POSTMAN

# The API-First Transformation


## Kin Lane

FOREWORD BY

Abhinav Asthana


# Hello World

# The API-First Transformation

# The API-First Transformation

Kin Lane

FOREWORD BY
Abhinav Asthana

**ART DIRECTION**

Shruthi Venkatesh

Hardcover ISBN: 979-8-9869518-0-5
eBook ISBN: 979-8-9869518-1-2

**PUBLISHED**

Postman
201 Mission Street
Suite 2375
San Francisco, CA 94105
+1 415 796 6470

www.postman.com

To John and Adam for lighting my API storytelling imagination with ProgrammableWeb. To Steve for funding APIStrat and API Evangelist for all the years. To my amazingly strong wife Audrey for enduring 12+ years of API blah blah blah while also teaching me the power of storytelling. To Abhinav, Ankit, and Abhijit for believing in my storytelling and providing me with a platform to bring my stories to life. To my Kaia Mhairi for ensuring I am still in the world to make all of this happen. And for Isaiah, we miss you kid.

# Contents

## PART 03 - OPERATIONS

# Foreword

Kin and I met almost a decade ago. Postman, the company I co-founded later, was just getting off the ground as a side project and as I learnt more about APIs, it was hard to escape Kin's writings on his blog - apievangelist.com. I started seeing references to the API Evangelist at conferences, in blog posts, and in the press. Soon after we were funded as a seed-stage startup, one of our investors sent a version of one of Kin's blog posts wanting to discuss how we think about the API landscape. I knew I had to get to know Kin!

Our early conversations taught me a lot and we stayed in touch. APIs were evolving at a rapid pace and as Postman grew to be more widely adopted, my belief strengthened that APIs were going to be the central building blocks of modern technology. However, explaining the value of APIs to technology leaders and business leaders was a challenge we started noticing. Both sides of the equation have to work together for successful API programs. Kin and I spoke about this often, and soon an opportunity opened up for us to work together formally with him joining us as our Chief Evangelist.

Kin's work at Postman expanded to talk to hundreds of customers, technology leadership, business leadership, ecosystem vendors, and the press. Kin is supported by an amazing team of technologists who contribute to several open-source initiatives like OpenAPI, GraphQL, JSON Schema, AsyncAPI, and many other areas. Kin's team also works with the developer community at large and helps them through tutorials, conference talks, and educational material.

A distillation of those lessons learnt is now in your hands in the form of this book. The common thread that you will find in the book is the notion of APIs being a strategic asset that the most forward-looking companies care about. They build and use APIs through a conscious strategy rather than just middleware, glue, or worse an afterthought. Well-built APIs are the foundation of strong technological powers, revenue drivers, and market cap multipliers. They also lead to happier and more productive development teams.

Building them and managing APIs well is an organizational skill that needs to be acquired and honed over time. While there are a lot of resources on the technology side, this book will help you see APIs from many perspectives. It'll also give you blueprints and ideas to help you think through your organization's API strategy. If you don't have one, this book will guide you in creating one.

That said, there is no substitute for actually executing a strategy. At Postman, we have used API-first strategies ourselves to help us scale from a 3-person startup to now what has become one of the fastest growing software companies in the world. I can attest first-hand to the advice that is shared in the book.

Abhinav Asthana
CEO and Co-founder, Postman

# Author's Note

I'm Kin Lane. If you've ever met me at a technology conference—or if you follow my API Evangelist blog or listen to my weekly Breaking Changes podcast, where I discuss the latest API topics with business and technology leaders across the globe—then you know I'm passionate about APIs. For reasons that will quickly become apparent as you read this book, I believe APIs are the single most important technological innovation in the world today.

I have spent the last twelve years evangelizing to companies, institutions, and government agencies about why APIs matter. In 2013, I served as a Presidential Innovation Fellow for the Obama administration, where I worked as part of the open data effort and assisted the Department of Veterans Affairs in streamlining their partner relations using APIs.

Today, I apply my knowledge as Chief Evangelist at Postman, where I talk with some of our largest enterprise clients about their API journeys. I also manage developer relations, lifecycle and governance, specifications and tooling, and our data, standards, and intelligence teams.

When my CEO Abhinav Asthana asked me to write a book combining my expertise with lessons I've learned from the customers and leaders I've interviewed on Breaking Changes, I leapt at the chance. While my work informs the road map for the Postman API platform, this is not a book about Postman. Instead, it is about stories I am hearing from companies that have shaped our technology and business landscape for the last 20 years–and are now defining its contours for the next 20. By seeing APIs from multiple perspectives, leaders in every industry will start to understand the unique benefits this technology has to offer.

Understanding APIs is powerful because they are the engine driving the great digital transformation within our companies and all around us. We can either sit back and allow this engine to pull us where it will, or we can learn to understand what it is, how it works, and how to direct its energy to our own advantage.

This book is for business leaders who choose the latter course—those who realize that the way information is created, distributed, and consumed will define tomorrow's organizations. If you learn how to execute APIs well, they will serve as a bulwark for your company and a source of value for years to come.

In this book, you will find the building blocks for constructing a successful future with APIs.

# Introduction: The API Opportunity

Every day, we are exposed to a digital firehose of information shaping our decisions. The shift to all things digital has been 50 years in the making, and the technologies precipitating it have taken many twists and turns along the way. But if you examine them closely, you will find that beneath the surface, they all have one thing in common—APIs.

APIs are the connectors that give meaning to long strings of ones and zeroes, translating disparate applications into products and services that bring value and convenience to customers—and, sometimes, riches to their creators. Consider these examples, a few of many:

- According to Fortune Business Insights, the global smartphone market size is projected to reach USD 792.51 billion in 2029, at a CAGR of 7.3%.[1]
- eBay's Buy APIs, which allows customers to place orders on third-party shopping sites, have generated 5 billion dollars of revenue.[2]
- Amazon Web Services generated $62.2 billion in 2021.[3]

All of these results were achieved through the skillful use of API technology. And the opportunities for creating value are by no means over—in fact, the API economy is still in the early stages.

Every organization stands to gain value from APIs, whether that means making internal processes more efficient, creating more innovative products, delivering better customer experiences, or all of the above. Only you can decide how to use the technology to your best advantage. What this book will do is equip you with the tools you need to get it done.

While we will delve into technical concepts, the book is certainly not written solely for

technology experts. We will purposely provide high-level information at every turn to help you fly over the dogma, tribalism, and vendor influence that often dominates API discussions. Neither will we dwell on any single technology or stage in the API life cycle. As a business leader, you need a broad perspective to make the right decisions for your company.

Despite what you frequently hear across the technology sector, there is no single strategy, technology, or operational approach to "doing APIs." Managing APIs well means ensuring your teams are equipped, educated, and incentivized to address the needs of your specific domain. While we have strong opinions on a handful of topics, such as governance and the API life cycle, our goal is not to convince you that we're right but to give you the ability to design solutions that meet your organization's needs.

The book is organized into three sections, which will help you:

- Create your API strategy,
- Assemble the right technology for your purposes, and
- Put the solutions to work in your operations.

The book is meant to be read, scanned, and picked up again after you have researched your design plans and bounced ideas off your teams. Information is often presented in easily-digestible, itemized lists that you can refer to again and again.

You will also find detailed descriptions of challenges and solutions, as well as references, guides, and other tools to help you plan the work you have ahead. And the narrative is interspersed throughout with real-life stories from top businesses and technology leaders, showing you how to apply their lessons to your own business, no matter what industry you are in.

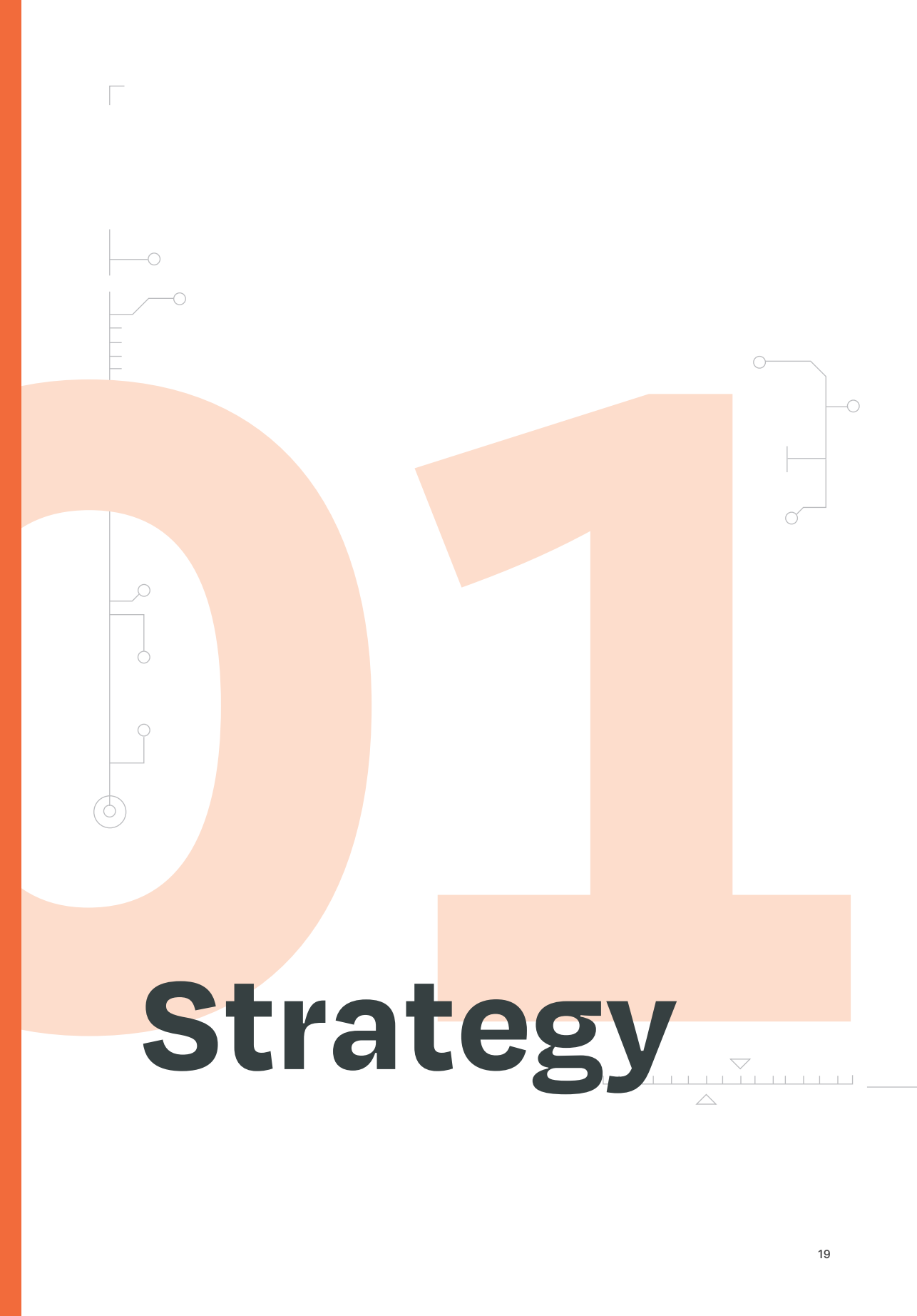
Despite its comprehensiveness, this book is not—and cannot be—a custom blueprint for your organization's API-first transformation. Only you can create that. Our goal is to arm you with knowledge and provide hundreds of ideas and examples to inspire you as you carve your own path in this promising, wide-open frontier.

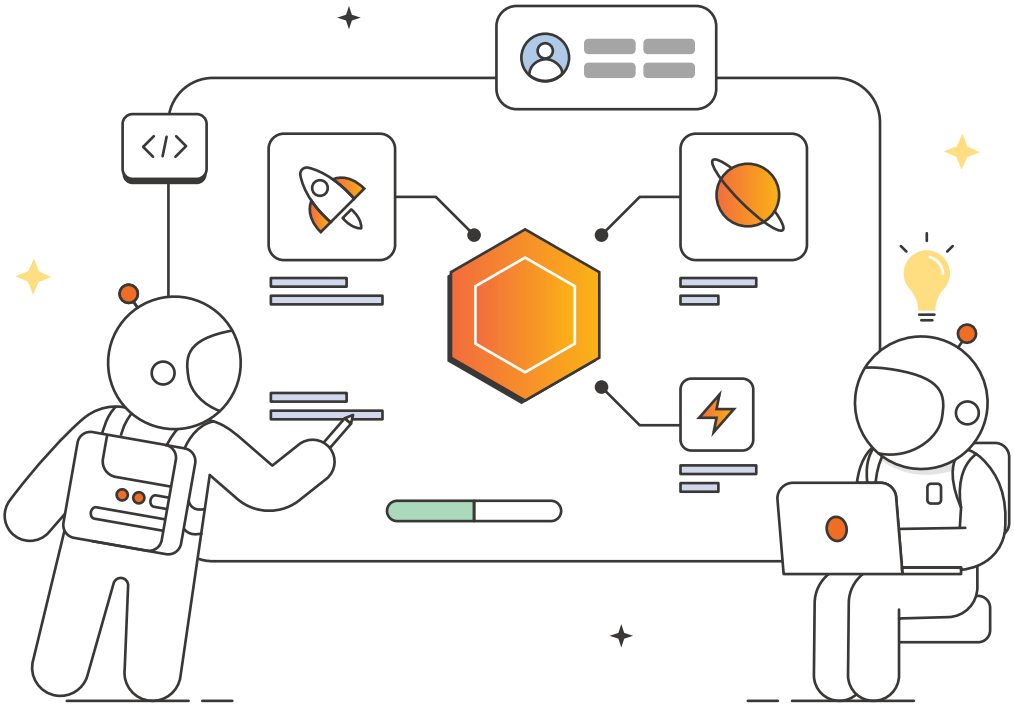
NOTES

1 "Smartphone Market Worth USD 792.51 Billion in 2029 | 7.3% CAGR," Global Newswire, 2022.

2 "How Ebay's Buy APIs Hit $5 Billion in Gross Merchandise Bought," eBay blog, 2021.

3 Amazon Reveals Its Most Profitable Business, Forbes contributor blog, 2022.


01
Strategy



# 01
# The Evolution of APIs

APIs are not new. They have been taking root for the last 50 years, but the evolution of business technology over the past two decades has created an unprecedented opportunity for them to dominate industries and create entirely new dimensions of trade. From the cloud to the gig economy, the internet is changing how we live and conduct business, with one common property across everything–APIs.

## 1.1 A Growing Market Opportunity

It took the last 30 years of the 20th century for the foundation of the digital economy to be laid. While we've seen some very significant shifts in business processes in the last 20 years—the introduction of the cloud and the gig economy, for example—this is just the start. Now that all of our digital resources, capabilities, and experiences can be defined as simple, scalable, synchronous, and asynchronous APIs, the future becomes more interesting, but also more volatile and competitive.

The growth of APIs, the expanding number of people working with them, and their reach across almost every industry reveal the scope of the opportunity. This isn't just about publishing the right set of APIs, it is about having the enterprise muscles to do APIs better than anyone else. This opportunity does not mean "Let's set up the right

APIs and we are done." It means honing the velocity, quality, and consistency of how your organization delivers and iterates upon your APIs.

### Technology, business, and politics

Because APIs were born in the technology realm, it can be easy to view them through a purely technical lens. It is also easy to see everything behind or in front of the APIs. But in reality, APIs are the technical, business, and political control valve for everything happening in the world now. This isn't hyperbole. APIs have been silently delivering every digital shift in the last 20 years, and while many have tuned into this reality over the last decade, most still see APIs as just a technical feat.

The business of APIs is how Amazon Web Services is becoming a greater revenue engine than Amazon proper. The politics of APIs is how Facebook was able to dominate social media with its Instagram and WhatsApp acquisitions. The API-first transformation isn't just about doing APIs, it is about doing them well and learning how to wield them in support of your business vision, understanding the politics of your industry, or creating entirely new categories.

Every enterprise today is doing APIs, but most of them aren't doing them well. They lack a formal strategy for how and why their teams are doing APIs. While you can study API pioneers like Stripe and Twilio for planning your API strategy, it is better to go all-in on your own API-first transformation and develop your organizational muscle for reliably delivering internal and external APIs. That is what will help prepare your company for the future.

## 1.2 Deconstructing the API

To understand the market opportunity, it helps to understand what an API is and why being API-first matters. In an internet-connected world, desktop, web, mobile, and device applications are designed for humans to use. APIs, on the other hand, are designed for systems, applications, and integrations to use. Websites and APIs both do the same things, such as return data, content, images, videos, and put algorithms to work. The only difference is that APIs don't return all the details needed to make things look pretty to the human eye—you only get the raw data and other machine-readable information needed to make the resources behind the scenes work.

### What are APIs?

Let's break down what an API is before we explore what they are capable of, learning a bit more about how we interface with them to programmatically apply digital resources and capabilities.

**(A)pplication** - You are "applying" something digital, often on a desktop, web, or mobile device, by connecting it to the Internet. Applications bring the web into our homes, workplaces, cars, and the public sphere.

**(P)rogramming** - Programming means allowing for something to be programmed, executed, and automated, making something that is repeatable and easy to use in software; allowing our physical and online world to be automated to do the things we can't or don't want to do.

**(I)nterface** - An interface is the point where two entities meet and interact, enabling communication between two systems.You apply digital resources, capabilities, and networking through servers to link to common, everyday objects, allowing the world around us to interact with us.

Application Programming Interfaces–APIs–are how you standardize, automate, and apply the digital resources and capabilities defining how the digital world works in the 21st century.

## How to think about APIs

When learning about abstract digital concepts, it helps to lean on real-world analogies. Here are two ways we like to help introduce people to APIs

**Restaurant Menu** - APIs have been compared to a restaurant menu, providing you a list of what is available and a set of instructions for ordering the items you want.

**Utilities** - APIs are often compared to your electricity, telephone, and plumbing systems, but their use extends to an endless mix of digital resources and capabilities that you access via digital interfaces.

Every digital resource, capability, and experience you encounter through your computer and mobile phone is API-driven. APIs define the raw digital resources your business produces and depends upon. Being able to produce and consume APIs effectively enables business to get done.

Your enterprise is increasingly defined by the control you have over your digital operations, and APIs define what you are capable of as an organization. Ultimately, it is up to you to define what APIs mean for your organization and the impact they will have on your industry.

# 1.3 Early Seeds

APIs emerged in the earliest days of digital computing back in the 1950s, later evolving to meet the needs of a variety of business sectors.

The computer and network origins of APIs began with these investments by the U.S. military, universities, and industries:

**Sage** - In 1954, the Semi-Automatic Ground Environment (SAGE) began its six-year development to be used as an early warning air defense system.

**Saber** - In 1964, IBM built upon SAGE to develop Semi-Automatic Business Research Environment (SABRE), an air travel reservation system for American Airlines.

**Arpanet** - The early stages of the Advanced Research Projects Agency Network (ARPANET) began in 1966—planting the seeds for what we now know as the Internet.

## Laying the foundation

Networks opened the opportunity for information to be shared across multiple physical locations.


**FTP** - In 1971, an MIT student named Abhay Bhushan published RFC 114 with the original specification for the File Transfer Protocol (FTP). to transfer files between two computers.


**EDI** - In the 1960s, electronic data interchange (EDI) was introduced to manage shipment supply chains in the U.S. Army by digitizing physical shipping manifests.

## Creating a business model

Over the years we realized we needed more structured approaches to exchanging business information, and began using networks that were rapidly becoming what we now know as the internet.


**Corba** - In 1991, the Object Management Group (OMG) introduced the Common Object Request Broker Architecture (CORBA) allowing applications to communicate using APIs.


**SOA** - In 1998, the idea of service-oriented architecture (SOA) emerged to provide more structured approaches to transmitting business data between networked servers.

As the web emerged, industry forces came together to formalize XML specifications in service of an SOA vision for the industry. But a simpler, lower-cost, and increasingly ubiquitous web would provide a much more powerful approach for delivering our digital resources and capabilities.

While taking shape for several decades, modern web APIs began to emerge in their current form during the e-commerce and social networking revolution of the early 21st century. They were pushing what was possible when it comes to buying and selling in the digital world, but also realizing that consumers are very social creatures, and we'd want to bring our friends, family, and followers along for the ride.

# 1.4 APIs in Commerce and Social Media

## APIs in commerce

Early web APIs escaped from the controlled service-oriented architecture (SOA) experiment in the enterprise and began to be applied to sales, products, affiliates, auctions, and the other expanding areas of the e-commerce shift.


**Salesforce** - From day one, Salesforce had a crude set of XML over HTTP web APIs to help the new startup compete against established thousand-pound gorillas, changing the conversation around how we manage our business relationships.


**Amazon** - The commerce powerhouse was using APIs to expand its network of sellers via affiliate marketing networks, but would forever change how we do business as part of its own API-first transformation, which resulted in what we now know as Amazon Web Services.


**eBay** - eBay saw early on the potential of APIs to transform the auction business. It also saw the value of ongoing investment in API-first transformation, helping the organization evolve beyond auctions to become the commerce giant it is today.

## APIs in social media

The digital experience was rapidly becoming a social affair, with images, links, and other digital resources becoming more shareable via APIs. APIs were also expanded to define our profiles, connections, and the networks where we share these images and links.

**Flickr** - The image platform saw the potential of APIs for making images available across the fast-growing world of blogging. It also saw early on how APIs enabled the next generation of business development through self-service offerings.

**Facebook** - The top social network platform has leveraged its original platform to build out a global platform of human interaction, gaining a competitive edge by featuring images with Instagram and messaging with WhatsApp.

**Twitter** - The entire Twitter empire was built in its API community. When the company began, it was a simple interface. Once the API was released, we got a mobile application, buttons, widgets, and everything else that has helped the network grow.

Commerce plus social media would provide the rich substrate we would need to begin building the new digital economy, but the system was missing other essential concepts needed to take us into the future. Selling products and connecting people were powerful uses of the web. Later, APIs would demonstrate the power of reducing common digital objects into API commodities.

Web APIs were increasingly used for industrial-grade purposes, making common building blocks of information technology (IT) available as simple XML or JSON APIs. APIs further commoditized essential technical resources for businesses, shifting how we deploy and pay for our IT infrastructure while making it more elastic and scalable to support the future of the business.

As the web economy was picking up momentum, the foundation was being laid for a new economy, setting the stage for a transformative moment in business that we call the cloud.

**EXPERT PERSPECTIVE**

## eBay responds to a changing commerce landscape


eBay was one of the first three web API pioneers. Alongside Salesforce and Amazon, eBay saw the potential of exposing XML APIs using the web early on. After sitting down with Tanya Vlahovic from eBay, it soon became clear

to me that the company didn't just dominate world auctions and commerce with APIs early in this century–it has continued to double down and invest in its API-first transformation.

At the heart of the eBay strategy is an OpenAPI, contract-driven approach to designing and operating APIs. eBay is using OpenAPI because it keeps documentation up-to-date, generating code libraries and other developer essentials using machine-readable specifications. eBay uses OpenAPI as the default contract for its relationship with its massive developer ecosystem, but the company is also rapidly expanding to use AsyncAPI as well.

You can see the API-first muscles eBay has developed over the last few years, not just in the number of APIs but in the pace of change evident in the company's developer portal. You can see the contracts for their APIs alongside the documents, with YAML and JSON links for each version. To the untrained eye, this looks like just part of the developer experience. But in fact, it is a reflection of the velocity and agility the company achieved by standardizing how they do APIs and iterating on them quickly, building their API-first transformational muscles.

Another sign of eBay's API-first muscles and confidence is how vocal the company is in telling the story of its journey. Tanya fully understands the importance of explaining how eBay does APIs and why they support open-source API specifications like OpenAPI through the Linux Foundation. She also sees the impact her team members have when they speak at API conferences and write guest posts on top technology blogs. eBay is seen as a thought leader because of its active API storytelling.

While many enterprise organizations are just starting to invest in their API-first transformation and keeping API efforts focused on greenfield projects, eBay is generating billions of dollars by being API-first in the core aspects of its business. This API-first investment has allowed the company to make a significant contribution to revenue, but is also allowing it to innovate and remain agile. In a shifting commerce landscape, eBay has a unique view of its user and developer communities.

# 1.5 Amazon: The API Moves to the Cloud

In 2006, Amazon forever changed how we do business on the web with the release of Amazon Web Services, demonstrating that you can deploy global infrastructure using simple web APIs. Amazon S3 and EC2 not only changed how we would do business, they showed us the way forward for delivering the industrial-grade digital building blocks we needed with APIs.

**S3** - Through its internalization of an API-first mentality, Amazon began to see their infrastructure in different ways, rethinking how they made storage available not just to scale their operations, but also by introducing Amazon Web Services, providing others with storage in the clouds.

**EC2** - On the heels of storage in the clouds, Amazon Web Services (AWS) launched [4]EC2 and elastic compute services for deploying different sizes of servers for hosting websites, APIs, and anything else you would need to host, paying only for what you used.

**RDS** - Continuing its expansion across essential IT infrastructure, AWS launched RDS, making common database platforms available as scalable APIs, providing MySQL, and eventually SQL Server, Postgres, and other fundamental data solutions to suit enterprise needs.

## The AWS origin myth

According to a famous blog post by former Amazon engineer Steve Yegge, sometime around 2002, Amazon CEO Jeff Bezos issued a mandate to his still-young startup online retail business, changing not just the DNA of Amazon, but the way we would define our businesses in the years ahead. He stated:

- All teams will henceforth expose their data and functionality through **service interfaces**.
- Teams must communicate with each other through these **interfaces**.
- There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, and no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
- It doesn't matter what technology they use.
- All **service interfaces**, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

The mandate closed with: **Anyone who doesn't do this will be fired. Thank you; have a nice day!**

This landmark API-first mandate at Amazon would change the trajectory of the online retailer, but also set in motion another division that would eventually rival its core business, and change how companies operate across almost every sector.

Time has shown that the AWS origin story is more myth than reality, but it doesn't matter. The AWS story is part of API folklore and is used to describe API-first transformations across almost every industry impacted by APIs today.

**EXPERT PERSPECTIVE**

## API elasticity in the clouds


Arun Narayanaswamy, Director of Engineering at *Amadeus Labs*, joined us on Breaking Changes to help define the symbiotic relationship between APIs and the cloud. Arun helped articulate how cloud-native developed from cloud APIs and how it will help us deliver the next generation of API infrastructure, which will power the APIs to scale operations in the coming years.

I enjoyed learning about how cloud-native elasticity is one and the same with the agility and nimbleness introduced by being API-first. Being successful with the cloud is all about embracing the technology that makes it so powerful, and changing how you design, deliver, operate, scale, and evolve your infrastructure to take full advantage of its vertical and horizontal scalability. Cloud elasticity is the fabric of agility for operating and evolving your APIs, while also automating the infrastructure management underlying them.

Arun shared a very pragmatic view about the pros and cons of operating in the cloud while continuing to use your own on-premises data center. APIs enable you to integrate and migrate between the cloud and your on-premises environment. You can burst some operations to the cloud, or use the cloud only for the APIs you need to be the most elastic and scalable. Arun shared many reasons why you might eventually move beyond your data center–for example, to save on costs or operate in specific geographic regions. But he also noted that there are many security, legacy, and other considerations that may

suggest maintaining your data center makes sense. Either way, APIs are playing a significant role in shaping the balance between the on-premises environment and the cloud. Companies must consider what works best for their teams.

One realization I had in speaking with Arun about the cloud-native layer of operations is that the elastic infrastructure we are using behind our APIs not only allows us to automate our operations, but the billing system for our cloud infrastructure also has APIs. Cloud billing in APIs means we are able to test, configure, automate, and develop with greater awareness of what it costs to operate and scale our API infrastructure. Being able to automate and optimize cost management of our API infrastructure will be essential moving forward, and will allow us to optimize the balance between what we spend on our cloud infrastructure and the value we gain from the digital experiences we deliver with our APIs.

The introduction of the cloud was a milestone, showing us that APIs could be used to deliver global infrastructure. The next milestone is realizing the role of being cloud-native in our API-first transformation, and the symbiotic relationships our APIs have with the cloud.

NOTES

4 "The Secret to Amazon's Success—Internal APIs," API Evangelist blog, 2012.

# 02
# APIs at Scale: From Mobile Devices to the IoT

In the early days of the internet, you trudged to the office and logged onto a company PC–only to wrestle with applications that didn't work very well or do very much. Now internet applications are part and parcel of our daily lives–we seamlessly float from one to another at work, at home, and everywhere in between.

How did we get here? It couldn't have happened without APIs. This chapter will show how they vastly broadened the spread of digital resources, starting with the first mobile phone and culminating in today's worldwide API economy.

## 2.1 Going Mobile

Mobile phones have been around since the 20th century, but in 2007 everything changed with the introduction of the Apple iPhone. These new mobile devices popularized a new category of Internet-connected applications we carry in our pockets, requiring an entirely new set of digital resources and unlocking an unlimited realm of possibilities and internet-enabled digital capabilities.

## Devices

Many types of mobile devices and operating systems have emerged in the last couple of decades, but the iPhone from Apple and Android OS continue to dominate the mobile landscape.

**iPhone** - In 2007, Apple forever changed how we build software with the introduction of the iPhone, which also introduced the mobile application ecosystem we now take for granted. This new system changed the way businesses make digital resources available.

**Android** - Following Apple's lead, in 2008, Google launched the Android mobile phone operating system. Eventually, this open-source software became the preferred OS for all handheld devices.

## Digital resources

After early mobile application developers realized they could deliver digital resources on mobile applications, waves of API providers emerged to provide the essential digital resources to power the growing spectrum of mobile applications. A few of many examples include:

- **Twilio (SMS)** - Twilio made it simple to send and receive SMS messages using a well-defined web API.
- **SendGrid (Email)** - SendGrid puts enterprise-grade email within reach of fast-growing startups.
- **Stripe (Payments)** - Stripe abstracted away the complexities of the payment system using APIs.
- **Google Maps (Maps)** - Google provided an API modeled after the hacks its own developers were using to create maps.
- **YouTube (Videos)** - Youtube conquered the web using APIs to allow users to embed their videos.
- **Instagram (Images)** - Instagram used APIs to bring images to mobile devices.
- **WhatsApp (Messaging)** - WhatsApp modernized global communication using simple APIs.

Our personal and professional lives have become increasingly defined by the APIs that power our mobile applications, link us to global markets and news, and connect us to our family and friends.

# 2.2 The API Economy

The phrase "API economy" is often used to describe revenue opportunities accrued from making APIs available to third-party developers. While this represents one slice of the economy, there is a much larger opportunity for API providers ahead. We are seeing waves of startups and enterprises leveraging an API-first approach to provide what consumers are looking for, adding entirely new layers to the global economy.

## Pioneering services enabled by APIs:

Uber

Ridesharing (ex., **Uber**) - APIs shook up the taxi industry, bringing new competition and forcing a very entrenched business sector to move forward with its own digital transformation. Uber's aggressive API-driven approach connected drivers with people who needed rides worldwide.

GRUBHUB

Food delivery (ex.,**GrubHub, DoorDash, Instacart**) - Grubhub has put APIs to work to enable food and restaurant delivery via mobile phone.We now take these services for granted. APIs also gave Instacart the edge to make delivering groceries and other goods a professional gig.

DRIZLY

Alcohol Delivery (ex., **Drizly**) - As with groceries and restaurant food, we now expect to have beer, wine, and spirits brought to our homes, shifting how we drink and socialize.

All of these services had been tried at one time or another, but it took the perfect storm of APIs, the cloud, our mobile phones, and a pandemic-driven shift in priorities to make them a normal part of how we live.

**EXPERT PERSPECTIVE**

## 7-Eleven manages a crisis and transforms its business model by becoming API-first


For my *Breaking Changes* interview with Shadi Fallah of 7-Eleven, I expected a mildly interesting conversation about the importance of APIs in powering mobile applications. I'll admit, I was biased out of the gate—to my mind, brick-and-mortar retail does not pencil out to a compelling API-first story.

I was wrong. After talking with Shadi Fallah for five minutes, I realized that not only did she have a clear grasp of APIs' capabilities, she had driven the retailer's move from API-aware to API-first.

Shadi shared with me the battles she fought to convince her company's leaders to begin an API-first journey. Like many traditional brick-and-mortar companies, 7-Eleven was starting to invest more in technology with the eventual aim of becoming a technology company that sells retail goods. Nevertheless, its main focus was maintaining its dominance as a retail commerce leader. Under Shadi's influence, the company switched to a platform approach to APIs, creating feedback loops with partners and consumers to help developers iterate faster to deliver the services people wanted.

Her timing couldn't have been better. As the world began to shut down during the pandemic, neighborhoods across the U.S. depended on their local convenience stores to obtain essential goods. 7-Eleven needed to step up to meet this need, and to prepare for further changes after the pandemic ended. An API-first approach allowed 7-Eleven to remain competitive while rapidly innovating to respond to shoppers' changing needs during this critical time.

But that's not all. I asked Shadi how the company handled competition with all the new grocery and alcohol delivery startups. She said that becoming API-first allowed the company to turn these competitors into partners, helping 7-Eleven navigate the regulatory complexities of delivering alcohol in 50 states and adapt in other ways to the new realities of the pandemic.

By becoming API-first, 7-Eleven was able to accelerate its behind-the-scenes processes and realize its long-term vision of becoming a technology company. And having a responsive API platform ensures that the company will be prepared for the future, whatever it may bring.

## 2.3 APIs Power the Internet of Things (IoT)

Once it was discovered that simple, low-cost web APIs would work for low-latency mobile applications, developers began using the same approach for sending messages and data between common everyday objects. APIs began connecting devices across our personal and professional lives, bringing the online world into the physical world and helping to automate and optimize how we do business.

Cameras, sensors, drones,and other internet-connected devices are being applied across a range of industries, introducing “smart” capabilities into existing real-world processes and using APIs to send and receive data from the cloud. IoT startups are tapping into APIs to make cloud infrastructure available, and iPhone and Android mobile applications are using APIs to power everyday objects.

**Manufacturing** - APIs are reshaping the manufacturing supply chain, linking the physical world with the digital world and moving the factory floor into the clouds. Processes are becoming more elastic, helping manufacturers keep up s online and offline, no matter where business is happening.

**Transportation** - Our cars, roads, and transit systems are being transformed through APIs, changing our experience as we commute to work and take trips with the family. Our lives have become more connected, no matter where we are.

**Energy** - APIs are modernizing our energy grid, digitizing monitoring and other processes to deliver the energy we need to live to our homes and businesses.

**Retail** - Commerce has moved online in the last 20 years, and the internet has crept into the physical, point-of-sale retail experience. Now the physical experience is becoming embedded with digital signage, sensors, cameras, and other connected devices to help shoppers find what they want and allow retailers to track their habits and purchases.

**Cities** - Our cities are being connected to the internet, bringing streets, parks, and public spaces online. Services can be improved by connecting the physical world to the cyber world, thanks to the APIs that stitch everything together.

**Healthcare** - Healthcare interoperability is a priority for hospitals, and public health agencies depend on APIs to bring the patient experience into the modern age, making sure healthcare records are available wherever they are needed.

**Agriculture** - As we work to feed growing populations, the agriculture industry is undergoing a wave of digitization. Whether they are used by drones, sensors, tractors, or the food distribution network, APIs are being applied in new and compelling ways to change how we grow and supply food.

The exchange of data through internet-connected objects is transforming every business sector. The internet-connected landscape of the future will be defined by low-cost web infrastructure, which can be used to power more efficient and reliable APIs.

EXPERT PERSPECTIVE

# The automobile experience of tomorrow


John Musser, Director of Engineering at Ford Pro, recently talked with me on Breaking Changes about the importance of APIs in the automobile experience of tomorrow. Like many traditional companies, Ford is rapidly becoming a technology company that happens to produce physical products–which are increasingly connected to the internet through APIs. The automobile experience of tomorrow will completely re-evaluate our relationship with the road, the communities we drive through, and our personal relationship with the car, as well as address wider questions about transit and transportation.

Today's automobile is defined by hundreds of sensors and internet-connected parts, from the engine to the entertainment system. APIs enable these sensors to work in concert, and they leverage the cloud to ensure that the automobile experience is not an isolated affair. From GPS systems to fuel gauges, the modern automobile is deploying APIs to make our vehicles smarter, more aware, and capable of doing more work for us. But this is just the beginning. As each API-driven automobile connects to an overall API-defined network of services, it is helping to deliver the transportation vision of the future.

Already, as we drive our vehicles, they are searching for the next gas station. While we are parked at work, APIs are making our trunks a delivery destination for the packages and groceries we'll need when we're ready to head home. APIs connect commercial fleets with the communities they serve, bringing greater efficiency to business operations. The API-driven experience of service providers, delivery drivers, and taxi cabs is perpetually optimized to improve customer service and help companies meet their business objectives, while also complying with local regulatory requirements.

At Ford, APIs aren't just about automobiles. The technology defines the company's core business model and supply chain, as well delivering a new driving experience for our personal and professional lives. I view Ford as a car company, but it was interesting to listen to the narrative around their API-first

journey and their belief that they are a digital experience company, transcending the automobile experience as they move into bike sharing, energy, and everything else involved with transportation.

It can be easy to focus on APi-first as a technology transformation, but as I've seen over and over in my conversations with enterprise leaders, its real value lies in what it does for businesses, industries, and the people they serve.

# 03
# Charting API Growth

APIs are growing exponentially across every industry and enterprise. In fact, so many APIs are created to support desktop, web, mobile, and device applications that no single company knows where all its APIs are!

There are many ways to quantify the growth of APIs and explain how, where, and why they are used. But instead of throwing a bunch of statistics at you, we decided to convey the information visually, in charts and graphs, so you can see at a glance what the numbers mean.

The images that follow were generated from data obtained from 20 million Postman users and 30,000 respondents contacted for our 2022 State of the API report.

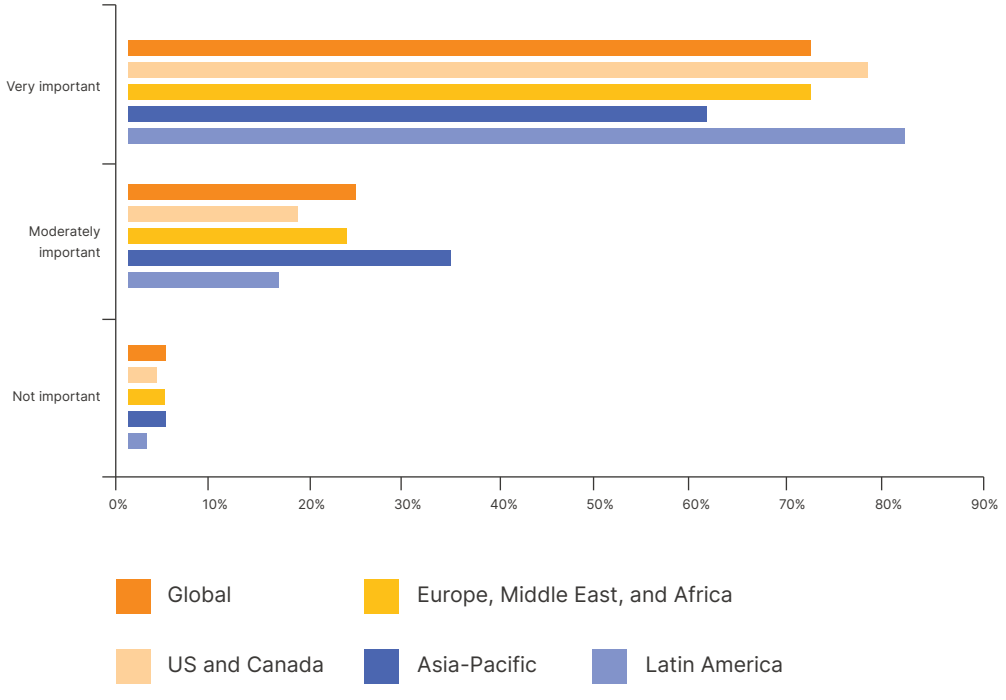
## 3.1 A Pandemic Accelerates Adoption

The pandemic introduced changes in how we live, work, and operate our businesses. In the enterprise world, it accelerated the API-first transformation across almost every business sector as companies scrambled to make a sudden, wide-scale shift to digital operations.

## Remote work

It has become clear that work processes implemented during the pandemic are here to stay and will need ongoing support.

One of the major changes is the demand for remote work. In the API economy, teams across generations believe they can accomplish everything they need to do from wherever they are.




## Investing in the future

There is also an appetite across teams for more investment in work-related APIs, revealing that people have become more confident in the ability of APIs to help them accomplish their daily tasks more efficiently.

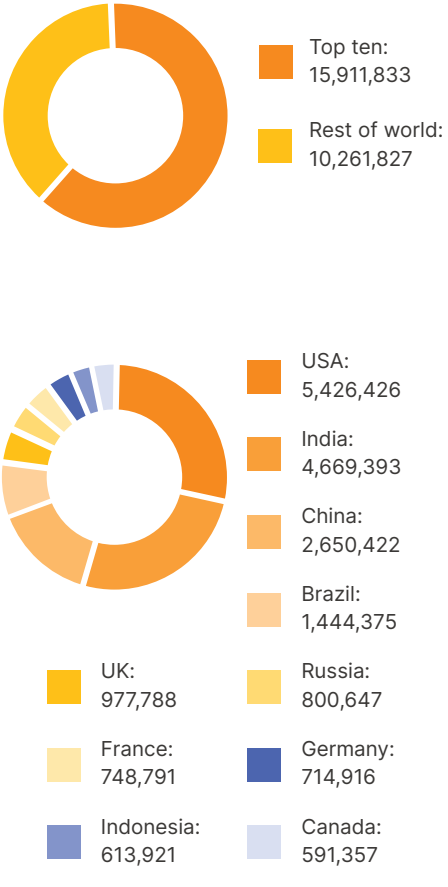
# 3.2 The Global Growth of APIs

More companies are developing, testing, and using APIs, as we can see from the expanding number of API collections across the globe. In addition, more types of workers are using APIs, especially in technology and business services, but in other sectors, such as finance and healthcare.

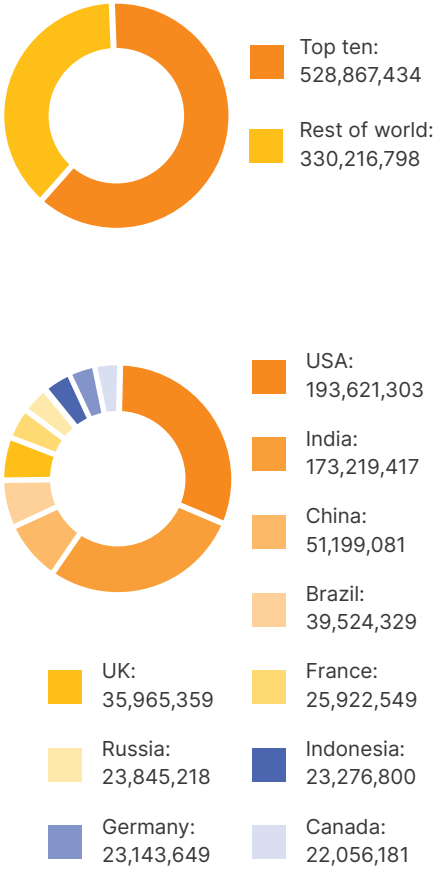
## Number of collections created

Collections have become a unit of representation for individual APIs, but also for the documentation, testing, and other essential elements of the API lifecycle. They reveal the executable, shareable, reusable representation of the business value defined by APIs. This global growth in collections reflects not only the greater number of APIs but, more importantly, the increase in productivity and the improvements in quality and governance required to reliably manage thousands of APIs.

NUMBER OF COLLECTIONS CREATED



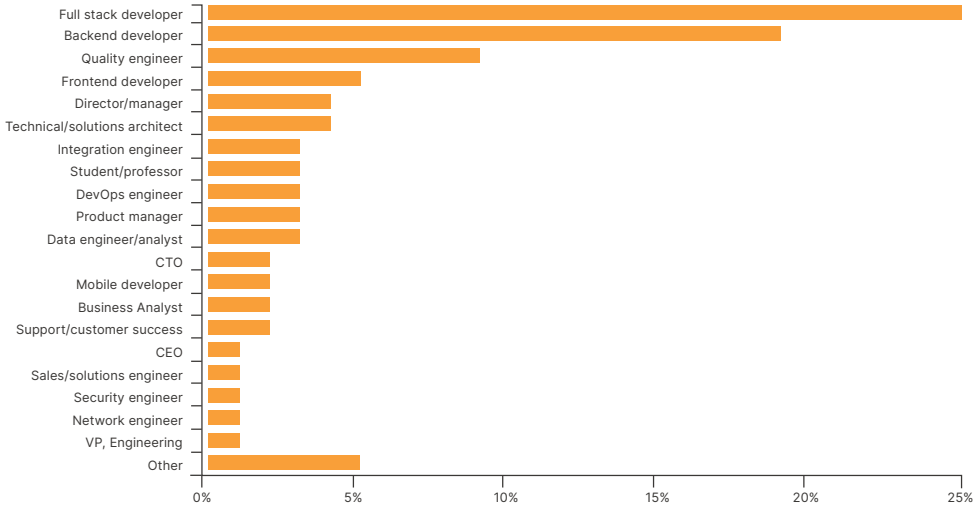

NUMBER OF REQUESTS CREATED



## Who works with APIs

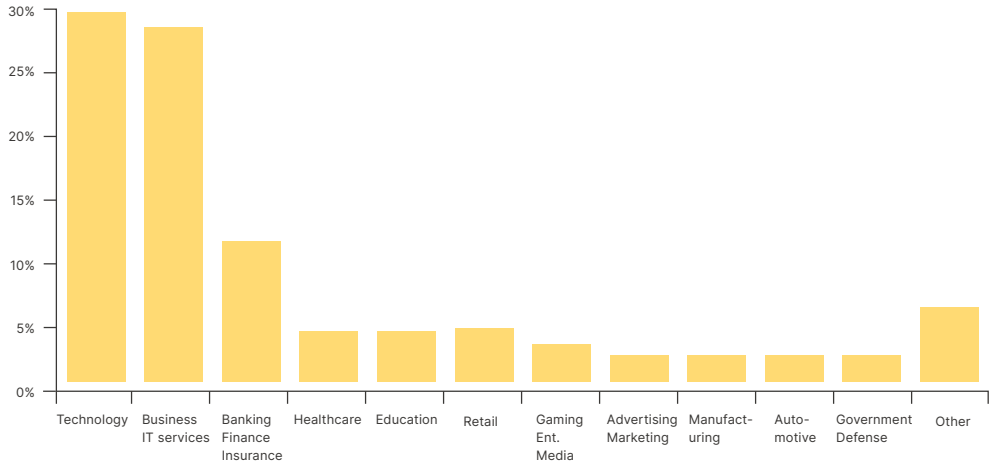
APIs have evolved from something developers do to something that nearly all business stakeholders are actively involved in. This evolution is rapidly expanding the roles of people involved in the API lifecycle, bringing APIs out of the shadows of IT and into the business spotlight.

There has been a steady uptick in the types of employees working with APIs, beginning with logical groups like quality and security teams, but also more product managers, analysts, and even sales teams, all of whom are putting APIs to work as part of their core tasks.




## APIs across industries

The API landscape has been expanding for the last decade, but in the past few years we've seen it shift to the mainstream. Of course, APIs are still dominant in the technology and business services sectors, but growth in financial services and healthcare has been exploding. Other industries, including education, retail, and even government agencies, are also getting into the API game.

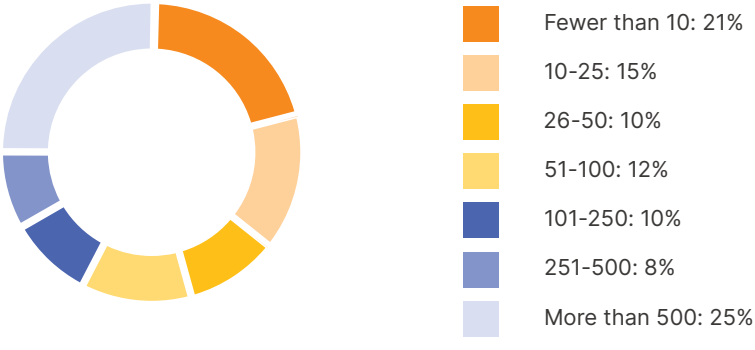



## Developers, teams, and APIs

Companies are hiring more API developers, and teams are spending more time working with APIs, which have become a top priority at organizations. Everyone is beginning to think more about APIs and use them to navigate everyday tasks.

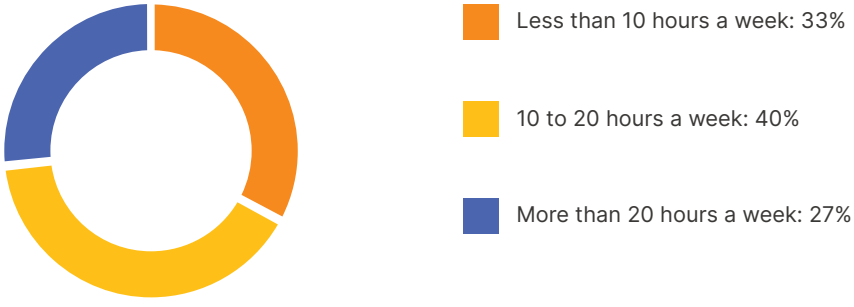
## More developers

The roster of API developers is growing as companies bring in people with a range of skills to manage the technology.



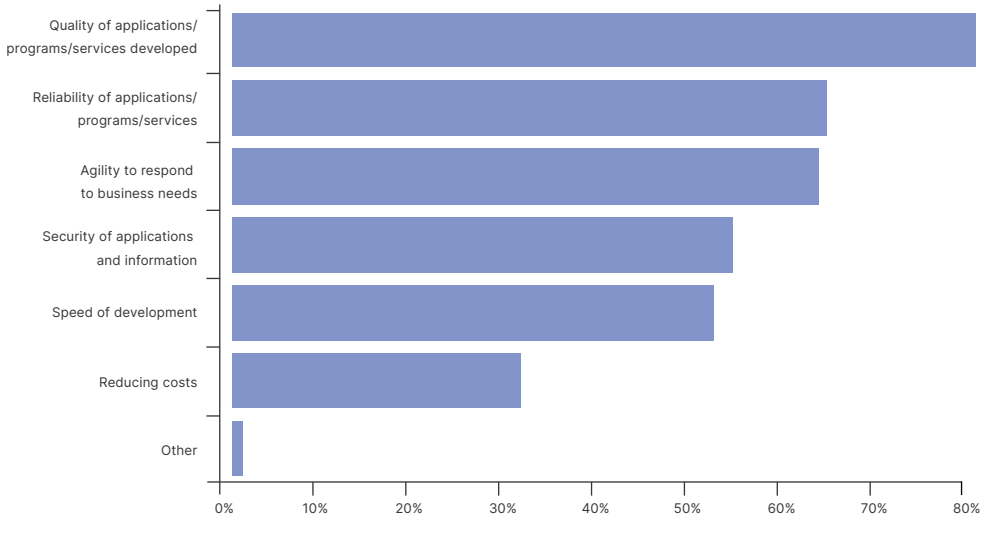
## More API users

Teams are now spending a significant amount of their time working with APIs.




## Development priorities

Teams are being told to focus on API quality, agility, reliability, and security, ensuring that digital resources meet business requirements.




## Public vs. private vs. partner

While there is a rapidly growing number of publicly available APIs, many companies still have a lot of anxiety about making APIs available to partners and third-party developers.

Private (used only by your team or your company): 58%

Partner (shared only with integration partners): 27%

Public (openly available on the web): 16%

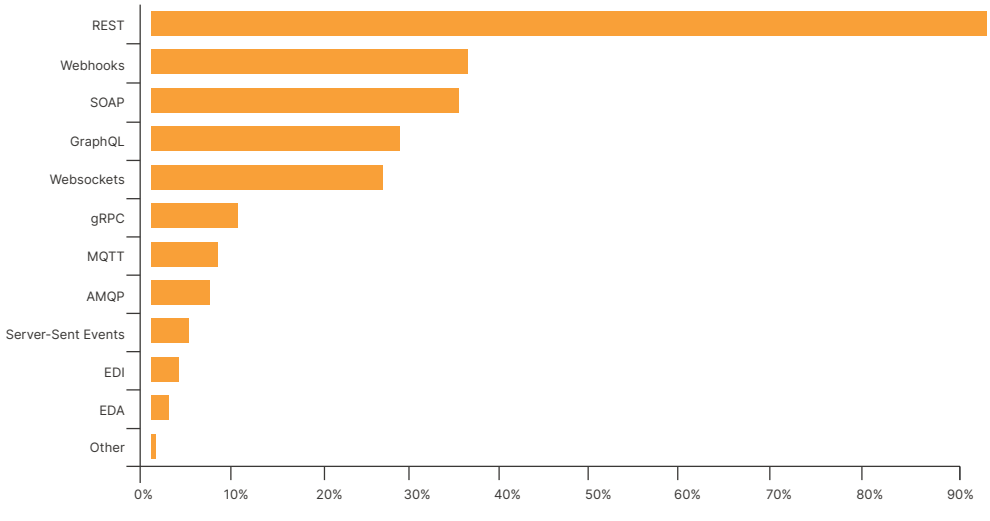
But companies moving from an API-early stage to a more API-aware stage are gaining confidence in exposing their APIs as they learn more about the controls they have through API gateways and other aspects of API technology.

# 3.3 API Technologies

A clear set of patterns and protocols has emerged to help enterprise organizations deliver APIs. A number of contract formats are available to help define the details of the API landscape. You can see the growing opportunities that APIs present by looking at the growth in contract specification communities, as well as the use of patterns and protocols.

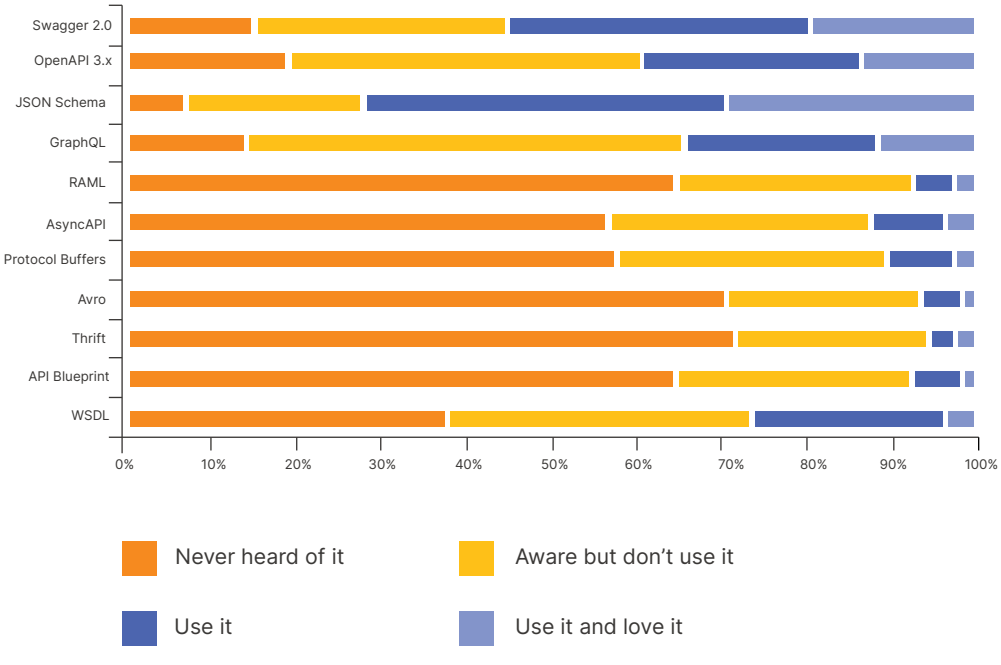
## Patterns and protocols

REST continues to be the dominant architectural pattern, but webhooks, WebSockets, and other patterns are expanding across enterprises in many industries.



## The top contracts

JSON Schema continues to be the leading contract in use across organizations, with Swagger 2.0 and OpenAPI 3.0 leading the pack in defining API access to valuable resources.



# 04
# Marketing and Selling Your API Products

APIs afford unique opportunities to help you understand your customers better, create marketing that speaks to their needs, and iterate your products to give them what they want when they want it. This chapter will show you step-by-step how to do it.

## 4.1 Determining Your Sales and Marketing Needs

Your sales and marketing in the API economy should center on staying change-resistant and future-proof. To sell effectively, you need a solid platform that will bolt onto your existing infrastructure and leverage the marketing and sales SaaS services you currently use. You need your digital factory floor and supply chain to instantly respond to your customers. Then you can turn your relationship with them into a real-time engine for understanding what they need and meeting their demands as quickly as possible.

Being API-first ensures that the feedback loop you have established with consumers provides the input to inform future sales and marketing campaigns. In other words, your marketing and resulting sales will inform the road map for your future APIs, as well as

improving sales and customer experiences today. So the sale isn't just about what currently exists—it is also about perpetually delivering consumers value.

It can be easy to see your APIs in terms of the applications they power, but the real API-first opportunities lie in speaking to the business needs of the enterprise.

While there are many directions possible once you have begun your enterprise API journey, here are the top level areas to consider:

**Industry** - Consider what would happen if you expose your digital resources, capabilities, and experiences to external partners or even competitors, or how these APIs might be applied in industries that you do not already directly engage with.

**Aggregation** - The aggregation of data in finance, healthcare, and other top industries has proved to be a hot area of investment. These industries are leveraging APIs to sell access to industry-wide global or regional data, making markets more visible and actionable.

**Opportunities** - Being API-first means that you can respond to new opportunities to sell your product in the moment, often before your competitors are able to respond.

**Reduce Costs** - A successful way to position your API products is to show how they help companies reduce operating costs by transcending legacy technology and improving their business processes.

**Switching Costs** - Positioning your APIs to help reduce the friction of switching services and systems is a strong way to ensure that your APIs will speak to enterprise business goals.

**Rigid Systems** - Make it easy for consumers to integrate your APIs. You should demonstrate how your APIs help make consumers' operations more flexible, composable, and less rigid and immovable.

**Modernization** - Your APIs should reflect what is coming next for consumers, speaking to the future, but also mapping to existing realities. Consumers need you to help them modernize the way they do business and respond to challenges.

**Innovation** - APIs should be used to reduce team workloads, helping carve out more time for innovation. They provide micro opportunities for teams to do new things on top of their existing work in a worry-free environment.

## 4.2 Reducing Friction for Your Customers

Once you decouple your API development from supporting specific web or mobile applications, you can begin to see wider opportunities for introducing new API products. This allows you to better position and sell API products that speak to the business goals of your consumers, rather than just meeting specific technical requirements of a project or application.

Do the work to make sure your API products reduce friction in existing business processes and make it easy for your consumers to modernize and innovate within their existing operations. Being API-first means you are able to rapidly deliver products that speak to real enterprise challenges, both in individual enterprises and across entire industries.

## 4.3 Effective Approaches to Marketing and Sales

As business continues to shift from physical to digital products, the way you approach sales and marketing is evolving. Alongside Cloud, SaaS, and other digital transformations, sales and marketing efforts for APIs are becoming more interactive, engaging, and immersive, speaking to the realities consumers face on a daily basis and ensuring that your products meet their needs.

**Experiences** - By making your API products available to consumers, you are selling digital experiences. By effectively providing reliable digital resources and capabilities, you can offer new experiences.

**Engagement** - Your sales and marketing efforts need to focus on micro engagement opportunities in alignment with your business goals, but they must also provide meaningful quantifiable engagement opportunities for customers using simple API products.

**Community** - API sales and marketing is about meeting consumers where they already are, connecting with consumers where they exist, and helping to augment existing business processes with new experiences.

**Portal** - Offering a portal window to consumers is a common way to make API products available securely through self-service. iPortals are a commonplace tool in API sales and marketing.

**Network** - API networks are increasingly augmenting existing portal experiences, providing access for private, partner, and public API consumers not just to documentation, but to mocks, tests, monitors, and other resources.

**Self-Service** - Internal, partner, and public API consumers are increasingly expecting API products to be self-service. Providing frictionless self-service purchasing and engagement reduces bottlenecks across operations.

**Discovery** - The discoverability of API products is the foundation of sales and marketing. Being able to find and onboard with APIs is the top layer of your sales funnel, with APIs sealing the deal.

**Consumers** - There are fewer "build-it-and-they-will-come" API opportunities today. The APIs making the most significant market impact are focused on the needs of a set of consumers, establishing a feedback loop with communities.

This approach to selling and marketing API products tends to focus on publicly available APIs, but some of these processes can work for reaching internal or partner consumers. This approach to sales and marketing isn't meant to replace or threaten existing practices. APIs are about setting a digital baseline for offering products to consumers, with an emphasis on mapping to their existing experiences and world view.

To support the next generation of sales and marketing, enterprises will need to look at existing approaches to evangelism and developer relations, helping bring attention to available API products, but also advocating for consumers. It is up to leaders to find the right balance across sales, marketing, and developer relations–one that reflects the personality and tone your company wants to establish. Finding the right alignment between your own business objectives and those of your customers is key to success.

# 4.4 Developing a Sales Strategy

APIs excel at doing one thing and doing it well. They become much more difficult to sell if you try to make them be everything to everyone. While crafting a sales strategy for your API products, it helps to be as precise as you can about how you will be making your products available. The shape, size, and positioning of your API solutions will vary widely depending who you are targeting, and will dictate a great deal about your approach to selling.

**Individuals** - Consider whether your APIs will be targeting individual developers, speaking to what they are looking to accomplish both in their role and as part of their personal interests outside of work.

**SMB** - To right-size your sales strategy, ask whether your API products are focused on the needs of small to midsize businesses and startups. Augment your sales team with the right experience for your API sales and marketing.

**Enterprise** - Your enterprise API consumers will have an entirely different set of needs, requiring you to do your homework on the mix of infrastructure and developer talent across teams.


**SaaS** - The software-as-a-service market has been evolving and maturing over the last decade, introducing a significant but often specialized opportunity for delivering APIs.

**Integrations** - APIs aren't always for desktop, web, mobile, or device applications. There is a significant opportunity for delivering and iterating upon the system-to-system integrations that businesses need to operate.

**Connectors** - The API-first transformation has produced many platform, application, and pipeline connectivity opportunities. Looking into them will reveal opportunities for API experiences that you can inject into solutions consumers already use.

**Bottom-Up** - Your sales strategy may need to speak g to the needs of consumers at the lower levels of an organization, spreading across the enterprise organically.

**Top-Down** - Your sales strategy may need to consider the realities of selling in a top-down fashion, reaching business and technology leadership. A top-down approach requires a different set of product packaging that speaks to the needs of decision-makers.

Being deliberate and concise in your API product sales strategy will help strengthen your position. It will impact all aspects of your API lifecycle and governance. There is a Venn diagram of considerations as you stitch together and evolve your sales strategy in this new era of self-service and reliable API products. The areas you invest in will shape how you sell and what the needs and expectations of your consumers will be—so make sure to plan accordingly.

Like other aspects of enterprise operations, sales is undergoing a shift, using historical practices that are still relevant, but also modernizing the sales toolbox with new constructs and practices. It is up to you to set the tone for your sales strategy by defining which of these areas matter the most to your objectives and which offer you the greatest opportunity for using API-first transformation to change how business is done today and tomorrow.

# 4.5 A Built-In Feedback Loop

Think of API marketing and sales as the digital version of just-in-time manufacturing, where marketing teams get people in the door by speaking to their immediate and future needs. When digital resources are modular and composable, and the life cycle around APIs is easily iterated, marketing teams can both speak to what already exists and hint at what will come next. Sales teams can map existing resources and capabilities to the ideal experiences consumers desire, instantly delivering what they need now and getting to work on developing what they will need next.

With an intimate knowledge of existing digital resources and confidence in the ability of the teams behind your APIs, marketing and sales workers can confidently obtain and retain customers' attention by helping them transition to what is coming next. An API-first marketing and sales strategy is not separate from the API life cycle. In an optimal situation–it informs the API life cycle, producing what is needed at the moment when you are talking with potential customers. There is still a massive divide between sales and developer relations, but the sweet spot for the enterprise lies in minimizing this divide as much as possible while staying true to your business goals and your consumers.

# 05
# Platforms and Digital Resources

To develop effective APIs, you need to think and act strategically. That means using a platform, rather than adding APIs willy-nilly. You also need to document and categorize your digital resources so you'll be ready to put them to work when the next API opportunity arises. You need to create solutions you can constantly iterate, making improvements as technology advances and user needs change. This chapter will show you how to accomplish these important tasks.

## 5.1 API Platforms

API platforms are software systems with integrated tools and processes that allow producers and consumers to effectively build, manage, publish, and consume APIs. They leverage existing infrastructure, as well as an ever-expanding and changing mix of API-driven services, to define, shape, and sustain business in a digital marketplace.

**Integrated** - API platforms are seamlessly integrated with your existing operations. They take the source control and CI/CD you use as part of the existing software development life cycle and marry it to your gateway and APM solutions. That will equip your teams with a modern API life cycle that is bolted to your existing investment in on-premises and cloud infrastructure.

**Discoverable** - APIs are naturally searchable and discoverable. You rely on your teams to always publish their artifacts and supporting metadata, with workspaces and repositories indexed and made available as a defaut..That enables your infrastructure to keep pace with the business, and allows teams to focus on the work that moves business forward, not just busy work.

**Collaborative** - An API platform brings teams and APIs out of the shadows, lowering the walls between business and IT groups and making both the producer and consumer side of the API life cycle a collaborative affair. With a platform, every aspect of a modern API life cycle is modular, portable, shareable, and executable. That helps to support more stakeholders across API operations and attracts the business expertise to bring APIs in alignment with meaningful business outcomes.

**Observable** - API platforms allow for any output across API operations to be gathered, measured, and made available via visual dashboards and reporting. That helps make very abstract APIs more tangible and connects them to business outcomes. Observability allows you to understand the overall state of enterprise operations and begin making changes to steer your business where you want it to go. That eventually results in a more agile and nimble enterprise.

**Automated** - API platforms deliver APIs, but they are also defined by APIs, making every part of the API life cycle automatable. Automation allows you to use workflows that are scheduled and executed from one or more cloud regions via CI/CD pipelines. That means you can respond daily to events occurring at scale across your organization, whether they are critical or mundane. You can use smaller teams and still provide them with the resources they need to meet the future demand of their industry.

**Governed** - Platforms bolt API operations onto our existing organizational infrastructure via standards like SSO and SCIM, and allow us to “see” all of our APIs so that we can ensure they are reliable, secure, and consistent. An API platform grounds not just the design of our APIs, but also the way we document, test, deploy, distribute, and observe them. That elevates governance beyond naming and ordering APIs, allowing platform-level control over the entire life of hundreds or thousands of APIs.

In the last decade, we’ve moved from using tens or hundreds of APIs to thousands. In the next decade we will move from thousands to hundreds of thousands, which can only be achieved with an industrial-grade API platform to carry the load.

A platform approach to delivering the API products your consumers need provides a foundation to build trust, making marketing and sales more impactful.

# 5.2 Defining your Enterprise Digital Resources

The modern enterprise is made up of hundreds or thousands of individual digital resources stored in databases and file systems, and put to work across a dizzying array of desktop, web, mobile, and other types of applications or integrations. You see the vast inventory of enterprise digital resources in browser URLs and in the digital experiences we encounter in our personal and professional lives. These resources shape our offline and online experiences each day as we engage at work and at home on our mobile devices.

**Users** - Every one of our user profiles is made accessible via APIs, so we can log in, follow, and be followed, allowing thousands or millions of users to engage via a single platform–on any device they choose.

**Messages** - We send messages back and forth every day on v SMS and other common messaging formats. It is the proprietary platforms of messaging APIs that allow us to engage with our closest friends, family, and coworkers in real-time.

**Images** - Every one of the social media images and photos on our mobile phone is published, stored, organized, backed up, and shared using APIs, providing us easy access to visual representations of the world around us.

**Videos** - Like images, videos are defined, published, and shared across the web and social media channels using APIs, allowing us to capture the world around us and weave these moments into our digital presence, blurring the line between online and offline experiences.

**Payments** - APIs allow us to send and receive money, dealing with anyone in the world. They serve as the basic financial ingredient that makes our world go around, funding the lives we lead online and offline.

**Documents** - The web is made up of digital documents in HTML, PDF, and other digital files. Thanks to APIs, we can not only publish and share them, but have them signed by multiple parties, allowing us to do business around the globe.

These are just a handful of the most common digital resources we all encounter each day. There are millions more. Your API catalog should represent all of the raw digital resources your enterprise depends upon, and your API networks should help you make these raw resources available to the consumers who need them the most.

Just as you do in the physical world, you need to have full control over your vital digital resources, and you must be able to effectively create and apply them as needed across operations.

## Always iterate upon your enterprise digital capabilities

A digital capability is simply a representation of something that can be accomplished digitally, defining a process, workflow, model, or algorithm that is valuable to internal or external consumers. Digital capabilities are often defined by one or many API requests or subscriptions that reflect a specific business capability. They may represent a single product offering or bundle multiple capabilities together for a specific digital product experience.

**Flows** - A workflow consists of an orchestrated and repeatable pattern of activity, enabled by the systematic organization of resources into processes that transform materials, provide services, or process information using API-defined resources.

**Actions** - Digital capabilities are designed to enable action of some sort, allowing online or offline processes to be triggered and setting additional workflows in motion. Actions also generate data along the way, producing intended outcomes across platforms or real-world devices.

**Events** - Organizations are increasingly defined by meaningful events that occur across internal and external operations.These events deliver the intended outcomes business and technical leaders are looking for, allowing for the orchestration of the desired outcomes that matter most.

**Models** - Artificial Intelligence and machine learning models are developed, iterated upon, and applied in increasingly modular ways using APIs. That's because APIs can access the data we have stored in databases and collected from daily operations and use it to iterate upon algorithms to make them more useful.

It should be clear by now that enterprise digital capabilities are defined by APIs. Business is set in motion through automation and orchestration across the native and third-party digital capabilities that each organization has access to. Being API-first allows you to define all of the private and open processes you apply across operations in private, secure, and reliable ways.

Your business operations depend on the synchronous and asynchronous delivery of many processes, putting your native, partner, and third-party resources to work. APIs allow you to define all of the internal and external digital resources you depend on. Your digital capabilities allow you to orchestrate and automate with these resources.

The collective power of your enterprise capabilities across domains—and your ability to apply and iterate upon them based upon how they are used—define what you are capable of as an enterprise. The API-first transformation is realized when you are able to define, use, and evolve your enterprise digital capabilities with maturity and control. How well you execute these processes will determine just how capable your organization truly is.

To summarize, digital resources provide the raw ingredients we need to power the business and digital capabilities bring APIs closer to achieving business outcomes that make sense to stakeholders. Transforming resources into business capabilities is remaking the enterprise landscape. The future will be defined by those enterprises with the strongest enterprise API life cycle muscles.

## 5.3 Making Your Infrastructure and Applications Composable

Modern software is about composing, assembling, evolving, and decommissioning business resources and capabilities to meet the ever-changing needs of the enterprise. Today's software not only allows us to build products, but to invent modular, industrial-grade Lego building blocks that power the web, mobile, and device applications we need to do business at the pace of the Internet.

**Resources** - Every digital resource produced and consumed by the enterprise is available for use somewhere. Mapping out all of the atoms of your digital transformation keeps the business in line with what it needs to operate and compete globally in a digital landscape.

**Capabilities** - Essential workflows and algorithms are defined in detail through many private, partner, and public APIs. These APIs document what the enterprise is capable of at any moment, while also being ready to adapt, change, and respond to entirely new market needs.

**Modular** - Enterprise resources and capabilities should be as modular as possible, driving re-use and collectively applying them at scale. You need to reduce business value down to the smallest reliable form, then make it available as a product to consumers via APIs.

**Distributed** - The enterprise is distributed geographically and organizationally, giving domains, groups, and teams the agency they need to deliver essential resources and capabilities while minimizing their dependency on other teams.

**Discoverable** - All the building blocks of the enterprise are discoverable, making digital resources and capabilities available to both business and technical stakeholders, as well as consumers. Everyone must have what they need to conduct business and compete.

**Self-Service** - Enterprise digital resources and capabilities are available via self-service and are visible only to intended audiences. Those who have access should be able to sign up and begin putting APIs to work with as few steps possible.

**Reliable** - Digital resources and capabilities must be reliable, providing users what they need without friction. Establishing and maintaining trust with consumers means reliably doing exactly what they need.

**Observable** - The ability to observe the usage of any digital resource or capability must be the default, no matter how many times something is reused and bundled with other services and no matter the scale of the deployment.

Modern software is composable because APIs exist just beneath the surface. APIs are how massive enterprise operations are made composable, reusable, and scalable. They do the hard work of reshaping the enterprise, using and re-using resources to provide the greatest business value possible.

A composable enterprise is the goal of an API-first transformation because this is where new revenue streams will emerge.APIs provide the agility and flexibility enterprises will need to compete and get ahead in today's digital marketplace. Composability is quickly becoming an essential enterprise trait.

Organizations further along in their API-first transformation journey will already have the digital resources, capabilities, processes, and practices in place to become agile and composable. The rest must internalize APIs in ways that will allow them to catch up.

# 06
# Maximizing Value for Users

In the end—and if you're doing it right, in the beginning and the middle, too—APIs are all about the business stakeholders and customers who use them. You can create APIs with the most elegant code imaginable and design them to integrate with hundreds of interfaces, but if you're not giving users the features they want, you're wasting your time (and your company's money). This chapter dives into what it means to deliver value and suggests some simple but highly effective ways of doing just that.

## 6.1 Treating Your APIs as Products

Treating APIs as products means doing the hard work to establish user empathy and ensure that they're easy to use and possess the shortest possible time-to-value. Good API products provide a complete API experience, with quality documentation, feedback loops, and support channels that give you the insights you need to iterate upon each new API version, helping to ensure that every release meets the needs of the widest possible audience.

**Consumer-Centered** - Make sure that the design, development, and operation of your APIs is as consumer-centered as possible.

**Experience** - Your API operations should focus on creating the most meaningful experience possible for your consumers, transcending the resources and capabilities being offered.

**Use Cases** - Make sure that your APIs are designed for specific business use cases that matter to your consumers. Make a commitment to understanding their processes.

**Feedback Loops** - Invest in your feedback loops with consumers, making it as easy as possible for them to provide feedback you can use as a road map for future iterations.

**Value Generation** - Ensure that your APIs focus on generating value for your consumers as well as your platform. API operations must benefit both you and your consumers.

**Measurement** - Define your metrics, gather data, and use it to make sense of how your consumers are putting your APIs to work making the information part of the API life cycle.

**Revenue** - Have a clear strategy for how your APIs will generate or support revenue generation, justifying their existence along the way.

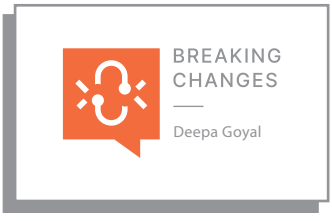
**Road Map** - Establish a road map on day one, and make sure you are keeping consumers informed of possible future developments to keep them along for the ride.

Treating your APIs as products will lift you from merely creating applications that are technically good to delivering solutions that meet consumer needs. Each new version is a dance with consumers, requiring you to find the right balance, helping them while also meeting your business needs.

> **Treat APIs as products, even if you don't plan to monetize them.**
>
> Gartner[5]

EXPERT PERSPECTIVE

# Speaking to the next generation of API product managers




One of the most important topics I have discussed with API practitioners on *Breaking Changes* this year has been treating your APIs as products and making sure you get feedback from product managers to drive your products forward across the API life cycle. So it was appropriate that we kicked off a recent season of the podcast with Deepa Goyal of PayPal, who shared her journey to become an API product manager and explained how important it is to produce content that speaks to the next generation of API product managers.

Looking back over her experience as an API product manager at PayPal, Deepa shared how little practical information was available to her about what it meant to be a product manager in charge of iterating upon APIs. Sure, there were plenty of marketing materials from API service providers saying you should treat your APIs as products, and similar content was being generated from the usual white male API pundits in the tech sector. But there was not comprehensive and practical guidance showing API product managers how to be successful as part of a modern API life cycle. Most importantly, there was a lack of knowledge and wisdom needed to train the next generation of diverse technical product managers and help them grasp the nuances of the API economy while showing them what is working on the ground today.

Today, with more experience under her belt, Deepa helped me better understand the business alignment that needs to be part of the API life cycle. That means more investment in API distribution and more thorough developer experience. At first I didn't quite see the sense in beginning the API life cycle discussion with product managers and the insights they obtain after an API goes into production, but now I see this is where the feedback loop with our consumers begins. Deepa provided a look into next-generation development experience, which includes documentation, but doesn't stop there. It also includes blogs, YouTube videos, and other channels that provide the onboarding and learning experiences that best suit consumers' needs.

As Deepa said, "What do they like? What serves them best?" She was completely focused on the intersection of business and consumer value. She discussed the importance not only of speaking to consumers on their terms, but of reducing friction with onboarding by measuring the "time to first call" for developers moving from discovery to an integration. I use this phrase a lot to help articulate the importance of developer experience. Deepa quickly elevated it for me by making it more closely aligned to business, expressing it as "time to first *value*." This phrase demonstrates for me the important lens product managers can bring to the discussion. Product managers help elevate our view above the very technical metrics we use and help us establish better alignment not just with our business, but with our consumers.

# 6.2 Identifying Outcomes: Jobs Theory and Interoperability

## Getting jobs done

Jobs theory, also known as "Jobs-to-be-Done" theory, provides a useful framework for designing, delivering, operating, and deprecating APIs. This theory complements an agile approach, helping to define the needs of consumers in the shortest possible time frame, iterate quickly, and respond faster than your competitors.

**Get Something Done** - People buy a product or a service to get something done. They don't care what a company's incentives are, they just want to accomplish their job.

**Jobs are Functional** - APIs must have a purpose that speaks to the needs of consumers, making and retaining an emotional connection.

**Reaching Maturity** - A Job-to-be-Done is stable over time. Your APIs should reach a level of maturity after several iterations, stabilizing into a reliable product for consumers.

**Agnostic Interfaces** - A Job-to-be-Done is solution-agnostic. APIs don't know or care about other APIs. Each one does one thing and does it well, focused on the job at hand.

**Measuring Right Thing** - Success comes from making a job, rather than the product or the consumer, the unit of analysis, focus on the value it delivers.

**Marketing Has Impact** - A deep understanding of the customer's job makes marketing and API iteration more effective.

**Get It Done on the Cheap** - People want products and services that will help them get a job done better, and when possible, less expensively, opening up a perpetual opportunity for APIs.

**Bring Value to Consumers** - People seek out products and services that enable them to get an entire job done on a single platform, making API integrations ideal for what they need.

**Delivering Success to Consumers** - Innovation becomes predictable when "needs" are defined as the metrics customers use to measure success for getting a job done.

A Jobs-to-be-Done methodology provides an unrelenting focus on the value delivered by an API. The key is providing what consumers need, then relying on your feedback loop to help guide the rapid iteration of your API products, always keeping in alignment with consumers. A jobs-based orientation provides the fuel for your APIs and helps you prioritize operations to move your enterprise forward.

A common organizational misconception is viewing APIs as technical details. Focusing on Jobs-to-be-Done helps us approach APIs from the business end, rather than losing sight of business goals in a forest of code.

## Making interoperability the default across operations

Ensuring interoperability–among distributed teams, federated groups, lines of businesses, and partners–is the desired state of existence for any organization. While the desire to operate as a single entity may prevail, the reality on the ground is usually distributed in both the physical and the virtual realms. Facilitating interoperability is one of the top reasons for enterprises to invest more in their API operations, ensuring that their business can connect in many different ways when necessary.

**Distributed** - The more distributed the enterprise is, the greater the demand for interoperability, which gives teams the ability to flex their muscles with APIs. As businesses depend more on distributed services,, we should give as much freedom and agency to teams as possible through APIs.

**Standardized** - We always lean on existing internet and industry standards, adopting, iterating, and contributing to them before we develop our own. However, in the absence of existing standards, enterprises should evolve their own common patterns to the status of standards–first across the organization,and possibly at some point, within their industry.

**Seamless** - All of our infrastructure and applications are part of a seamless web of business interoperability. We are producing and consuming APIs as rapidly as the business dictates, trying out new services, then working to integrate them with our API platform. We are using APIs, then iterating and evolving beyond them whenever possible. We should see all digital resources and capabilities as seamless and interoperable building blocks.

**Event-Driven** - Enterprise systems should be designed to respond to the most meaningful events occurring across our platforms. And the platforms we depend on must allow operations to respond in real time to what is happening across the market. We must allow not just applications, but our enterprise infrastructure to identify, subscribe, and respond to change as it is happening across the seamless platforms we use to do business.

Interoperability in today's digital landscape isn't just nice to have, it is an essential part of doing business, keeping up with changes across the global marketplace. Interoperability doesn't put you at a disadvantage with your competition–in fact, it does the opposite, helping you develop the muscles you need to outmaneuver competitors and develop entire new categories of doing business.

## 6.3 Delivering Meaningful Experiences

Providers who are further along in their API-first transformation have teams who are proficient in developing and maintaining digital resource APIs, and are rapidly delivering the meaningful and composable digital capabilities that power the business. Examining the digital experiences powered by APIs enables you to effectively map your APIs to real-world experiences.

**Discovery** - API-driven experiences should be easily discovered in the areas consumers frequent. Each API product should speak to the intended experience for targeted consumers, making it as easy as possible for them to find in the moment when they need it.

**Evaluation** - Each digital experience should be easy to test, requiring minimal gates to step through to understand its value. The more you make your experiences available for consumers to evaluate, the more you will learn from them.

**Success** - Defining what success means for each digital experience helps make APIs much more effective. Having a clear understanding of successful outcomes will help reduce the surface area of each API that is used to shape experiences.

**Effort** - As an API producer and provider of digital experiences, it is important that you empathize with the effort you require from your consumers. Understanding this effort will help you develop empathy and optimize the experience.

**Seamlessness** - Digital experiences should always integrate with and augment existing consumer experiences, requiring as little from consumers as possible. Once value and trust have been established, you can evolve the experiences while remaining integrated.

**Awareness** - Effective digital experiences help build awareness amongst consumers, informing and educating them incrementally. Awareness is about investing in your consumers and working to build trust via a useful digital experience.

**Engagement** - Sticky and beneficial experiences focus on direct engagement with consumers, providing watchable and forkable experiences. A focus on engagement provides some of the metrics you will need to understand how effective you are.

**Iteration** - Experiences should adapt and evolve based upon the availability of new resources and composable capabilities. Digital experiences should reflect ongoing needs and desires of consumers, shaping what comes next.

Most important of all, digital experiences provided by APIs have to be pleasant and beneficial. Making an experience memorable and leaving a positive mark on the consumer is what the API-first transformation is all about. The aim is making your API products fit consumer needs.

Digital experiences represent a more mature API economy, one where we are beginning to get sophisticated with technology, but are also improving our understanding of how and why we are doing APIs. That will help us bring digital products closer to meeting consumer needs while keeping them aligned with overall business goals, evolving and changing to suit both as circumstances dictate.

**EXPERT PERSPECTIVE**

## Solving the right problems with API feedback loops


By the time I sat down with Jessica Ulyate, Platform Associate Director of Product at HelloFresh, I had been getting schooled on product management for several months. I was very keen on getting her definition of product management in API operations. Jessica gave me a very succinct response: Product management is simply the intersection of business, development, and consumers. For me, this line sums up the most meaningful  discussion we can have about delivering and iterating upon our API resources, capabilities, and experiences.

Echoing what I have been hearing across many other conversations with leading API practitioners this year, Jessica placeda lot of emphasis on the API feedback loop,, and specifically, the role product managers play. They can help you understand what your customers need and what your business needs, then help you iterate sensibly with this balance in mind. Product managers must understand where the business is headed, where the industry is headed, and what consumers need at this intersection. Then they negotiate with development teams to deliver the most meaningful solutions possible at the current moment. Jessica soberly states that there is no gold star for product managers at the end of the process. It's a never-ending game–which is a challenge, but it's also what makes API product management so very interesting.

Another dimension Jessica shared, which also reflects what I am hearing in other conversations, was the importance of investing in a platform group to support your development teams. I asked Jessica what her definition of platform operations was, and t once again, her answer was simple: "We help developers create value faster." Platform operations is all about removing

obstacles for developers, reducing friction and allowing them to focus on doing the things that bring value to your business. It is not about setting up technology environments, struggling with infrastructure, or dealing with any other aspect of operations that is known and has been done before. Jessica's view of platform operations helped me better understand the need to help development teams focus on business value. That, in turn, will help product managers do a better job of balancing business and consumer needs.

Earlier in her career, Jessica would have said you need to have technical chops to be a product manager.But after being repeatedly humbled by folks who possessed the right domain knowledge, she's changed her tune. Technical knowledge won't always give you the lens you need as you are sifting through customer feedback loops, or the metrics you use to measure success. What you must do is strike the right balance between business development, operations, and the consumers your platform serves.

## 6.4 Investing in Low-Code and No-Code Opportunities

Low code/ no code refers to an ever-growing opportunity to deliver applications without writing code–or at least minimizing the amount of code that needs to be written. This makes producing and consuming APIs much more inclusive, allowing those who are not fluent in writing code to benefit. But it also helps those who are proficient in coding by making them more productive in their work. Low-code/no-code business and technology stakeholders have a variety of ways of engaging with API operations and the businesses they support.

**Inventory** - To enable low-code/no-code development, provide business and technical stakeholders with a robust inventory of internal, partner, and third-party resources they can use when creating low-code/no-code integrations and applications, enabling them to do whatever is needed across services.

**Flows** - Low code/no code offers the ability to create and evolve common business workflows using many different APIs, allowing them to be used by technical and business stakeholders to perform common business tasks e without getting their hands dirty in the code.

**Authentication** - The complexities of authentication and authorization should be abstracted away, making it easy for anyone to be able to navigate the secrets needed to authenticate with one or many APIs, and solve authentication problems along the way.

**Visual** - With low-code/no-code, you make it a visual experience to navigate API inventory, build workflows, and configure authentication. Anyone can stitch together many different APIs, allowing business or technical stakeholders to take advantage of using them.

**Runtime** - Low- and no-code applications provide a simple runtime any user can employ to run their workflows, providing the compute power to make each individual request, handle responses, and execute workflows in the order suggested by the user and executing the jobs they need done.

**Automation** - You have the ability to schedule and trigger specific events, setting in motion the desired workflow. Low- and no-code allows anyone to automate common business tasks throughout the day, so that teams can do more with less.

**Collaboration** - Low- and no-code improves collaboration and sharing, providing feedback and allowing anyone to iterate upon the functionality of each workflow. That makes business integrations and orchestration a team affair, something everyone learns from.

**Observability** - People using low code/no code applications can see how their workflows and automations are running. They can also see the results of their work and how it fits into context with what other users are doing, cultivating a higher awareness of how the organization functions.

Low-code/no-code isn't just for non-technical folks. It will help enterprises deal with the demands of their digital future. There is a perpetual shortage of developers, and extending the ability to develop applications to more people will expand the pipes behind the applications and integrations that business stakeholders depend on, giving them the . resources and capabilities they need to be successful.

NOTES

5 "Top 5 API Lessons for Software Engineering Leaders," Gartner blog, 2021.

# 07
# The API-First Difference

API development is different for every organization. If you're not sure where you are on the API maturity curve, then making decisions about what to do, when to do it, and why can be bewildering. This chapter will describe the API journey and show you some of the capabilities to strive for. Then it will help you figure out where you stand and what kind of results to expect as you go along.

## 7.1 The API-First Journey

The API-first transformation is a journey. Every company is doing APIs today, but many are not aware of their scope and are not making them a priority across technical and business groups. At Postman, after speaking with hundreds of enterprises, we tend to see three distinct phases of the API journey: API-early, API-aware, and finally, API-first–the optimal state.

In addition, we use 13 different schemas for API-first transformation to determine where an enterprise stands in its individual journey. Examining the organization through these lenses reveals not only the organization's level of API maturity, but how it can move and respond to APIs as a single entity. This analysis reveals the business benefits available to the enterprise from becoming API-first and helps to alleviate the friction of adopting APIs to power business applications.

It is easy to find yourself drowning in the sprawling enterprise API landscape. You begin doing APIs for system-to-system integrations, producing and consuming APIs for web applications, and then scrambling to use APIs to meet the needs of mobile and device applications. The thirteen cross-cutting aspects of your technology operations provide a grounding for quantifying the journey–otherwise, you would find your organization with less control over direction. All of the blueprints you will find in this book are mapped to optimizing in these thirteen areas, providing the potential seeds for growth across critical areas that will contribute to positive business outcomes.

The API-first transformation will take time. You must have a strategy to define where you are going, and it is important to regularly pull off to the side of the road and assess where you are. You won't get very far unless you have a way of defining your collective destination and a framework for assessing progress. A lack of collective awareness about how APIs are produced and consumed across the enterprise and the absence of a framework for defining operations and the change occurring across operations will leave you lost. Seeing API-first as a journey and knowing where you are helps make this massive undertaking more manageable. If you do this well, you can have fun along the way, and I guarantee you will learn a lot as you progress in your personal and professional API-first transformation.

# 7.2 Why Becoming API-First Matters

## Future-proofing your organization

All organizations want to minimize risk and make the future as predictable as possible. Becoming API-first will help you do that. It helps reduce your legacy baggage, ensuring that your teams are always iterating and innovating to embrace the future as it arrives. API-first is about using APIs to rapidly iterate and map out change as it is happening, allowing teams to respond to whatever comes their way. That can help you resolve challenges ahead of the competition.

**Innovation** - As an enterprise, you are able to make innovation part of your operational DNA, with teams actively flexing their innovation muscle, experimenting, and engaging with consumers to find the business value in what comes next. You are modernizing legacy infrastructure and delivering the services the business needs today, while also innovating and learning what is needed for tomorrow.

**Agility** - Teams are able to respond quickly to market shifts, taking knowledge from regular day-to-day operations, but also experimenting via labs, and other innovative techniques. Then they respond with new products that meet a changing world. The process involves starting with well-oiled enterprise API operations, then responding to whatever comes next with small, quick, and well-informed iterations, using the digital resources and capabilities needed to accomplish goals.

**Velocity** - Teams know what they are capable of, and are able to increase or decrease velocity as required by the business and consumers. Having just the right velocity needed to respond to shifts in the market can move you ahead of the competition. Velocity will take a lot of training and experience, but with the right investments across the API lifecycle, teams will have what they need to respond to whatever comes their way.

**Change** - Change is embraced in an API-first environment. It is expected, welcomed, and seen as the right way. Teams are well-versed in defining, designing, and delivering API-driven change, and leveraging the feedback loop from these iterations to inform what is next. A Jobs-to-be-Done attitude equips everyone with a focus on the value present in change, and their training and confidence will prepare them for the work required.

**Control** - Leaders have the control they need not only to comprehend the state of operations, but to understand it in relationship to external consumers. They have observability and control, making adjustments when needed as agile and responsive teams move in the right direction. Being API-first shows enterprises what they need to do to move forward and shows them the direction they need to take to lead the way.

**Adaptive** - API-first organizations are able to adapt to whatever is thrown at them, evolving to serve changing markets and completely reinventing specific domains of the enterprise if required. An API-first approach perpetually finds the optimal state of doing business, delivering the products markets are demanding today, while also being prepared for what is needed to lead the way tomorrow.

There is no way to predict the future, but APIs give us a proven way to respond to any number of possible futures that may arrive tomorrow. APIs don't necessarily reveal exactly what we need to do business in the future, but operating in an API-first state means we will be able to quickly respond, evolve, and step up to confidently respond to what the future holds. Being API-first is how companies are staying relevant in the face of a changing demographic and business landscape.

## Enabling unlimited expansion

APIs are the building blocks of modern digital transformation. They provide the reusable, composable, and scalable units of value that are defining digital resources across every industry. APIs give us the ability to deploy and operate essential services across multiple clouds and multiple regions, allowing enterprises to redefine and rebuild their business in a much more resilient manner. APIs give us the digital capabilities we need to sustain the existing business, but also to define entirely new sectors we haven't yet imagined.

**Resources** - The enterprise has the required digital resources at its disposal for assembling the applications, integrations, and automation needed, but it also has the API factory floor, giving it the capacity to deliver the next generation of resources.. APIs are the building blocks of any future world we want to create, allowing us to manifest the products we know markets will need.

**Capabilities** - There are thousands of business workflows in motion across the enterprise at any given second. With APIs, they become discoverable, well-defined, and executable by business and technical stakeholders. That allows for the rapid creation, maturing, and scaling of known and unknown workflows that the enterprise will need to do business tomorrow.

**Scale** - Combined with the elasticity of the cloud, an API-first digital platform provides the scale required to build an entirely new world, providing an unlimited supply of digital resources and capabilities, as well as the teams to deliver new products. Products can be rapidly created, iterated upon, and then scaled to meet the demand of markets within weeks or even days, not years, providing the just-in-time scale needed to compete in a digital world.

**Multi-Cloud** - The next era of cloud computing will not be isolated in a single cloud. The enterprises that have achieved a world-building level of operations are finding success operating across all of the cloud platforms. They are investing in the talent, processes, and infrastructure needed to wrestle with the nuances of each cloud platform, but leveraging APIs to abstract away the differences, allowing them to successfully operate in all of the top cloud providers whenever business demands it.

**Multi-Region** - API-first enterprises are applying their new-found API-first agility and velocity to stand up and operate API infrastructure in regions closest to their consumers. They are responding to regulation and data nationalism, but also reducing latency by operating closer to the edge of their business networks. APIs can help organizations define their operations by domain, but also by geographic region, optimizing the platform to provide a better consumer experience.

**Regulated** - An API platform lends itself to efficient management of highly regulated enterprises, responding to internal as well as external influences in a balanced way. Layering government regulation into a mix of governance tools gives leaders more control over how the enterprise responds to markets and regulators, while staying focused on the needs of consumers. The API platform pushes regulation to a state in which the enterprise has the ability to shape it as a positive force in how the company does business.

## Becoming a category-defining power

The agility associated with a high-performing API-first approach to business was demonstrated by Amazon in developing Amazon Web Services, which changed how the company does business. API-first startups like Stripe and Twilio provide similar examples,. setting into motion the framework for gig economy businesses such as ridesharing and food delivery, among others. APIs allow both startups and enterprises to rapidly iterate while maintaining a feedback loop with consumers, keeping the trajectory and velocity of APIs moving in the right direction and increasing the potential for creating entirely new ways of doing business.

**Domains** - Well-defined domains within the enterprise don't just result in more productivity, quality, and a governed API life cycle. They can transcend their platforms and communities to create entire new industry-level domains. They allow organizations not only to lead, but to set the rules for how the business works.

**Products** - An API platform factory floor is capable of delivering entirely new products that markets want and need–as well as those they may be currently **unaware** that they need. That's because teams constantly evolve the products consumers need, responding rapidly to changing market demands through rapid iteration of APIs and the applications they support.

**Standards** - The design, schema, and approaches adopted across the enterprise can become so mature and useful to business that they begin to be emulated and adopted in other external APIs. They can become the de facto standard within an industry, not through a traditional standards body, but by a forward-thinking company's leading by example to deliver what is needed.

**Velocity** - An API-first company achieves levels of enterprise-wide velocity that other competitors can't keep pace with. They have no chance of catching up, because by perpetually optimizing the digital factory floor, large enterprises build enough momentum to set the stage for an entirely new way of doing business.

**Pulse** - Through an extensive network of feedback loops with consumers across thousands of internal, partner, and public APIs, the enterprise is able to take the pulse of what is needed across an industry, or multiple industries. That allows a single business to redefine existing industries or create entirely new ways of doing business.

API-first organizations will shape the business ie categories of the next fifty years. Enterprises that are able to transform their API-first operations into new business models are the ones that will bring us the next big thing.

## 7.3 Where Are You in Your API-First Journey?

There are three categories of businesses in today's digital marketplace:those that are unable to keep up, those that are effective at competing and leading existing industries, and those that are defining entirely new ways of doing business. How much you invest in your API-first transformation will define which of these categories you will be in, and will dictate the future of your enterprise. We would all like to be in the third category. BBut to get there we must do an honest accounting of where we are today, and what it will take for us to achieve an API-first state.

## API-Early

Enterprise organizations early in their API journey are already doing APIs–they just aren't, doing them in a strategic, observable, or governable way. They exist in a pretty chaotic and undefined landscape of digital services that are powering a mix of web, mobile, and device applications, while cobbling together integrations across internal and external systems.

- **Strategy** - There isn't much of a strategy to be found, and while all teams are doing APIs, they are left to their own devices for designing and operating APIs.
- **Discovery** - There is no central place for finding APIs at any point in the API lifecycle, resulting in a lot of redundancy in the APIs behind applications.
- **Applications** - Desktop, web, and mobile applications are the priority when it comes to development, but the APIs behind them operate in the shadows, unseen except by anyone except developers.
- **Visibility** - There is a lot of anxiety about exposing APIs externally, even to trusted partners, because the organization lacks experience in securing and managing them over time.
- **Quality** - The quality of APIs is very low and generally unpredictable, leaving trust in teams and the APIs they produce low in the eyes of internal, partner, and public consumers.
- **Security** - There is a centralized security group, but it has difficulty securing what is unseen.There is no standard for security across teams, but they continue to launch APIs as needed.
- **Productivity** - Teams spend a lot of time reverse engineering existing APIs, trying to debug them and learn how they work in order to evolve.Most new APIs are delivered via code-led approaches.
- **Velocity** - The speed at which teams produce new APIs or enhance existing ones is very unpredictable, making planning and road maps difficult to communicate and achieve.
- **Observability** - Teams do not have much visibility into the health of individual APIs, let alone APIs at scale across the organization or I the operations behind the APIs in production.
- **Governance** - There is no consistency present across APIs. They all use different patterns, with a lack of standards leaving leaders unable to control the state of operations.
- **Standards** - There is a lack of awareness of internet and industry standards.There is no group helping to define and evolve internal organizational standards across teams.
- **Regulations** - Government regulations are seen as a never-ending source of friction and obstacles the enterprise must overcome, always preventing operations from moving forward.

- **Innovations** - There is no room for innovation. All teams are expected to spend all of their time putting out fires, and there is no time and no resources to spend developing new ideas.

API-early enterprises find themselves in a perpetually reactive mode facing the demands of applications, projects, and seemingly endless incidents that arise in production and in supporting end-users.

Being API-early means there are huge opportunities in laying the foundation for change. But it will take the right mix of roles across the enterprise to transform into becoming more API-aware.

## API-Aware

Once an organization begins to wake up to the potential of being API-first, it starts investing in a strategy for producing and consuming APIs in alignment with business objectives. The company becomes more aware of the APIs that already exist and the life cycle that is moving them forward (or possibly holding them back).

- **Strategy** - Teams have come together, acknowledging that an API strategy is needed.A formal document has been drafted, but leaders are just getting started in deciding how to proceed.
- **Discovery** - A central API catalog exists within the enterprise and for public engagement, but the APIs are never quite uptodate, not reflecting what is happening in production.
- **Applications** - Teams building applications have begun doing more planning and communication before developing web or mobile applications, resulting in better coordination.
- **Visibility** - An API management solution exists to handle the publishing of public APIs, but the process for securing them is undeveloped.
- **Quality** - Some teams have gotten a handle on testing their APIs, but there's still a lot of work to do when it comes to test coverage and overall API reliability.
- **Security** - The organization has a central security group and is doing API security reviews, but teams producing the APIs see the process as too slow and heavy-handed.
- **Productivity** - Teams are able to discover APIs developed across the organization and are exposed to development by other teams, increasing productivity, but these processes still need more work.
- **Velocity** - APIs are more discoverable by teams, including where teams are getting work done. The organization has invested heavily in hardening CI/CD workflows, releasing new iterations more often.

- **Observability** - Teams are using existing APM solutions to understand the health of their APIs. They are routing monitoring data into APMs, building dashboards, and understanding the state of their APIs.
- **Governance** - A centralized group has come together and begun defining guidelines, standards, and rules teams can use to help produce more consistent APIs.
- **Standards** - More IETF, IANA, and other Internet standards are making their way into the design of APIs, and more reusable patterns are beginning to be shared across teams.
- **Regulations** - Regulatory compliance is beginning to become less daunting now that the enterprise landscape is coming into focus. The organization has a map showing teams where data is located.
- **Innovations** - Teams are beginning to find more bandwidth due to productivity gains. They are starting to feel they have more time to play around with new technologies.

API awareness breeds more API awareness. Organizations find their initial investments in discovery, quality, and governance begin to snowball, leading to more collaboration and coordination, with tooling, processes, and knowledge and shifting the enterprise towards becoming API-first.

The API-aware stage is where you begin to see the transformational properties of APIs at the domain and organizational level. APIs can do a lot in a single application or integration, but it is at the collective organizational level that they will have the greatest impact on how the enterprise operates.

## API-First

API-first organizations understand very well that applications are built as an interconnection of internal and external services through APIs. They realize that APIs define how enterprise digital resources and capabilities are made available to internal and external consumers, powering business applications as needed.

- **Strategy** - There is a living strategy defining the goals of enterprise API operations, outlining what the API lifecycle looks like across teams. Leaders regularly update the strategy.
- **Discovery** - The organization has an active catalog of private, partner, and public API resources. The teams behind them and the operations around them are discoverable, helping to facilitate use.
- **Applications** - Desktop, web, mobile, and device applications are regularly developed using common, reusable, and secure APIs. That speeds development, while keeping processes consistent.

- **Visibility** - APIs regularly begin as private, internal-only constructs, but once complete, they are able to quickly be exposed to partners or even the public, based upon the demand for resources and capabilities.
- **Quality** - Nearly 100% of APIs are developed to meet the minimum bar for expected quality. They meet SLAs, and minimize the risk of breaking with each release, always providing the performance expected.
- **Security** - The organization hasn't suffered a breach in years. APIs, no matter where or how they are used, enjoy the same level of security teams provided earlier in development.
- **Productivity** - Every API has a dedicated workspace and repository where teams can easily find APIs and the work going on around them. Teams have the tools they need to iterate and deploy APIs.
- **Velocity** - Teams release often with fewer failures, and are able to move from idea to release in a shorter period of time, enabling domains within the organization to respond more quickly.
- **Observability** - There is observability across nearly 100% of APIs. Teams can observe both individual APIs and across all APIs, and can see the infrastructure used to deliver them.
- **Governance** - Leaders properly govern teams and APIs across a well-known API life cycle, resulting in APIs that are consistent, documented, reliable, and easily found by consumers.
- **Standards** - Internet, industry, and organizational standards are alive and well in the APIs that teams design, and team members are actively participating and contributing to standards.
- **Regulations** - Teams easily achieve compliance with privacy and interoperability rules, reducing the time needed to respond to ad hoc privacy requests, while ensuring that all APIs are interoperable.
- **Innovations** - Teams regularly spend 20% or more of their time learning new technologies, experimenting with new types of APIs, and finding ways to optimize the way APIs are delivered.

API-first doesn't mean all challenges have gone away. But it does mean that teams are better equipped to deal with anything that comes their way, and spend less time in a reactive state. API-first means that the API infrastructure behind desktop, web, and mobile applications is a priority. Even business units make sure the SaaS solutions they adopt have APIs to ensure interoperability and provide opportunities for automation.

# 7.4 Gauging Results

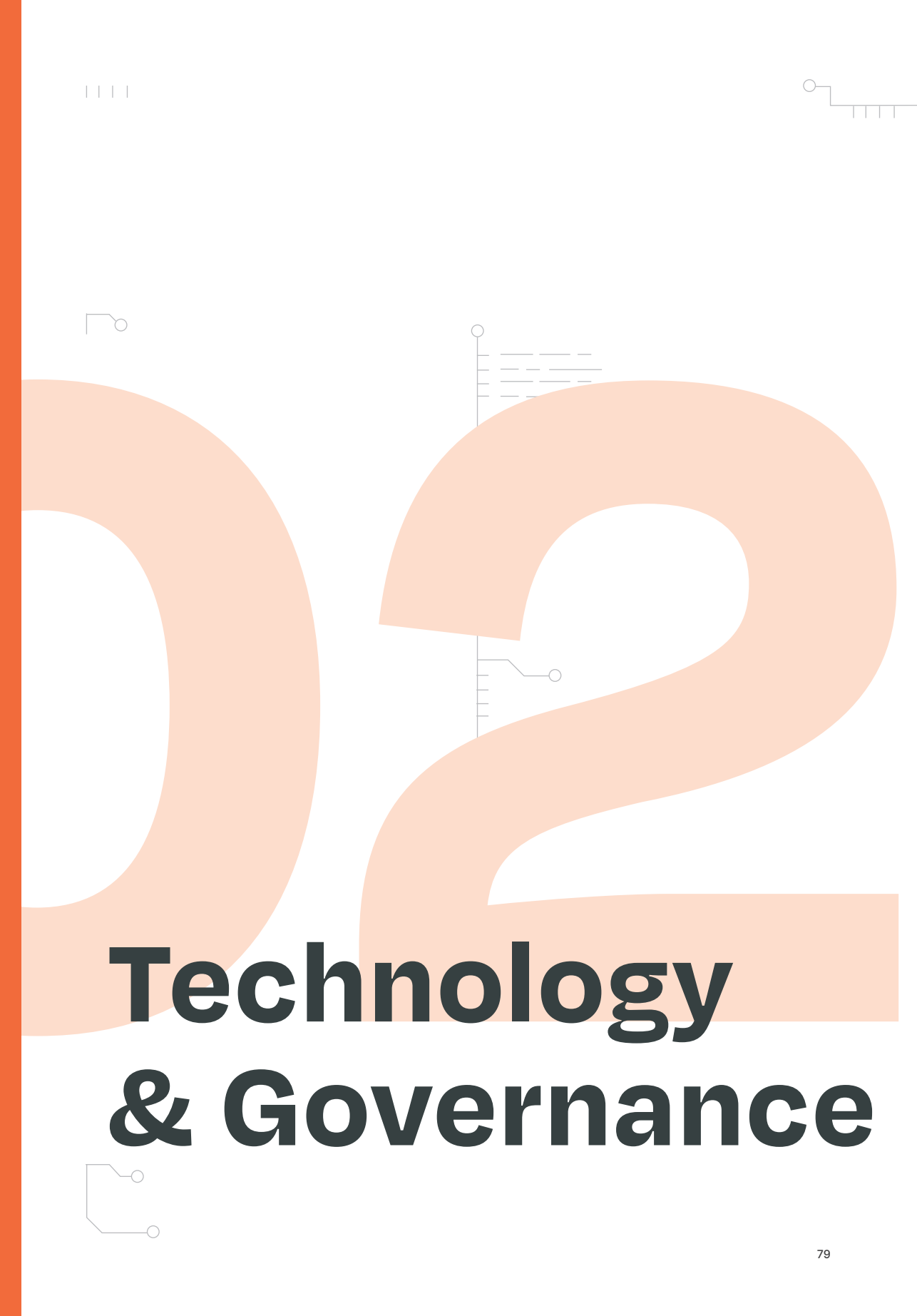
The API-first concept plays out in different ways for different organizations. Some see it as a destination to continually strive for, knowing they will never entirely attain it entirely, but will gain enormous benefits along the way. Others see it as a way of dealing with current and future challenges. There is no perfect, utopian API-first state. Your outcomes will be very much in alignment with your expectations, so it makes sense to understand this early on and set the right benchmarks for your teams.

To be honest, the results you will see as part of your API-first transformation will be very disappointing early on. The transformation will seem too massive to handle, and you'll suffer regularly from doubts, not quite knowing where to start or whether you are making an impact. You will live in a sort of groundhog day of API-first transformation, saying the same things over and over and experiencing perpetual déjà vu with process friction and challenges across teams. Even after you begin to pick up momentum, you'll experience many low days where it seems like your projects are just too much for your organization to handle.

But once you begin finding your rhythm with a few right-sized API resources, you will begin to see the potential of being API-aware. Your teams will have much more shared knowledge about the API lifecycle, and they'll begin using a common vocabulary in API contracts, such as OpenAPI and AsyncAPI. This is where you will begin to see the API-first opportunity and understand what you need to do to move your enterprise in the direction you want it to go. With a little more API momentum in your teams' feet, you can expect better results in transforming your organization and boosting your business results.

The results you see on the ground will reflect the appetite your leaders and teams have for change and the scope of operations you have chosen to transform. It is common for those of us in the API space to recommend starting small. But that approach doesn't work for everyone. In recent years I have begun recommending that leaders right-size their approach. Sometimes that means you need to go big. It depends on the results you're looking for. Many business leaders I speak to say they simply can't afford to go small, perhaps in part because the Covid-19 pandemic has injected more urgency in our API-first transformations.

No matter how large or small your steps, it's important to view the API-first transformation as a journey–one that requires leaders to regularly step back and assess how far the organization has come, while always maintaining a shared set of goals that will lead to a brighter, more agile future.



# 02 Technology & Governance

# 08
# The Essential Elements of API Technology

Whether you're debating REST or HTTP protocol or OpenAPI or AsynchAPI contracts, you have many decisions to make for structuring your APIs and the infrastructure supporting them. This chapter will help you make sense of the alphabet soup of API technology and choose the solutions right for your organization.

## 8.1 API Infrastructure

Today's digital business landscape is defined through a handful of Internet protocols for requesting, responding, publishing, and subscribing to machine-readable contracts. This toolbox of contracts increasingly defines the backbone of the API economy, and developers are rapidly iterating upon it and inventing entirely new dimensions.

REST, web, or HTTP APIs dominate the conversation when you talk with technologists. While these ubiquitous API patterns will continue to be the bedrock in every industry, newer, more specialized types of contracts, have emerged to help us manage internal, B2B, and B2C operations.

Web-scale businesses use web protocols to move digital resources around, automating operations using well-defined digital capabilities. They continue to iterate on the experiences that matter to your partners and third-party consumers. However, there is another dimension to consider: You need to apply this same discipline to the APIs you are consuming each day.

Your infrastructure toolbox needs to be diverse. It must reflect the latest protocols, patterns, and standards while aligning with your business goals and your consumers' needs. And your teams need to be educated on the pros and cons of each tool in that toolbox.

As we've stated previously, there is no single API solution for your business needs. There is a growing suite of API infrastructure that works well in some situations and not so well in others. But collectively, these tools can get you 90% of the way to where you want to be. Your team needs to do the work to make sure they are aware of the benefits and tradeoffs of each piece of infrastructure in your toolbox, and you need the domain expertise to properly apply the tools across your operations.

API infrastructure today is contract-driven. That means every API and the infrastructure behind it should be defined as a machine-readable contract. API contracts are the platform glue for all the infrastructure dependencies across your operations. Without them, the experience consumers have with your API resources and capabilities will never be quite consistent. And without contracts for the infrastructure behind your APIs, you will never be able to operate your digital API factory floor reliably.

Look at your API architecture through the same lens you use to judge your WSDL contracts. The architectural decisions you made 20 years ago are still visible across the landscape. You will also have to live with the decisions you make today for many years, so it's important to have clear intentions behind the infrastructure you put in place.

## 8.2 Internet Protocols, API Contracts, and Specifications

Various internet protocols are in use across the World Wide Web, allowing regional and local area networks to transport digital resources and capabilities around the globe to service digital commerce. The most well-known l is the hypertext transfer protocol, or HTTP, which is the backbone of the web as we know it., But HTTP is getting a makeover, and other leading protocols are also gaining acceptance across the API landscape.


**HTTP 1.1** - The Hypertext Transfer Protocol, or HTTP, is a protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless protocol that can be used for many tasks involved in distributed object management systems.


**HTTP/2** - HTTP/2 is a major revision of the HTTP network protocol. It was derived from the earlier experimental SPDY protocol originally developed by Google, then adopted by the HTTP Working Group of the Internet Engineering Task Force.


**HTTP/3** - HTTP/3 is the third major version of the Hypertext Transfer Protocol used to exchange information on the World Wide Web, alongside HTTP/1.1 and HTTP/2. HTTP/3 always runs over QUIC, providing a next-generation internet approach.

**TCP** - The Transmission Control Protocol is one of the main protocols of the internet protocol suite. It originated in the initial network implementation, in which it complemented the Internet Protocol. Therefore the entire suite is commonly referred to as TCP/IP.

**MQTT** - MQTT is a lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP; however, any network protocol that provides ordered, lossless, bi-directional connections can support it.

The protocol you select when delivering an API will set the tone with consumers for what is possible and will provide constraints depending on the patterns employed. This will determine what is possible across the API lifecycle, shaping your overall journey and business outcomes.

APIs, whether internal or external, are built on these fundamental internet protocols. APIs are the next iteration of the web and take advantage of the low cost of internet infrastructure to stitch together the digital experiences we plan when defining and shaping our businesses.

HTTP 1.1 provided us with the base we needed to get through the last 25-plus years of doing business on the web. HTTP/3 will be the protocol that delivers the future, with TCP and MQTT continuing to deliver the industrial-grade networks we need for the enterprise applications, systems, and devices that define our physical and digital supply chains. These protocols provide us with the transport layer we need to power our digital businesses, but there are other contracts and patterns we'll also need to get the job done.

## Understanding the role of API contracts

On top of the protocols and patterns employed by API producers, a variety of machine- and human-readable contracts have emerged to help govern not just the technical, but also the business and legal aspects of the API producer and consumer relationship. These contracts are used to make sure producers and consumers are on the same page, providing a single source of truth for each version of an API made available.

## Specifications

A handful of specifications have emerged that help us describe the surface area of our synchronous and asynchronous APIs. These specifications employ a mix of protocols, patterns, styles, and formats in machine- and human-readable formats, which we can use to define the contracts available between producer and consumer.

**OpenAPI** - The OpenAPI specification, formerly known as Swagger, provides a machine- and human-readable format often used for describing HTTP 1.1, web, or REST APIs. It provides a contract for describing the relationship between API producer and consumer.

**AsyncAPI** - The AsyncAPI specification is a sister specification to OpenAPI, but focused on event-driven APIs across HTTP, TCP, MQTT, and potentially HTTP/2 and HTTP/3. It provides a contract for describing the relationship between publishers, subscribers, and brokers.

**JSON Schema** - The JSON Schema specification is used for validating the models of data sent and received from APIs. It is also used by OpenAPI and AsyncAPI as vocabularies for modeling and validating the objects available via different types of APIs.

**Protocol Buffers** - Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data–think XML, but smaller, faster, and simpler. They are often used for high-speed internal APIs or B2B APIs for partners.

**Collections** - Collections are a machine-readable format from Postman for describing the surface area of APIs, then executing in service of the API lifecycle, providing mock servers, documentation, testing, and other developer needs across enterprise API operations.

**GraphQL** - GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, giving clients the power to ask for exactly what they need and nothing more.


**WSDL** - Web Services Description Language, or WSDL, is an XML-based interface description language used for describing the functionality a web service offers. It provides a machine-readable description of how a service can be called and what kind of data it returns.

API contracts provide both the technical and the business details of the relationship between API producer and consumer, moving everyone to a shared understanding of what can be expected with each version of an API. That includes the quality of service and the digital resources and capabilities that will be made available. The success of each API will depend heavily upon the balance between producer and consumer but also upon the business and technical aspects of conducting commerce online.

**EXPERT PERSPECTIVE**

## Contract-First at Goldman Sachs


Chander Shivdasani, Vice President of Goldman Sachs' Marcus banking products and services division, recently talked with me on *Breaking Changes* about the important role APIs are playing in helping the financial company deal with change and modernizing legacy operations. When I asked for more details about the company's approach to defining and delivering the next generation of APIs with OpenAPI, I learned that they insist on calling it a contract-first approach rather than design- or definition-first. With this language, they express the importance of a contract-first approach for bringing business stakeholders to the table and improving the alignment between their API evolution and business objectives.

Goldman Sachs' contract-driven approach to OpenAPI reveals the role OpenAPI plays in defining each business contract established between an API and its consumers. This business nuance of defining API contracts is more than just a word. It denotes greater alignment between the technical and business aspects

of API operations, narrowing the divide that commonly exists. These contracts provide the language needed to deliver entirely new services while modernizing legacy infrastructure, identifying dependencies, and helping Goldman Sachs remedy years of technology deficits.

The API contracts at Goldman Sachs are being used to establish a common vocabulary across teams, helping to ensure that everyone is on the same page. This contract vocabulary is helping the financial company stabilize, automate, and optimize across the API lifecycle and apply consistent governance across teams. The contract-driven approach is also helping the firm power federation to deliver existing business services and respond quickly to emerging market needs. A federated API approach benefits from centralized guidance and governance, while keeping teams autonomous and helping the company navigate geographic and regulatory concerns.

Being contract-first is helping Goldman Sachs iterate and innovate, but also allows it to respond to privacy regulations like GDPR and navigate data sovereignty, all while embracing a federated approach to doing business across multiple geographies. While there is still a lot of work to be done in modernizing operations, by bringing more business stakeholders into the conversation, the contract-first approach is allowing Goldman Sachs to move the enterprise forward, while I responding to new and innovative market opportunities. As a result, the well-established financial institution can behave more like a fintech startup, showing the impact that an API-first transformation can have in helping enterprises deal with any change that might come their way.

## 8.3 Using OpenAPI as your Standard Digital Business Contract

The OpenAPI specification, formerly known as Swagger, gives you the ability to describe the surface area of your HTTP 1.1 APIs using JSON or YAML. OpenAPI provides a robust way to describe what is possible with each API, defining the surface area of each request and response.

**Info** - You have a place to define common metadata for an API, such as a name, description, licensing, terms of service, and contact information, helping to ensure that all APIs have enough metadata available to articulate a purpose across the API life cycle.

**Servers** - You include a list of servers for every instance of API, possibly across multiple regions or stages of development. That allows consumers to quickly find an instance of an API they can use to meet their needs and apply it as a resource.

**Paths** - API consumers can take different paths to access resources and capabilities, similar to browsing the web. But in this case, they are navigating the API landscape, looking for the resources and capabilities needed to power applications and integrations.

**Operations** - Define the specific operations that can be accomplished using a specific path. Operations provide the ability to read, write, update, delete, and perform other actions on API resources, setting different capabilities defined as part of each API in motion.

**Parameters** - Provide a defined set of parameters that can be used to change the state of API responses. Provide key/value pairs for common things like pagination or search, but be specific, depending on the objects returned with API responses.

**Responses** - Describe the HTTP Status Codes, headers, and media types returned with each response, helping the consumer understand the structure and state of the response and providing consumers with as much information as possible about responses.

**Schema** - Provide JSON Schema descriptions of request and response bodies, allowing the responses to be validated and helping automate validation at the gateway to ensure the highest quality possible for consuming APIs within any application.

**Security** - Describe the type of authentication required for accessing an API and provides a machine-readable description of the API keys required. OAuth, JWT, and other types of security protocols help automate the authentication layer of API usage within clients.

OpenAPI isn't just for documentation or code generation. It is the standard business contract you apply to every digital resource you use in applications and integrations across the enterprise. It provides teams with a common vocabulary to describe the relationship between API producers and consumers.

# 8.4 Mapping the Event-Driven Enterprise with AsyncAPI

The AsyncAPI specification provides the ability to describe the surface area of your multi-protocol APIs using JSON or YAML. The open source specification provides a robust way to describe what is possible with each API, defining the surface area messages and channels. It can then be used as the source of truth, showing what is possible when publishing and subscribing to each asynchronous API.

**Info** - Provides a place to define common metadata for an API, such as a name, description, licensing, terms of service, or contact information, helping to ensure all APIs have enough metadata available to articulate a purpose across the API life cycle.

**Application** - An application is any kind of computer program or group of programs, allowing for the view of a producer or a consumer, a microservice, an IoT device (sensor), or possibly a mainframe process that will be publishing and subscribing to messages.

**Producer** - A producer is a type of application connected to a server that creates messages and addresses them to a channel or publishes them to multiple channels, depending on the server, protocol, and use-case pattern applied as part of an API implementation.

**Consumers** - A consumer is a type of application connected to a server via a supported protocol that consumes messages from a channel or multiple channels, depending on the server, protocol, and use-case pattern in an API implementation.

**Message** - A message is the mechanism by which information is exchanged via a channel between servers and applications.The payload containing the data, defined by the application, MUST be serialized into JSON, XML, Avro, binary, or another format.

**Channel** - A channel is an addressable component made available by the server for the organization of messages, enabling producer applications to send messages to channels and consumer applications to consume messages from channels.

**Protocol** - A protocol is the mechanism (wireline protocol OR API) by which messages are exchanged between the application and the channel. Example protocols include, but are not limited to, AMQP, HTTP, JMS, Kafka, MQTT, STOMP, WebSocket.

AsyncAPI provides business contracts for the many different channels you can publish or subscribe to across the enterprise, defining the events that matter to operations while also helping to ensure that the events and the messages passed along are well-defined and make sense to consumers.

As the API landscape has expanded across multiple protocols, AsyncAPI has emerged to quantify the landscape, helping us document, mock, test, and manage this growing layer that is a part of how we run our business. It ensures that the enterprise is delivering and responding to the most meaningful events.

## 8.5 Modeling and Validating Your Business Using JSON Schema

The JSON Schema specification provides a machine- and human-readable way of describing digital objects used as part of API requests and responses, as well as the messages we publish and subscribe to our more asynchronous APIs. JSON Schema allows us to describe the structure of our digital resources and capabilities so that we can validate them during development and production.

**Objects** - Objects are a way to define digital structures, providing a machine-readable way to describe meaningful concepts exchanged online using different types of APIs and passing data in a way that makes it easy for teams to have a shared understanding.

**Properties** - The individual characteristics of an object, providing the details that give an object meaning and value. Properties may describe the name and email of a person object, or the name and description of a product object, providing a logical set of properties that applications can understand.

**Property Names** - Each property has a name, allowing each individual characteristic of an object to be described in a way that makes sense to consumers, providing a shared meaning of a specific aspect of a digital object available via an API.

**Property Description** - Each property can also have a description, providing much more detail about what the object property will contain. Descriptions convey the meaning and purpose behind why the property exists, and how API consumers can use it in applications.

**Property Type -** Allowing each property to be defined as a string, number, object, or other common or custom type, formally articulating what it can be expected to contain, helping us to be very strict or loose when it comes to making data available in objects.

**Property Patterns** - Patterns allow for regular expressions to be used to precisely articulate what a property should contain, providing a universal way of precisely describing the contents, ordering, and structure of the data available via each object property.

**Required** - Defining a list of the properties for each object is required whenever you are moving objects around synchronously or asynchronously via APIs, helping to describe the minimum amount of information needed to define an object made available.

JSON Schema is how the digital bits we pass around the web each day get validated behind the scenes, making sure the requests we make and the response received meet the expectations of both API producers and consumers, giving us the real-time validation we need to reliably do business at scale.

Both OpenAPI and AsyncAPI use JSON Schema to model the payloads of both our synchronous and asynchronous APIs, providing a rich way to define the digital resources and capabilities of the enterprise. JSON Schema validates that business is happening as we expand at scale across our operations.

## 8.6 Internal Operational Contracts as Protocol Buffers

Protocol Buffers (Protobuf) is a free and open-source cross-platform data format used to serialize structured data. It is useful in developing programs that communicate with each other over a network or for storing data. The method involves an interface description language that describes the structure of some data and a program that generates source code from that description for generating or parsing a stream of bytes representing the structured data.

**Serialization** - Protocol buffers provide a serialization format for packets of typed, structured data that are up to a few megabytes in size, suitable for both ephemeral network traffic and long-term data storage, and extending with new information.

**Services** - A service is an individual system that supplies a digital resource or capability, providing a granular unit of business value that can be used internally within the enterprise or made available to partners in a secure but highly performant manner.

**Messages** - Messages are digital communications for sending serialized and structured, record-like, typed data in a language-neutral, platform-neutral, extensible manner, providing a highly efficient way of communicating between systems within the enterprise and with partners.

**Message Types** - Protocol Buffers allow you to define any type of message you will need to make your digital resources and capabilities available, giving you full control over how data will be structured, interpreted, and consumed by internal and partner developers.


**Language Compatibility** - Messages can be read by code written in any programming language, providing a high-performant way to make data available across many different platforms. A robust add-on ecosystem allows for a long tail of integrations.

Protocol Buffers provide a solid contract for defining the relationship between your internal services, allowing for large volumes of high-quality and well-defined data to be exchanged across the enterprise. In some cases, gRPC is used externally with partners, providing high-volume business-to-business services outside the enterprise firewall.

Protocol Buffers are a very strict and efficient approach to delivering industrial-grade digital plumbing for the enterprise. But don't confuse these APIs with the simpler web APIs that are the fundamental building blocks of internal and external APIs. Make sure you are right-sizing your approach to APIs, and possess a clear definition of what it will take to deliver what your API consumers need before choosing gRPC as the solution.

Not all developers will find working with gRPC intuitive, so make sure your teams have the right education and skills to be effective when producing and consuming gRPC APIs. When it comes to the right implementations, gRPC is very effective at delivering high availability and powerful APIs. But in the wrong situation, they can be overkill and introduce more friction for developers.

# 8.7 Making the API Lifecycle Executable Using Collections

Postman collections are a machine-readable specification for saving API requests in a portable, executable, and documented way, allowing one or many API requests to be organized by folder, then shared or published for use by others. Collections provide an executable unit of value as defined by each API's source of truth. They are made available in a format that can be used to document, mock, test, secure, and automate many different APIs.

**Folders** - Each collection has the ability to define one or many folders and organize API requests into each folder, making collections more intuitive and easier to use.

**Authentication** - Collections allow the authentication to be defined for any API consumed, providing most of the top authentication formats used to secure APIs.

**Documentation** - You can document your APIs, as well as the tests, automation, visualizations, and other use cases for collections, ensuring API operations are documented.

**Parameters** - The parameters and default values can be provided for each API request, helping define query and path parameters that shape each API request and response.

**Headers** - Collections allow for HTTP headers to be passed along with each request, shaping the transport of each API request and response made with collections.


**Body** - Enabling the ability to add JSON, XML, Text, and other types of data payloads as a body of the request, securely sending data (when encrypted) as part of a request.

**Responses** - The details of a response, including the HTTP status codes, headers, network information, response time, and the other technical details of the API response for APIs.

**Scripts** - Collections allow for folder level, as well as pre-request and post-request scripts to be applied, providing scripts that get executed when collections are executed manually.

OpenAPI and AsyncAPI provide a source of truth for each API. A collection provides a derivative of that truth for a specific stop along the API lifecycle. This relationship provides a versatile way of generating documentation, mock servers, tests, and other essential building blocks.

Collections can be combined with CSV or JSON sets of data to represent specific business use cases and outcomes. Pairing collections with data helps to ensure that each individual API contract accurately reflects each and every possible business outcome associated with the contract. It allows for new levels of reliable automation across enterprise API operations.

Using collections to automate the API lifecycle is the only way we will be able to properly scale our operations from the hundreds or thousands of APIs we depend on today to what we will need tomorrow.

# 8.8 Embracing your Legacy Web Services Using WSDL

WSDL is an XML format for describing network services as a set of endpoints operating on messages that contain either document-oriented or procedure-oriented information. The operations and messages are described abstractly, then bound to a concrete network protocol and message format to define an endpoint. While few new APIs use WSDL, they are ubiquitous for web services across common enterprise systems.

## Documents

WSDL contracts are defined as documents, giving flexibility for the type of document and the message, as well as the port type, and binding defining access and communication.

**Types** - Defines the XML Schema data types used by the web service, specifying a specific, existing, and well-known schema that provides a shared understanding.

**Message** - Defines the data elements for each operation, allowing just the contents of the API request to be accessed, focusing on only the payload, not transport.

**Port Type** - Describes the operations that can be performed and the messages involved, shaping what type of communication is used for communicating.

**Binding** - Defines the protocol and data format for each port type, setting the stage for how API communication will occur, with the desired protocol and port.

### Port type

The port type element defines a web service, the operations that can be performed, and the messages involved, providing for a very versatile way of delivering upon WSDL contracts.

**One-Way** - The operation can receive a message but will not return a response.

**Request-Response** - The operation can receive a request and will return a response.

**Solicit-Response** - The operation can send a request and will wait for a response.

**Notification** - The operation can send a message but will not wait for a response.

While most greenfield APIs will not be using SOAP as a pattern, it is likely we will still see web services that use SOAP and expose a WSDL contract for the next fifty years. This longevity will make it an essential part of the API-first transformation occurring across our organizations and the industries we operate in. WSDL will always describe our legacy infrastructure, but it is also an important part of our transformation, something we'll be emulating and learning from for years to come.

## 8.9 Investing in the Architecture your Operations Need

It can be very tough to keep up with the pace of change in software delivery today. It can be very challenging to develop the awareness, skills, and discipline you need for individuals, a team, and the enterprise. Each organization has its own history, culture, constraints, and trajectory, but there is always a kind of zeitgeist in the architecture powering APIs, which in turn shapes how we deliver applications and integrations.

Just beneath the desktop, web, mobile, and device applications we depend on daily to do business, REST or web APIs dominate, providing the essential resources, capabilities, and experiences our users expect. This is where you always begin with your API-first transformation, but a healthy transformation will also include a diverse set of protocols and patterns that are needed in the diverse range of situations the

enterprise is struggling with across the digital landscape. As with the overall strategy for your API-first transformation, there is no one approach to investing in and leveraging architecture to deliver your APIs.

It is very common for each iteration of API infrastructure to declare that it is replacing what came before, but most of the time, the latest API architecture is either augmenting or living side-by-side with existing, more established architectures. One way to navigate these cycles of evolution is to establish a strategy for defining and guiding your API-first transformation. Otherwise, you might find yourself with blinders on regarding any single piece of API infrastructure that comes along. I see this problem today in conversations about REST vs. GraphQL vs. event-driven and other areas when the real conversation should be about REST *and* GraphQL *and* event-driven.

There are valid pros and cons to each of the leading API architectural patterns, and to effectively operate your enterprise, you need to be aware of as many patterns as possible, developing your team's knowledge and muscles across as many of them as you can. There are proven patterns for a variety of business use cases and industries, and you should avoid trying to reinvent the wheel and embrace what already works well. RESTful APIs are always the base of enterprise architecture, but GraphQL, Websockets, gRPC, and other proven patterns can be used to stitch together the digital quilt enterprises need to power their digital experiences at scale around the globe.

This API-First Transformation book will provide an overview of the API architectures that matters most today and give you the building blocks for applying them as part of your strategy. However, it is up to you to build muscle memory and discipline across your teams to maximize your infrastructure investments.

# 09
# Patterns and Protocols

There are many tools for configuring and integrating your APIs. In this chapter, we will describe some of the most important, explaining their advantages and disadvantages to help guide you to the best choices for your enterprise. We'll also take a peek at up-and-coming technologies like machine learning and real-time data analysis, which are rapidly changing the corporate API landscape.

## 9.1 The Most Important API Patterns

There are many common patterns in use across the API sector, but the foundation of the modern API toolbox continues to be REST. with GraphQL, WebSockets, and increasingly gRPC added to the mix. The line between pattern and protocol is often a blurry one, but there are a handful of well-known patterns that act as the cornerstone for API applications and integrations, depending on which industry or layer of the enterprise an API is operating in. While there are other patterns to consider, these five API matter the most to enterprise organizations today.


**REST** - Representational state transfer is a software architectural style created to guide the design and development of the architecture for the World Wide Web. REST defines a set of constraints for how the architecture of a distributed system should behave.

**GraphQL** - GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API. It gives clients the power to ask for exactly what they need and nothing more.

**WebSockets** - WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and is used heavily for financial APIs.

**gRPC** - gRPC, also known as Google Remote Procedure Call, is an open-source remote procedure call system initially developed at Google in 2015 as the next generation of the RPC infrastructure. It has since become a desired patent used internally with HTTP/2.

**Microservices** - A microservice architecture–a variant of the service-oriented architecture structural style–arranges an application as a collection of loosely-coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight.

**Webhooks** - Webhooks use APIs to trigger events. Instead of making calls to APIs, webhooks occur when different events occur across operations, pinging other systems or sending data to help make actions more event-driven and real-time. With simple web APIs.

Some of these patterns merely provide a style to follow. Others are standardized formats and protocols that provide a set of agreed-upon constraints. You can apply a specific protocol or a mix of transport protocols to consumers, brokers, and other stakeholders.

## Finding the right pattern for the job

Each of these patterns has strengths and weaknesses, but collectively they provide you with a rich and increasingly proven set of tools you can consistently use across teams. A diverse mix of standardized APIs allows you to do business at the scale and speed you need. Make sure your teams get to know each of these API patterns, then regularly exercise their knowledge on APIs in production until they find which patterns work best in different situations.

It can be easy for technologists to get caught up in the API patterns they have most experience with, or the ones that work for their particular applications and integrations. However, as an organization, it is important to Invest in a diverse API toolbox. That way you avoid technological dogma and apply the pattern that makes the most sense for the

job at hand. Having a diverse set of patterns in your API toolbox will help ensure that your enterprise is as adaptable as it can possibly be.

# 9.2 Using REST and GraphQL

## Defining essential enterprise resources using REST

Representational State Transfer, or simply REST, is the dominant pattern across enterprise API operations because of its simplicity and use of the HTTP protocol. REST provides a foundational and ubiquitous way for making digital resources and capabilities available across the widest possible range of developers in the desired applications and integrations. REST brings a common set of characteristics that can be easily applied across hundreds or thousands of APIs.


**HTTP** - RESTful APIs are built on top of the same technology that brings us the World Wide Web, making them a low-cost and widely understood transport protocol for making digital resources to developers across almost any programming language.

**Uniform** - Delivering uniform interfaces for consumers makes onboarding and integration of RESTful APIs much easier for API consumers. They can readily apply the value represented in the enterprise API catalogs they have access to.

**Resources** - REST focuses on uniformly defining digital resources, providing the raw ingredients that can be used in digital applications. It provides the vocabulary developers need for delivering meaningful use cases.

**Methods** - RESTful APIs take advantage of HTTP methods to help standardize the actions you can take with the digital resources defined, providing a common vocabulary for complementing resource-oriented nouns with useful verbs.

**Synchronous** - By using HTTP, RESTful APIs, you emulate the web, allowing applications and integrations to make requests for digital resources and wait until they receive a response, providing users what they need at the moment.

**Stateless** - Simple APIs using a RESTful pattern do one thing and do it well, without requiring wider awareness of the state of the application making the API call. They leave the state of the user experience to be defined by the application in question.

**Cacheable** - Since RESTful APIs use HTTP, they can benefit from the capability of web documents, significantly improving the performance of applications that may be requesting data, content, and other media that may not change very often.


**Simple** - REST works as an API pattern because it is simple, making digital resources, capabilities, and experiences available to consumers in a way that can be discovered, understood, and put to use with the least amount of work possible.

REST does have limitations. It won't be the right solution for every project, but it does provide the essential baseline for API operations. REST APIs are a natural evolution of the web, but instead of making digital resources and capabilities available to humans via a browser, they make these same digital resources available via mobile and desktop applications. They also allow other systems to use these resources in different types of automated applications and integrations.

## Making the data landscape actionable using GraphQL

GraphQL provides the ability to apply queries, mutations, and subscriptions defined within a query language to synchronous and asynchronous APIs. GraphQL adds another type of contract to our API toolbox, allowing us to offer a very flexible query language enabling developers to get the data they need to use in any application.

In conversations, I have found enterprise organizations using GraphQL to support their mobile and single-page applications. I also see them deploying an internal graph to abstract many different APIs and data sources into a single layer of the enterprise that developers can query and work with.

**Operations** - You have a document describing various types of operations, such as queries, mutations, subscriptions, and any fragments, that define what is possible with the API.

**Document** - GraphQL has two types of documents: execution documents and schema documents. They determine what types of queries are possible and how consumers can use them.

**Selection Sets** - Located within an execution document, selection sets are sets of fields that make up the content contained within curly braces, describing the desired fields associated with a query.

**Fields** - Fields are object-specific attributes that can be requested and return a value, providing the atomic unit of any combination of data made available via an API.

**Fragments** - There are three types of fragments. Named fragments allow us to reuse fields, Type conditions allow us to conditionally select fields, and Inline fragments don't have a name defined inside the selection set.

**Directives** - There are four directives defined in the spec: @skip, @include, @specifiedby and @deprecated. They change the way a section of the document is executed.

**Mutations** - Different types of operations enable different state changes for data, providing the ability to read, update, and achieve other states.

**Subscriptions** - These are long-lived requests that allow the server to send the client events as they happen, allowing consumers to subscribe to the data they need.

GraphQL is an optimal solution when you have a large amount of data with a wide surface area and developers familiar with GraphQLwho are building a variety of applications with specialized needs for specific domains of data. GraphQL provides you with a powerful API layer for your distributed enterprise landscape, augmenting your RESTful and other types of APIs. That helps you stitch together all data in a meaningful API layer.

# 9.3 Using WebSockets and gRPC APIs

## Delivering more real-time APIs using WebSockets

As explained above, WebSocket is a computer communications protocol standardized by the IETF in 2011 and providing full-duplex communication channels over a single TCP connection. WebSocket is distinct from HTTP, but is designed to work over HTTP by using a handshake that uses an HTTP Upgrade header to change from the HTTP protocol to the WebSocket protocol using TCP.


**HTTP** - WebSockets' relationship with the HTTP protocol makes it something that compliments RESTful APIs with a higher volume, speed, and real-time alternative, helping shift APIs into a higher gear to deliver what consumers demand.

**TCP** - When API consumption requires a much higher volume of real-time connections, it makes sense to switch from HTTP to TCP, providing a much more industrial-grade pipe for moving digital resources where they are needed and responding faster to events.

**Asynchronous** - WebSockets provide the ability to publish and subscribe to digital resources in real time, allowing consumers to publish and subscribe to messages in real time by requesting and waiting for a response to get what they need.

**Real Time** - Interactions with WebSocket APIs occur in real-time, allowing consumers to publish and then subscribe to events as they are happening, providing faster access to a firehose of information they need to support applications.

**Event-Driven** - WebSocket APIs do not require consumers to consciously make requests when they want information. They can just subscribe to events that are already occurring across operations, making the API experience more impactful.

**Ad Hoc** - It is common for WebSocket APIs to be used as a catch-all for whatever schema you might need across the enterprise, making for a pretty ad hoc and sometimes chaotic mix of schema from many applications and integrations.

**Volume** - WebSocket APIs are definitely designed for noisier APIs, working well for financial, market-related, and other real-time data scenarios. They provide a set of garden hoses or fire hoses consumers can publish and subscribe to for their work.

**Scale** - This API pattern is something you can use to scale some of your higher- volume API channels, augmenting other patterns in your API toolbox. You can use it for a more precise set of applications to meet the needs of a precise set of consumers.

A WebSocket API provide an effective way for you to stand up a private, partner, or public API that handles a significant volume of specialized or ad hoc API traffic. But most WebSocket implementations lack schema normalization, management, and observability, making them prime targets for standardization before they become a sprawling landscape of schema noise.

You will find WebSockets in the financial sector, but their use is rapidly expanding across other industries, adding a publish and subscribe layer on top of the ongoing enterprise chaos.

## Delivering the business backbone using gRPC APIs

gRPC is an open-source, high-performance remote procedure call (RPC) framework that can run in any environment, efficiently connecting services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is a solution for delivering industrial-grade distributed computing to devices, mobile applications, and browsers to backend services. Right out of the box, gRPC brings numerous benefits to your operations and your approach to delivering APIs.


**HTTP/2** - HTTP/2 is the next iteration of the HTTP protocol.Derived from the SPDY protocol developed by Google, it has been the preferred choice of transport for gRPC, though that will likely shift to HTTP/3 now that it is becoming ready for prime time.

**Protocol Buffers** - gRPC uses Protocol buffers, Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data, to help deliver extremely efficient API resources for use in a variety of applications.

**Performance** - gRPC is known for being very performant, leveraging the latest protocols and serialization approaches to ensure that APIs function as the high-speed backbone for internal and B2B systems across any business domain.

**Code Generation** - With gRPC, you get a variety of client code out of the box in all of the top languages, providing high-quality code for developers to use in developing their applications using the industrial-grade API pattern.

**Strict** - gRPC makes it difficult to deliver sloppily-designed APIs, providing a strict approach to defining the contracts that govern API producers and consumers. That helps stabilize the APIs we depend upon across the enterprise so we can offer reliable services.

**Streaming** - Like WebSocket and other real-time APIs, gRPC streams data, providing an always-on connection for applications to use when sending or receiving high volumes of data as it is created, published, and made available via APIs.

**Robust** - gRPC is your backbone API infrastructure, providing what you need to connect your critical systems. It offers the performance and reliability you need for your most important, revenue-driving business operations.

**Internal** - If you have the resources for it, using gRPC makes sense for powering your internal API infrastructure with some possible external partner instances. But mostly, gRPC should provide the API backbone you need to drive the core of your operations.

gRPC provides a very solid technical solution for delivering and maintaining infrastructure and backbone APIs within the enterprise and possibly with partner and B2B solutions. gRPC is not an entry-level pattern like REST. While it can provide streaming APIs like WebSockets, gRPC is a much stricter and more disciplined approach to delivering API infrastructure, providing yet another solid pattern you can apply as part of your API-first transformation.

## 9.4 Decomposing and Decoupling with Microservices

The benefits of APIs have manifested in many ways but few have made an impact close to that of microservices, which have redefined the enterprise. The microservices journey has helped us understand who we are as organizations and provided the vocabulary for defining the technical, commercial, and human elements of doing business in The Internet Age. Microservices have helped us decompose the monolithic business infrastructure that has accumulated over the last few decades and begun to evolve our operations.

**Composability** - Microservices are designed to be stitched together and consumed in a variety of ways meeting a mix of business needs. They provide the digital building blocks enterprises need to operate, grow, and adapt to business needs.

**Templates** - Having standardized templates for designing and deploying microservices helps ensure that microservices are consistent, intuitive, and easy for consumers to use, no matter which teams were responsible for bringing them to life.

**Synchronous** - Synchronous microservices provide a simple and intuitive way to deliver solutions for a single digital resource or capability, providing a very modular and composable way for developers to implement the digital services needed.

**Asynchronous** - Asynchronous microservices provide a more event-driven approach to delivering resources and capabilities, allowing developers to publish and subscribe to a variety of channels that help define business operations.

**Risk** - There is risk involved with breaking digital resources and capabilities into very modular and potentially distributed microservices.Teams should be made aware of what is needed when managing a potentially sprawling landscape of digital services.

**Reward** - With the right enablement, robust microservices offer a big payoff in flexibility and agility. They can make enterprise operations more nimble, responding to any changes that may come along.

**Lifecycle** - Delivering hundreds or thousands of APIs consistently across an organization requires a common and well-known life cycle to be employed across teams, ensuring everyone is on the same page for delivering and iterating upon APIs.

Microservices are not a silver bullet that will solve all our enterprise challenges, but they do provide a proven approach for breaking up and reorganizing our digital resources and capabilities across business domains. Microservices give teams a clear definition for a class of API that is only used internally, powering what they need to deliver APIs, applications, and integrations.

# 9.5 Making Operations More Event-Driven with Webhooks

API requests trigger some sort of event in a system adding or updating data. Webhooks do the reverse, making a call to any URL when one of these events occurs. A webhook is the trigger that results when different types of API requests or message publishing occurs, which can set in motion any other API-driven action, making operations more event-driven.


**URL** - Each webhook has a URL property, allowing it to have the address of the resulting call that should be made when each event is triggered, automating any workflow.

**Payload** - A payload provides data to send with each event triggered, sending static or dynamic data to the URL for each webhook and adding more context to the event.

**Event** - Events provide a meaningful name and description of each moment triggered when an API call is made.

**Analytics** - Webhooks require the ability to see events that have been triggered, including errors, retries, and other important information needed to manage events at scale.

**Emails** - It is common to allow emails to be sent to one or many addresses when an event is triggered, adding another layer of system or human messaging.


**Logging** - Like other API infrastructure, the logging of webhook calls is a common part of system operation, providing a record of every webhook transaction that occurs.

**Notifications** - For some events, there may be an additional need to send a notification using another API, or some native application mechanism.

**Retry** - Not all webhooks will execute successfully, so it is common to build in the ability to retry one or many times, allowing outcomes to be eventually realized.

Webhooks are one of the most useful but overlooked and underutilized tools in the toolbox. They are the poor man's event-driven architecture. Like APIs, webhooks use simple, low-cost web infrastructure, but they use it to help make APIs more resilient and event-driven.

The most common way webhooks are used is as a relief valve for API producers when API consumers poll an API too many times. A webhook tells API consumers to stop requesting an API when looking for new or updated information, indicating we'll tell you when something changes or when a specific event occurs.

Webhooks can make operations much closer to real-time with very few additional resources and skills required. With proper management tools and analytics, API providers can significantly improve the overall efficiency of their API operations.

EXPERT PERSPECTIVE

# A pragmatic view of API design at Domino's Pizza


One of the trickiest stages of the API life cycle is deciding how to design our API resources, capabilities, and experiences. It is anxiety-inducing to navigate all of the dogma and opinions even for a single API design pattern like REST, let alone making your way across multiple protocols and patterns like WebSockets and GraphQL. So I was very pleased to learn from James Pozenel, Solution Architect at Domino's Pizza, about the company's very pragmatic approach to API design.

Few debates about API design are more heated than those about how to name and order your API paths. RESTful API design, which uses HTTP, provides simple, intuitive names for URL paths (i.e., "/images" or "/books"). Then you use HTTP methods like GET, POST, PUT, and DELETE to obtain, add, update, or delete your data. This methodology is widely known as CRUD APIs (Create, Read, Update, and Delete), and provides the fundamental building blocks we use for delivering API resources. The challenge is that GET, POST, PUT, and DELETE won't give you verbs rich enough to articulate every capability and experience you are looking to deliver with your digital resources.

The RESTafarians who have dictated how we *should* design our APIs have opposed mixing noun API resources with complex verbs using the URL. However, James and his team put a lot of thought into the pros and cons of adding complex verbs to noun resources like "/images/upload," "/book/publish," or in the case of Domino's, "/pizza/delivery." REST dogma tends to emphasize embracing the constraints of REST, so James is breaking the rules. But that is fine. James' team has found a much richer vocabulary for describing the digital resources Domino's needs to get business done globally while continuing to evolve and iterate on the digital capabilities and experiences they need to dominate the pizza conversation around the world.

This pragmatic approach to designing APIs gave James' teams a much richer vocabulary and allowed them to add a role-based access control layer, leveraging HTTP paths to determine which capabilities and experiences consumers have access to. This domain-driven approach to security, combined

with the company's pragmatic approach to API design, makes for a very rich, secure, and flexible way of delivering digital capabilities and experiences. The company's approach gives it a flexible way of defining what the API-first transformation looks like for a physical business operating across the globe to make sure pizza is delivered to homes and businesses. It can be easy to get caught up in API design debates, but if you focus on what your business needs and what matters to your consumers, you will find the balance you need to succeed.

## 9.6 Synchronous and Asynchronous APIs

APIs provide a way for humans to interact with each other. They are also the systems we depend upon to talk to each other on our behalf. As shown by the diverse range of API architecture listed above, different types of API-driven experiences dictate different types of interactions between our applications and integrations. In a digital world, API interactions are usually synchronous or asynchronous, setting and creating exactly the experience consumers are looking for in their applications.

Most of the web resources we use each day are defined as synchronous API interactions. A human consciously makes a request via an application for some data, content, media, or other digital resource. As we navigate our way through websites and mobile applications, we click buttons and links, or swipe and use other gestures to make API requests, which synchronously wait for a response to be fulfilled. These activities all happen behind the scenes of the websites and applications we use, and are orchestrated into a single digital experience.

When we need more real-time experiences in our applications, we choose asynchronous APIs, which allow applications to publish messages to APIs via channels and subscribe to channels to receive the messages needed. These asynchronous approaches allow for more seamless and engaging interactions, especially for high-volume streams of data that change frequently. Asynchronous API patterns are ideal for powering event-driven experiences. They help keep data segmented in different channels, bringing more order to the high volumes of information that will be published and subscribed to.

Synchronous and asynchronous API interactions each have strengths and weaknesses in powering applications. It is common for enterprise organizations to lay a base of raw digital resources and capabilities using synchronous APIs and microservices but then deliver richer, more real-time experiences using asynchronous APIs. It is also important

to remember that asynchronous APIs can have real-time characteristics via WebSockets and other protocols and patterns but also via simpler and lighter-weight webhooks. Teams should empathetically consider the needs of API consumers and decide whether synchronous or asynchronous APIs would be best.

If you are unsure which type of pattern is most appropriate for your applications, begin with synchronous. You can always begin with a layer of simple web APIs, then augment them with webhooks when APIs end up with a higher volume of data and polling from consumers. Unless you have definitive examples demonstrating how asynchronous patterns will benefit your applications, you should start with simple and much lower-cost synchronous APIs using HTTP. You can evolve and iterate upon your needs as demand grows.

## Synchronous interactions

The web allows us the ability to request HTML pages via URLs and receive a response that contains what we are looking for. It was a low-cost approach that has been applied to the ability to synchronously GET, POST, PUT, or DELETE digital resources and capabilities via the internet.

## Request

Every synchronous API interaction has a base set of properties that help keep resources secure. They also allow consumers to control the transport of their requests and tailor the requests to their needs, while sending data and media along as part of the body of each request.

**Authentication** – Require everyone making a request to provide authentication, ensuring that everyone using an API is supposed to be using it and keeping your environment secure.

**Parameters** - Allow each request to be customized by providing a set of parameters along with each request, transforming the response the API consumer is looking for.

**Headers** - Define and shape the transport of the request by setting the headers of the request used by the network and the server to handle the request and response.


**Body** - The machine-readable XML or JSON request body is sent as part of the request, providing the information submitted with each request.

Synchronous API requests provide a simple and low-cost way to request data, content, media, and other digital resources, helping standardize and scale the way API consumers access and pull digital resources via an interface that makes sense worldwide.

## Response

Each synchronous request has a response, returning the data, content, and media requested by the API consumer and possessing an HTTP status code to communicate success or failure.


**Status Code** - Standard HTTP status codes articulate the success or failure of a request in a way that any machine can be programmed to understand during handling.

**Headers** - The headers of the request define how the transport and shaping of the request was handled, allowing for the receiving system to understand how it arrived.


**Body** - The response data or message returned as part of the API request returns JSON or XML code that can be used in any application or integration using the API.

Responses return what each consumer was requesting, providing the raw data, content, or media that can then be rendered in applications and system-to-system integrations.

## Asynchronous interactions

When you are looking for a more real-time streaming experience for applications, you should consider asynchronous interactions to deliver the experiences consumers expect.

## Publish

Asynchronous APIs enable publishing messages to a particular channel via an asynchronous connection. That provides a robust way to publish large volumes of data via messages to a channel.

**Protocols** - We select HTTP, HTTP/2, TCP, MQTT, or some other common protocol that is used for an application to asynchronously publish data to internal systems for wider usage.

**Channel** - We publish messages to a specific topic (sometimes called a channel) to provide a context for each message published as part of the application using each API.

**Message** - The JSON or XML message that is published submits information to a system asynchronously, making it available for consumption by other systems via asynchronous API.


**Schema** - The schema is the structure of the message, standardizing how data is being published to the asynchronous API and ensuring that there is a standardized schema available to validate.

Depending on the protocol, an application may just publish and forget after it sends its message, but in some implementations, you may also simultaneously subscribe to the channel.

## Subscribe

Using asynchronous APIs, applications can subscribe to a variety of channels, receiving messages as they become available within a system and ensuring data is where it is needed as actions occur.

EXPERT PERSPECTIVE

# Investing in real-time and streaming experiences


When trying to understand where the world of APIs is headed, it is helpful to speak with the venture capital firms investing in the tooling that shapes how we do business. To expand my awareness of developer tools, I sat down on *Breaking Changes* with Nnamdi Iregbulem from Lightspeed Venture Partners to learn why the company is investing in the real-time and streaming API realm.

Nnamdi spends his time studying what is happening across enterprise organizations with streaming and real-time APIs. He is confident that the explosion in making data available across our operations and integrations is just the tip of the iceberg. Enterprises are making huge investments in modernizing their databases, data warehouses, and data lakes, and they are expanding their

DataOp teams. Teams are investing in infrastructure to deal with the large volume of real-time data they have to make sense of each day, leveraging APIs to stream data and also manage the infrastructure behind them.

These real-time data pipes and stores are increasingly important to enterprise operations, but investing in machine learning to make sense of the data really represents what is coming next. Nnamdi sees a massive need for training machine learning models on the real-time streaming data companies are generating.This isn't our grandfather's machine learning. Like APIs, today's machine learning systems are taking a very modular approach to training models as business occurs. The result is MLOps, a term describing how we effectively train the models, and ModelOps, which describes how we automate the management and workflows associated with applying and evolving models.

As the data and machine learning landscape expands across the enterprise, there are more tools for introducing automation and observability, both across data operations and across the machine learning layer that is being honed on top of the data. That helps both data and the machine learning models trained on top of it more visible, tangible, and understandable to teams. Visualizing data is nothing new, but visualizing how machine learning is being automated, iterated, and applied represents an entirely new market. According to Nnamdi, DevOps, DataOps, MLOps, and ModelOps continue to change the developer tooling landscape.

What I liked about Nnamdi's view of the real-time streaming API landscape is that he took notice of the growing power developers are acquiring as a result of these shifts. He identified the importance of the skills that will be needed to continue delivering real-time streaming data experiences enterprises are going to need in coming years. That makes skills involving asynchronous, streamlining, real-time, and event-driven APIs a very hot commodity today.

# 10
# Managing Your APIs

Your enterprise resources, capabilities, and experiences are all defined as APIs. The tools you use to manage them will determine how you balance the needs of your teams with those of your partners and customers. When you achieve the right balance, you can build and constantly improve upon your digital capabilities, benefiting everyone.

## 10.1 Optimizing API Infrastructure Management

Most enterprise API infrastructure has emerged over time, providing the resources and capabilities for a single web or mobile application, or possibly an integration with an external partner.

Standardization, efficiency, visibility, and attention to the enterprise legacy are often afterthoughts in this endless march forward, and few organizations have a strategy for managing the API-first transformation. In this book we intend to elevate your view of how you manage your API infrastructure so that you can perpetually work towards achieving the right balance for your teams.

While engaging with business and technical leaders undergoing digital transformation, we occasionally hear that APIs are just one part of what is needed. That's because many executives are looking at things through a business lens, with only a light understanding

of the systems, infrastructure, applications, and networks that will be needed today and tomorrow. Yes, there are many systems, applications, and processes involved in digital transformation, but APIs are the key element in every one of them. The cloud has APIs. Mobile phones have APIs. The IoT has APIs. Networks have APIs. Infrastructure has APIs. Business systems have APIs. Heck, even APIs have APIs.

APIs are the piece that gives you the control and visibility you need. Your enterprise resources, capabilities, and experiences are all defined as APIs–and so are those of your partners. APIs are how you automate and orchestrate your business operations in a digital world. It is common for people to have blinders on when it comes to business and technology outside their wheelhouse. Database people see databases. Mobile app developers see mobile apps. To really see APIs and have them in your wheelhouse means that you have awareness across every aspect of your business operations. To see APIs means you have visibility and control over the SaaS and other software you use (and if your software doesn't have APIs, you shouldn't be using it).

There are already many blind spots in business operations today. With so much of the business landscape defined as very abstract APIs that exist behind the more visible and tangible applications that use them, we end up with an exponential number of digital blind spots. An API strategy lays the foundation for how you will manage your API infrastructure beyond any of the applications, integrations, and automations that put APIs to work. Without a strategy for producing and consuming APIs, it becomes much more difficult and costly to optimize the digital pipes behind the applications you depend on, and you will be at the mercy of vendors for your operations. Managing your API infrastructure is the key to stabilizing your API-first transformation.

## Avoid creating an ad hoc API landscape

In the fast-paced enterprise business environment, there is a never-ending attraction towards defining digital resources and capabilities as one-offs to support specific projects, applications, and integrations. That leaves you with a legacy trail of APIs that are often redundant, overlapping, and in need of investment to remain reliable and continue delivering value for the business.

## Common distractions for API operations

There are plenty of distractions that cause organizations to lose their way of delivering, sustaining, and evolving their API landscape. Identifying common distractions is the first step in elevating your APIs into a single strategy that can move operations forward.

**One Project** - It is easy to see APIs as a subset of a project with clear start and completion dates, rather than part of a larger enterprise system and something that should be discoverable and usable beyond the project at hand.

**One Application** - The needs of a specific web or mobile application are often prioritized over the larger enterprise system, resulting in redundant or shadow APIs, which are invisible because they are obfuscated by apps.

**One Partner** - It is common to feel the need to respond to partner requests individually rather than seeing them as part of a larger system, resulting in unnecessary repetitive work. Defining a standardized set of partner resources would help speed up interactions.

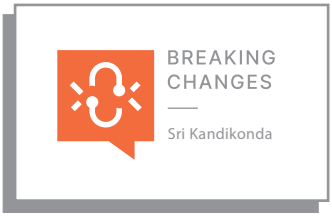
**One Integration** - Integrations between two internal or external systems often end up as one-off solutions, contributing to an ad hoc landscape rather than a ready-to-go integration toolbox for all business needs.

**One Customer** - Some customers are louder than others and may enjoy an outsized role in decisions, resulting in APIs that do not serve the wider needs of a customer base. They also lead to an ad hoc API business landscape and more overhead.

In the moment, each potential distraction might seem like a priority. Teams will be incentivized to deliver the project, application, or integration that meets the specific needs of a partner or a customer, resulting in redundancies and inefficiencies across operations. But with a centralized API strategy and well-defined API lifecycle and governance structures in place, your APIs can collectively be harnessed as part of the wider enterprise, with fewer distractions along the way.

**EXPERT PERSPECTIVE**

## The difference between projects and products at FedEx




Continuing the ongoing discussions I am having with leading API practitioners, I had Sri Kandikonda, Principal Product Manager at FedEx, join me on *Breaking Changes* to talk about the evolution of API product management and why it is so important to businesses today. I was eager to learn how FedEx sees digital products. We started our conversation by discussing the fundamental difference between viewing APIs as projects or products.

I have heard several compelling definitions of product management, but Sri shared with me the fundamental difference between products and projects. He compared a product to a baby that needs to be nurtured and carefully managed over a period of time. Most importantly, he said your products must align your business model, provide clear revenue or value streams, and solve real problems for your consumers. In API circles, we have been too focused on rolling out APIs.Sri said it is critical that we start with the why, not the what. Each API product must solve a real-world business need; otherwise, it will never matter, no matter how well-designed or well-implemented it is.

Sri walked me through the landscape of the different types of APIs at FedEx, from microservices to public APIs, noting that not all APIs are worthy of being treated as products–some entail too many concerns about maturity, visibility, and other matters. Sri also said the concept of APIs as a product applies both internally and externally, adding that what matters most is that your APIs are customer-centric, focused on solving real business problems and provide a rich and active feedback loop with consumers.

Sri also shared the importance of self-service. If API consumers can't immediately see the value of your API products and easily begin putting them to work in their applications and integrations, they won't mean much. It's important for developers to have empathy with consumers to discover the experiences that matter most to them, then produce documentation, content, and other resources that speak to that experience. Then, get out of the way.

Sri's view of the API landscape and API product management shows what is missing across most API operations. I am finding there is an insatiable appetite for knowledge in this area. Sri said the subject of product management is ubiquitous across university curriculums, and with the evolution of SaaS and technical product management, learning about API product management will be the next logical step. Let's hope so because bringing API products into alignment with enterprise goals and consumer needs is essential for doing business online today.

## 10.2 Using APIs Across Many Types of Applications

APIs deliver resources and digital capabilities across multiple types of applications, powering many different web, mobile, and device services, as well as providing system-

to-system integrations and automations. The way enterprises see applications and integrations begins to evolve once they begin shifting towards API-first operations, pushing the boundaries of what applications mean and how they become part of business operations.

## Types

APIs have powered applications since the beginning of computers. The current breed of web APIs found its roots in aggregating content for web applications, then quickly expanded to mobile devices, back to the desktop, and even to the network beneath our applications.

**Desktop** - APIs are how desktop applications on our Windows and Mac PCs exchange data with the server, making hundreds or thousands of calls each day to create, read, update, and delete information as needed.

**Web** - Websites began as simple HTML documents, but have become a dynamic mix of many internal and external API calls. These are stitched together to provide the desired online experience within a specific domain, providing a richer web experience than was possible before.

**Mobile** - Mobile phones allow websites and applications to be accessible in our hands, turning a device for voice or messaging into a rich application ecosystem that sends and receives data across many different cloud platforms.

**Device** - Once developers realized that APIs could power mobile applications, they moved on to making internet-enabled televisions, thermostats, automobiles, and other IoT products.

**Network** - As more of the infrastructure we depend on for the web moved to the clouds, the network connecting servers and clients has also become API-enabled, making the network composable and configurable and changing how we operate applications.

These constructs all shape what many think of as an “application,” but APIs and the automation and orchestration they power are rapidly changing the notion of how we “apply” our digital resources and capabilities both online and offline. The size and scope of applications is getting smaller, but their evolution is picking up velocity, moving at the pace of consumer demand.

The definition of an application is the action of putting something into operation. As you progress in your API-first transformation and begin elevating your API strategy above

any single application, your definition of an application will begin to evolve to reflect what matters most to your teams and your customers.

## 10.3 Strengthening your Partnerships with APIs

As more business is conducted online through partner SaaS applications and other external services, the urgency to make internal APIs available to partners has increased. The line between inside and outside the firewall is becoming blurred as making digital resources available to trusted external partners becomes a priority.

**Purpose** - You must offer partners a clear menu of enterprise resources and capabilities through APIs and self-service, making operations streamlined and repeatable across relationships.

**Onboarding** - With APIs, you can reduce partner onboarding time from weeks or months to hours or days, automating the process as much as possible. That will reduce friction when qualifying partners and give them access faster.

**Access** - It is common to give early and exclusive access to API resources and capabilities to specific r groups of partners before you offer them to the general public.

**Innovation** - To incentivize partner innovation and provide preferred access to fuel the development of new and interesting products and services. Lean on partners to deliver applications, integration plugins, and other interesting business use cases.

**Exposure** - APIs in partnership scenarios expose your organization to new opportunities for responding to their needs in any situation.

**Marketing** - The digital resources and capabilities made available via APIs provide what you need to develop effective marketing campaigns. Being API-first allows your teams to reach new levels of marketing automation and do more with less, generating a bigger impact.

**Branding** - There is an opportunity to extend your brand across external communities, leveraging APIs to aggregate content across social networks and push messages out through partner platforms.

**Communications** - Feedback loops are a natural part of modern API operations. Gathering feedback from partners informs the API roadmap, helping to drive business forward.

Self-service APIs equipped with a well-defined onboarding process provide a proven way to securely connect businesses with partners, enabling synchronous and asynchronous digital transactions via APIs.

An organization-wide API strategy allows you to respond to partner needs without advance preparation. Investing in your API-first transformation supports partner relationships by making your digital resources and capabilities available to partners, allowing them to build the next generation of experiences—which in turn strengthens your own business.

**EXPERT PERSPECTIVE**

## Powering partner creativity across NBA teams


It is no secret that APIs power partnerships. But I was pleased to learn that by being API-first, the National Basketball Association (NBA) is able to engage with all of its disparate teams, fostering creativity and autonomy, while still successfully operating as a single brand. We hear a lot about using a federated approach to API gateways and governance and how it helps address the sprawling enterprise landscape, but the NBA provides a solid example of how taking a federated approach to APIs for content and media can have a massive impact on business.

Chris Van Patten, Director of Digital Product Development for the NBA, joined me on *Breaking Changes* to talk about the association's recent shift to an API-first stance for content management and how it helped to meet the digital needs of individual teams. First, I asked Chris what API-first meant to him. He said it all begins with the contract. You have a content management system (CMS) on one side and one or more front-ends on the other side, and the API contract manages everything in-between. This API-first approach allows the NBA to strike a balance between having the access and control it needs while still meeting the needs of each team and its fans. Each team is unique and autonomous, but still part of the NBA, which has the ultimate say in how content and media are presented across the network of web and mobile properties.

The NBA provides us with a powerful example of how API-first can be applied through federation. There are 29 separate teams, each with its own web and mobile properties. While the NBA can dictate and control many things, the organization and its fans are better off when each team is allowed to be as autonomous and creative as possible–while still representing the parent brand. For Chris, being API-first means balancing the central organization with team needs across a very complex mix of content, media, schedules, and other constraints that define professional sports today. Being API-first and API contract-driven allows the NBA to deal with the complexities of its operations, establishing alignment with business goals while allowing for innovation to occur centrally and across teams. And that's really what APIs are all about.

My conversation with Chris reinforced my thinking about how we can take advantage of APIs using a federated model, using it not just for data, but to better meet our content and media needs. It reminded me that all the nuances of data, applications, content, media, and algorithms we use are really just the base for how we deliver the next generation of experiences to empower our partners.

## 10.4 Leveraging APIs for Integrations

The enterprise digital landscape is increasingly made up of internal and external interconnected systems. Modern web APIs define how these systems interoperate, and how they can evolve and be automated. APIs are connecting all parts of enterprise operations, leveraging internal and external APIs to stitch together the expanding digital landscape required to do business today.


**SaaS** - APIs enable the SaaS solutions we depend on to run our businesses to run seamlessly with existing enterprise operations, allowing aspects of business to be outsourced while still maintaining control over usage.

**Infrastructure** - The existing infrastructure likely already possesses APIs, providing a huge opportunity for more automation and orchestration across the software already in place. This is your low-hanging fruit.

**Interoperability** - APIs define how distributed and federated parts of enterprise operations are made interoperable, and how acquisitions, partners, and industry- level interoperability is set into motion. APIs reduce the challenges involved in working across business divides.

**Syncing** - Syncing data across operations and with external partners and services is a regular part of business operations. APIs enable you to do that as efficiently and cost-effectively as possible, allowing for more alignment across business domains.

**Migrations** - APIs are how data and objects are migrated between servers, clouds, partners, and other constructs of our regular operations. They ensure that small or large amounts of data are made accessible, so that they can be migrated to a new location for optimal usage.

**Automation** - The only way an organization can remain competitive in this digital landscape is through the automation of common business processes. APIs define those processes and the digital resources they depend on, providing the information t we need to automate them.

**Orchestration** - APIs provide the knobs and levers that can be scheduled, triggered, and pulled to keep business moving forward, producing just the right performance for our operations.

The average enterprise landscape has thousands of connected servers and platforms. APIs determine how they are made whole (or left operating in isolation). APIs are essential to the interoperability of a distributed enterprise and the industry it operates in.

**EXPERT PERSPECTIVE**

## Baking APIs into top platforms at Shutterstock


When I sat down with Alex Reynolds, the former Vice President and General Manager for Platform Solutions at Shutterstock, I expected a pretty straightforward API story about a public API portal and developers building applications for the image and video company. Though this was certainly part of our conversation, what really caught my attention was how the company is baking their API solutions into other leading platforms, providing next-level API adoption.

From what I gathered, Alex's experience in business development and his platform vision contributed to the company's approach, and it is proving to be a very winning formula. Shutterstock didn't have a "build it and they will come"

mentality with their public API portal. Instead, the company got to work building the trust of leading platforms like Wix and Facebook, baking its image and video solutions into the business workflows of website operators, marketers, and other people already using these platforms on a daily basis.

Shutterstock's team approaches the development and iteration of their APIs with a heavy emphasis on partner engagement, workflows, and feedback loops. The digital media company injects itself into essential business processes and workflows, then leverages partner feedback loops to experiment and iterate to better meet the needs of consumers. Shutterstock is essentially maintaining a real-time business development feedback loop that is baked into other leading platforms. The business value generated by partner relationships is important, but the feedback loop from workflows on partner platforms is where the company's future business value lies.

The experiences Shutterstock's raw API resources are delivering via partner platforms are moving the API program forward. The feedback loop and product focus surrounding their API operations power the road map, informing teams when iterating upon each API but also creating new APIs that provide machine learning, analytics, and other essential resources for use in the next generation of APIs. Partnerships have become the cornerstones of Shutterstock's API-first transformation, allowing the company to keep its finger on the pulse of consumer activity across a broad spectrum.

## 10.5 Reshaping your Legacy Systems with APIs

APIs are widely used to address challenges with legacy infrastructure.Teams are finding success with more modular approaches, refactoring order systems to work with modern applications and integrations, and in some cases doing away with legacy infrastructure altogether. APIs are helping enterprises deal with technical debt and future-proof operations against further technology debt. They do that by ensuring that systems are smaller, more modular, and capable of evolving and deprecating without the overhead associated with legacy systems.

**Monolith** - Teams are using APIs to decouple and redefine monolithic legacy systems, reverse-engineering and implementing microservices and APIs to modernize legacy systems. They are breaking business down into more manageable components that can evolve.

**Microservices** - Modular, single-use, synchronous and asynchronous microservices are redefining legacy systems, allowing them to become more distributed with components that are easier to evolve and reuse.

**Facades** - APIs create facades that provide modern interfaces for applications, working to evolve or deprecate infrastructure. That allows technical debt to be abstracted away so it can be sustained or deprecated.

**Gateways** - Gateways provide an industrial-grade approach foro standing up a modernized stack in front of legacy systems. They provide a single modernized entry point for all API infrastructure without exposing back-end systems.

**Proxies** - Proxies intercept and map out the traffic in legacy applications. Proxies, APM, and other solutions can map out exactly what legacy systems do, applying the information to define and deliver facades and microservices.

**Deprecation** - There may come a time when legacy systems can be deprecated, relying on proxies, gateways, facades, and microservices to chip away at how systems are used in applications and modernize enterprise infrastructure.

**Sustainment** - While not desirable, there are many situations where legacy systems may not be able to be deprecated for some time. Organizations must undertake an honest assessment of how gateways and other approaches can be used to sustain legacy infrastructure.

APIs are essential for modernizing legacy infrastructure, providing interfaces that can be used in applications, while abstracting away legacy solutions undergoing modernization. APIs can help modernize existing infrastructure without disrupting applications and integrations dependent on the digital resources and capabilities delivered by legacy solutions.

Ideally, the modernization of your legacy systems will not be a separate project from your regular operations. Early in your API-first transformation, you may have to dedicate resources to mapping out and redefining your legacy infrastructure landscape. But as you move forward, this task should become part of every team's regular work, and everyone should be addressing legacy technical debt along with the new.

EXPERT PERSPECTIVE

# Perpetually addressing legacy technology is part of being API-first


While chatting with Rico Cordova, Head of Content Engineering for Samsung TV Plus, I was eager to learn more about his approach to dealing with the technical debt built into the company's regular development processes. Rico is laser-focused on developing highly optimized teams who can deal with anything thrown at them. He believes that when you are building new systems, you are equally responsible for taking care of your legacy technology.

Rico is a veteran when assembling high-performing teams capable of developing the microservices and APIs needed to deliver digital media at competitive speed. I was impressed with the way he has his teams document APIs and processes, making sure everyone has the education and skills they need to deal with change and move at the velocity this industry demands. Rico believes your technical debt is directly related to the velocity at which your teams can move, so it was critical for teams to address their technical debt in real time as part of their regular work.

Think about it. The work required to address technical debt is a regular part of defining, designing, and delivering new resources and capabilities. This means you don't have to trace and discover your dependencies, or have entirely separate teams map out the legacy landscape and work to modernize it. Each team understands that their technical debt is directly related to their forward motion.

To do new and interesting things, you must address what has come before. Rico also shared a healthy view of what legacy means, reflecting on why we see our technical legacy as a negative thing. He ultimately agrees that bad technical debt is the result of inadequate real-time processes that are woven into operations and remain unaddressed.

I spend a lot of time thinking about the relationship between APIs and change management.Talking with Rico gave me a whole new perspective on how an API-first approach helps us address our technical legacy. I have long seen how

you can use APIs to deprecate, then modernize your legacy systems. But Rico's views shifted how I see the responsibility of each team, not just for dealing with technical debt, but helping to make sure the organization's technical legacy isn't seen as a negative.

## 10.6 Open Source Solutions

Open source tooling has become the bedrock of every enterprise organization. Open source standards and specifications are rapidly providing the nutrients needed for a healthy and vibrant API life cycle. If your organization simply sees open source as a way of bypassing the costs of commercial solutions, then you aren't realizing the full business potential of open source.

Most people associate open source with tooling, but there are a range of elements that can make it change the way that you do business as part of your API-first transformation.

**Tooling** - Open source tooling and packaging e is widespread across enterprise organizations, but teams rarely measure how it is used to stitch together procedures across operations.

**Specifications** - OpenAPI, AsyncAPI, JSON Schema, GraphQL, and other API specifications have become standard machine-readable contracts for conducting business through self-service and automation, helping to stabilize our operations.

**Standards** - Internet and industry standards are beginning to stabilize finance and healthcare–the top two industries. They provide more structure for these industries in the EU and U.S. and offer a model for other countries and industries to follow.

**Content** - Open source doesn't stop with tooling, specifications, and standards. It is increasingly used to produce content and media, changing the way enterprises produce and consume APIs across teams.

**Licensing** - The legal licensing for tooling, specifications, standards, and content is how most people define the quality of open source. While licensing agreements do set the tone for what is possible, they are just the starting point for real value.

**Contributions** - Open source solutions are only as strong as their contributors. Their worth depends on the passion, skills, and the free time of people who know the domain the open source is applied to and have invested in understanding what is being built.

**Collaboration** - Without collaboration and coordination in open source tooling, specifications, standards, and content, you end up with something that is rarely active and alive, and won't be around for very long.

**Communication** - Communication between open source contributors and consumers can be difficult because the usual gates associated with proprietary offerings don't exist, making open channels essential for health.

Every enterprise is an open source consumer. You will begin to see the real benefits of open source across your teams when you become a contributor. You are the one with domain expertise within your industry, and that could really benefit open source tooling, specifications, standards, and content it uses.

Don't let the lawyers at your company scare you off from participating in open source communities. Do the work to bring awareness across teams, and also among leadership. At the very least, start by making sure you are aware of how open source is currently being used.

**EXPERT PERSPECTIVE**

## Providing for the next generation of open source needs


Like every other aspect of technology, the world of open source tooling, specifications, standards, and content is rapidly expanding and evolving. I recently spoke with Pia Mancini, Co-Founder and CEO of Open Collective, about what open source maintainers and contributors need today and where things are headed. Open Collective is at the heart of funding and providing governance for open-source projects. In my opinion, it reflects the front lines of future open-source development.

At first glance, Open Collective appears to be a way for open-source maintainers to raise funds. This is definitely at the heart of what Open Collective does, but doing it across many jurisdictions across the globe is what makes it a powerful platform for open-source solutions. According to the organization's website, "Open Collective is a legal and financial toolbox for grassroots groups. It's a fundraising + legal status + money management platform for your community." It is a game changer for open-source maintainers living outside the markets where their sponsors and contributors operate, opening the door to the funds they need to keep operating and expanding.

What I didn't understand about Open Collective is that the money is just the beginning. The organization also provides a governance framework that groups can use to establish a more solid platform. Most open-source maintainers do not have the experience required to govern their communities. Open Collective provides an opportunity for them to tap into a well-defined, ready-to-go scaffolding, enabling them to govern and navigate the complicated arena of open source. Most people do not go into open source with plans for organizing and operating what they are building—they are just looking to deliver a technical solution that meets the needs of existing or potential consumers. Open Collective provides organizational structure, letting open source maintainers focus on doing what they do best.

You can see some of what the Open Collective presents in the Apache and Linux Foundations. What you don't see is how the organization is responding to the increasingly diverse needs of open-source maintainers, contributors, and sponsors around the globe. As open source continues to expand, it is becoming increasingly diverse. As technology extends further into every area of life, we are seeing open source involved in conflict zones like Ukraine or Afghanistan. It is at the front lines of almost every social justice issue you read about in the news. This is a reality that Pia and the Open Collective are tackling head-on, realizing that the old days of open source are rapidly fading, and its future lies in interacting with diversity and social justice issues around the world.

## 10.7 Managing API Access

APIs are abstract and difficult to see, creating anxiety about who has access to put digital resources and capabilities to use. While a significant amount of discussion in the last decade has been about public APIs, the majority of APIs are only available privately, existing in the shadows behind our web and mobile applications.

## Access

It is critical for organizations to effectively control the visibility of their APIs, quickly and confidently moving them from private team use to make them available to partners or third-party developers. This ability will play an outsized role in the velocity of enterprises doing business today.

**Private** - Being private means keeping APIs and the operations around them private and available to stakeholders on an invitation-only basis.

**Team** - You can limit access to APIs, workspaces, documentation, and other elements of the API life cycle to the teams who will be producing or consuming them internally. Eventually, you may decide to make them available to partners or public consumers.

**Group** - In many organizations, APIs are only available to a specific group, domain, or legacy tribal boundary. Access is limited based on lines of business and the needs of applications and integrations within a single group.

**Partner** - You can also choose to expose APIs, documentation, mock servers, environments, and testing to trusted external partners. That will allow them to view or contribute to producing or consuming APIs, using workspaces and repositories to engage across the life cycle.

**Public** - It is common to make workspaces, APIs, and other elements available to the public, applying necessary authentication and access controls. That allows anyone to watch, fork, learn from, and work with the APIs around them.

API visibility can be anxiety-inducing if enterprises do not have organizational-wide API management, authentication, access controls, and other security practices in place. APIs are all about striking the right balance between access and control, while keeping everything in alignment with business needs and consumer expectations.

It is common for teams to believe that internal, private, team, or group APIs do not need the same level of investment in design, documentation, testing as other elements of API operations. However, if teams are consistent in how they treat APIs, the transition from private to public becomes much less anxiety-inducing. The organizational transition from API-early to API-first is all about the mastery of tools and processes needed to confidently move from private to public with API resources in as short a time as possible maintaining the highest levels of quality, reliability, and security.

# 11
# Developing Governance and Standards

API governance is established by the organization's leaders, who must set a clear strategy for all API operations and make sure all teams understand and follow the rules. The goal of governance is to help teams deliver consistent APIs across a consistent API life cycle, no matter what type of API products they may be developing.

## 11.1 Managing API Governance

Your governance sets the rules your teams will use to work with, roll out, and manage APIs across the enterprise. Here are some of the factors leaders consider in creating a governance strategy and making guidelines available to the teams who will use them.

### The elements of governance

**Shape** - The shape of governance depends in part on the existing organizational apparatus. You must always ensure that API operations are in alignment with the business.

**Domains** - Governance means carving operations into logical bounded contexts that can be used to define and shape how teams operate.

**Guidelines** - Formal documentation, wikis, or other documents define your governance and enable teams to do the right thing as part of their work.

**Maturity** - You should have a clear definition of what constitutes API maturity, while allowing for different levels of maturity to coexist with a balanced set of expectations.

**Standards** - Teams should have a strong and ever-evolving awareness of standards that exist inside and outside the enterprise, and a strategy for how they will be applied.

**Templates** - Provide as many reusable templates as you can to help demonstrate and apply patterns, standards, and other elements to APIs and the operations surrounding them.

**Rules** - Establish sets of linting rules that can be applied at design time to guide the creation of standardized APIs and applied across the entire API life cycle.

**Policies** - Define standard source control, CI/CD, gateway, and other policies to help govern API operations, standardizing the configuration and shape of API production.

**Centralization** - Consider which parts of governance should be centralized, developing a single body within the enterprise to help guide governance.

**Federation** - Consider which parts of governance should be federated, relying on teams to define, shape, and lead when it comes to their own enablement

**Design Reviews** - Formal reviews examine the design of APIs, providing self-service. Peer reviews also help API producers consider the big picture when designing.

**Quality Reviews** - Formal reviews help ensure that all APIs are fully documented and properly tested.

**Security Reviews** - Formal security reviews look at the security of each API, ensuring that encryption, authentication, authorization, and other security elements are in place.

**Enablement** - Governance on the ground floor enables teams to do the right thing throughout their regular work.

It is important to remember that capital “G” governance is only a concept that lives at a high level of the enterprise.On the ground floor is where lowercase “g” governance operates. Governance is about enabling API teams to do the right thing without having to think twice or work very hard. Leaders provide the standards, processes, and tools to help them do that.

## Finding the right shape for your governance

The structure of API governance must be established at the highest levels of the organization–otherwise, it will just be a low-level vision realized by a handful of teams. To have the greatest impact, governance needs to reflect the particular organization that is applying it. To do that, leaders must invest in resources, time, and people to develop a system that will guide teams in learning about, understanding, applying, and reporting on governance rules, as well as providing feedback on what is working and what is not.

**Structure** - Provide a clear structure for how governance will be executed, balancing a top-down with a bottom-up approach to looking at it.

**Leadership** - Select a group of business and technical leaders, bringing together a mix of skills and domain expertise.

**Guidelines** - Provide details on the standards for designing APIs, but also for documentation, testing, and other aspects of team API operations.

**Domains** - Establishing domains within the enterprise will allow for logical separation of business concerns in accordance with the dictates of domain experts.

**Groups** - Establish a logical separate of teams, grouping them by domain, line of business, project, or another bounded context that makes sense.

**Teams** - Define the team of people behind API operations, providing names, roles, and other relevant details about who they are and what they will be contributing to the team.

**Workspaces** - An API workspace strategy lays out the API factory floor for the enterprise, establishing naming conventions and other patterns to enable teams.

**SingleSign-On** - Provide teams with an authentication scheme allowing them to log in with a single ID across the multiple services they will need for work.

**System for Cross-domain Identity Management (SCIM)** - Leverage the SCIM standard for automating the provisioning and deprovisioning of team member accounts.

**RBAC** - Role based access control should be applied at the authorization layer of an API, but also to the API operations around it, helping govern who has access to operations.

How you shape your governance will set the pace for your teams, enabling them to move forward, deal with change, and feel like they are part of defining and evolving the shape of governance. By making teams a part of the governance discussion, you will ensure that your guidance is realistic and easily adopted.

## The parts of API governance that need to be centralized

Centralizing practices and information across the enterprise organization helps streamline and standardize teams' work. Centralization provides the consistent nutrients teams need to be successful and allows disparate groups to produce consistent and valuable services.

**Excellence** - Set up a center of excellence, bringing together all necessary knowledge, skills, and practices and working to evangelize them across domains and teams to facilitate awareness, participation, and feedback.

**Expertise** - Create a central group of leaders with different areas of expertise. Make sure they meet regularly to identify and evolve the knowledge and practices teams need to be successful.

**Leadership** - Establish clear leadership for centralized governance, taking a lead role in demonstrating how and why API governance matters. You can do that without ever having to say the word "governance" or being seen as an enforcer.

**Domains** - Thoughtfully carve the enterprise into logical domains that reflect, but transcend tribal boundaries that emerge from lines of business and legacy acquisitions. Establish clear articulations of your business domains.

**Vocabulary** - Define the common vocabulary used within domains, using the language teams will use when designing APIs. Also engage with consumers of those APIs, making producing and consuming APIs as intuitive as it can possibly be.

**Rules** – Craft and evolve linting rules across domain contracts and artifacts, encouraging teams to apply them. They can even be applied as part of policies used across gateways and other stops along the life cycle to help stabilize API operations.

**Enablement** - Provide the best possible services, tooling, standards, and other resources for teams, making it easy for them to deliver and operate consistent APIs, no matter which team created and owns them.

**Feedback Loop** - Foster an active two-way feedback loop with teams, encouraging feedback on governance. Allow teams to dictate the evolution of centralized guidance as it is applied on the ground across federated API teams.

How centralized your governance is will vary according to your company's structure, culture, and industry. This is the reality on the ground. It is also the reason you should have ongoing conversations with teams as part of your governance feedback loop.

## 11.2 Improving Organization with Domain-Driven Design (DDD)

The enterprise is a complex landscape of digital activit.Domain-driven design, or DDD, has emerged to help enterprise architects and other stakeholders define, organize, and communicate about all of the digital resources, capabilities, and experiences across the business. DDD allows business and technology leaders to establish better alignment and clarity across operations, providing the vocabulary, patterns, and standards that enable teams to build reliable, consistent software using APIs.

**Models** - Models are software abstractions that describe business logic. Developing models helps narrow the divide between code and a description of business operations and value, providing us with a way of quantifying the digital resources we use across APIs.

**Bounded Contexts -** We define enterprise operations as large models by dividing them into different bounded contexts, while being clear about their interrelationships. That provides meaningful segmentation that transcends legacy tribal boundaries.

**Command Query Responsibility Segregation (CQRS) –** A CQRS approach allows you to use different models for updating or reading information, providing more abstractions, and in some cases, more complexity.

**Ubiquitous Language** - Ubiquitous Language is the practice of building a common, rigorous language between developers, users, and business stakeholders, ensuring that everyone involved with API operations is on the same page and uses a common vocabulary.

**Microservices** - Using microservices means designing software applications as suites of independently deployable services. These services can include business capabilities, automated deployments, and business intelligence. Microservices provide decentralized control of languages and data.

**Dependencies** - DDD helps provide an honest and clear view of the dependencies between APIs, mapping the technical ways they work together, while also providing visibility.

**Patterns** - With DDD, you identify existing patterns across the APIs and microservices in use. Then you can spread the use of these patterns across teams, while working to introduce new healthy patterns into regular team workflows.

**Experts** - Domain experts should always have a seat at the table in modeling business operations. They help segregate responsibility and define the Ubiquitous Language you use. Including them ensures that the business domain always has a voice.

Domain-driven design isn't just about these tangible elements.lit is about enabling your teams and articulating the domains you are operating in. That will translate into the overall design and experience of your APIs. DDD represents the order (or chaos) that exists across API operations.

EXPERT PERSPECTIVE

# Events, workflows, and automation in the energy industry


One of the common questions I hear from enterprise teams is how to define and design their APIs. But as we stated in the opening of this book, there is no single successful way to do APIs. Only you possess the domain knowledge needed to define and design APIs that matter for your business. I saw an example of this in action during my *Breaking Changes* conversation with Robert Flowers, Senior Product Owner at Duke Energy. Robert is a domain expert in the energy space, and his view across the business, technical, and regulatory landscape provides an inside look at API-first transformation in that industry.

Robert walked me through all of the elements of a healthy API-first transformation in progress. It started with modernizing legacy systems and applying domain-driven design principles to establish alignment between business teams, development teams, regulators, and the energy industry. Then Duke Energy began defining all their digital resources using RESTful synchronous APIs, actively mapping out all meaningful events that are occurring and making the information available via asynchronous APIs. Duke is investing in a data mesh to define and connect all the sources of data across the enterprise, producing a single layer teams can use to discover digital resources when developing applications and integrations. Robert and Duke Energy clearly understand the importance of laying a strong base foundation for the operations that they've built and how doing that is critical to their overall API-first transformation.

Everything Duke Energy is building enables the workflows and automation the company will need to operate, compete, and innovate in the energy sector. Duke Energy is heavily investing in the schema, contracts, change management, and governance required to deliver the capabilities they need to compete in tomorrow's energy market and deliver the physical and digital experiences their partners and customers need.As a by-product of this investment, they will be able to more easily respond to industry regulation that contains to shape this important layer of our society. Duke Energy still has a lot of work to do in its API-first transformation, but the important thing is that they

are doing the work, laying the foundation. Robert and his team fully understand the value of the data they possess. They realize that having productive teams and high-quality, well-governed, and consistent APIs will allow them to respond to events as they occur and build the workflows and automation they need to optimize business operations.

One seismic shift I've noticed over the past couple of years is that companies that have been doing APIs for a while–those that have a solid base of digital resources and well-honed processes behind them–are now ahead of the game. These companies are beginning to deliver more meaningful experiences for their consumers because they have a base of resources available to do it.

## 11.3 Providing Guidelines and Guardrails for Your Teams

Governance guidelines are commonly used to inform all stakeholders in the API life cycle about API strategy. They express how teams should be pushing APIs across a common API life cycle. Guidelines help govern not just the design, but every other stop along this life cycle. They are essential to getting teams on the same page as they produce APIs across operations.

**Strategy** - The API strategy distills the plan into something that can be articulated across the organization, speaking to both technical and business groups.

**Protocols** - It's important to establish a toolbox of protocols available to teams, providing a baseline for protocol selection and application.

**Patterns** - A catalog of common patterns provides examples of how to define, design, develop, and operate APIs across a standardized API life cycle for all teams.

**Standards** - Document all standards available for designing and developing APIs, offering a fully fleshed-out set of common standards that match each domain.

**Templates** - Offer a catalog of common templates, providing examples of how teams can define, design, develop, and operate APIs across a standardized API life cycle.

**Errors** - Provide standardized and well-defined guidance for handling errors, ideally following Internet standards.

**Security** - Make security policies and practices a part of guidance, providing tools and other resources that shift security left and enable teams to apply best practices early.

**Lifecycle** - Document the API life cycle in motion across the enterprise, providing a common vocabulary for describing each stop along the life cycle for teams.

**Tooling** - Providing a catalog of the tools available to service every stop along the API life cycle, including a mix of platform, third-party, and open-source solutions for teams.

Your governance guidance should start simple. If you need help, look at the guidelines of top public API providers, then define what matters most to your operations and communicate that to your teams. Then keep iterating upon it, learning from your teams and your consumers. Continue updating your guidance and keep evolving and responding to concerns that matter.

Remember, none of your developers will care about API governance. A lack of governance really boils down to a lack of support and enablement for teams on the ground floor. If your API governance falls short, make sure you review the resources and tools you have provided to your teams.

## 11.4 Defining and Communicating API Maturity

Not all APIs are created equal. Still, many things we apply to our public APIs can help prepare our less mature internal APIs for consumption beyond internal teams. Without a clear understanding of what constitutes maturity across various APIs, teams are left trying to do the best they can in the moment. However, once we have a structured way of defining and communicating what API maturity means, we can better comprehend where each of our APIs stand.

**Version** - An organization-wide approach to versioning and communicating APIs provides a foundation for defining maturity. APIs tend to mature as they change, and versioning facilitates community maturity.

**Consumers** - One of the best ways to mature and harden an API is for people to use it. Each consumer of your API provides one more potential vote in your API maturity ranking, helping validate that you are heading in the right direction.

**Metrics** - Possessing, reporting upon, and regularly questioning the metrics you are using to define success of your APIs will shape how you define maturity. A lack of measurement means you are unlikely to move forward in ways you know about and can learn from.

**Feedback** - Receiving feedback from your API consumers, then incorporating that feedback intelligently into your roadmap is how you "hear" your APIs. The more mature your feedback loop is, the more mature your APIs and your operations will be.

**Reliability** - Having a baseline of testing, security, and governance for your APIs provides the reliability you need to convey maturity to your consumers. Nobody will consider your API mature if you break contracts and do not perform as expected.

**Lifecycle** - One of the things that will help you offer more mature digital resources, capabilities, and experiences is having a well-defined and well-executed life cycle. Without a shared understanding of each step, your teams will never reach maturity.

**Observability** - Observability means being able to understand and control the state of your APIs using the existing outputs of your operations. It is key to API maturity. If you can't measure and see your APIs and the operations around them, you'll never stabilize your APIs.

**Governance** - You must eEstablish a baseline for the consistency of your APIs and the life cycle around them to mature the API products you deliver across domains, and teams. Becoming more consistent is an important part of achieving API maturity.

Just as achieving maturity means different things to different people or businesses, achieving API maturity depends on your overall business strategy and operations. Still, mature API operations do share some common characteristics. They also lack the chaos and unreliability of less mature API operations. Discussing API maturity will help you realize what you need to do to deliver more reliable API products that meet the needs of your applications and integrations.

# 11.5 The Importance of Standards across the API Life Cycle

Standards keep us from reinventing the wheel with our digital resources and capabilities. They also help ensure that our APIs interoperate with other systems as much as possible. A wide variety of general and precise API standards exists, providing us with solutions ranging from the backbone of the internet-level considerations to common healthy patterns to use for making APIs intuitive and easy to use.

When you are doing business on the web, it doesn't make sense to compete on the naming and ordering of the interfaces you produce, consume, and depend on across your applications and integrations. You already depend upon industry standards like HTML for websites and SMTP/POP for email. It doesn't make good business sense to manage your digital resources, capabilities, and experiences in a proprietary way. Standardizing encourages interoperability and reuse within your organization, as well as within your industry. It doesn't give your competitors any advantages, and it will allow you to move faster.

Standardization of your API operations begins with the protocols and patterns you already use to design and deliver your APIs. After that, it is an ongoing evolution of your infrastructure, practices, policies, and the education you provide your teams throughout your API-first transformation. Standardization does not mean finding a set of standards, applying them, and then you are done. Using standards means being aware of what is and is not working in your API infrastructure. Using established standards helps make things more consistent and interoperable. Exploring standards encourages you to consider how you can further standardize based upon what you are already doing.

It is common for people to think of standards bodies like IETF and IANA, or industry-level standards. These are formal examples of standards, but we are also talking about how you define your API life cycle and apply governance across your operations, as well as the design of each API.

Standardization is about properly defining how you do what you do, sharing that across as much of your organization as possible, and treating your standards as living and evolving over time. Standards stabilize how you do APIs so that teams can move faster, with higher levels of quality and reliability.

Begin the standardization portion of your API-first transformation by documenting the standards you are already using, such as HTTP, JSON, and OpenAPI. Then seek low-hanging fruit for designing, delivering, and operating your APIs, continuing to invest in standardization across your operations.

# 11.6 Internet and Industry Standards

APIs are the next evolution of the web. A number of existing internet standards should be applied regularly as part of API operations, as early in the life cycle as possible.

**Internet Assigned Numbers Authority (IANA)** - The Internet Assigned Numbers Authority (IANA) is a standards organization that oversees global IP address allocation, autonomous system number allocation, root zone management in the Domain Name System (DNS), media types, and other Internet Protocol-related symbols and internet numbers.

**Request for Comments (RFCs)** - RFCs are a publication in a series from the principal technical development bodies for the internet, most prominently the Internet Engineering Task Force. They are authored by individuals or groups of engineers in the form of a memorandum describing the methods, behaviors, research, or innovations that apply to the working of the internet and internet-connected systems.

Do not reinvent the wheel for designing and transporting your digital resources and capabilities. It is extremely likely that there is already a standard out there to support what you are doing.

## Industry Standards

Data and other interoperability and communication standards have existed at the industry level since the birth of computers and the internet. More recently, they are evolving to support modern approaches to delivering APIs behind web, mobile, device, and other types of applications.


**PSD2/PSD2** - This is a European Union Directive administered by the European Commission to regulate payment services and payment service providers throughout the European Union (EU) and the European Economic Area (EEA), providing a common set of industry API standards for financial enterprises and providers of tools and services to follow when delivering API infrastructure.

**Fast Healthcare Interoperability Resources (FHIR)** - This standard describes data formats and APIs for exchanging electronic health records (EHR), providing a common set of digital objects and API paths for accessing digital healthcare records using modern API infrastructure.

PSD2/PSD3 and FHIR are just the first two industry standards that speak directly to how we design and deliver our APIs. The fact that they emerged from the top two industries is a sign that other sectors are likely to follow their lead.

# 11.7 Organizing Your Standards

## What to standardize

Every organization should have a set of standards for API operations. There are plenty of redundant aspects of doing APIs that are easy to make consistent–things like vocabulary, pagination, sorting, filtering, usage of request bodies, HTTP methods, and error handling. A little standardization in these areas can go a long way towards stabilizing the development and delivery of r your APIs, making your consumers' lives easier along the way.

**Vocabulary** - Establish a common vocabulary for naming things, then sharei that vocabulary across teams to standardize and optimize the way you talk about APIs. Getting teams to use the same language throughout their work will help reduce the friction associated with forward motion.

**Headers** - Headers are key-value pairs of data that can be passed back and forth as part of API requests. They conform to the HTTP standard and rely on the IANA registry of headers to define the routing and prioritization of requests made to APIs when using HTTP as the transport protocol between client and server.

**Pagination** - Pagination provides a standardized way of navigating large sets of data and content via an API. It limits the results returned with each request, but provides consumers with visibility for navigating i results and shaping their API requests to achieve optimum outcomes for API producers and consumers.

**Schema** - Establish a common schema for each domain, using internet and industry standards when possible, but then standardize your own schema, stabilizing common objects and versioning and evolving them for reuse.Then reference them in contracts and use them for validation, documentation, testing, and other parts of the API life cycle to stabilize how data moves inside and outside the enterprise.

**Variables** - Define a consistent set of variables that can be used to abstract away common properties for use across different APIs. You can define things like base URL, headers, secrets, and environmental, collection, or global values that are needed across many different APIs.

Internet, industry, and organizational standards provide the base for the consistency and interoperability across enterprise operations. These standards make APIs more intuitive and help them speak a common language, reducing friction for consumers who put them to work in applications and integrations.

This is by no means meant to be a complete list of things you should be standardizing in your organization. This blueprint is just designed to remind you that you need to be standardizing as part of your regular operations. The items listed here reflect the most common areas I see among API-aware and API-first organizations that are finding success. By employing some simple standardization measures, they are accelerating the velocity of their API-first transformation.

## Providing standardized components and templates

Common, reusable, and standardized templates enable teams to move faster while delivering more consistent and reusable APIs. That reduces time, money, and friction downstream. Reusable components often begin with design patterns that can be applied during the define and design stages of the API life cycle, but can expand rapidly to include almost every other stop along the lifecycle.

**Simple** - Provide simple templates that reduce the cognitive load for learning new standards and patterns to produce APIs.

**Modular** - Keep templates modular and reusable, daisy-chaining concepts together into large patterns, workflows, and processes for moving APIs forward.

**Reusable** - Design your standardized components to be reusable, allowing them to do one thing and do it well, then be applied in many known and unknown uses.

**Starter** - Offer entire starter kits to teams when starting a new API, providing a complete example of the preferred way for designing, developing, and operating APIs.

**Contracts** - Maintain a catalog of complete contracts showing implementations of different types. This allows for easy editing and sets new APIs into motion.

**Components** - Leverage the components object for OpenAPI and AsyncAPI contracts, providing a rich set of templates that can be used to rapidly design new API contracts.

**Extensions** - Use extensions for OpenAPI and AsyncAPI, going beyond what each specification can do and providing templates teams can use across the API life cycle.

**Rules** - Set template rules for linting different artifacts, helping jumpstart the use of common rules and the development of new ones to help govern operations.

**Policies** - Provide standardized policies and starter templates to apply across the API life cycle, helping to centralize policy management while federating usage.

Templates help enable teams to do the right things across the API lifecycle, helping provide the common parts and pieces of delivering APIs. They help teams focus on the unique experiences your enterprise provides to consumers via APIs.

Think of API operations as a bucket of Lego bricks, where the reusable components are the red, blue, and yellow bricks and your API strategy uses these bricks in a variety of strategic kits or templates to deliver the business outcomes leaders are looking for. All teams have to do is build templates, allowing other teams to assemble new and interesting experiences using raw bricks.

EXPERT PERSPECTIVE

## How standards reduce friction for industries


One thing you learn when speaking with API veterans like Alexei Akimov, former Head of API at Adyen, is the impact standards can have across your organization, your industry, and the entire world. Alexei knows firsthand how internet and industry API standards reduce costs, help companies deal with regulation, and serve as the lubricant across your API operations. He explained how API standardization has changed the payments industry and has enabled global commerce across every country and region working with Adyen.

Alexei worked on the front lines of the payment API at Adyen, supporting the community through the company's public portal by publishing documentation and other resources. Alexei saw the friction API consumers encountered when

putting their payment APIs to work in applications. He witnessed how standards like OpenAPI helped get everyone in his team on the same page. He also saw how industry standards like PSD2 got everyone on the same page across an entire geographic region. If you want to learn what is working with standards and what is not, the payments industry is a good place to start.

PSD2, which standardizes how payments work across European countries, shaped how Alexei did APIs at Adyen. We discussed how PSD2 is influencing payment API standards outside of the European Union, in South America, Southeast Asia, and other regions that are aiming to standardize payments across providers. These regions aren't required to use PSD2, but standardization opens up opportunities for more frictionless interoperability between countries. The organic expansion of industry API standards around the globe is a cue for other industries to tune in and learn about how API standards can shift the landscape.

Alexei coherently described the impact of standards like OpenAPI, PSD2, JSON Schema, and others. But he also talked about the role of open source tools, which are exponentially amplified when supporting standardized specifications like OpenAPI, PSD2, and JSON Schema. Open source API tooling that leverages open source standards is what makes the API economy work. It is how we reduce costs, move quicker, and compete on the things that matter. Standards help us reduce complexity and increase our ability to manage not just hundreds, but thousands of APIs at scale.

I consider the financial space to be on the front line of determining how API standards will be defined across almost every other industry. The only other sector even close is healthcare. Other industry and trade organizations would be wise to study what is happening in finance and get to work on their own industry standards–then do the hard work of iterating and making them work at scale.

## 11.8 Applying Rules to API Operations

Machine-readable rules in YAML or JSON can be used to lint or validate any other YAML or JSON artifact. Rules allow for common or specialized procedures to be established to help check for consistency in the design, development, deployment, and management of APIs. Rules help codify standards and health practices for delivering APIs into the API life cycle across domains and teams.

**Name** - "Name" here means the name of the linting rule, describing what it applies to and how it makes the design of an API or the operation around it more consistent.

**Description** - A verbose description states what a rule does, providing as much detail as possible about how the rule standardizes one small part of operations, helping to stabilize teams.

**Given** - A given is the property being targeted for linting, identifying a specific aspect of a contract that needs attention and focusing the attention of linters on this particular part of each API.

**Then** - "Then" is the criteria for evaluating the contract being limited. It provides the logic for the rule being applied.

**Formats** - Rules are designed to be applied to different types of contract formats, including Swagger, OpenAPI, AsyncAPI, and others.

**Documentation URL** - An external URL can be used for documentation and educational resources associated with a specific rule, turning every rule into a potentially teachable moment.

**Rulesets** - Organizing multiple rules into sets enables you to apply them all at once, organized by domain or other bounded context. Rulesets make rules easier to apply.


**JSONPath** - JSONPath is the specification used as part of the "given" and "then" properties to target specific sections of a JSON contract, providing any level of scope when linting contracts.

Rules help us distill all the things we need for usability, consistency, and stability into modular procedures that can be applied as sets or individually during design–and more importantly, throughout the API life cycle. In some cases we can gate rules at the pipeline level, ensuring we are always producing the best APIs we can, no matter what team produces them.

The leading approach for defining and executing rules is Spectral, an open source solution developed by an API service provider named Stoplight. It is common today to define Spectral rules to apply as part of the API design process, linting or validating the OpenAPI contract for an API. However, Spectral can be applied to any JSON object, making it easy to apply rules to any part of the API life cycle.

# 11.9 Embracing Federation

Successful API transformation often takes advantage of federated approaches to governance. Federation acknowledges that the enterprise will always be defined by many smaller groups, domains, or states of business operation. These groups will need a certain degree of autonomy, but they can also leverage centralized structures, standards, and resources.

**Principles** - Principles strengthen the realization that API operations are perpetually realized and executed in a federated way across domains and teams, responding to the changing needs of the enterprise and the markets it serves.

**Tooling** - Tooling means perpetually defining the tools used in different domains and teams. You should allow teams to explore new tools, but have a formal process for weaving them into the overall API platform and stabilizing their use.

**Lifecycle** - Map out the life cycle employed across teams, identifying the common areas, but also the unique variations that should either remain specialized, or be shared across teams and centralized as part of governance.

**Bounded Context** - Define the existing organic lines that exist between teams and groups, then get to work understanding how these lines can be reshaped over time to optimize the vocabulary, standards, and life cycle teams use.

**Policies** - Craft policies centrally, then disseminate them across federated teams, including teams in the process of versioning and establishing a feedback loop between centralized and federated policies.

**Automation** - Equip teams with the artifacts and tools they need to automate governance, so that it can be centrally defined and consistent. Then enable teams to do the right thing with governance by automating redundant aspects.

**Observability** - Provide the observability necessary to "see" what is happening within and across domains, groups, and teams, helping ensure that APIs and the operations around them are as observable as possible, no matter who is behind them.

**Champions** - For federation to be successful, it requires the hard work of champions embedded within teams, groups, and domains–people who are willing to do the hard work of sharing knowledge, policies, and practices centrally and across enterprise teams.

Like centralization, there are upsides and downsides to federated governance. You will need to constantly evaluate and recalibrate what is done in a federated environment or centralized across teams. Federation provides the opportunity for us to embrace the sprawling organic nature of our operations and bring order to the way things are done at scale across teams.

# 12

# Reviewing Your Design and Governance

Reviewing the design and security of your APIs before they go into production ensures not only that they'll meet your standards, but that they'll be safer and easier for consumers to use. So how do you do that? This chapter offers tips and suggestions that will save you a lot of trouble down the road.

## 12.1 Design and Quality Reviews

### Applying design reviews as part of the life cycle

API design reviews make you give pause before any API goes into production, making sure every API meets your organization's design standards. Designers and developers should have a wealth of resources to help them design the best possible API before it ever gets submitted. Each API submitted as part of an API design review process should possess the necessary artifacts and elements needed to properly evaluate its design.

**Workspace** - Groundi API design reviews with a dedicated API workspace where you can find everything you need to review the state of the proposed API design from teams.

**Contracts** - There is always a contract available for the API, allowing self-service automated review, but also to make sure the teams and reviewer are on the same page.

**Documentation** - Provide documentation for the entire surface area of an API, allowing human reviewers to understand the API, and teams to articulate the value.

**Mock Server** - Publish a mocked representation of APIs under review, allowing everyone involved to understand what an API does by playing with the mock server.

## Process

Once you're ready and have all the needed artifacts and elements, your API should be subjected to a well-defined process for reviewing its design and providing feedback, including whether it is ready for production.

**Self-Service** - Provide as much of the design review process and feedback as possible in a self-service format, with modular services, tooling, and rules for teams to use.

**Timeline** - Establish a timeline for how long design reviews should take, holding all stakeholders accountable achieving the desired outcomes within a specified timeframe.

**Feedback** - Provide formal, documented, and constructive feedback for teams about the design of their API, but also allow teams to provide feedback on the process.

**Outcomes** - Defining the outcomes of a design review, regularly assessing the process and the reality on the ground with teams to make sure reviews provide value.

Design reviews should not be seen as yet another gate developers need to get through. They provide an opportunity for educating teams about best practices. They also establish a feedback loop with teams to help you learn where they need help.

## Injecting quality reviews across APIs in development

A quality review, either manual or automated, can assess the overall quality of each API implementation, looking at how the API is being deployed and sustained. Quality reviews can be done at a certain milestone in API development, or exist as a checklist that teams can follow as they define, design, and deploy each API.

**Workspace** - A quality review of a workspace can check that there is a logical name and description of the work occurring, while also considering visibility, RBAC, and other common concerns.

**Repository** - Linking Git repositories to API workspaces helps accommodate different sources of truth and development workflows, while helping keep API artifacts, services, and tools available for use across the API life cycle.

**Documentation** - Validating that documentation is available for each API helps reduce friction with consumers, ensuring they have adequate information about naming, descriptions, errors, authentication, and other aspects that could cause friction.


**Examples** - A quality review can check to see if there are examples available for each individual API path, channel, or other dimension, providing simple examples of what consumers can expect when integrating APIs into their applications.

**Mock Server** - Having a mock server makes it easy for onboarding and testing APIs, reducing friction for consumers who may want to play around with them as they consider adoption.

**Contract Testing** – Require every API to possess contract testing for 100% of the surface area. That will provide you a nice baseline of quality, ensuring that every API respects the contract that was agreed upon by API producers and consumers.

**Performance Testing** - Check to see if performance tests are included in quality reviews. They will help you further establish a baseline for quality.

**Monitoring** - A quality review can check API monitoring, for example, requiring that every API be monitored on a regular schedule. Results can be published to existing APM solutions or viewed via your API platform reporting.

Quality reviews can simply be a checklist to remind API developers and other stakeholders of what each API needs. They can also provide an opportunity for teams to review each other's APIs, or leverage a centralized group to review every API as it moves to production. Quality reviews ensure that APIs are operated consistently across teams. They also give you an opportunity to engage with teams about how they deliver and iterate upon APIs and discover what they need to improve their processes.

# 12.2 Security Fundamentals and Reviews

## Security fundamentals

A security review provides an opportunity to pause and ensure that teams are thinking about safety early in the process. Cybersecurity is too important to be simply a concern of the gate team fear before an API moves into production.

There are many security considerations teams should think about in the defining and designing phases to provide development teams, and eventually users, with more secure APIs.

**Encryption** - Make encryption the default for all APIs, covering the transport layer, but also storage and databases behind APIs.You should have a solid encryption plan from the start.

**Authentication** - Use common standards for authenticating API consumers using any API, reducing the complexity for them at this layer.

**Authorization** - Consider an added authorization layer that defines whichAPI-driven resources and capabilities each consumer will be able to access once they start using your APIs.

**Role-Based Access Control** - Apply RBAC to all of the elements of API operations, defining who can edit or read artifacts, documentation, testing, and other elements.

**Contracts** - Each API possesses a complete contract, including full details of the authentication and authorization procedures. The contract acts as a menu of security features for each API.

**Environments** - Evaluate the development, staging, sandbox, and production environments teams will use and determine their security strategy.

**Documentation** - Include security fundamentals as part of the documentation for each API, making sure consumers are always fully aware of the controls in use.

**Tests** - Provide collection security tests–modular, reusable, executable, and fully documented tests for all of the most common vulnerabilities your teams will face.

There is plenty more your security team will be considering when it comes to API security, but these fundamentals should be the baseline for your operations. Without these elements, it becomes very difficult to properly secure your APIs at scale. These are the building blocks that enable teams to deliver more secure APIs. Without having to become security experts themselves, they can enjoy the support of centralized security resources.

## Checking for the OWASP Top 10

When you're injecting a required security review of an API, consider the OWASP Top 10 as the starting point, running the following checks against every API before certifying it ready for production:

**Broken Object Level Authorization** - APIs tend to expose endpoints that handle object identifiers, creating a wide attack surface level access control issue via specific objects.

**Broken User Authentication** - Authentication mechanisms are often implemented incorrectly, allowing attackers to compromise authentication tokens to gain access.

**Excessive Data Exposure -** Developers tend to expose all object properties without considering their individual sensitivity, relying on clients to perform the data filtering.

**Lack of Resources & Rate Limiting** - Too many developers fail to impose any restrictions on the size or number of resources that can be requested by the consumer in any given time period.

**Broken Function Level Authorization** - Access control policies may be too complex with different hierarchies, groups, roles, and an unclear separation of resources.

**Mass Assignment** - If you bind client-provided data to data models without proper properties filtering, it could allow attackers to traverse and explore by guessing your structure.

**Security Misconfiguration** - Security misconfiguration is commonly a result of insecure default configurations, or incomplete or ad-hoc configurations by teams.

**Injection** - Injection flaws, such as SQL, NoSQL, and command injection, occur when untrusted data is sent to an interpreter as part of a command or query with a request.

**Improper Asset Management -** APIs tend to expose more endpoints than traditional web applications, making proper and updated documentation extremely important.

**Insufficient Logging & Monitoring** - Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, opens doors for attackers.

The OWASP Top 10 API vulnerabilities provide us with a baseline that should exist across 100% of the APIs in production, whether they are for internal, partner, or public consumers. There will be other security concerns as well, but if you make this list the default across your operations, you will significantly improve the reliability of your APIs.

This stage of the life cycle tends to get the most attention in technical conversations about delivering APIs–doing the work to bring each API to life. A well-defined API life cycle makes the development stage as efficient as possible, with developers doing what they do best.

## Test

This area of the API lifecycle is focused on streamlining, documenting, and automating the testing of the underlying contract for each API. It is about understanding the overall performance of the API and its integration, and running other types of tests. Testing should include automation to properly scrutinize the entire surface area of each API, as well as the operations surrounding it, helping teams to do more work and consistently contribute to the quality of all APIs.

**Collections** - Use collections to define one or a sequence of many API requests, establishing a modular, collaborative, and executable artifact across API testing.

**Scripts** - Define folder-level, pre-request, or post-request scripts that will run as you configure requests, validate responses. Automate testing to run as you iterate and respond to consumer behavior.

**Contract** - Use OpenAPI and AsyncAPI contracts to ensure that 100% of the surface area of an API is tested and behavior reflects the contract between producer and consumer.

**Performance** - Test Specific paths for each API in multiple regions to make sure if the API, gateway, and network provide desired performance.

**Mocking** - Use mock servers generated from the API contract, then augment them with examples for specific use cases, testing for specific outcomes.

**Data** - Inject CSV or JSON data as part of the testing process, making sure API requests reflect specific business workflows and outcomes.Testreal-world API-driven scenarios.

**Monitor** - Schedule testing monitors to run on a schedule reflecting the business needs of the API, but also the type of test being run, allowing teams to automate testing.

**CI/CD Pipeline** - Bake tests into CI/CD pipelines, ensuring that all tests run when APIs are being built and no API goes into production without tests.

While there are always nuances in testing an API, the majority of testing can be realized with these common elements. A consistent approach to testing across all APIs helps ensure quality and provides a consistent template for what testing is, allowing it to be used across all teams.

Standardizing how teams test APIs is essential for achieving the reliability you will need across the thousands of APIs you will use to do business in the future.

## Secure

This stage is about securing the access and operations surrounding each API, ensuring that only those who should have access are able to make requests and publish messages. Security is about establishing an organization-wide approach API authentication and encryption are applied. It is also about how APIs are fuzzed and scanned for vulnerabilities. You must provide teams with everything they need to secure each API and the operations around it, consistently securing the expanding ßAPI landscape.

**Authentication** - Authentication helps ensure APIs are accessed only by those who should have access, allowing API producers and consumers to easily apply rules consistently.

**Authorization** - Once a user is authenticated, the authorization layer will make sure they only have access to approved resources.

**RBAC** - Role based access controls should be applied at the authorization layer of an APIand to the API operations around it, helping govern who has access to operations.

**Encryption** - Ensure that all API requests are encrypted, and maket reading encrypted messages is as easy as possible.

**Environments** - Have a solid map of the development, staging, and production environments across all APIs in operation. That will help you manage API deployment more consistently.


**Variables** - Provide a well-defined vocabulary of variables that abstract away the common aspects of authentication and authorization, helping standardize the way we engage with APIs.

**Secrets** - Add a layer on top of environmental variables specifically for managing secrets, making sure you have clear visibility and control of secrets and tokens being applied.

**OWASP Top 10** - The OWASP Top 10 is a standard for API producers, covering a broad consensus about the most critical security risks for web APIs and providing a consistent checklist for teams to follow.

There are many layers of security for producing and consuming APIs.Organizations are increasingly making security a priority earlier in the API life cycle, instead of after an API goes live. This evolution is often described as shifting left, or investing in security earlier in the life cycle and equipping API teams with proven approaches to delivering more secure APIs.

API security doesn't stop at authentication and authorization of each API. It should include data in transport, the operations around each API, and security concerns for both producers and consumers. A known API life cycle, combined with well-defined security practices, gets teams up to speed with what matters most to secure APIs.

## Deploy

Once an API is ready for deployment to a staging or production environment, there should be a repeatable set of elements at work to move enterprise operations forward at scale. The deployment orchestration of APIs across teams helps optimize the API factory floor across enterprise domains, making every step forward deliberate and repeatable.

**Source Control -** Use source control to manage code and artifacts used to deploy an API, providing a single location where you can find everything behind each version of an API, ideally with multiple branches to accommodate many API contributions.

**CI/CD Pipeline** - The pipeline ensures that the deployment of an API to each stage is as repeatable as possible, with tests and other essential needs of the API build process.

**Gateway** - Publish contracts, extensions, and other configurations to the API gateway, deploying an API into a staging, then a production environment if all tests pass in the pipeline.

**Policies** - Require all APIs to be deployed according to a standardized set of plans, with consistent policies limiting access to resources, applying proper security, and ensuring a consistent deployment across all teams.

**Releases** - Establish a formal release for each version of an API, documenting the changes being deployed and the communication about them, keeping consumers informed by rolling up all the branches and changes into a single release.

**Stages** - Allowing multiple stages to be deployed, providing development, staging, production, and potentially other environments for deploying and testing. That will enable APIs to be reliably deployed into production with the highest possible quality.

**Environments** - Apply commonly managed environments with a coordinated variable strategy for testing and automating configuration as part of the pipeline, helping abstract away the technical details and secrets of API environments.

**Observability** - Once an API has deployed, what observability do you have over the build process? What about observability of the API once it is available? You need to ensure that you see what is happening during deployment, just as you do for e all other stages of the API life cycle.

Deployment means different things to different organizations. What is important is that there is always a source of truth; a repeatable build process; and a standard set of releases, stages, environments, and plans to deploy APIs consistently across teams and domains.

Like the software development life cycle (SDLC), the API life cycle should be well-defined and repeatable. The big difference is, the API lifecycle should also possess a handful of nuances that help reliably deliver API infrastructure using API gateways, policies, and observability practices.

## Observe

Observability provides teams with the ability to “see” APIs and the operations around them using a set of common metrics, helping provide the data they need to move each API forward independently of others.

**Activity** - You must understand the activity across an API platform to understand how APIs are being moved forward, configured, and evolved over time by tapping infrastructure outputs, ensuring there is provenance for change throughout the API lifecycle.


**Logs** - Actively use  logs for source control, CI/CD, and the gateway. They provide the outputs you need to understand the velocity of individual APIs, as well as velocity across domains.Tap into existing logging outputs to observe the state of each API.

**Traces** - Leverage traces added to clients, SDKs, gateways, and other functions. They will help you make sense of the API landscape and how APIs are putting backend infrastructure to work, using unique identifiers that are passed to APIs and tracked all the way to backend systems.

**Monitors** - Establish monitors for all contract, performance, security, and governance tests. These tests provide the results you need to understand the state of APIs and all API operations. Use collections to define the outputs you need to understand the state of your APIs.

**APM** - Route all outputs from API operations into your existing APM solutions, tapping every output across the API life cycle to understand the health and state of the platform through the infrastructure you have already invested in.

**Dashboards** - Set up dashboards within the APM to understand the health of individual APIs and the life cycle around them, providing a visual way for anyone to observe the state of any API and how it is being operated.

**Reports** - Providing team, API, documentation, testing, and other reporting, showing what teams are doing across API operations, and how the lifecycle is unfolding across teams, using native platform reporting that speaks specifically to API operations.

**Notifications** - Use notifications and alerts to observe changes with each API, as well as events that occur across the life cycle. They will help you understand consumer activity, as well as what is happening across teams and other stakeholders.

Observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs. You need to tap into all of the outputs available across the API platform and the infrastructure used to move APIs forward across the API life cycle into production.

Because our infrastructure has APIs, we can use collections to define each aspect of API operations as a machine-readable, executable, and granular output that routes data into our APM solutions. That helps make 100% observability an achievable goal for your teams.

## Distribute

An API does little good if it can't be found and put to use. Distributing an API, as well as the supporting operations around it, ensures that consumers can find it when building applications and integrations. It also means, teams will be less likely to create redundant APIs.

**Portal** - API deployed into production should be published to the central portal, providing centralized access to internal or external consumers through a single doorway that can be supported as part of overall API operations.

**Catalog** - The metadata for each API should be updated and kept in sync, ensuring that all relevant information consumers will need is available in the catalog and keeping the catalog up to date.

**Network** - API should be published to the private, partner, and public networks where API consumers are via the platforms they are already using. Tap into existing API consumers so your developers can meet them where they are.

**Workspace** - The workspaces around an API should also be made available via the network, providing teams and consumers access to API operations.. Expose, not just the APIs, but also the work, artifacts, and resources behind them.

**Visibility** - The visibility of each APIic should be deployed deliberately and confidently, making t sure each API is available to consumers, and striking the balance between access and control to meet desired business outcomes.

**Content** - Publish blog posts, videos, and other content to support the distribution of APIs, articulating the value an API delivers in a way that is discoverable by consumers. Remember to include rich keywords and other necessary metadata.

**Buttons** - The documentation, tests, and even the workspace behind an API should be made available via blog posts, videos, wikis, and other resources available to support APIs. Use embeddable buttons that consumers can activate with one click.

**Search** - APIs and the operations around them should be indexed and made available via search. That gives teams, business stakeholders, and consumers the ability to search for and find what they need before building new APIs, applications, or integrations.

The distribution of APIs plays an essential part in both production and consumption. Developers' lack of attention to distribution impacts discovery and feedback loops for APIs, limiting usage and depriving them of valuable information they could use to iterate and provide consumers with features they want.

Developers can distribute APIs, but it is best done by product managers, marketers, and others who are more attuned to business needs. They will be sure to select the right metadata and ensure that r portals, catalogs, networks, and content are closely aligned with business goals.

API consumers need all the help they can get with observability so that they can regularly see and understand what is happening with their integrations. APIs are abstract, and it can be difficult to see individual and community use. That's where you can step in and help.

While API consumers will care a lot more about their own API consumption, the usage and feedback by others within the community will also play a role in how they view a platform, and will influence decisions they make moving forward.

## 12.3 What Good Governance Looks Like

An API-first enterprise understands that good operations are defined by enabling teams to do the right thing on the ground floor. Good governance establishes a clear list of tools developers need for producing and consuming APIs, and makes sure they have access to them. Governance should augment the work of various teams involved in bringing APIs to life and putting them to work in the applications and integrations that power today's businesses.

**Platform** - Set up your API platform, beginning with your source control, CI/CD, gateways, and APM. Then evolve it towards other stops along the API life cycle, taking full advantage of the APIs behind your infrastructure to stabilize your developers.

**Integrations** - Lean on industrial-grade integrations to bring together the services you need to make producing and consuming APIs as seamless as possible for all stakeholders. Realize that APIs aren't just for applications–they are also needed behind APIs.

**Tooling** - Empower developers with the tools they need, providing bedrock tooling like source control, CI/CD, and IDE. Encourage exploration of new commercial and open source tooling, providing tracks to formalize team adoption.

**Collaboration** - Modern API development teams aren't locked away anymore. Producing and consuming APIs is a team effort across business and technical groups, providing a huge opportunity to improve productivity through more collaboration.

**Automation** - Help teams automate at every turn, documenting and then automating common processes that do not need human execution and intervention. Equip teams to bake workflows into the CI/CD pipeline as part of their work.

**Observability** - Make observability available to teams, allowing them to understand the health of their APIs and how they compare to other teams' work. Encourage teams to learn from other teams and allow them to pick up new skills.

Remember, the success of your governance is a direct reflection of the enablement you provide to teams. Nobody on the ground floor cares about governance, but they do want to be supported in doing the right thing. That support is much easier to provide with an API-first platform approach. When teams are fully equipped with the right training and tools to do the right thing in the moment, you will begin to see productivity and quality shift into a higher gear.

Teams across API operations will not always be aware of your strategy or care about the long-term impact of API governance. What teams want is to be enabled to solve the problems they have been tasked with. With good governance in place, you can make their workdays easier and more productive.


# Operations

# 13
# Alignment and Life Cycles

API operations often sprawl across the enterprise, led by different teams with conflicting ideas about how to manage them. As a result, many organizations fail to understand their API consumption, and their API products may be marred by operational and security problems.

By understanding the API producer and consumer life cycles and organizing your API operations into well-defined phases, you will reduce friction and complexity and gain a comprehensive view of APIs throughout your organization—while still giving your teams the freedom they need to innovate. This chapter shows you how to do it.

## 13.1 Aligning Your Organization for APIs

Dividing your organization into intentional operational areas is one of the first steps of your API-first transformation. It will shape how your digital resources and capabilities are made available and provide a grounding force for your API life cycles and governance. Organizing your API operations into deliberate bounded contexts reduces complexity and eliminates the unknowns across your digital enterprise landscape.

**Domains** - Translate the traditional lines of business or tribal boundaries that have emerged over the years into more deliberate spheres of activity and knowledge. That will help you establish bounded contexts for operations, reducing the cognitive load while delivering meaningful API products.

**Groups** - Establish designated groups within the enterprise that reflect specific business domains or provide cross-cutting contributions to the API life cycle. That will ensure that API operations–and their underlying infrastructure, workspaces, and teams–are logically grouped to deliver the most effective outcomes.

**Teams** - Establish a profile for every human or automated persona that contributes to the API life cycle. Providing a context for individual stakeholders will accelerate the API lifecycle and help move APIs forward at scale.

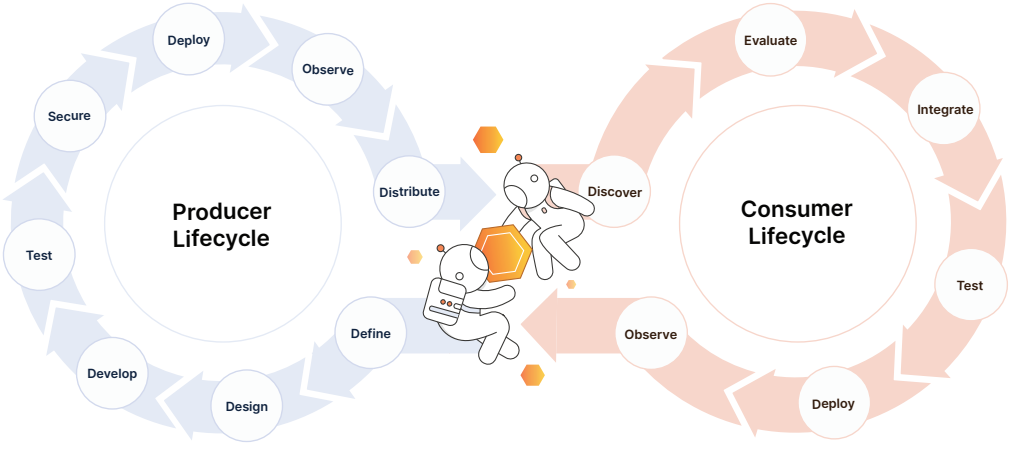
**Roles** - Assign a specific role to each team member, establish access controls to keep teams from stepping on each other's toes, and stabilize the changes made across operations.

**Single-Sign-On** - Provide teams with an authentication scheme allowing them to log in with a single ID across the multiple services they will need to deliver APIs across the API life cycle.

**System for Cross-domain Identity Management (SCIM)** - Adopt a System for Cross-domain Identity Management, or SCIM, a standard for automating the provisioning and de-provisioning of team member accounts across API operations. A SCIM will make it easier to manage the large number of users required to deliver and operate your APIs.

API operations can be sprawling, creating friction and challenges among people and teams. Much of the friction can be alleviated with a little organizational alignment. Team composition changes regularly, with t members constantly coming and going. That's not necessarily bad, but you need structure and automation to ensure people are doing what they should be doing across the API life cycle. Organizational alignment also helps reduce friction over time between development and business teams as your company shifts to meet the needs of your industry.

## API Producer and Consumer Equilibrium




# 13.2 The Producer Life Cycle

Every API begins with an API producer who makes some digital resource or capability available via their desired protocol. In the typically chaotic, ad hoc approach organizations use to deliver APIs for one-off projects and applications, APIs vary in how they are brought to life and sustained to support an application or integration. Every enterprise API strategy centers around identifying the many different life cycles in motion across operations and then working to bring them all into alignment through a common, well-known API lifecycle.

Ask about your organization's API life cycle, and you will likely get many different responses. The narrative we've been told about the API life cycle for the past decade has been heavily skewed toward the API producer. Vendors selling their solutions, supported by industry analysts, were very interested in imparting their vision of the API life cycle to the industries impacted by APIs. This narrative would be sufficient if we used a single vendor across the entire API life cycle, but the reality is that teams are using many different vendors.

There is no single definition of the life cycle for delivering APIs. It is up to your organization to settle on a common understanding of each step and the business and technical details required. Just as you do with design and governance, you need to work towards achieving consistency in your API life cycle. This is not an easy thing to do. Begin by agreeing on a simple outline of each stop along your API life cycle. Then invest in fleshing out the details, and consider adopting rules for as many parts of the process as possible.

The lifecycle blueprints in this book are meant to show you the most common stops along the API life cycle and provide some details to consider as you assemble your own API life cycle definition. These blueprint elements are just a handful of the technical details you can use to map out the processes across your teams and domains. They are meant to provide you with suggestions for how you can iterate to improve your processes based on what is happening. The objective is to establish a common way for teams to talk about what the API lifecycle is, and begin to establish alignment in the way APIs are delivered.

Having an API life cycle definition for your operations offers a grounding force, not just for how you deliver APIs, but how you talk about them. The grounding of the API life cycle is where to begin if you want to improve productivity across your teams, increase the quality of your APIs, and make API governance possible.

## Define

APIs are a very abstract digital concept, loosely wrapping a variety of text, documentation, artifacts, and code that define what an API is capable of doing. A thoughtful API life cycle begins by sitting down with all stakeholders and finding a common way of defining each API and how it will move forward over time.

**Requirements** - What are the requirements for the API? You need to define the business value it will bring to help you guide development and operation. Stakeholders - Identify business and technical stakeholders, including any external partners and consumers who might need to be involved.

**Domain** - What domain will an API be operating in?, Define the vocabulary, standards, and other patterns that developers at design and development teams will use.

**Regions** - Identify the region(s) where an API will operate so that you can comply with regulations and other business requirements and ensure that APIs are as close to consumers as possible.

**Teams** - Line up who will be working on an API, bringing together designers, developers, technical writers, QA specialists, and other roles who will be involved in moving your APIs forward.

**Roles** - Define who will have access to what in terms of editing, viewing, and working with APIs, and the operations that move them forward.

**Workspaces** - Set up the workspaces where teams will be designing, developing, and managing APIs, then iterating upon them and managing multiple versions.

**Change** - Establish the underlying approach for managing change with an API, keeping the versioning, communication, and other elements in alignment with centralized governance.

**Road Map** - Create a road map for each API, planning for the future from Day One.

A well-executed definition of the API life cycle lays the important groundwork that will contribute to its usability. You are providing the API with the nutrients it needs to operate. The definition stage may seem like an easy one to skip, but in reality, it is the most important place to begin, laying the foundation for almost every other stop along the way and contributing to velocity.

## Design

In the design phase, you invest in the informed design of the surface area of each API, shaping how it works, the protocols and standards it uses, and how each API will conform to wider rules across the organization. Investment in the design of each API, as well as the overall practice of design across operations, fine-tunes your operations and brings operations into alignment.

**Patterns** - Select which patterns you will use–such as REST, GraphQL, and other common varieties. Patterns will standardize how your APIs work and help designers and developers choose the right tools for the job.

**Protocol** - Make a sensible decision about which protocol to use designing an API, picking the right solution for the project and for the consumers who will be using the product.

**Standards** - Understand the internet, industry, and organizational standards you will use when designing the API, making sure that each API is as consistent as possible.

**Schema** - Establish a schema for all of the objects used as part of requests, responses, publishing, and subscribing when integrating an API into applications.

**Contracts** - Put contracts like OpenAPI and AsyncAPI to work in defining the surface area of each API, providing a machine-readable contract to guide teams' work.

**Versioning** - Change is inevitable. You need a clear plan for how an API will be versioned, leveraging a common pattern for managing change.

**Rules** - Identify which linting rules you will need to help ensure the API is following central governance and remaining as consistent as possible across teams.

**Editor** - Establish a common, consistent way of directly or visually editing the artifacts used in the API design process, helping clarify the carry load of the design.


**Examples** - Try to provide examples of each part of an API, making actual examples part of each contract that can be used for documentation and mocking.

**Mocks** - Generate a mock representation of an API using its contract, providing an example of what the API will do in production. That will help with the design process and later on, with onboarding.

Design is often discussed in terms of RESTful or web APIs, but this is a universal look at design that can be applied to both synchronous and asynchronous APIs. Design is applied across a variety of API patterns, so you should define a common way to eventually describe and discuss design across all APIs.

## Develop

This stage is about bringing an API to life. That means assembling all the infrastructure needed and generating any skeletons, using frameworks, assembling, and hardening an APIs functionality. The Development might take a design-led approach or a code-led approach. Either way, as long as it follows a common life cycle, it can benefit operations.

**Compute** - Establish a baseline for the underlying API compute, choosing among virtual servers, containers, and serverless to power each API.

**Database** - Provide the data storage and querying requirements for an API, leveraging a centralized database or establishing a database for use by this single API resource.

**Storage** - Define the centralized storage for an API, planning where objects, images, and other files will be stored, retrieved, and managed as part of API usage.


**DNS** - Apply a consistent approach to using DNS for each API, following a larger domain strategy but allowing for flexibility and redundancy in accessing each API.

**Encryption** - Ensure that encryption is the default in transport and storage, and make sure each individual API applies these security controls from the beginning, not as an afterthought.

**Frameworks** - Generate frameworks as the scaffolding for each API, leveraging consistent approaches, avoiding redundant work and using the API contract to produce as much code as possible.

**Gateways** - Set up the gateway to prepare for an API deployment. Do the initial work to set up everything needed at the gateway layer, shifting as much of this work as far to the left in the life cycle as possible.

**Annotations** - Use code annotations to auto-generate the contracts you need to document, test, secure, and govern the API, leaning on a code-led approach to APIs.

**Integrated Development Environment (IDE)** - Maximize productivity across teams by providing further enablement via their trusted IDE, helping to increase developer productivity.

**Source Control** - Establish a Git repository for managing all the code and artifacts for an API. That will help you establish source control for each API early in the API life cycle.

This stage of the life cycle tends to get the most attention in technical conversations about delivering APIs–doing the work to bring each API to life. A well-defined API life cycle makes the development stage as efficient as possible, with developers doing what they do best.

## Test

This area of the API life cycle is focused on streamlining, documenting, and automating the testing of the underlying contract for each API. It is about understanding the overall performance of the API and its integration and running other types of tests. Testing should include automation to properly scrutinize the entire surface area of each API, as well as the operations surrounding it, helping teams do more work and consistently contribute to the quality of all APIs.

**Collections** - Use collections to define one or many API requests, establishing a modular, collaborative, and executable artifact used across API testing.

**Scripts** - Define folder-level, pre-request, or post-request scripts that will run as you configure requests, validate responses, and automate testing to run as you iterate and respond to consumer behavior.

**Contract** - Use OpenAPI and AsyncAPI contracts to ensure that 100% of the surface area of an API is tested and behavior reflects the contract between producer and consumer.

**Performance** - Test specific paths for each API in multiple regions to make sure the API, gateway, and network provide the desired performance.

**Mocking** - Use mock servers generated from the API contract, then augment them with examples for specific use cases, testing for specific outcomes.

**Data** - Inject CSV or JSON data as part of the testing process, making sue API requests reflect specific business workflows and outcomes. Test real-world API-driven scenarios.

**Monitor** - Schedule testing monitors to run on a schedule reflecting the business needs of the API, but also the type of test being run, allowing teams to automate testing.

**CI/CD Pipeline** - Bake tests into CI/CD pipelines, ensuring that all tests run when APIs are being built and no API goes into production without tests.

While there are always specific nuances to testing an API, the majority of testing can be realized with these common elements. A consistent approach to testing across all APIs helps ensure quality and provides a consistent template for what testing is, allowing it to be governed across all teams. Standardizing how teams test APIs is essential to achieving the reliability you will need across the thousands of APIs you will use to do business in the future.

## Secure

This stage is about securing the access and operations surrounding each API, ensuring only those who should have access are able to make requests and publish messages. Security is about establishing an organization-wide approach to how API authentication works and how encryption is applied, as well as secrets, roles, and the way APIs are fuzzed and scanned for vulnerabilities. You must provide teams with everything they need to secure each API and the operations around it, consistently securing the expanding API landscape.

**Authentication** - Authentication helps ensure APIs are accessed only by those who should have access, allowing API producers and consumers to easily apply rules consistently.

**Authorization** - Once a user is authenticated, the authorization layer will make sure they only have access to approved resources.

**RBAC** - Role-based access controls should be applied at the authorization layer of an API and to the API operations around it, helping govern who has access to operations.

**Encryption** - Ensure that all API requests are encrypted, and make reading encrypted messages as easy as possible.

**Environments** - Have a solid map of the development, staging, and production environments across all APIs in operation. That will help you manage API deployment more consistently.


**Variables** - Provide a well-defined vocabulary of variables that abstract away the common aspects of authentication and authorization, standardizing the way teams engage with APIs.

**Secrets** - Add a layer on top of environmental variables specifically for managing secrets, making sure you have clear visibility and control of secrets and tokens being applied.

**OWASP Top 10** - The OWASP Top 10 is a standard for API producers, covering a broad consensus about the most critical security risks for web APIs and providing a consistent checklist for teams to follow.

There are many layers of security for producing and consuming APIs. Organizations are increasingly making security a priority earlier on in the API life cycle instead of after an API goes live. This evolution is often described as shifting left, or investing in security earlier in the life cycle and equipping API teams with proven approaches to delivering more secure APIs.

API security doesn't stop at the authentication and authorization of each API. It should include data in transport, the operations around each API, and security concerns for both producers and consumers. A known API life cycle, combined with well-defined security practices, gets teams up to speed with what matters most for securing APIs.

## Deploy

Once an API is ready for deployment to a staging or production environment, there should be a repeatable set of elements at work to move enterprise operations forward at scale. The deployment orchestration of APIs across teams helps optimize the API factory floor across enterprise domains, making every step forward deliberate and repeatable.

**Source Control** - Use source control to manage code and artifacts used to deploy an API, providing a single location where you can find everything behind each version, ideally with multiple branches to accommodate many API contributions.

**CI/CD Pipeline** - The pipeline ensures that the deployment of an API to each stage is as repeatable as possible, with tests and other essential needs of the API build process making API deployment as repeatable as possible across teams.

**Gateway** - Publish contracts, extensions, and other configurations to the API gateway, deploying an API into a staging, then a production environment if all tests pass in the pipeline.This gives you a repeatable way of managing gateways.

**Policies** - Require that all APIs be deployed according to a standardized set of plans, with consistent policies, limited access to resources, and proper security controls, ensuring a consistent deployment across all teams.

**Releases** - Establish a formal release for each version of an API, documenting the changes being deployed and the communication about them. Keep consumers informed by rolling up all of the branches and changes into a single release.

**Stages** - Allow multiple stages to be deployed, providing development, staging, production, and potentially other environments for deploying and testing APIs. That will allow APIs to be reliably deployed into production with the highest possible quality.

**Environments** - Apply commonly managed environments, with a coordinated variable strategy for testing and automating configuration as part of the pipeline, helping to abstract away the technical details and secrets of API environments.

**Observability** - Once an API has deployed, what observability do you have over the build process? What about observability of the API once it is available? You need to ensure that you see what is happening during deployment, just as you do for all other stages of the API life cycle.

Deployment means different things to different organizations. What is important is that there is always a source of truth, a repeatable build process, and a standard set of releases, stages, environments, and plans to deploy APIs consistently across teams and domains.

Like the software development life cycle (SDLC), the API life cycle should be well-defined and repeatable. The big difference is, the API lifecycle should also possess a handful of nuances that help reliably deliver API infrastructure using API gateways, policies, and observability practices.

## Observe

Observability provides teams with the ability to "see" APIs and the operations around them using a set of common metrics, helping provide the data needed to operate and move forward each API independently of others.

**Activity** - You must understand the activity across an API platform to understand how APIs are being moved forward, configured, and evolved over time by tapping infrastructure outputs, ensuring there is provenance for changes throughout the API lifecycle.


**Logs** - Actively use logs for source control, CI/CD, and the gateway. They provide the outputs you need to understand the velocity of individual APIs, as well as velocity across domains. Tap into existing logging outputs to observe the state of each API.

**Traces** - Leverage traces added to clients, SDKs, gateways, and other functions. They will help you make sense of the API landscape and how APIs are putting backend infrastructure to work, using unique identifiers that are passed to APIs and tracked all the way to backend systems.

**Monitors** - Establish monitors for all contract, performance, security, and governance tests. These tests provide the results you need to understand the state of APIs and API operations. Use collections to define all of the outputs you need to understand the state of your APIs.

**APM** - Route all outputs across API operations into your existing APM solutions, tapping every output across the API life cycle to understand the health and state of the platform through the infrastructure you have already invested in.

**Dashboards** - Set up dashboards within your APM to understand the health of individual APIs and the life cycle around them, providing a visual way for anyone to observe the state of any API and how it is operated.

**Reports** - Provide team, API, documentation, testing, and other reporting, showing what teams are doing across API operations and how the lifecycle is unfolding across teams. Use native platform reporting that speaks specifically to API operations.

**Notifications** - Use notifications and alerts to observe changes with each API, as well as events that occur across the life cycle. They will help you understand consumer activity, as well as what is happening across teams and other stakeholders.

Observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs. You need to tap into all the existing outputs available across the API platform and the infrastructure used to move APIs forward across the API life cycle into production.

Because our infrastructure has APIs, we can use collections to define each aspect of API operations as a machine-readable, executable, and granular output that routes data into our APM solutions. That helps make 100% observability an achievable goal for your teams.

## Distribute

An API does little good if it can't be found and put to use. Distributing an API, as well as the supporting operations around it, ensures that consumers can find it when building applications and integrations. It also means teams will be less likely to create redundant APIs.

**Portal** - An API deployed into production should be published to the central portal, providing centralized access for internal or external consumers through a single doorway that can be supported as part of overall API operations.

**Catalog** - The metadata for each API should be updated and kept in sync, ensuring that all the information consumers will need is available in the catalog and keeping the catalog up to date.

**Network** - APIs should be published to the private, partner, and public networks where API consumers are, via the platforms they are already using. Tap into existing network effects for API consumers so your developers can meet them where they already are.

**Workspace** - The workspaces around an API should also be made available via the network, providing teams and consumers access to API operations. behind APIs. Expose not just the APIs, but also the work, artifacts, and resources behind them.

**Visibility** - The visibility of each API, whether it is private, partner, or public, should be deployed deliberately and confidently, making sure each API is available to consumers and striking a balance between access and control to meet desired business outcomes.

**Content** - Publish blog posts, videos, and other content to support the distribution of APIs, articulating the value an API delivers in a way that is discoverable by consumers. Remember to include rich keywords and other necessary metadata.

**Buttons** - The documentation, tests, and workspace behind an API should be made available via blog posts, videos, wikis, and other resources to support APIs. Use embeddable and actionable buttons that consumers can activate with one click.

**Search** - APIs and the operations around them should be indexed and made available via search. That gives teams, business stakeholders, and consumers the ability to search for and find what they need before building new APIs, applications, or integrations.

The distribution of APIs plays an essential part in both production and consumption. Developers' lack of attention to distribution impacts discovery and feedback loops for APIs, limiting usage and depriving them of valuable information they could use to iterate and provide consumers with features they want.

Developers can distribute APIs, but it is best done by product managers, marketers, and others who are more attuned to business needs. They will be sure to select the right metadata and ensure that portals, catalogs, networks, and content are closely aligned with business goals.

# 13.3 The Consumer Life Cycle

The value of an API is defined by those who consume it. The direction and velocity an API takes is a shared journey between the API producer and consumer. Business value is not created simply by producing an API. API producers and consumers must have a shared understanding of what is happening, and where things are going. The greatest challenges across the API life cycle and API operations occur when producers and consumers fall out of alignment and friction is introduced into the relationship.

API consumption is about better defining how internal and external consumers put the API you produce to work in their applications and integrations. Good API producers are also API consumers. You cannot talk about producing APIs without talking about API consumption. This dance requires empathy. If you fail to deliver APIs that get used, you likely lack the empathy to deliver what consumers need. The health and viability of your API operations depend on having a properly defined API consumption life cycle, as well as a properly defined API producer life cycle.

The API consumption life cycle begins with knowing who your consumers are. You know who they are. You know where they spend their time. You regularly engage with them to understand their experience and needs. These API consumer conversations are hard to do consistently or at scale if you don't have a common definition of the consumer life cycle. Without an ongoing understanding of your API consumers, your API capabilities and experiences will begin to fall flat, missing what is needed. Without an active feedback loop with the right segments of your API consumers, your API will lack the nutrients it needs to iterate, evolve, and deliver the success you are looking for.

The API consumption life cycle discussion continues with creating a plan for how you consume infrastructure, partner, and third-party APIs. Without a consistent definition of your API consumption life cycle, each API integration you do will be a special snowflake–a custom affair. But if teams have a shared vocabulary for talking about the API life cycle, they can follow a consistent approach for putting external APIs to work across operations in a reliable and repeatable way. Every business application, system, and infrastructure element you put to work in your business should have an API, or you shouldn't be using it.

It is easy to ignore your API consumers when you are an API producer. But it is very painful to be ignored when you are consuming a third-party API. Recognizing and understanding your consumers' pain is the only true way you will develop the empathy you need for your API-first transformation.

## Discover

Discovery means enabling API consumers to find exactly the right API they need for

their application or integration. It also means giving consumers everything they will need to onboard with an API. ng Consumers need to assess the overall quality and reliability of the API, then get to work integrating it into their use case, no matter what language they are using.

**Search** - Consumers should be able to search for APIs on the interfaces they are already using, allowing for the discovery of API information in a manner relevant to their work.

**APIs** - The contracts and other artifacts that define the surface area of an API, including authentication and authorization, should be discoverable as part of regular operations.

**Documentation** - Up-to-date and accurate documentation for all APIs should be easily discoverable by teams, with human-readable details describing what is possible with each API.

**Tests** - Contract, performance, integration, and user acceptance tests should be made searchable by consumers.

**Workspaces** - Alongside Git repositories, you should include private, partner, and public workspaces as part of discovery, indexing the places where all work occurs for each API.

**Teams** - The teams behind APIs, and any partner or public contributors, should be made discoverable alongside documentation and other data, encouraging engagement.

**Workflows** - Common workflows using APIs should be made discoverable, helping business and technical stakeholders implement those they need.

**Changes** - Multiple versions of each API should be made discoverable, indexing each API release and the communications surrounding it to help consumers easily get up to speed.

Every moment consumers spend looking for an API, trying to find the latest version, and understanding what it does restricts the forward motion of a team, domain, and organization. Clearly, consumer discovery is an API priority to invest in.

Discovery entails a number of challenges for enterprises. As with other types of content and media, API consumers are unlikely to spend much time looking for APIs before they go elsewhere, or begin building what they need themselves. It is essential to view the concept of API discovery from the perspective of your consumers.

## Evaluate

Once consumers discover an API, they need time to evaluate whether it meets their needs. Do yourself a favor by providing a streamlined, hands-on experience, allowing consumers to play with real functionality and learn about resources and capabilities by doing something, rather than just reading about it. Giving them the information they need to make an educated decision about whether an API will satisfy their needs makes them feel empowered and helps them understand how the API fits into the business solution they want to build.

**Explore** - Enable consumers to explore as much of the surface area of an API as possible, perhaps without even authenticating, so they can learn what is possible.

**Execute** - Provide the ability to execute each request, response, publish, and subscribe, making sure that learning about an API is as hands-on as it can possibly be.


**Examples** - Make sure there are always examples for each element of an API, allowing API contracts to be mocked and providing rich documentation showing how they work.

**Documentation** - Provide rich documentation with useful descriptions, examples, and other information to help consumers get started using each API.

**Workflows** - Go beyond reference documentation and provide actual business workflows consumers can use to accomplish their workplace goals.

**Demonstrate** - Show consumers how the API works, providing tutorials, videos, stories, and other content demonstrating what is possible when you integrate APIs into applications.

**Forking** - Make your documentation, mock servers, tests, and workflows forkable, enabling consumers to fork artifacts that help them onboard and start using your APIs faster.

**Feedback** - Make it easy for consumers to provide feedback as they evaluate an API, capturing as much data from their experience as possible to inform your API roadmap.

Investing in the evaluation stage of a consumer-centric API life cycle helps reduce consumers' time to first call, or more importantly, time to first transaction. You are making it as easy as possible for consumers to begin generating value in their applications and on the API platform.

There are many signals consumers consider when evaluating an API. This is the stage of the consumer life cycle where you are most likely to lose partners and public users, who may have other options for APIs. Your internal users may have other options as well. But if they don't, make it easy for them instead of creating friction.

## Integrate

There are many ways for consumers to develop against an API. It used to be enough to provide snippets or libraries in a variety of programming languages, but increasingly, consumers want access to artifacts, collections, and other ways of visually developing their integrations or applications, pushing the boundaries of traditional desktop, web, or mobile applications.

**Contracts** - Machine-readable contracts like OpenAPI and AsyncAPI make integration as simple as importing the contract for an API, authenticating, and making the API calls you need to move data between systems, providing an artifact that defines consumption.

**Collections** - Provide forkable and executable collections that describe specific parts of an API to help consumers accomplish a specific digital capability. Provide a buffet of capabilities for consumers to choose from for their API integration needs.

**Automation** - Take advantage of automation opportunities, allowing collections to be scheduled and baked into the CI/CD pipelines and common business capabilities to be executed. Business and technical stakeholders can do more with less through API automation.

**Workflows** - Provide ready-to-go low-code and no-code options for executing common business workflows, allowing multiple internal, partner, and public APIs to be daisy-chained into solutions that will help business and technical stakeholders integrate better.

**Snippets** - You can generate lightweight code snippets in a variety of programming languages. They will do most of the heavy lifting for consumers integrating APIs in the language of their choice, automating the more repetitive aspects of API integration.


**SDKs** - You can generate complete software development kits, abstracting away the authentication and other complex aspects of deploying APIs. You always want to reduce the workload for consumers, easing integration of your APIs into their applications.

Today's API integrations come in many shapes and sizes, requiring a mix of approaches to satisfy the needs of the widest possible consumer audience. We are moving to a more modular, automated, and low-code/no-code environment when it comes to stitching together the thousands of APIs we need to do business today.

The ways consumers integrate API resources, capabilities, and experiences into their applications, automations, and other types of integrations is rapidly expanding. This rapid expansion is raising the bar for API producers, who must reduce friction and lighten the integration load with proven solutions.

## Test

API testing should be shared with consumers, opening up the possibilities for user acceptance testing. That will lead to greater trust and understanding of the reliability of a platform, and will strengthen the relationship between API producer and consumer. The results of API testing, including dashboards and visualizations, can be shared with consumers via workspaces and repositories, and alongside API documentation and portals. Go the distance–API testing plays an important role for both API producers and consumers.

**Availability** - Share uptime and availability information with consumers on a dashboard, providing transparency around the operation of the platform they will depend on for their applications and helping provide a historical accounting of availability.

**Contract** - Exposing contract tests and even the results of scheduled contract test runs will increase consumer awareness of API contracts and how their validation can be used in applications and integrations.

**Performance** - Share performance tests and results of scheduled performance test runs, increasing transparency about API performance while demonstrating how your platform has considered performance and is taking steps to improve upon it.

**Security** - Share your overall security policy and security test results, creating trust with consumers.

**Usage** - Explain that any type of consumer testing is not considered part of the usage they pay for and doesn't show up as part of rate limits. That will encourage consumers to test an API and play a role in the feedback loop for your platform.

**SLA** - Provide an honest service level agreement (SLA) for each API, detailing expectations for services. Allow for beta, experimental, and less mature APIs, while demonstrating the production-ready grade of other APIs.

To strike a balance between producers and consumers, your baseline testing should occur via a platform, bringing consumers into the conversation about API quality and holding producers accountable for platform quality.

Visibility and transparency at the testing level builds trust and ensures that API producers are truly considering the pain of consumers. Good API producers are also API consumers. They have been there, and felt the friction of onboarding themselves.

## Deploy

The deployment of API integrations comes in many shapes and sizes, and the notion of what an application is has shifted over time. Putting APIs to work used to be simply about web and mobile applications or system-to-system integrations. Today, deployment in the world of API consumers ranges from writing custom code for an application to low-code/no-code syncing between two separate API platforms.

**Source Control** - Place all manually developed or automatically generating client code in a repository, providing a source of truth for the code and for any artifacts that are needed to define the deployment and operation of any API integration.

**CI/CD Pipelines** - Implement the continuous integration portion of CI/CD, automating how applications and integrations are deployed and, making the deployment of API integrations, applications, and other use cases something that is always repeatable.

**Collections** - Leverage a Postman collection as a modular, shareable, and executable definition of an application, stitching together many different API calls across internal and external API sources to apply digital resources and capabilities in a specific way.

**Serverless** - Use serverless layers for deploying integrations, orchestrations, and automating API resources and capabilities, tapping into ephemeral compute to deploy integration code that accomplishes specific business outcomes.

**Runners** - Acknowledge that some collection applications will be manually run by different team members using runners, organizing different types of integrations and applications by workspaces and letting different stakeholders manually put them to work.

**Monitors** - Schedule the run of collection-defined integrations. Use the collection, combined with environments that employ a variable strategy, to accomplish any API-driven applications and integrations on a schedule from any cloud region.

**Workflows** - Take advantage of complex workflows to put APIs to work, iterating through multiple series of API calls to enable business and technical stakeholders to design, save, and execute the scenarios they need to accomplish business each day.

**Webhooks** - Respond to different API events using webhooks to trigger collection- defined integrations, applications, and workflows. Engage with users via different platforms while responding to their activity using modular API deployments.

To help today's consumers deploy their API integrations, producers need to think beyond web and mobile apps. Make deployment across a spectrum of possibilities a simple, one-click affair, using minimal code.

## Observe

API consumers need to "see" API operations and learn how their API consumption, or the consumption across a community, influences their own applications and integrations. Observability is fast defining the difference between healthy API ecosystems, and not-so-healthy API communities, and healthy and not-so-healthy organizations.

**Watches** - Keeping track of the watches on workspaces, APIs, and collections to understand who is tuned into what is happening. Use watches as a metric for the number of consumers, contributors, and internal and external stakeholders who are tuned in.

**Forks** - Track who is forking repositories and collections, using the fork count as a metric for engagement and knowing who your consumers are. Track engagement via workspaces, repositories, and collections to learn how consumers are using your APIs.

**Feedback** - Engage with API producers and consumers, understanding the conversation is around each API or group of APIs. Observe discussions about digital resources and capabilities.

**Notifications** - Use in-app, email, or SMS notifications to engage with a platform and keep consumers part of the forward motion of an API, collecting metrics for observability.

**Usage** - Provide dashboards, reporting, and other visuals to help consumers understand their platform usage. These features will keep consumers engaged, helping them play an active part in the community and observe the activity that matters.

Use existing platform outputs to keep consumers informed, but also make their engagement more observable for producers and other consumers. That will help contribute to the overall health and viability of the ecosystem surrounding each API internally or within the external community.

API consumers need all the help they can get with observability so that they can regularly see and understand what is happening with their integrations. APIs are abstract, and it can be difficult to see individual and community use. That's where your help comes in.

While API consumers care a lot more about their own API consumption, usage and feedback by others in the community will also play a role in how they view a platform, and will influence the decisions they make moving forward.

## 13.4 Moving Past Tribal and Vendor Dogma

While there should be a common understanding for the API life cycle across an organization, it is fine to have multiple entry points into the life cycle, as long as the end result is documented, tested and governable. In this industry, it can be too easy to insist on one right way of doing things–as technologists, we tend to look at things through a Boolean lens. But enterprise operations are much more human than that, and require a mix of approaches to bring alignment across teams.

API design-led is often held up as the gold standard for API development and the definition of what the API life cycle should be. However, on the ground, this belief is often torn up as teams try to tackle modernizing legacy systems, iterating on different types of APIs with other teams who possess a mix of skills for delivering consistent APIs. API design-first represents an optimal state for API operations, but it doesn't always reflect the skills teams possess. This reality tends to create friction as API design-led teams try to develop awareness and synergy across many different teams, resulting in pushback in the API-first transformation.

Moving to API design-led too fast can entrench existing tribal boundaries among teams, making it more difficult to move forward. While planning your API-first transformation, take inventory of the approaches teams are using to bring APIs to life and iterate upon them. Meeting teams where they already are is always the best place to start.mapping out processes and defining possible next steps may well reduce friction among teams. While many variations exist, there are four main processes to use to match the needs of teams on the ground.

API design-led should always be your destination for teams. After getting to know where your teams stand, you can begin thinking about how to help them see the potential of being API design-led. They will not be ready for a complete shift, but can adopt some of the benefits introduced by this approach while still using code-led or proxy-first ways to generate contracts, helping define and design APIs better without undergoing overly stressful behavioral changes. Technological change is much easier than human change.Acknowledging the scope of what is needed to get teams from code-led to design helps quantity the work needed. That will help you understand how much work and time it will take to move the organization forward at scale, or at least in incremental steps over time.

# 14
# Approaches to Creating APIs

There are several different methods for creating or modifying APIs. How are they different, and does it really matter which one you choose? This chapter will explain the most important variables and show you how to use your existing infrastructure to move toward becoming API-first.

## 14.1 Making Decisions: Lead with Design, Code, Prototype, or Proxy?

### Design-led: the optimal approach

Using a design-led approach means you define and design your API before you begin writing any code. You use an API specification to develop an API contract, mock and document the contract, then iterate with stakeholders upon the design of the resources and capability. Once you have reached agreement about what the API should do, you can develop tests to verify that the specifications you desired are in fact in production.

**Workspace** - Work on a new API always begins in a dedicated workspace. Make sure there is a single place where teams can find artifacts and the work that exists behind each API.

**Contract** - Every API has a machine-readable contract describing the surface area of the API, providing an understanding between API producer and consumer to guide operation.

**Mocks** - The contract for an API is perpetually used to generate mock servers, helping make the API design as realistic as possible by matching specific use cases with examples.

**Document** - Documenting means generating human-readable documentation from an APIs contract, ensuring that it is accurate and uptodate for each API as it is being designed.

**Feedback** - Provide a feedback mechanism for all stakeholders to use with the current design of an API, helping guide teams to add new features, capabilities, and experiences.

**Iterate** - Aggregate feedback from consumers and other stakeholders.Identify sensible changes to the API, then iterate on the contract, updating mocks and documentation.

**Test** - Once the contract for an API has been established, and there will be no more iterations to this version, you can produce contract tests to validate that an API delivers upon the original intent when in production.

**Develop** - Hand off the API contract, providing teams what they need to bring an API to life in development. Move to staging and tests before it goes into production.

A design-led approach helps produce artifacts and gets all stakeholders and teams on the same page for what is needed to produce an API. It saves time by allowing teams to iterate upon an API without having to produce code. While design-led can get a bad reputation for creating more work than code-led, in reality it saves a significant amount of time throughout the life cycle.

## Code-led: an acceptable approach

It's common to develop a new API or add functionality to an existing one by writing code. A code-led approach is about producing the desired behavior by writing code in a local or shared development environment. Once you agree upon the API's shape and behavior, you produce a machine-readable contract. This contract will then be used to provide a set of tests that will automate the validation of assertions stakeholders make throughout the development process.

**Workspaces** - Even when an API is developed in a code-led manner, you should set up a dedicated workspace to provide a single place to find all work.

**Annotations** - Use programming language annotations to add the necessary metadata to the code behind an API in order to generate a machine readable contract.

**CI/CD Pipeline** - Use your CI/CD pipeline to turn annotations into machine-readable contracts.Translating your code into a contract allows you to power documentation, using the forward motion of AIIs to produce the contracts you will need to define them.

**Contract** - Produce a machine-readable contract for your API, leveraging code and your existing software development life cycle to create the contract needed to govern each API.

**Document** - Generate human-readable documentation from an APIs contract, ensuring that is accurate and up-to-date.

**Feedback** - Gather feedback on the design of an API, tapping into existing channels to understand what consumers and other stakeholders need from it.

**Iterate** - Update the code base based upon feedback, adding capabilities, and building and updating the machine-readable contract, which then updates elements downstream.

**Test** - Produce contract tests for APIs. Add tests to your CI/CD pipeline to ensure that each API continues to reflect the agreed-upon contract, with no surprises in production.

Code annotations are the key that links a code-led approach with some of the benefits of design-led. API design still exists when writing code, and annotations help express the design as machine-readable API contracts. Using the existing software development life cycle to produce API contracts, you can t power documentation, testing, and other benefits often associated with a design-led approach. Code-led lets you build upon the existing skills of teams, translating their work to produce some of the benefits design-led champions like to showcase.

## Prototype-led: very efficient

This approach to delivering APIs involves building a prototype of an API, opting to not build a contract first, but mocking and documenting the desired functionality using a collection. With a prototype-led approach, you produce as much of the functionality as

you can, iterating upon the design of the API with stakeholders before producing a machine-readable contract, as well as the tests you'll need to verify that your API behaves as expected in production. A prototype-led approach will appeal to some teams as the quickest way to move from idea to development once everyone is ready.

**Workspace** - Whether you're creating a new API or enhancing an existing one, a workspace is always the place to begin, ensuring there is a single location where teams can find all ongoing work.

**Collection** - We hand-craft a collection to describe the surface area of the API, defining the requests and example responses to make our API as real as possible.

**Mocks** - The contract for an API is perpetually used to generate mock servers, making the design of the API as realistic as possible by matching specific use cases with examples.

**Document** - Generating human-readable documentation from an APIs contract is important. It ensures you have accurate and up-to-date information about each API as it is being designed.

**Feedback** - Provide a feedback mechanism for all stakeholders for the current design of an API, helping guide producers forward.

**Iterate** - Aggregate feedback from consumers and other stakeholders.Identify sensible changes to the API, then iterate on the contract, updating mocks and documentation.

**Test** - Once the contract for an API has been established, and there will be no more iterations to this version, you can produce contract tests to validate in production, ensuring that the original intent of the API is accomplished.

**Contract** - Once you've effectively prototyped your API, iterate upon the design using the prototype. Then you can choose to generate a contract from the collection prototype, getting to the same place where you would be with a design-led approach.

API contracts are an essential part of API operations, but sometimes it makes sense to produce a contract only after you have iterated upon an API with a prototype. Not every team will be proficient in crafting an API contract to use for mocking. Sometimes it makes more sense to produce a collection, and iterate upon the API first. Then when they're ready, teams can use the collection to generate a machine-readable contract that can be used throughout the rest of the API lifecycle.

## Proxy-led: managing your legacy infrastructure

Proxy-led is the process of reverse engineering of an existing API by proxying the requests and responses sent, or the published and subscribed messages delivered. That generates a collection from actual behavior within a desktop, web, mobile, or device application. The intent of proxy-led is to produce a collection that describes the surface area of an API, then mock and document this behavior before you produce a machine-readable contract and tests to validate behavior moving forward.

**Workspace** - Whether you're creating a new AP or enhancing an existing one, a workspace is always the place to begin, ensuring there is a single location where teams can find all of the ongoing work.

**Proxy** - Run the traffic for a desktop, mobile, or device application through a proxy to reverse engineer the traffic, mapping out the surface area of the APIs behind them.

**APM** - Pull the traffic logged in of APM solutions as evidence of API paths, parameters, and headers, as well as request and response bodies, or the messages being published as part of event-driven APIs.

**Collection** - We hand-craft a collection to describe the surface area of the API, defining the requests and example responses to make our API as real as possible.

**Mocks** - The contract for an API is perpetually used to generate mock servers, making the design of the API as realistic as possible by matching specific use cases with examples.

**Document** - Generating human-readable documentation from an APIs contract ensures you have accurate and up-to-date documentation for each API as it is being designed.

**Test** - Once the contract for an API has been established, and there will be no more iterations to this version, you can produce contract tests to validate in production, helping ensure the quality of APIs.

**Contract** - Once you've effectively prototyped your API, iterate upon the design using the prototype. Then you can choose to generate a contract from the collection prototype, taking the evidence gathered to produce the API contract needed over time.

A proxy-led approach allows you to define the API landscape and contracts that already exist and automate the way you determine the "diff" between your contracts and what exists in production. A proxy-led approach helps your API catalogs, portals, and network keep pace with what is happening on the ground across operations, perpetually synchronizing hundreds of thousands of API contracts with the reality in production.

## 14.2 Life Cycle Essentials

There are only minor differences in efficiency and velocity between a design-led, code-led, proxy-first, or prototype-led approach. The major gain in efficiency and velocity comes when all APIs possess contracts and up-to-date documentation, are fully tested, and are governable. Then you can bring them l into a common, well-known API life cycle, no matter how teams have begun their journey.

**Life Cycle** - A common, agreed upon API life cycle emerges across teams. They understand the various ways for entering it, and can discuss processes for every stage of the life cycle.

**Contracts** - You must have up-to-date and accurate machine-readable contracts available for all APIs, no matter how the API lifecycle is entered, to ensure that business value can be validated.

**Discoverable** - Every API is discoverable, including its metadata, the contracts defining what is possible, and the operations surrounding the API in production.

**Productivity** - Teams can move forward at their desired velocity. The company, while not compromising quality, keeps teams as productive as possible.

**Quality** - Contract and performance tests cover as close to 100% of the surface area of APIs as possible, ensuring a baseline of quality moving forward.

**Observable** - Every API and the life life cycle around it is observable, providing you awareness and control over all aspects of the life cycle, and the way APIs are used.

**Governance** - APIs are discoverable, reliable, consistent, and delivered in a standardized format, no matter which team is developing them. That helps govern the forward motion of the enterprise.

**Velocity** - Achieving organizational, domain, and team velocity moves the enterprise forward. Finding just the right speed for the moment is the key to taking your business where it needs to go in the coming years.

The goal of the API life cycle isn't to restrict teams to one way of delivering APIs, but rather to establish a common vocabulary, allowing everyone to get on the same page and determine together what is important across the API life cycle.

Declaring what must always come first or saying there is a single way to deliver APIs is counterproductive and will undoubtedly slow down your API-first transformation. Ultimately, the approach isn't the destination.Our goal is to efficiently and effectively deliver high- quality APIs that are discoverable, and have an up-to-date API contract that can be used across a well-known API life cycle.

What matters most when trying to shift your API-first transformation towards a design-led approach is meeting your teams where they are. This is the only way you will be able to truly understand the state of teams' work and determine how far you need to go to become API design-led.

## 14.3 Maximizing Your Infrastructure Investment

A modern API life cycle is best delivered as a layer on top of your existing software development lifecycle (SLDC), maximizing the infrastructure investment you have already made. APIs are the way you deliver more composable software.The API life cycle isn't a replacement for what already exists– it is about optimizing and refocusing the resources your teams are already using to support your API strategy.

The best way to stitch together the API platform you will need to scale your operations is to augment your existing Git workflows with an API workspace and enrich your CI/CD pipelines with API testing, while beginning to automate management of your API gateways with your existing build processes. Once you have this set up, you can begin piping your API life cycle into your APM solution to make APIs and the operations around them observable with existing infrastructure.

While you will need many other pieces of infrastructure to work seamlessly with your API platform, workspaces, source control, CI/CD, gateways, and APM solutions provide the cornerstones. These existing investments are now being retrofitted to support your future of efficiently delivering high-quality APIs, acting as the API factory floor for your digital supply chain. Your existing software development life cycle has served you well to this day. All you need to do to take your organization into the future is to augment it

with API workspaces and gateways. That will provide the automation, observability, and governance you need to scale from hundreds to thousands of APIs across your business.

To pick up speed with your API-first transformation, you need to build on top of your existing software development lifecycle. You need to map this existing life cycle with the API contracts, schema, collections, and other API artifacts in your API workspaces–syncing these artifacts with your existing source control, so that it can be used for testing in the CI/CD pipelines, and deploying and configuring policies at the gateway layer. These API contracts are what puts the API into your software development life cycle, providing what you need to standardize and stabilize your existing software development workflows.

Ultimately, it doesn't matter whether the source of truth for your API lives in your Git repositories or your API workspaces.What matters is achieving alignment between your software development lifecycle and the iteration, documenting, testing, and deploying of machine-readable API contracts to meet the needs of the enterprise. You have already made 75% of the investment needed to pick up speed with your API-first transformation. All you need to do now is augment it with the remaining 25%, which will transform your existing software development workflows with the API contracts you need to do business today.

# 15
# Workspaces, Source Control, and CI/CD

What can you do with API workspaces? Likely, a lot more than you think. You can also use source control and CI/CD to hasten your API-first transformation, improving quality and velocity every step of the way. This chapter will show you how.

## 15.1 Workspaces

API workspaces are to the API lifecycle as Github repositories are to the software development lifecycle. A symbiotic relationship between Git repositories and workspaces has emerged.

### Contents

**APIs** - One or more APIs (defined as OpenAPI, GraphQL, WSDL, Websockets, or gRPCs) provide a machine-readable contract that can be used to guide the rest of the life cycle.

**Collections** - Collections define documentation, workflows, mock servers, tests, and other automations as portable and executable units of value to be used across the API life cycle.

**Environments** - Environments are key and corresponding value storage that is used to define the base URLs, secrets, and other variables you need to use across multiple APIs.

**Monitor** - A scheduled cloud runner can execute different collections, running tests, automated workflows and orchestrations, and automating applications and integrations.

## Visibility

**Private** - You can make workspaces accessible only internally within the enterprise, applying RBAC to the workspace, APIs, collections, and environments to maintain order.

**Partner** - Or you can open up workspaces to trusted external partners, sharing the work behind each API for collaboration to enable producing and consuming APIs in a shared workspace.

**Public** - You may decide to invite the public to view, watch, and fork APIs and the collections they contain, making not only the APIs, but the operations around them publicly available.

## Engagement

**Watch** - Watch your workspaces, APIs, and collections, pushing notifications to consumers whenever there is a change to help bring producers and consumers closer.

**Activity** - Activity includes any action across the team involving configuring APIs, documentation, mock servers, environments, monitors, and other API elements.

API workspaces are the cellular makeup of your enterprise.They provide access to your APIs, but also give you a window into productivity, quality, and governance memory, making sure your enterprise moves forward at the speed you need to do business in a digital world.

EXPERT PERSPECTIVE

# Everything you need for an API is available in the workspace


When you listen to Tim Velasquez, Software QA Manager at Werner Enterprises, you will begin to see workspaces very differently. And you will also see how important the role of QA is in an organizational shift. Tim walked me through the quality layers his team brings to each API through its workspace. He also explained how having the right QA processes in place will continue to accelerate quality and velocity as new versions are iterated.

For Tim, an API workspace isn't just a place where you publish API contracts, documentation, mock servers, tests, and other essential elements of API operations. Theworkspace is where you find everything you need to understand what an API does, and how you can either iterate on the next version or put the API to work in an application, integration, or automation. At Werner Enterprises, a workspace isn't just where you find an API, it is where you find the highest-quality version. That's because Tim's QA team has done its job to guarantee that each API is the best possible version of that API it can be.

Picture it. Within each API workspace you will find the API contract, documentation, mock servers, environments, and a full suite of tests to ensure the highest-quality API experience. With each version of the API, you repeat this process, iterating upon what already exists while ensuring the API continues to deliver a high-quality experience. Over time you are hardening and maturing your API and improving the overall experience, providing one digital resource, capability, or experience you need to operate. What Tim's team is doing is providing the nutrients you need to ensure that each API is a healthy cell in the overall enterprise organism. When you do this across hundreds or thousands of APIs, you end up with a healthy and vibrant ecosystem of cells across multiple domains.

Beyond delivering APIs of the highest quality, think about what Tim's approach to API workspaces does for teams. Team members can go into development not knowing anything about an API. Then they land on the workspace, and boom–they can readi the workspace overview, explorie documentation, run tests, and

use other resources to quickly get up to speed without searching or guessing. Everything they need is easy to discover and readily available in one workspace. Later, when this team iterates upon an API, everything they changed is added to the cellular memory of the API and its supporting workspace.

Workspaces are a one-stop-shop not only for teams, but for API consumers as well. And over time, they continue to improve team efficiency, allowing developers to do more and more in less time.

# 15.2 Source Control and CI/CD

## Source control

To move the organization forward along an API-first path, start by leveraging your existing source control for tracking and managing changes to code, and include the machine-readable artifacts you produce across the modern API life cycle.

Source control is key to managing collaboration among stakeholders and consumers across the multiple versions of an API that might be in production at any given moment. Source control allows us to extend the existing software development lifecycle to deliver and govern each API across a well-known API life cycle.

**Organizations** - Establish organizations for source control that are in alignment with your enterprise-wide API strategy. Align domains, groups, and organizations to facilitate the enablement of teams delivering and operating APIs.

**Repositories** - Make sure you have a clear mono or distributed repository strategy for the API life cycle. That will provide a source of truth to guide r team decisions about how source control repositories should be used to support APIs.

**Folders** - Include in your API strategy guidance instructions about how folders within source control should be structured and used to consistently organize code and artifacts. That will give you a consistent approach for learning what is needed across many repositories.

**Artifacts** - Make sure essential machine-readable artifacts–such as OpenAPI, AsyncAPI, JSON Schema, and other contracts–are available across the API and software development lifecycle for use in documentation, testing, and much more.

**Feedback** - Leverage source control feedback loops as part of the wider API life cycle feedback loop. Gather feedback from stakeholders throughout the evolution of code and artifacts, and ensure that feedback can be gathered at all stages of the life cycle.

**Integration** - Seamlessly integrate your source control into all aspects of the API life cycle, using Git or APIs to make your source control central where the API work is happening. Do not neglect to invest in the GitOps portion of your API factory floor.

**Automation** - Put automation to work, ensuring that your evolving code and artifacts for producing or consuming APIs is as standardized and repeatable as it can be. That will allow teams to do more with less while achieving high levels of quality.

Git repositories can be aligned with API workspaces, but they also provide a well-known and seamless relationship with workspaces. This relationship between repositories and workspaces helps ensure that API contracts, collections, and other artifacts are present not just throughout the API life cycle, but in the underlying software development life cycle, enhancing the way your teams work.

## CI/CD

Tapping into continuous integration and continuous deployment, better known as CI/CD, will help you build, test, and automate integrations with internal and external APIs, and also deploy APIs that can be used for applications and integrations. CI/CD allows you to leverage the native pipelines of your source control–or add a commercial or open source CI/CD solution, seamlessly tying API production and consumption with the repeatability you need across teams.

**Pipelines** - Use CI/CD pipelines to ensure that the API lifecycle is always repeatable and builds the highest-quality APIs possible Your pipeline provides a consistent build process for the deployment of each API you deliver.


**Variables** - Tap into a life cycle-wide strategy for defining variables applied as part of the pipeline build process, ensure that the naming, use, and evolution of these variables occurs at the strategic level, and that they are applied consistently at the CI/CD level.

**Collections** - Run contract, performance, security, and other types of collections as part of your pipeline using open source Newman. That way, you will be standardizing and validating as your APIs are built, applying a variety of tests, rules, and policies to improve quality.

**Environments** - Use standardized guidance to provide teams the development, staging, and production environments they need, keeping CI/CD pipelines alongside collections to ensure that each environment is well-defined as you feed the pipelines.

**Observability** - Tap into existing outputs for your CI/CD pipelines and for piping data into your API platforms, APM, and other observability and reporting systems. That will provide visibility for the testing that occurs as teams build APIs.

A modern API life cycle is built on top of existing enterprise investment through continuous integration and continuous deployment processes. You layeri incommon artifacts like OpenAPI, collections, and environments to ensure quality while maintaining velocity across operations.

CI/CD pipelines are the way to automate the API life cycle, using collections to execute contract, performance, security, and other types of tests. CI/CD pipelines are also ripe for automating every other stop along the API life cycle–such as publishing documentation to a portal, applying policies at the gateway layer, and injecting governance into the build process across teams.

Of course, you can test your APIs at the CI/CD layer. But what teams need to realize is that this same testing can be applied not only to the instance of an API, but also to the operations around it.

# 16
# Gateways, Performance, and Scaling

As APIs proliferate within the enterprise and around the globe, sharing them with teams, partners, and consumers is becoming more complex. Fortunately, an array of commercial and open source solutions is available to help you navigate the new landscape.

Innovative gateway models and cloud-native architecture provide unprecedented opportunities for automation, customization, and speed, while real time application performance monitoring lets you know whether your operations in Zambia are really hitting the mark. This chapter will show you some of the cool new tools you can use to make your APIs work better and faster, setting you up for a brighter future in a fast-changing world.

## 16.1 API Gateways

The first API gateways were offered as a component within larger API management platforms. As the industry evolved, a new wave of gateways was introduced to provide solutions for a changing landscape. Some gateway providers are now supplementing their standalone offerings with additional tools, often centered around a management approach more conducive to internal API life cycles.

## The most common API gateway capabilities

Gateways, which have become essential to enterprise operations, offer a mix of open source or commercial capabilities. Here are a few of the most important:

**Authentication** - You need to ensure that consumers are authenticated before they access an API resource or capability. Authentication improves consistent security across all APIs behind applications, protecting the resources and capabilities exposed.

**Authorization** - Authorization controls which resources and capabilities consumers can access.

**Contracts** - You can use OpenAPI, AsyncAPI, and JSON Schema to shape the deployment and management of APIs. The contract defines the resources and capabilities made available, as well as the backend system mappings using spec extensions.

**Routing** - A common capability for gateways is to route traffic to a specific backend service, or possibly to an external service, playing traffic cop for all the API requests made. Routing obfuscates the backends of APIs from consumers building applications.

**Plans** - Organizing APIs and their consumers into standardized, but sometimes customized access plans allows you to govern which APIs are available, helping align APIs to business domains and objectives and keeping the API catalog current.

**Policies** - Machine-readable policies define the configuration and constraints applied to APIs and their consumers as they access digital resources via each gateway, providing standardized constraints across all APIs.

There are plenty of other capabilities gateways offer, but these are the leading ways they are used to expose valuable digital resources, capabilities, and experiences. These capabilities provide a baseline for quality, security, and consumer access, striking a balance between access and control over digital resources, capabilities, and experiences.

## Characteristics of a modern API gateway

No more can a single vendor control access to all internal resources and capabilities. Today's multi-gateway landscape possesses a range of characteristics suitable for different needs and outcomes, reflecting the ever-expanding demands consumers are placing on internal and external APIs.

**Centralized** - Many enterprises have a single centralized gateway handling all traffic coming from outside the enterprise via a single entry point. That provides an industrial-grade way of handling traffic coming in and out of the enterprise.

**Federated** - It is increasingly common for enterprises to support a federated gateway approach for making APIs available across domains, acquisitions, and lines of business, and helping to manage traffic in and out of the sprawling enterprise landscape.

**Regional** - The deployment of regionally specific gateways has emerged to respond to increased regulation and data sovereignty rules in many regions. Regional gateways also bring resources and capabilities closer to the consumer.

**Vendors** - It is common for enterprise organizations to have API gateways from multiple vendors, providing a mix of gateway solutions teams can use when securing APIs or managing the access to digital resources and capabilities within a specific cloud.

**Cloud** - Many enterprises now use multiple clouds, leading teams to runcloud-specific gateways. That means teams need to learn specific approaches for publishing cloud- native APIs.


**Open** - Open-licensed gateways, as well as openly licensed contracts, policies, and other artifacts, are increasingly common in enterprises, providing a low-cost and interoperable approach to managing gateway APIs.

The gateway landscape today is farmore modular, distributed, and standards-driven than it was just a decade ago. API gateway offerings continue to evolve to meet an increasingly diverse landscape of protocols, patterns, and network considerations.

These API gateway characteristics aren't just exposing APIs, they are helping us ensure that our APIs are available where and how consumers need them. These elements reflect the needs of the modern enterprise for delivering the digital resources, capabilities, and experiences they need to move at the pace of today's global operations. Today's API gateway has shape-shifted to reflect what teams need for navigating a sprawling enterprise landscape.

EXPERT PERSPECTIVE

# Federating the API gateway landscape


API gateways have become a commodity today, with a wealth of open source and commercial providers offering solutions. When you study the API gateway provider landscape, you begin to understand why Dekel Tankel, Global Field CTO for Tanzu at VMware, says that federated API gateways are the future. The advent of microservices has given t enterprise developers greater power, and teams are looking for autonomy and agency. A federated approach can deliver it, while also helping organizations scale.

With the power shift, developers have asserted more control over the gateways they depend on for delivering critical microservices. While many organizations continue to use a single centralized gateway owned by traditional IT groups, there is often a federated landscape of micro, regional, and other gateway incarnations. This gateway sprawl employs a mix of open source and commercial solutions, rendering the future of APIs a multi-gateway affair. Teams nowadays don't need all the capabilities that came with the central API gateway IT installed. Instead, they have chosen much smaller, lighter-weight API gateways.

With its Tanzu gateway offering, VMWare doubled down on a federated API gateway approach, leveraging API contracts to power the gateway and encouraging a federated approach to gateway policy management. VMWare's offering spoke to the needs of technology leaders by providing a contract- and policy-driven approach that could be centrally defined. But the company didn't turn a deaf ear to developers, creating a microgateway approach to meet their needs, while enabling them to also take advantage of centralized contracts, patterns, and policies. This API gateway pattern resulted in part from the last 20 years of gateway evolution, but it also reflects the new enterprise landscape.

A federated API gateway approach delivers a successful formula for API governance, providing centralized governance that also translates to enablement of capabilities on the ground floor. This type of architecture embraces the sprawling reality of enterprises, while still allowing them to centralize policies and practices, standardizing the delivery of APIs across many domains, teams, and geographic regions.

VMware's progressive approach to API delivery meets developers where they are, providing them agency and autonomy while still holding them accountable for adhering to centralized policies, governance,and industry regulations. In other words, a federated API gateway landscape helps organizations balance technology requirements with their business needs and their developers' demands for agility as they roll out APIs at scale.

## 16.2 Application Performance Management (APM)

As part of the API-first transformation, organizations look to existing APM solutions to monitor and manage the performance of APIs and the infrastructure and operations around them. All outputs from every stage of a well-defined API life cycle – and the software development lifecycle beneath it–are fed into existing APM solutions. This approach leaves no part of API operations untapped when it comes to detecting and diagnosing the performance and quality of API operations.

**Collections** - Postman collections are the modular, shareable, and executable way of defining the outputs across API operations. Collections make any platform more observable, allowing any aspect of API operations to be measured and evaluated.

**Environments** - Machine-readable environments define the common elements of API environments, including URLs, tokens, keys, and other variables that can be applied in a variety of settings, providing an efficient way to make APIs more observable.

**Operations** - You can define collections for the operations behind APIs, making source control, CI/CD, gateways, and other stops along the API life cycle observable. You can use collections to define the output, and a monitor to publish into the APM solution.

**Coverage** - You can standardize the coverage of tests for contract, performance, security, operations, and governance by using collections and scheduling with monitors. That provides you as close to 100% coverage across APIs as possible, ensuring that every part of operations is observable.

**Monitors** - Monitors are cloud collection runners that can be scheduled and run in different cloud environments. A variety of configurations is possible, allowing you to automatically monitor any aspect of API operations.

**Results** - The results from collection runs against different environments can be routed into common APM solutions, making the health data of APIs and operations available for further processing. Data can also be included in observability dashboards.

**Dashboards** - APM solutions make not only individual APIs, but the infrastructure around them more observable.That means you can view source control, CI/CD, and gateways in ways that make more sense to your business.

Your APM solution is the observability window across your APIs and your operations. Every output from your API operations should route into your APM to provide you with a dashboard view of all the technical details surrounding your APM management. It also supplies your product managers and leaders with the view they need. Your APM is key to understanding the state of your enterprise. Use it to build the awareness you will need.

## 16.3 Embracing the Benefits of Cloud-Native

The cloud was the third evolutionary shift in the API universe–right after commerce, and social. The cloud has been available via APIs since its inception, so it makes sense that there is a symbiotic relationship between being API-first and being cloud-native. The cloud enables APIs, and APIs enable the cloud. Using the cloud is an essential part of the API life cycle, API governance, and the management of API operations at a global scale.

The cloud provides many benefits, and there are also many paradigm shifts that come when delivering APIs in the cloud. Here are a few of the key elements of cloud-native that you should be thinking about as you invest in your API-first transformation:

**Elasticity** - Cloud platforms can scale much more easily than traditional data centers, leveraging virtualization and automation to meet the demand of consumers and allowing all types of infrastructure to be elastic and flexible.

**Concurrency** - APIs can be scaled out horizontally across multiple identical processes, as opposed to vertically, scaling-up a single large instance. That provides a more balanced approach for delivering elasticity across your services.

**Configurations** - The standing up and configuration of APIs can be done via APIs, allowing for more automation and repeatability. It's all part of the elasticity, concurrency, and availability of API services and products, which can always be configured.

**Disposability** - In the cloud, all API infrastructure is disposable. You can stand it up and reconfigure as needed, ensuring that teams can confidently dispose of one or many instances of an API, then quickly return to the desired state if they choose.


**Logging** - The capacity for logging API activity is greater in the cloud. Every event and activity that occurs can be logged, making the collection, aggregation, archiving, and observability of API logging a default part of API operations across teams.

**Administrative** - All the administrative tasks you need to operate APIs are available via APIs. That gives you another class of APIs you can use to manage, automate, and orchestrate APIs, ensuring that your API operations are always moving forward.

**Costs** - Defining, calculating, and managing the costs associated with API infrastructure in the clouds is much simpler and can easily be automated. You can use cloud billing APIs to obtain a tighter grip on what you spend to operate your APIs.

**Regions** - A cloud-native approach allows you to deploy APIs in geographic regions around the world, increasing the reach and performance of your APIs and significantly hastening your API-first transformation.

The cloud provides the elasticity, concurrency, configurability, and disposability you need to efficiently deliver and iterate upon the thousands of APIs you will need to do business in coming years. While there will always be situations where on-premises makes sense, moving to the cloud enables you to meet the needs of the future while simultaneously managing your legacy.

Deploying new infrastructure is not a one-and-done deal. You will need to stand up, tear down, and perpetually iterate and evolve your infrastructure. The place where you will do that is the cloud.

EXPERT PERSPECTIVE

# Standing up and tearing down operations for each race


One of the benefits of operating in the cloud is that things are much more elastic, allowing you to stand up and tear down the infrastructure behind your APIs as needed. There are many business situations where this elasticity comes in very handy, but few compare with the situation Formula One faced. Ryan Kirk, Lead Cloud Architect and Team Manager at Formula One, shared the company's experience with me, showing how embracing the cloud radically changed the way the company does business.

For 15 years, Formula One shipped a data center around the world–stood it up, and tore it down for every race. Then they'd ship it to where it was needed next. When the company began exploring the cloud, executives initially had an "If it isn't broken, then don't fix it" attitude. But once they started playing around with the cloud they began to see the benefits of its elasticity and flexibility. The cloud provided a whole new way to support races, without continually standing up and tearing down the data center.

Another key ingredient in Formula One's cloud journey was the adoption of a DevOps culture. It was clear that the cloud provided new opportunities for automation and deployment of infrastructure, using repeatable CI/CD workflows across teams. Their teams wanted to automate repetitive tasks to become more productive. The cloud would also reduce friction as the company evolved DevOps to include DevSecOps–embedding security controls earlier in the software development cycle.The Formula One team was looking to abstract away procedures like changing out the API keys used in the CI/CD pipeline as part of deployment. These are things that developers don't care about, and with the cloud, they can easily be defined and automated.

The cloud changed how Formula One defined its virtual data center for races, making infrastructure, services, and self-service available so that teams could stand up and tear down what they needed, as needed. Teams began treating their infrastructure as code. That allowed them to achieve the scalability, redundancy, and geographic distributions the company needed.

Formula One teams can now provide the privacy and security required to support their high-profile events, while achieving the “speed” they need for standing up and supporting the capabilities they need to manage each race. Teams have more confidence in delivering infrastructure,, while automating as much as possible, gaining more time for innovation and optimization. Repeating these successful processes gives Formula One a digital muscle memory that makes the company faster, more agile, and more responsive–always ready for the twists and turns that lie ahead.

## 16.4 Deploying APIs Across Regions

Enterprises are increasingly deploying APIs across multiple geographic regions to move digital resources and capabilities closer to customers and comply with regulatory and data sovereignty requirements. As physical borders evolve to become digital borders, organizations have many new considerations as they expand their operations.

**Deployment** - API producers are increasingly deploying their APIs into multiple regions, and must ensure that gateways to backend systems meet customer and regulatory needs in specific geographies.

**Availability** - Some enterprises are only offering a portion of their digital resources, capabilities, and experiences to specific regions. They are designing and delivering APIs based upon what different regions need.

**Operations** - Organizations are zooming out from individual APIs and thinking about their overall operational needs. That could mean full deployment of APIs, or making only part of the catalog available to specific regions, depending on the broader scope of business needs.

**Revenue** - Invest in the research to understand the revenue opportunities you would gain by delivering APIs to a specific region. Work to understand the needs of target consumers, then provide an initial set of APIs to meet their needs.

**Governance** - It is important to build into your governance strategy some consideration of regional needs; that is, the f rules, policies, and other standardization you will need to support each individual region.

**Legal** - Make sure you understand the legal requirements of operating in a specific region. Examine local laws, regulations, and the legal details of engaging with consumers.

**Health** - Dashboards will help you understand the regional health of your APIs, while also observing health across the enterprise.

**Support** - Before expanding to new regions, make sure you can support those operations. That means, providing multilingual and localized content, documentation, and any other resources your consumers will need for success.

The World Wide Web is global, but in recent years network and data sovereignty concerns and nationalism have introduced more regulation that enterprises must consider when delivering APIs in specific regions. While regulation is a top concern, the main driver for regional expansion is pushing resources, capabilities, and experiences closer to consumers to better meet their needs.

**EXPERT PERSPECTIVE**

## Moving the gateway to the edge at Cloudflare


If you want to understand where API operations are headed in the next decade, I recommend tuning into what Cloudflare is up to–not just with their DNS offerings, but with their new API gateway solution. I recently sat down with Patrick Donahue, Vice President for Product at Cloudflare to discuss the company's strategy for the newly-released gateway. Cloudflare's approach, in contrast to that of incumbent API management and gateway providers, reveals where the world of APIs is headed in coming years.

Cloudflare has been pushing the DNS conversation forward for the last decade, but it has also invested in serverless. Now, the company is investing in an API gateway that is poised to shift the conversation about how we produce and consume APIs. Cloudflare operates in almost 300 cities in over a hundred countries, strengthening their DNS offerings, but also allowing them to push serverless computers to the edge. This investment has laid the foundation for their API gateway offering, which will push APIs to the edge across all of these regions, radically reshaping the API landscape and enterprise operations around the globe.

During our conversation, the Cloudflare API gateway was still in beta. The company was tapping into feedback from a handful of its most trusted customers, trying to determine their needs in terms of performance, regulatory compliance, and other factors. Imagine having a product feedback loop with customers who operate across the globe to learn what they need at the gateway layer. Cloudflare can now iterate and evolve the gateway to meet the needs of customers at scale everywhere,, essentially putting its finger on the pulse of the API economy as it evolves.

Cloudflare's unique view of the API landscape will translate into a better API gateway and will help the company develop better awareness of what is needed at the front lines. The company will also extend this awareness to customers through public API landscape reports.

It is safe to say that APIs–and more specifically, API gateways–have been commoditized since 2015. But what Cloudflare is doing bakes APIs and API gateways into the fabric of the web, changing its nature. While a significant portion of the web will continue to be allocated for human beings, increasing portions of it are being dedicated to web, mobile, device, and network applications and system-to-system communications. That will help all of us , automate more and extend our enterprise operations to a truly web scale.

## 16.5 Change Management

The API landscape is always changing. When you are API-first, this is a good thing. You have control over the rate of change and can effectively communicate changes to your consumers. Being API-first allows you to leverage change to your advantage, iterating upon your APIs as needed, not just to keep up with the pace of change, but to move it forward yourself.

**Life Cycle** - Without having a clear definition of the API life cycle, and without being able to discuss it with your teams using a shared vocabulary, change will become exponentially more difficult for you.

**Source of Truth** - You must always have a designated source of truth where code and artifacts are stored, evolved, synced, and forked, and where they are used across teams and contributors. Everyone must be working for the same truth.

**Contract-Driven** - Open source specifications like OpenAPI, AsyncAPI, JSON Schema, GraphQL, Protocol Buffers, and WSDL are used to govern the production and consumption of APIs, ensuring there are machine- and human-readable contracts in place.

**Versioning** - You need to apply a semantic, date-based, or other versioning strategy to API contracts and artifacts, helping communicate change as APIs are being produced to keep consumers up to speed.

**Source Control** - Use existing source control to manage change across API operations, making sure artifacts are available to show you what change looks like. That will minimize breaking changes, while keeping API contracts in alignment with deployments.

**Releases** - Provide organized and communicated releases for APIs, pausing for a moment to identify, consider, and discuss changes. Provide a clear milestone that can be shared between producers and consumers of each API.

**Provenance** - Without a record of the past, it is very difficult to know where you are going. That makes architectural decision records, change logs, and other provenance solutions a critical part of optimizing API velocity.

**Automation** - Contracts, versioning, and API release should always be as automated as possible. Use existing source control and CI/CD, but also be sure to leverage monitors for scheduling and orchestrating the way change is managed across all APIs.

Change is inevitable. APIs allow you to standardize and communicate change as it happens, helping API producers and consumers stay in alignment. As a result, consumers are more likely to keep pace with each version of APIs they are using.

Without a well-defined API life cycle, machine-readable API contracts, and a repeatable and automatable release process, seeing and managing change will be much more difficult.

EXPERT PERSPECTIVE

# Communicating and incentivizing change at Stripe


Without a doubt, managing change for your API-producing teams can be difficult. But shepherding your consumers through change is exponentially tougher. To learn how a leading API provider does it, I talked with Chris Traganos, Developer Advocacy Engineering Managerat Stripe. As you'd expect, Chris has a front row seat for understanding change in the payments industry. Stripe is very vocal and deliberate about its versioning strategy. When you first integrate with its API, you are basically "pinned" to that version until you deliberately request a new one. Stripe is very consumer-friendly–it doesn't force users to move forward with each version. It continues to support all of its versions and makes sure each new release is backwards compatible. It also communicates changes to consumers and shares its change management strategy. This approach reduces friction and has become extremely successful, but it does create a challenge: How do you incentivize API consumers to upgrade to a new version, where you know they could benefit from new capabilities if they'd just try it.

The problem is, API consumers won't always budge, even when you dangle new and exciting features in front of them. Why? Because they worry about the cost and time of changing their integrations. To deal with this reality, Stripe is investing heavily in open source client SDKs and other integration solutions. That will help consumers easily upgrade with each release without breaking their applications. It will also allow new consumers to quickly integrate Stripe's APIs in the languages of their choice.

By combining its easygoing approach to versioning with an investment in open source technology, Stripe will make keeping up with its latest innovations a frictionless experience for consumers. This company truly understands that your progress as an API producer is directly related to the progress of your API consumers. And investment in change management via APIs will help you increase your velocity.

# 16.6 API Portals

Internal, partner, and public API portals are now common across the enterprise landscape, having been introduced as part of an earlier SOA investment, or more recently, as part of an API management investment. Portals play a critical role in onboarding and engaging with users. But like other parts of the API life cycle, they are evolving, and often struggling to keep up with the changes happening across API operations.

**Internal** - Internal API portals exist as part of legacy investment in API management solutions. They often needmore work and engagement to ensure they are keeping pace with the rate of change.

**External** - Publicly available API portals are becoming a common fixture of enterprise public websites. They provide a public doorway to digital resources and capabilities available for use in business operations.

**Networks** - Networks provide an opportunity to connect your private, partner, and public APIs to a larger network of developers. That will help automate discovery not only of APIs, but of the mock servers, tests, and other aspects of API operations.

**Documentation** - Human-readable API documentation has become a staple of portals, providing HTML views of what is possible with APIs. Documentation describes how to use them, providing examples that demonstrate potential and speak to developers.

**Onboarding** - Portals are the doorway to API consumption. They show consumers what is possible and how to get started putting APIs to work as part of applications and integrations.

**Community** - One way to breathe more life into your API ecosystem is to build, attract, and cultivate community within your own API portal. You can also link g to other networks to engage with developers in the communities they frequent.

**Support** - Providing support to API consumers offers a rich opportunity for building trust and ensuring they successfully put your APIs to use.

**Discovery** - A well-maintained portal provides a rich opportunity for helping producers and consumers discover new APIs when they want to develop new APIs of their own, plan new applications and integrations, or put resources to work.

**Experience** - Portals aren't just about providing documentation and support. They are about delivering meaningful and enjoyable experiences for consumers and making digital resources and capabilities more accessible for developers.

Portals are an established aspect of API operations, and are making incremental shifts to keep up with the pace of change across operations. These shifts are helping to deliver the experiences today's consumers expect,, reducing their time to first call and making sure they always have access to the latest and greatest experiences.

# 17
# Roles, Discoverability, and Analytics

As APIs become more embedded in enterprise operations, their use is extending far beyond the IT department. Executives, product managers, marketers, operations and security specialists–heck, even lawyers–all have a stake in determining how APIs are designed and what they can do. And of course, so do your consumers.

But all these users are powerless unless you give them behind-the-scenes visibility into your APIs and the operations surrounding them. Without visibility, you're also missing out on valuable feedback. This chapter delves into the nitty-gritty about what observability really means and why it's so important, offering you tips at every turn to make your APIs more powerful by making them more discoverable.

## 17.1 Roles

A growing number of job roles have become involved in API operations, from defining their capabilities to distributing them and iterating upon the design to bring them into alignment with consumer needs and business objectives. For companies further along

in their API journey, APIs are no longer just an IT-led effort, but one that requires a mix of skills and cross-cutting roles to move APIs forward in a productive, reliable, and consistent manner.

**Product Manager** - APIs are increasingly being treated as products. Project managers keep the road map for APIs in alignment with consumer needs and the goals of business stakeholders, including those in sales and marketing.

**Software Architect** - Organizations are leaning on software architects to define the technical details of defining and designing APIs. They help establish common contracts for schema r that API developers can use.

**Developer** - Specialized backend developers focus on designing, developing, and delivering consistent and intuitive microservices and APIs that meet the needs of consumers and align with wider business objectives.

**Test Engineer** - The development of contract, performance, and other types of tests required to deliver high-quality, reliable APIs is increasingly implemented by test engineers. They are often part of a skills QA team that understands API nuances.

**Information Security** - The information security role goes beyond application security and other security teams.These people understand the specialized needs for securing APIs, whether they are applied as private, partner, or public applications and integrations.

**Release Management** - It's important to centralize the release of APIs, ensuring that they are deployed in a repeatable and consistent way and helping load-balance the journey from development to production across developer teams.

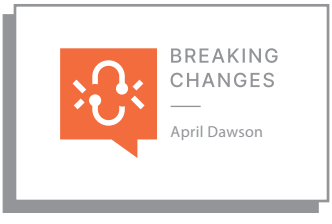
**Operations** - Once APIs are in production, SREs, DevOps, and other operations roles are stepping in to ensure they are observable and that the life cycles surrounding them are also observable across all relevant roles.

**Product Marketing** - After APIs are ready for consumption, product marketers step in to drive distribution. They make sure that APIs are discoverable and that consumers engage with each version.

While some of these roles may already be familiar to your team, it's essential to understand where they fit in the API life cycle and how they work in concert to consistently and reliably deliver new APIs and iterate upon them. You should maproles for each stop along the API life cycle, then educate stakeholders about the mix of responsibilities. That will help you establish clear ownership across teams, reducing friction at every stage of the life cycle.

**EXPERT PERSPECTIVE**

## API literacy for lawyers at North Carolina Central University




As APIs become ubiquitous behind the desktop, web, mobile, and device applications we depend on in our businesses, employees are gaining more exposure to them. That includes lawyers, as I learned in speaking with April Dawson, Associate Dean of Technology and Innovation and Professor of Law at North Carolina Central University (NCCU).

Through its Technology Law and Policy Center, the university is working to equip the next generation of lawyers to understand APIs and use them in their work. This isn't just about teaching APIs to the lawyers who will be working at the intersection of law and technology–it's about equipping every lawyer with an awareness of APIs. I have seen universities invest in an API curriculum for computer science students, but focusing on a specific sector of our society and giving all students a base-level understanding of APIs? That was new to me, and it definitely speaks to the wider API-first transformation happening across many business sectors.

Having an awareness of APIs will help lawyers more effectively access court-related information through legal APIs, such as UniCourt and the Free Law Project, but the ramifications go beyond that. Law graduates are increasingly working on municipal, county, state, and federal policies that regulate or influence how all of us use technology. In effect, that means April and NCCU are working to produce the next generation of technology policy makers. By better understanding APIs, these lawyers can help the country create more robust laws that can be applied in ways that are feasible and make sense.

What is happening at NCCU is just one example of how APIs are entering the mainstream. Most leading universities have some sort of API-related curriculum for technology students, but it could also help others succeed in their careers. The growth of low code/no code applications and other methods that do not require programming experience allow non-specialists to integrate, orchestrate, and automate APIs that can help guide their work and increase productivity. As a result, I expect to see rapid expansion of API education programs in the future.

# 17.2 Making Your Operations Discoverable

An API-first landscape is discoverable by default. API operations are known because APIs, their artifacts, and the operations around them can be viewed in workspaces and repositories that are kept in sync. Discoverability is about providing an ongoing snapshot of the state of API operations that is always indexed and searchable via private, partner, and public networks and the workspaces where API development occurs.

## A discoverable state

APIs are very digital and abstract, making them difficult to "see." Providing discoverability of APIs and the operations around them help prevent the APIs from operating in the shadows of applications.

**Latest** - Show the latest goings-n across operations, helping to surface interesting and relevant APIs and ongoing API work across domains and teams.

**Activity** - Show the configurations and changes are made across API operations, revealing what team members are doing in real time across workspaces.

**Search** - Provide a universal search for internal and external APIs that people can use to build applications, use in integrations, and develop automations.

**Browse** - Make API operations browsable by, workspace and other metadata, helping teams explore operations.

**Notifications** - Push notifications to team members and consumers, sending them useful information via native, email, and other channels.

**Suggested** - Provide suggestions for teams and consumers for relevant APIs, automations, and other resources and capabilities that might meet their needs.

**Feedback** - Allow stakeholders to provide feedback about API operations and how they are used in applications. That will give your teams ideas for possible enhancements.

API discovery must keep up with the pace of business, which means that everything must be indexed natively as part of the underlying API platform.

While each API has its own API discovery considerations, this blueprint is about discoverability across all your APIs It's important to make discovery a default state that doesn't require additional work by teams and is able to keep pace with change as it occurs.Expanding on which elements are discoverable

Making API operations more discoverable is the top challenge enterprises face today. Too many make API documentation the only discoverable element. For any kind of API–private, partner, or public–teams, tests, and essential building blocks of APIs operations should be discoverable by default. The elements below represent the points of collaboration that matter most to API-first transformation.

**Domains** - Make sure domains are discoverable, allowing business and technical leadership to find everything they need for single or multiple domains.

**Teams** - Ensure that the teams behind APIs are discoverable, allowing consumers and other stakeholders to see the humans behind the operation and support for each API.

**APIs** - API discovery means making available all the artifacts that define each API, allowing those with access to search for contracts and other machine-readable definitions.

**Documentation** - Requireall APIs to have up-to-date documentation that is indexed and has visibility matching the needs of targeted private or public API consumers.

**Mock Servers** - Provide a sandbox and specific business use case mock servers for your APIs, making it easy for users to onboard without having to authenticate or sign up.

**Tests** - Reliable APIs provide access to tests, including contract, performance, security, governance, and other types you run against an API, providing transparency and building trust.

**Environments** - Publish machine-readable environments for the development, sandbox, staging, and production phases, keeping configuration variables discoverable.

**Monitors** - Provide access to the monitors that are testing, automating, and orchestrating with APIs, helping to show how API operations works and making processes more transparent.

This indexing of API operations becomes the institutional memory of the enterprise, providing an ongoing map of operations. Digital indexes possess everything you need to demonstrate the value an enterprise produces each day. They represent the digital resources and capabilities an enterprise has at its disposal.

API discovery isn't just about helping consumers find the APIs they need It is about ensuring that your API operations are discoverable. It isn't enough to provide API documentation through your developer portal.You need to ensure that all APIs and their documentation, mock servers, tests, environments, and every other aspect of your operations can be found.

# 17.3 Analytics and Reporting

An API-first landscape needs to be transformed into a visible landscape, taking every output available across all APIs, the infrastructure behind them, and the governance overlaid on top of them and including it as part of the overall feedback loop within each domain. This work is about testing each instance of an API, its surface area, and the infrastructure used to deliver it, using collections as the universal observability connector that allows you to see across API operations.

**Domains** - Provide observability into the domain, rolling up teamwork, workspaces, APIs, and every other element of operations into simple dashboards and reporting. That will help business and technology leaders understand the state of API operations domain by domain.

**Teams** - Offer insight into how teams are working and performing, considering the technical and the human aspects of API operations. Then you can understand which teams are effective and which need more assistance across the API life cycle.

**APIs** - Make testing, security, governance, and consumption of APIs observable using dashboards and reporting solutions, allowing observation of individual APIs, as well as observation at scale of other dimensions important to business.

**Life Cycle** - Provide observability for the define, design, develop, testing, secure, deploy, and observe stages of the API life cycle, helping teams and leaders observe the digital API factory floor and understand the state of operations.

**Security** - Provide observability into the security across APIs, making efforts to shift security left in the API life cycle. This will also allow teams to see the scanning, authentication, encryption, and other aspects of securing API operations.

**Governance** - Automate the observability of governance, helping quantify how discoverable, reliable, and consistent APIs are across teams. You will also be able to observe how well your teams understand (or don't understand) the API life cycle and how it aligns with governance.

You can't control the direction of the enterprise unless you observe the state of operations at scale. APIs give you the opportunity to define the value generated across the enterprise each day and see what is needed to move in the right direction.

Your API platform should possess native reporting that helps you effectively "see" your API operations. However, you should also use your existing APM solution to provide the analytics and reporting you need to understand the state of your enterprise operations. Beyond your native and APM analytics and reporting, push your API life cycle vendors to provide you with the reporting capabilities you will need to understand each stop along your API life cycle. Better yet, push them to have APIs, so you can pipe results into your APM using collections.

# 18

# Improving Productivity, Quality, and Security

To ensure that your APIs are consistently high-quality and secure, you need to have the right tools and use them the right way. If you do that, you will also make your teams far more productive. This chapter will show you how.

## 18.1 Productivity

How do you achieve productivity in an API-first world? Through well-defined workspaces that possess everything you need to engage with an API throughout its well-defined life cycle. An API-first approach organizes the enterprise API factory floor into workspace beehives that contain the artifacts, documentation, mock servers, environments, monitors, and other building blocks teams will need to move the enterprise forward.

**Workspaces** - Everything you need to engage with APIs should be available via collaborative workspaces. The workspaces must contain e access controls to prevent undesired outcomes while ensuring that your APIs are discoverable, and so is everything else needed to sustain and evolve APIs, allowing for turnover of teams without any disruption to work.

**Collections** - The collection is a machine-readable, executable, and documented unit of work, providing everything you need to define a unit of business value in an API-first world. That includes definitions, documentation, mock servers, testing, and the automation you need to validate, scale, and empower your teams to do more with less.

**Life Cycle** - You must have a known life cycle that allows all contributors to deliver the best possible APIs in a short amount of time. That means standardizing how APIs are delivered to optimize productivity, and making sure developers have the education and training they need for navigating the life cycle, as well as getting teams on the same page across enterprise domains.

**Documentation** - Everything across the life cycle must be documented. That includes not just the reference documentation for your APIs, but onboarding docs, workflow docs, and your mocks, tests, and other automation. Documentation turns workspaces into institutional memory across the enterprise.

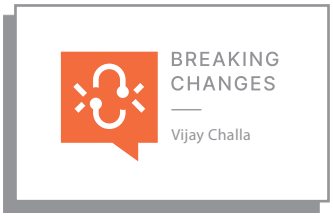
**Discovery** - All API operations must be discoverable, making teams, APIs, and operations available via search and discovery and ready for use across any stage of the API life cycle. Allow teams to find what they need when they need it, helping make API operations not only more discoverable but also capable of self-service, increasing team productivity.

Teams aren't productive in a chaotic environment where they can't find what they need, there is no common vocabulary for how things work, and they constantly lack documentation. Workspaces help ground the life cycle, and collections provide the atomic units of enterprise memory over time, helping teams move forward as they iterate products and increase the value your organization produces. Why do API operations tend to be so chaotic?

It's mainly because APIs and the operations surrounding them are often very abstract and technical. The more we can do to make API operations visible, discoverable, and well documented thin workspaces, the less chaotic things will be.

EXPERT PERSPECTIVE

# Boy Scouts of America does more with less by being API-first




When I saw the brainstorming session for a proposed episode of Breaking Changes with the Boy Scouts of America (BSA), I expected a simple story about how APIs power mobile applications at the non-profit organization. I was pleasantly surprised to learn that the BSA is actually API-first in everything they do. They depend on their ongoing API-first transformation to do more with fewer resources, while providing the best possible experience for membership and partners.

Vijay Challa, former CIO of the BSA, explained the importance of domain-driven design principles, which the organization applies to all of its API resources. Domain-driven design means they've put a lot of thought and planning into their schema and vocabulary for things like scouts, events, badging, and the other moving parts of what we all know as the Boy Scouts. Vijay's team understood that doing the upfront work around the design of domains would pay off during the design phase and other stages of their API life cycle, saving them time and money down the road.

The BSA employs an API contract-driven approach to defining, designing, and then delivering their APIs. They recognize the efficiencies they gain using a contract-driven approach, and how it helps with documentation, mocking, testing, security, and other stages of the API life cycle. API contracts allow the BSA to deliver more reliable and secure APIs across their internal and external applications. Their thorough understanding of APIs and the importance of a contract-driven approach has helped them with their ongoingAPI-first transformation, maintaining momentum with smaller teams and fewer resources. Teams are all on the same page, working as a collective group to move the organization forward.

Security and privacy are top priorities for the BSA. Because they have done such a good job of defining their API resources and capabilities, they are able to strike a balance between providing access and achieving the control they desire. Vijay described the organization's sophisticated role-based access

layer, which goes beyond defining who can access what to include a Know Your Developer (KYD) strategy, modeled after Know Your Customer (KYC) rules. This fine-grained control and awareness helps the BSA operate without friction across 50 states with many different partners and applications.

It is really interesting to see such a successful example of an API-first transformation and how it helps organizations operate more efficiently and do more with less. The BSA's story is all the more compelling because its work has such a meaningful social impact.

## 18.2 Improving API Quality

Achieving high API quality across enterprise operations requires individual sets of standardized tests that are applied across every digital resource and capability. Collection-driven testing allows for any type of test to be defined in a modular, shareable, and executable manner. API-first is about investing in a self-contained and documented set of tests for contracts, performance, integration, and any other variable you can imagine that can be used to automate quality assurance across operations.

**Testing** - Contract and performance testing should be present across as close to 100% of the organization as possible, but this is just the starting point. Tests for I integration, user acceptance, and other processes can be layered on from there. Make sure that all APIs and teams make testing a default part of how they work–no compromises.

**Security** - All APIs use encryption by default. Leverage standardized authentication and authorization and test all APIs against the OWASP Top 10 list of vulnerabilities, setting the baseline for security across nearly 100% of APIs across domains. Then add other security practices, setting a baseline for all teams to apply in their work.

**Governance** - A base seat of governance rules and policies must be in place. Translate governance into enablement early in the life cycle, applying rules and policies to shape behavior at the source control, CI/CD, and gateway layers of the API life cycle, helping both enforce the rules and enable teams.

**Runners** - All tests are self-contained as collections. They are documented and shareable, but also executable using local and cloud runners. That allows tests to be manually executed by not just QA and developers, but by other technical or business stakeholders, helping make sure that everyone contributes to quality and is held accountable.

**Pipelines** - Testing, security, and governance are baked into the CI/CD pipelines that keep production moving forward, establishing a common, well-known regime of collection-defined testing that is baked into the API build process. That will prevent unwanted behavior when API operations are in production.

**Monitoring** - All testing, security, and governance is modular and reusable, available for manually executing using runners and baked into the CI/CD pipeline. It is scheduled via monitors across the regions that matter to consumers, helping ensure that quality is automated and executed as a regular part of team operations.

**Observability** - Every test–including security audits and governance tests–publishes results into the enterprise APM, ensuring that every API and the operations behind it are observable both individually and , collectively, across teams and domains. Make sure the state of enterprise quality is observable to leaders through dashboards and reporting.

There is no single solution for maintaining quality across API operations. But if you make testing, security, and governance modular, machine-readable and executable via collaborative workspaces, you can more easily move towards a collective understanding of what quality means across teams. That understanding will then begin to manifest itself across the enterprise. Investment in testing elevates quality across the APIs behind enterprise applications and integrations, providing a more unified front and building trust with end users through reliability.

## 18.3 Shifting Left - Securing Your APIs Early

One of the common buzz phrases you hear in API security is “shifting left.“ It simply means you should be doing more to secure your APIs earlier in the API life cycle, rather than waiting until after you’ve deployed your API. If you look at the API life cycle as a linear left-to-right motion, shifting left equates to pushing things out earlier. Shifting left is essential for security, but you shouldn’t stop there–there are many other elements you might want to consider shifting left as well.

**Life Cycle** - It can be difficult to know exactly what "shift left" means if you don't have a shared definition of the API life cycle. Once you begin to nail down a vocabulary to describe it, shifting left becomes much more feasible.

**Testing** - Testing should not only occur after an API is up and running. More teams are finding it beneficial to begin crafting tests before an API has been deployed, using design-led or other earlier approaches to testing.

**Security** - It is feasible to move security to the earlier define and design stages of the life cycle. Shifting security left also helps teams develop more secure APIs before a security review is done.

**Governance** - It is not ideal to begin with governance by enforcing rules via the CI/CD system. Shifting governance to the define, design, and development stages of the API life cycle helps to deliver consistent APIs.

**IDE** - A developer's integrated development environment (IDE) is a great place to shift testing, security, and governance left in the API life cycle, providing teams with CLI, extensions, and other essential enablement tools.

**Reviews** - Design, quality, and security reviews provide an opportunity to shift processes left in the API life cycle, centering reviews around API workspaces, then making reviews self-service, automated affairs.

**Education** - The API life cycle provides a significant opportunity for making API education more modular and snackable. In addition to shifting left, you will make g API and life cycle literacy available at every stage of the process.

**Strategy** - While you're down in the weeds with each API at different stages of the life cycle, you have an opportunity to connect the dots of the tactical activities to where they fit into the wider domain, enterprise, or industry API strategy.

Shifting left is often accompanied by a "shield right" philosophy, which means that you have a feedback loop for handling things when they go wrong. For example, if there is a security breach, you have procedures for how to respond, address, and communicate information about the failure.

A balanced "shift left and shield right" approach acknowledges that you need to plan earlier in the API life cycle to address some of the common challenges of API operations, but you also need to respond, evolve, iterate, and grow based upon successes and failures.

# 18.4 Platform-Level Automation

It is impossible for humans to keep up with the pace of today's API operations. That means organizations need to make an ever-increasing investment in automating all the API operations behind our web, mobile, and device applications. Luckily, there is a very modular, collaborative, and executable way of automating API operations across teams. You can provide whatever unit of automation needed to define and execute anything across operations that can be done through an API.

**Collections** - Postman collections provide a modular, portable, and executable unit of value that can be applied across every stop along the API life cycle, defining individual API requests or workflows from multiple API requests that can be automated.

**Environments** - Machine-readable environments provide a well-planned set of variables that define development, staging, sandbox, and production environments, helping standardize automation so it can be applied across multiple potential environments.

**Runners** - Collection runners allow teams to run collections. They can see each step run, display results, and iterate through different business workflows. You can automate common tasks technical or business stakeholders can run.

**Pipelines** - CI/CD pipelines allow you to automate using APIs, beginning with testing, security, and governance as part of deployment and integration, and also providing any other automation workflow that needs to be accomplished upon a pipeline run.

**Newman** - Postman Newman provides an open-source command line runner that can run collections locally via the CLI, but also in CI/CD and cloud environments. It provides an engine for API automation that can run collections anytime in any environment.

**Monitors** - Collections can be automated on a schedule, running ongoing workflows across regions, different environments, and other configurations. Collection-driven automation helps you not only monitor, but execute business capabilities.

Automation is the only way we will keep up with the future. It is essential to shift automation left across API teams while still empowering consumers. Automation enables both API producers and consumers to do more with less and keep up with today's fast pace of business.

All API infrastructure should have APIs. API requests can be defined as very fine-grained or coarse-grained collections. The collections can be manually run with runners and scheduled via cloud monitors and CI/CD pipelines. Collections provide an unprecedented opportunity for defining all the knobs and levers you need to pull APIs forward throughout the API life cycle as executable and automatable units of value.

Every stage of the API life cycle should be automated, pushing well beyond test automation. The same kinds of automation you employ across the third-party APIs you depend on to do business, you can also apply to automating your own API operations. Automation is the only way we are going to be able to scale our businesses to meet the demands of a global digital marketplace. It allows teams to do more with fewer resources and focus on what they do best, automating the rest.

# 19
# Educating Your Teams

More teams outside of IT are working with APIs—but do they really know what they're doing? Unless you teach them, the answer is probably no. Without knowledge of sound practices for API design, production, and delivery, teams will throttle efficiency and impede API product quality. This chapter will show you how to provide them the thorough, ongoing training they need to do their best work.

## 19.1 Keeping Up with Training

If you want your teams to produce and consume APIs effectively, providing ongoing education is essential. You should offer as much in-person and virtual training as you can. Start by showing your teams how to document API operations, then teach them how to include guardrails and make operations as repeatable as possible to achieve maximum efficiency. Make sure your training covers all aspects of API operations, and keep it going to stay current as technology changes.

**Workspaces** - The foundation for any enterprise begins with example and starter workspaces. They demonstrate the most desirable approach to designing, developing, and operating APIs. They don't just tellg teams, but show them the ideal state, imprinting their memory with good models.

**Repositories** - Education about the API life cycle will have to be layered on top of your existing software development lifecycle. You can leverage repositories, README, and other common elements to introduce learning opportunities.

**Documentation** - The documentation available for each API allows you to educate consumers and other stakeholders, providing concise descriptions of what APIs are capable of doing.

**Guidelines** - Publish guidelines for all aspects of API operations, helping define the API life cycle. And don't forget to describe the details, tooling, and other resources available for each individual stop along the life cycle, making sure everything teams are expected to do is included.

**Blueprints** - Stabilize common practices, distilling processes into simple blueprints to outline essential concepts teams will need to be successful. Establish a common vocabulary and set of practices across teams.

**Workshops** - Conduct virtual and in-person workshops whenever possible, providing training for every part of API operations. Derive your curriculum from your wider API strategy, equipping different job roles with what they need to contribute to operations.

Ongoing education is vital for keeping teams up with the ever-changing technology landscape. It also ensures that new team members gain access to the enterprise knowledge they need.

Provide education across every stage of the API life cycle. Workspace overviews, repository READMEs, and documentation are the best places to start. This organic approach makes everything a learning moment, that requires everyone to be both a teacher and student.

EXPERT PERSPECTIVE

# Education as an ongoing part of API operations


API operations are about 25% technology, 25% business, and 50% people. That means education and training is a constant concern for leadership. After all, employees come and go, and unless you provide continuing education, you will soon be right back where you started. In talking with Sophie Rutard, former Head of APIs at Euler Hermes, I learned how critical educating teams is for the company as it strives to transform the insurance industry using APIs.

Euler Hermes is the world's leading provider of trade credit insurance, which ensures that business invoices are paid. Euler Hermes produces APIs that seamlessly integrate trade credit insurance into any business workflow or application. The company depends on the regular flow of API products to meet the needs of its partners, so it's critical for teams to know at all times how to design the API products needed.

To keep employees equipped to define, design, and deliver the consistent APIs that support the business, Euler Hermes employs a mix of approaches to educating teams.

First, the company has its most mature teams to define best practices and provide guidelines and rules for API design and governance. Then, architects and the governance group began identifying and working with teams who aren't proficient in defining, designing, and delivering consistent APIs. The centralized governance groups at Euler Hermes provide webinars and training to teach good API design practices, but they also make API education a regular part of the API life cycle. To incentivize teams to design better APIs, the governance group avoids policing guidelines at the CI/CD pipeline and instead, educates teams during design time and as part of design reviews. That has helped reduce friction for teams of all skill levels.

Euler Hermes provides a compelling example of how to make education a natural part of the API life cycle. The company provides training where teams are building their APIs, and also makes it part of the interactions teams have

with centralized architecture and governance groups in design reviews. This approach to API education is useful for any enterprise, but in heavily regulated industries, it's critical. With industry regulation baked into API design and delivery and sustained across the API life cycle, every team is equipped to do the right thing as they work.

It is common for enterprises to deliver API education before building APIs, then check to see whether teams are doing the right thing when the APIs are deployed via CI/CD pipelines. But as Euler Hermes shows, by far the best approach is to spread API education and awareness throughout the life cycle.

# 20
# Regulations and Privacy

Data privacy regulations are spreading like wildfire across the globe, and with each new incarnation they become stricter and more complex to integrate. But there's some very good news for companies that are ahead in their API-first transformation—for them, managing even highly detailed regulations like GDPR will be a snap. This chapter shows you why.

## 20.1 Compliance Regulations

Over the past decade, authorities have applied regulations to APIs for banking, healthcare, and other industries. Rules like GDPR and the California Consumer Privacy Act are shaping the way we do business on the web and mobile devices. It's essential to keep up with laws as they evolve, and APIs provide an excellent way of doing that. As regulations continue to proliferate, they will be a top priority for enterprises in the years to come.

**Privacy** - APIs are defining the way businesses respond to emerging privacy regulations. Companies further along in their API-first transformation are finding it easier to comply because they have a map of the data landscape and can quickly access and delete any data in question.

**Interoperability** - Standardized APIs for industries like finance and healthcare are leading the conversation about government-regulated interoperability. This is just the tip of the iceberg. Legislation is working its way through governments to target additional industries for regulation.

**Automation** - APIs are automating the notification and reporting of regulations.Government agencies are publishing APIs of compliance rules and allowing businesses to submit required data using APIs. This opens up a new realm of automated regulation and deregulation that can keep up with the pace of business today.

APIs are defining the enterprise digital landscape–and the way government agencies will regulate it. They are using the same approach enterprises use to understand and regulate businesses in coming years. Though there are other areas in which regulation and policy-making are shaping how we do or APIs (or decide not to do them), privacy, interoperability, and automation are the top three areas of compliance regulations defining how we do business with APIs.

## Your API-first transformation shapes how you see regulations

The state of your API-first transformation will define your experience with existing and future API regulation. If you do not have your API infrastructure well-defined and mapped out, it will be very difficult to respond to privacy rules and shift your operations to be automated and interoperable in response to regulations.

Of course, your enterprise shouldn't be doing APIs solely to comply with legal requirements. You should be doing them to better achieve your business objectives, which in turn will make regulatory compliance much easier. For example, you can deliver and manage APIs in regions around the world to bring resources, capabilities, and experiences closer to your customers, while also complying with regional regulations. You can map out all the schema you use across your operations to optimize how teams deliver digital resources–which in turn will allow you to quickly identify PII, PHI, and other sensitive data you will need to manage for regulatory compliance.

If you are further along in your API-first transformation, you will have a better understanding of how API and schema standards can help you reduce cost and improve business agility by being interoperable. All three of the areas of regulation covered in this book reflect practices enterprises already use to do business. Facilitating privacy, interoperability, and automation are things you should be doing already. Once you encounter regulatory obstacles, they become much easier to navigate and respond to if your API muscles are in place.

EXPERT PERSPECTIVE

# Product management to bridge business, technology, and compliance


The enterprise divide between IT and business groups has a long history. It has been common for business groups to throw requirements for applications or integrations with API features over the wall to IT for development. IT then cranks out some code and throws it back over the wall to be used by business groups and partners. In my Breaking Changes conversations, I am always looking for interesting stories about how this divide is bridged. and I was pleased to find a good example in talking with Antwain Daniels, Vice President of Product Management at Wells Fargo.

Antwain prides himself on being a bridge between business and technical groups. He perpetually works to create alignment not only between businesses and developers, but also among partners, regulators , and other stakeholders impacted by the bank's technology. Antwain reflects a new wave of product managers who have relationships with business teams and developers, understand the needs of partners, and critically, understand the importance of bringing compliance teams along for the ride. Collaborating with business stakeholders isn't something developers typically enjoy, but their work is much harder when they try to understand industry regulatory and compliance requirements on their own.

Antwain has the knowledge, expertise, and personality to connect the dots across overlapping parts of the enterprise and act as a human feedback loop. He believes it's a product manager's job to create, translate, cultivate, and aggregate feedback from business, developers, compliance, partners, and consumers–then distill all of that into the best requirements possible to help move products forward. To do this, the project manager must know all the details across all stakeholders and reduce any obstacles or friction along the way so that the organization can deliver the most useful digital products possible.

Though Antwain modestly says there are plenty of people like him out there, that's unfortunately not what I'm seeing when I talk with enterprise leaders. In

fact, product managers' inattention to the API life cycle–from setting APIs in motion to collecting feedback and creating common ground rules-–is the greatest deficiency I see across enterprise API teams. Without an API-first product manager like Antwain, teams are more likely to be out of alignment with the business, out of compliance with regulations, and very likely unable to deliver the features consumers are looking for in applications.

By bringing teams together, Antwain feels companies can eventually close the divide between business and developer groups, healing a rift that has plagued enterprises for years and bringing alignment and forward motion to the parts of operations that matter most.

# 20.2 Privacy

When your business runs on APIs, and your APIs are defined as contracts, you know where all of your PII, PCI, and PHI data is located. Modern privacy regulation focuses on giving consumers access and control over their personal information. APIs are how these privacy controls are defined and fulfilled.

## Privacy rules

There are two major sets of rules changing how we do business online. One comes from the European Union and the other from the state of California. Both are having an impact beyond their borders.


**GDPR** - The General Data Protection Regulation is a privacy and security rule. Though it was drafted and passed by the European Union (EU), it imposes obligations on organizations anywhere if they target or collect data related to people in the EU. The law gives consumers more control over their personal information and limits companies' ability to collect, store, and sell it.


**CCPA** - The California Consumer Privacy Act of 2018 (CCPA) gives consumers more control over the personal information businesses collect about them. CCPA regulations also provide guidance on how to implement the law. CCPA reflects the precedent set by GDPR, and like GDPR, it has been influencing policy around the globe, including other privacy regulations in the United States.

## Types of data to consider

There are two types of data at the center of privacy regulation. These types of information have the largest impact on how companies manage the personal details of users across operations.

**Personal Identifiable Information (PII)** - PII law covers any information pertaining to the identity of an individual, whether the data is directly provided or can o be reasonably inferred by indirect means.

**Payment Card Industry (PCI)** - The Payment Card Industry Data Security Standard is an information security standard for organizations that handle major credit cards.

## The role of API contracts

API contracts define and shape your API operations, but they can also help you understand and manage privacy across your operations and help you follow GDPR, CCPA, and other rules.

**JSON Schema** - This specification allows you to define all the digital objects you use across APIs and the applications they provide. It gives you the vocabulary for defining the objects that possess PHI, PII, and other sensitive data.

**OpenAPI** - This specification uses JSON Schema to define objects, then provides a machine-readable access map of your digital resources, including all of your synchronous APIs in use across enterprise operations.

**AsyncAPI** - This specification uses JSON Schema to define objects, then provides a machine-readable map of the various events that occur across operations and their asynchronous APIs behind them.

New regulations have set a precedent allowing r end users the right to access their data from any platform. APIs are how users will access their data and allow third-party developers to access it as well. APIs are essential to internet privacy and will continue to play a role in privacy regulations around the globe.

EXPERT PERSPECTIVE

# Know where your PII is by being API-first


Messaging provider Twilio, an API-first company, is a poster child for doing APIs well. The company has set the bar for producing APIs that don't just meet the needs of developers, but make it simple for them to put APIs to work in applications. Talking with Peter Shafton, former Vice President of Architecture and Research at Twilio, it didn't surprise me to learn that the company's API-first approach to platform management set it up for easy implementation of evolving privacy regulations.

Twilio works with SMS and telephony, so it already operates in a heavily regulated industry in the U.S. But due to the scope of business the company does in Europe, the introduction of GDPR became a top priority. f To help Twilio make sense of this new regulation, Peter was sent to Europe to attend GDPR workshops. In the early days of GDPR, few were sure what the impact of the ruling would be. Twilio made it Peter's job to understand the requirements and determine Twilio's response.

At the workshop, another company was asked how they would respond to a GDPR privacy request. The company representative said they would have to look through over 80 databases in a number of different groups to understand exactly where a user PII might exist, adding that the effort would take a significant amount of time and might not be successful. When Twilio was asked the same question, Peter said the company would simply make a series of API requests across several locations known to contain PII. When you are API-first, as Twilio is, you know where all your data lives. You can easily identify PII properties and you know how to query, access, and delete or archive them, satisfying regulatory privacy requests quickly without contacting multiple parties.

GDPR in the EU and CCPA in California represent a significant slice of the global economy. There will be two types of companies responding to these regulations. The first group will be unsure how many databases containing PII it has and will have a great deal of trouble finding them and implementing the

rules. The second group–companies like Twilio–will be able to quickly make a series of API requests to satisfy regulatory privacy requests. Which scheme sounds more appealing to you?

Privacy regulation is likely to become stricter, raising the API stakes along the way. If you are further along in your API-first transformation, you will view privacy regulation as a natural part of doing business, rather than a set of costly roadblocks.That's because being API-first elevates user privacy and security with or without privacy regulation. It simply makes good business sense.

# 21
# Interoperability and Automation

API-enabled interoperability and automation go hand-in-hand. Only by using both can enterprises, industries, and regulatory authorities keep up with the fast pace of today's markets. This chapter will show you a few of the ways you can use these capabilities to your advantage.

## 21.1 The Benefits of Interoperability

Regulation often aims to stabilize an industry and improve the flow of information. How? By driving interoperability among businesses in a particular industry–or among companies doing business with the government. APIs also reduce operating costs for businesses and incentivize a reusable, plug-and-play approach to delivering and operating software.

**Portals** - One important aspect of industry interoperability is having familiarity with the portal landing pages consumers use to onboard with APIs.

**Workspaces** - As we've stressed previously, it's important to provide a single location where people can find not just APIs and documentation, but visibility into the mocking, testing, monitoring, and other elements of APIs.

**Documentation** - You need to publish consistent, up-to-date, interactive, and simple documentation to inform consumers across many different providers.

**Onboarding** - The onboarding process across many different APIs plays an important role in ensuring interoperability not only of the technical interface, but the human interface.

**Contracts** - Contracts provide a common approach for defining technical details about the relationship between API producers and consumers across many disparate providers.

**Standards** - To further stabilize the experience across APIs, define the interoperability standards used in their design, development, and operation.

**Testing** - Make portable, documented, and executable contract tests available, ensuring that any stakeholder can validate whether an API is compliant.

**Certification** - Offer a formal certification program for API providers, applying industry standards for API certification.

**Change** - The constant change and evolution of APIs should be part of your plan.Establish a formal approach to versioning, communicating change, and updating the roadmap.

**Policies** - Establish common policies to shape identity and access management, source control, CI/CD, gateways, APMs, and other parts of enterprise API operations.

The business imperative for interoperability is becoming clear. It doesn't just assist with regulatory compliance, but brings you the agility and flexibility you need to compete aggressively in areas that matter the most to markets. You should never compete by offering proprietary interfaces to your digital resources, capabilities, and experiences. Instead, compete by offering unique experiences, while perpetually innovating and responding to markets. Interoperability will help you do this.

Providing interoperability across your increasingly sprawling enterprise landscape, as well as across the growing number of partners you depend on, should be the natural state of your enterprise. You shouldn't choose to adopt industry API and schema standards because the government tells you to, but to benefit your own business by lowering the overhead for onboarding new partners, teams, and digital opportunities as they emerge.

EXPERT PERSPECTIVE

# Standardized APIs required to do business with the government


The European Union's regulations for the payments industry–PSD2, and now PSD3 –have dominated conversations about API regulation for the last five years. Now, in addition to this financial industry specification, a new specification for healthcare has evolved: the Fast Healthcare Interoperability Resources (FHIR). On Breaking Changes, I sat down with Kelly Taylor, former Director of Digital Services for the State of Colorado, to learn about the impact this API specification I is having on healthcare interoperability at the state and federal levels.

Kelly began this journey at the federal level, pushing the specification forward as part of the Blue Button API at the Centers for Medicare & Medicaid Services (CMS). FHIR later became the interoperability standard for accessing the healthcare records of the 6.2 million people receiving Medicare or Medicaid. In 2021, CMS and the Department of Health and Human Services (HHS) began issuing a series of rules that essentially told healthcare providers that if they wanted to do business with CMS, they would need to have a FHIR-compliant API to ensure interoperability. This set of healthcare rules was not only the first set of healthcare API regulations, it became a precedent for API regulation across industries in the U.S.

I have been working with Kelly since his time at CMS, but for this Breaking Changes episode, I was most interested in the impact of the CMS regulation at the state level–specifically in Colorado. Kelly shared that the state government and healthcare providers were working together to determine how compliance would work. They soon realized there was a lot more required of APIs than just adhering to a common standard. In addition to securely managing patient healthcare information, healthcare providers needed to understand how to publish documentation, provide onboarding for consumers, and support the needs of developers who would be putting FHIR-compliant APIs to work in applications and system-to-system integrations.

The CMS API rules are still being rolled out, and the federal agency has relaxed some requirements in light of the Covid pandemic, so there isn't much data yet on how states are faring with compliance. Still this healthcare API regulation provides an intriguing glimpse of r how future API regulation might look across other industries. As federal and state governments and public and private healthcare providers work together to make sense of API regulation, the rest of us can tune in to learn about the positives and negatives of this approach to mandating interoperability and common standards.

## 21.2 Keeping Up with Automation

As regulation increases, the need to automate notification, reporting, and evolution of policies and rules will increasingly be done via APIs. We see regulators not just requiring APIs and API standards as part of regulation, but making APIs available as part of the regulatory process itself.

**Policies** - Businesses are increasingly learning about their regulatory obligations via web and mobile applications, and APIs can provide self-service access to regulatory policies and rules. w APIs also make the policy notification process event-driven and closer to real-time, keeping pace with modern digital markets.

**Reporting** - APIs are already becoming commonplace for automating regulatory reporting. Their low-cost web technology helps the government keep up with the pace of industry change and the growth and scale of enterprises, bringing more observability to industry regulation.

**Responses** - Automation is creating an API-driven feedback loop that can inform future decisions about regulation or deregulation. By leveraging an API-first approach, authorities can find the optimal regulatory balance for specific industries, ensuring that rules are a force for positive market change and do not harm businesses unnecessarily.

API automation is the only way t enterprises can keep up in today's markets. It is also the only way for governments and industries to regulate markets and ensure an optimal regulatory balance.

To comply with incoming API rules, we must rely on API automation to be notified, report, and respond to regulations of any shape and size. Only then can the dance between government and the industries it regulates proceed at the scale necessary to keep up with today's pace of business. A couple of examples: In the U.S., you can visit the Regulations.gov API to learn about regulatory policy. You can submit workspace safety compliance reports through the Occupational Safety and Health Administration (OSHA) portal. API automation is both how governments will regulate industries and how industries will respond to regulation–making regulation a natural part of the API economy.

## Powering de-regulation with APIs

So far we have discussed the role of APIs in industry regulation, but they can also play a part in deregulation. By helping authorities understand the impact of their rules, APIs may identify areas where getting out of the way would be helpful.

Regulation is needed to strike the right balance in markets, and APIs are increasingly becoming a natural part of how markets work. In addition to seeing API-first transformation at the enterprise level, we are now beginning to see it at the industry level. Transformation at this scale is exactly what we need to move markets into the digital age.

# Your API Operations: The 30,000-Foot View

The blueprints and narratives we just explored in this operational section should give you a robust set of building blocks for retooling your enterprise factory floor and supply chain to support your digital transformation. Your physical office and factory floor is easy to define, but your digital office and factory floor is more abstract. It will take a significant amount of work for you to "see" it, and even more work to establish a shared understanding across your organization and with partners and external stakeholders. The goal of this book is to provide you with a solid foundation for bringing your API factory and digital supply chain into focus across your teams.

You are doubtlessly already producing and consuming APIs–you just aren't doing it at the operational level. You and haven't yet stitched together the industrial-grade platform you will need to build the future. You have source control, CI/CD, gateway, and APM solutions in place. Now you need to make the life cycle across all of your APIs more visible, tangible, and discoverable–by using workspaces. This will become the base of your digital factory floor, which you can use to organize your teams by domain and organize your digital resources, capabilities, and experiences across private, team, partner, and public workspaces. All of this work will bring order and consistency across the API life cycle, for both producers and consumers.

Once you have established a base foundation for your API operations– with hundreds or thousands of workspaces available to manage all of the APIs you produce and consume–you can get to work optimizing and automating your API operations by using APIs. You see, your workspaces, APIs, documentation, mock servers, testing, environments, and monitors all have APIs behind them.That allows you to define, automate, and orchestrate across your entire organization. Automating the API lifecycle is the only way you can keep up with the pace of business today. The problem is, you can't automate anything unless it is well-defined. Having a consistent definition of the API life cycle is fast becoming essential for growing and scaling your operations.

So take the blueprints and supporting elements from this section and use them to get started. Begin by identifying the elements you already have in place, then pick one or two additional capabilities to invest in. Keep going until you have made your operations visible and shareable across teams, providing a common awareness of how APIs can be produced, consumed, and evolved over time. With that shared understanding, business and technology leaders will gain a much better grasp of enterprise operations, enabling them to steer the business in the right direction in a time of accelerating change.

# Closing Thoughts

Describing today's complex API landscape is no easy task. My goal with this book was to give you three doors for approaching it: the strategic, the technological, and the operational, with accompanying blueprints and stories from enterprise leaders to help you get your bearings and plot your own course to a successful API-first transformation.

I wish I could join you for every step in your API journey. I truly do. The best part of my job is the conversations I have with Postman customers and guests on Breaking Changes. Because I learn so much from these discussions, I try to spend 50% of my week talking with enterprise representatives. They have helped me more than any book or codebase could ever do to understand in depth the realities of doing APIs at scale.

The book has been through several iterations and has become a living document for me and my team. And the process isn't over yet—not by any means.

At any point along your journey, I invite you to email me or participate in the Github repository and workspaces we are using to power this ongoing work forward. Your feedback is very important—and we may be able to offer additional insights that didn't make it into the book. You can be sure we will incorporate your responses and observations to make the next version even better.

We want to get it right because APIs—and the applications and automations they enable—are the most important pieces of technology in the world today. APIs represent the relationship between producers and consumers, the intersection of business and technology, and the ability of governments to translate policy into action. They are the building blocks we will use to create innovations, and the manuals that will help us make sense of all the fast-moving changes around us.

I hope this book has helped you see the big picture more clearly and incorporate the tools that matter most to your organization. Our collective journey has only just begun. I am eager to learn about your personal journey as you make your way through the API ecosystem. So dive in—and please share your experience.

# The API-First Transformation

API-first companies create innovative products and build them much faster than those starting from scratch. This book gives business and technology leaders all the blueprints and building blocks they need to achieve their own API-first transformation—accompanied by inspiring stories from 30 top-brand executives to light the way.