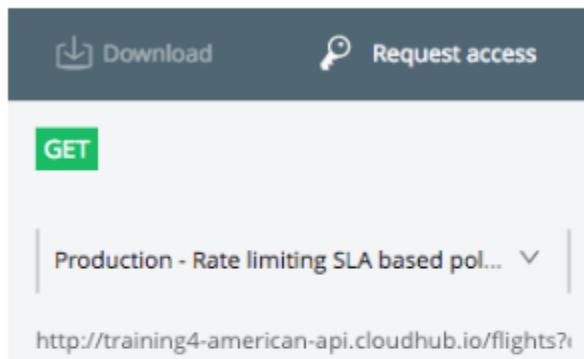
61. Look at the response; you should get the example flights that are to SFO.

The screenshot shows the API response details. At the top, there are tabs for "Parameters" and "Headers", with "Parameters" being the active tab. Below this, a section titled "Query parameters" has a checked checkbox labeled "Show optional parameters". A dropdown menu is open, showing "LAX" as the selected value. To the right of the dropdown is a blue "Send" button. Below the response area, the status "200 OK" and time "754.20 ms" are displayed, along with a "Details" link. At the bottom, there are download and copy icons. The main content area displays a JSON response:

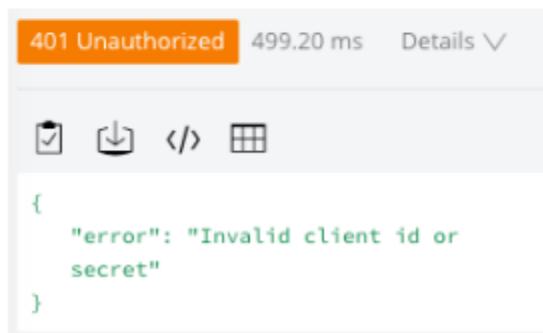
```
[Array[2]
-0: {
  "ID": 1,
  "code": "ER38sd",
  "price": 400,
  "departureDate": "2017/07/26",
  "origin": "CLE",
  "destination": "SFO",
  "emptySeats": 0}
```

Make requests to the API using an API proxy endpoint

62. At the top of the API console, change the endpoint to Production – Rate limiting SLA based policy.
63. Look at the endpoint URL that is displayed.

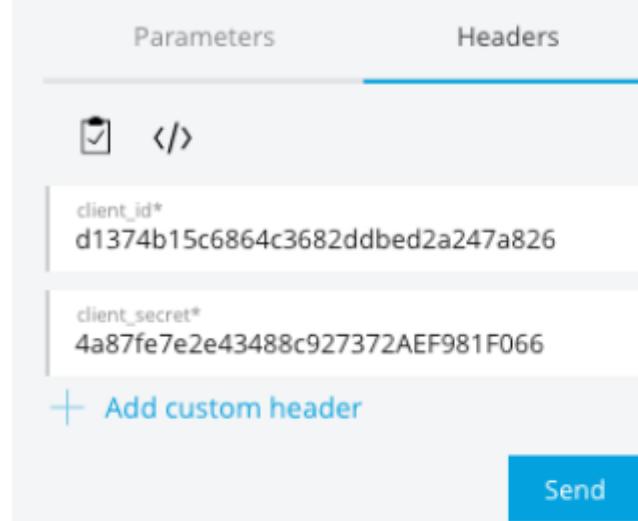


64. Click Send; you should get a 401 Unauthorized response.



65. Select the Headers tab.

66. Copy and paste the client_id and client_secret values from the course snippets.txt file.



67. Click Send; you should get a 200 OK response with only flights to LAX.

200 OK 1053.50 ms Details ▾

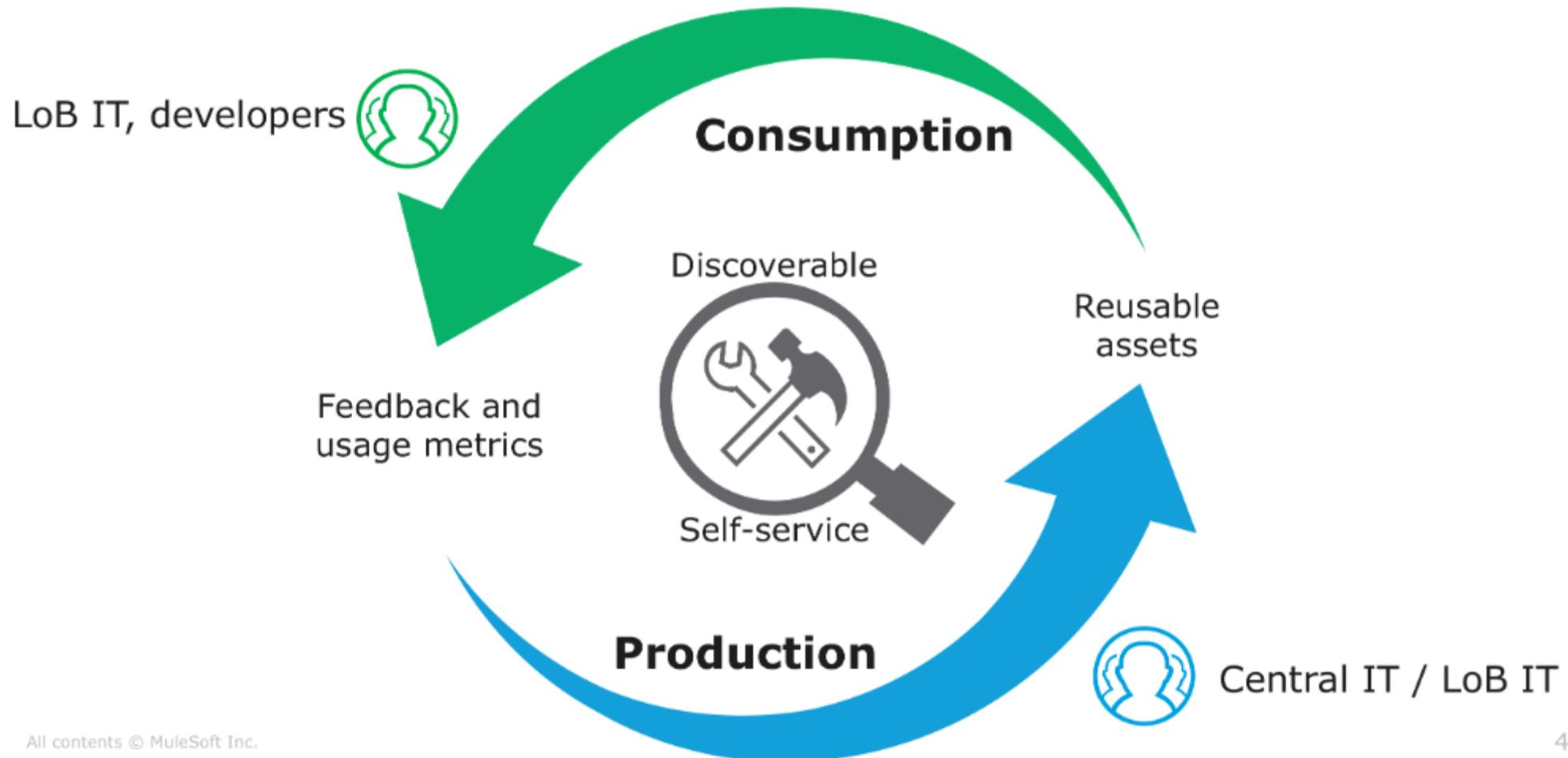
[Array[3]]

- 0: {
"ID": 1,
"code": "rree0001",
"price": 541,
"departureDate": "2016-01-20T00:00:00",
"origin": "MUA",
"destination": "LAX",
"emptySeats": 0,
"-nLane": 1}

Successfully creating application networks using API-led connectivity

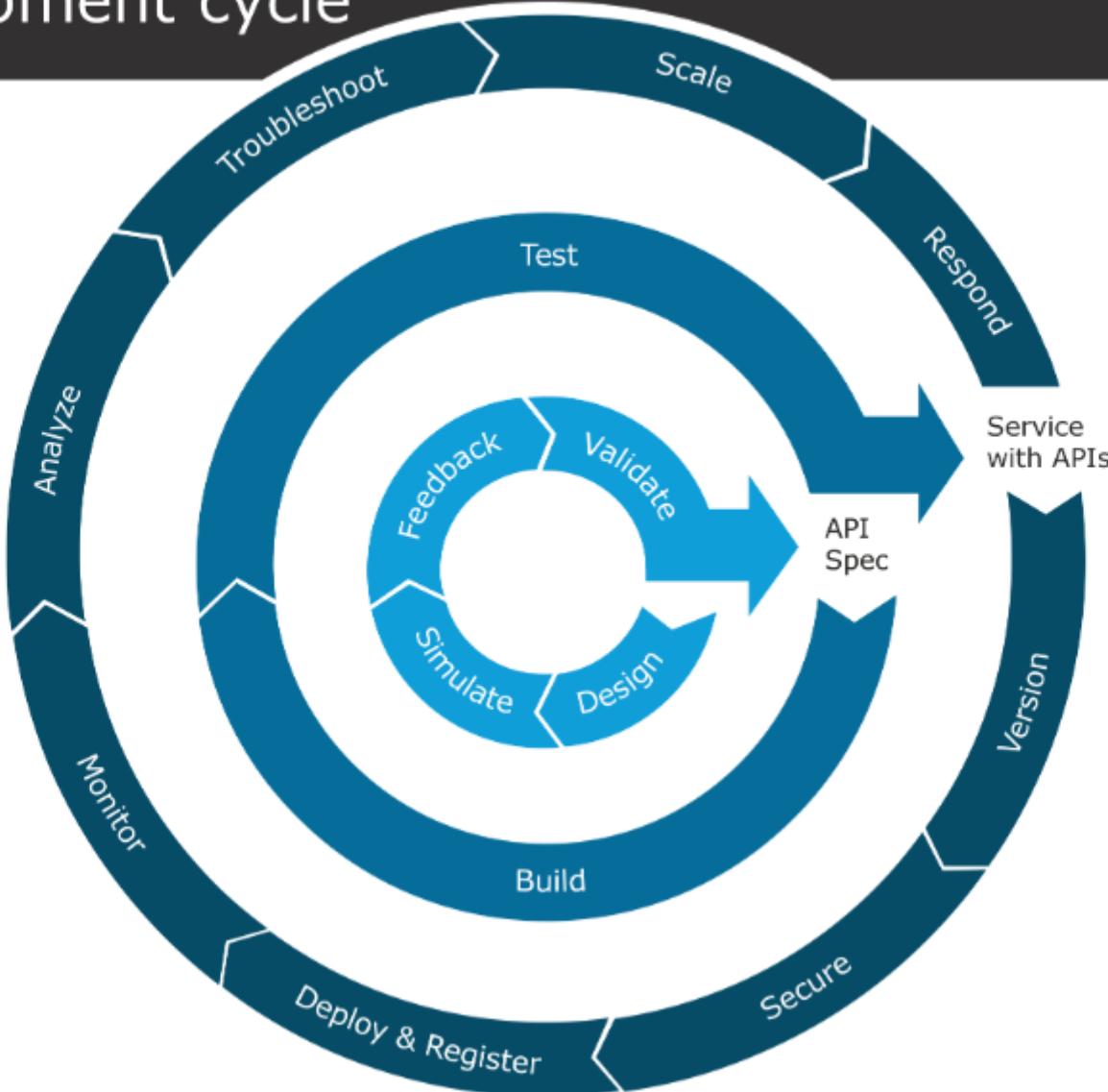


Producing discoverable and consumable assets is key



- 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

API development cycle



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

- 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



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

