# Intro to APIs: What Is an API?

Kin Lane
October 5, 2020· 14 mins

## What exactly is an API?

If you want to learn about APIs, you've come to the right place! API stands for application programming interface. APIs are the little pieces of code that make it possible for digital devices, software applications, and data servers to talk with each other, and they're the essential backbone of so many services we now rely on.

Digging deeper, an easy way to understand the definition of an API is to think about the applications that you use every day. In an internet-connected world, web and mobile applications are designed for humans to use, while APIs are designed for other digital systems and applications to use. Websites and APIs both do the same things, like return data, content, images, video, and other information. But APIs don't return all the details that are needed to make things look pretty for the human eye—you only get the raw data and other machine-readable information needed behind the scenes to put the resources being delivered to work, with very little assistance from a human.

### What is API integration?

"API integration" is a pretty common Google Search term, and we have good news. The whole reason APIs exist is to support integration. API integration is simply the connection between two (or more) applications, programs, services, or systems, using APIs. Applications use APIs to send and receive data and content between each other. Keep reading for a history of APIs, what they're used for, examples, and more.

## History of APIs

Web APIs got their start by putting the "commercial" in ".com," powering commerce startups looking to change the way we do business on the web. They took advantage of this new medium to make products and services available to customers via a single website, and as they worked with partners, they sought to automate much of the commerce that was powering the web and included juggernauts like Salesforce, eBay, and Amazon. In 2004, a shift in the API landscape began to emerge as a new breed of API providers started to pop up, offering ways to share information with local and global social networks, led by the likes of Facebook and Twitter. With a strong start in commercial and social applications, APIs continued to grow as everything moved to the cloud, became much more mobile, and provided the foundation for next-generation devices. To get the full scoop on the history of APIs, check out this Postman blog post.

## What are APIs used for?

What are APIs used for? Lots and lots and lots of things, including:

- APIs power desktop applications.
- APIs are behind most web applications.
- APIs make mobile applications possible.
- APIs are the integrations for no code solutions.
- APIs connect devices to the internet.
- APIs define the networks—or the information passed between applications, systems, and devices.
- APIs even connect everyday things like automobiles, doorbells, dishwashers, and wearable devices.

Read more about what APIs are used for.

## Why should you care about APIs?

Curious about why you should care about APIs? Here’s a very brief list:

- APIs help you access the data you need to get your work done and do daily tasks—whether you’re a business user, a student, or using an application just for fun.
- APIs make it possible to integrate different systems together, like Customer Relationship Management systems, databases, or even school learning management systems.
- APIs help different departments, teams, and groups become more agile.
- APIs help organizations, schools, government agencies, and nonprofits strengthen relationships with other organizations, research institutes, and agencies.

Need more reasons? Keep reading about why you should care about APIs.

## How do APIs work?

APIs work by sharing data and information between applications, systems, and devices—making it possible for these things to talk with each other.

Sometimes the easiest way to think about APIs is to think about a metaphor, and a common scenario that a lot of folks use is that of the customer, a waiter, and a restaurant kitchen: A customer talks to the waiter and tells the waiter what she wants. The waiter takes down the order and communicates it to the kitchen. The kitchen does their work, creating the food, and then the waiter delivers the order back to the customer.

In this metaphor, a customer is like a user, who tells the waiter what she wants. The waiter is like an API, receiving the customer's order and translating the order into easy-to-follow instructions that the kitchen then uses to fulfill that order—often following a specific set of codes, or input, that the kitchen easily recognizes. The kitchen is like a server that does the work of creating the order in the manner the customer wants it, hopefully! When the food is ready, the waiter picks up the order and delivers it to the customer.

## Why use APIs? Reasons to use APIs

What are the reasons to use APIs? Almost too many to count, but to truly answer this question, we asked thousands of developers and professionals why they choose to produce or consume APIs in our annual State of the API report. Here are their top answers:

- *Integration with internal and external systems:* One of the top reasons developers use APIs is to integrate one system with another system. For example, if you want your Customer Relationship Management (CRM) system to integrate with your Marketing Automation system, you could use an API to enable the two systems to talk to one another so that you automatically send a marketing email when a sales representative adds a new prospective customer to the CRM.
- *Adding or enhancing functionality of internal and external systems:* Another common reason that developers use APIs is to add or enhance functionality of internal systems as well as external systems. For example, if you have an internal system that tracks vacation days for managers, you might use an API to allow employees to request days off from their email.
- *Adding or enhancing functionality for customers:* Sometimes when you add or enhance functionality, it's about improving customers' experiences and helping them interact better with your organization. For example, if you worked for a food delivery company, you might use an API to automatically notify customers when their meal is approaching their house.
- *Speeding up software and system development:* APIs allow developers to code and deliver functionality as microservices, instead of big, monolithic applications. By breaking this functionality up, developers can actually speed up software development and system development by eliminating dependencies and reducing the overhead involved in code reviews, testing, and more. Another way that APIs help accelerate development is by allowing frontend and backend teams work in parallel. In other words, a frontend developer can work on creating the frontend of a system, what users and customers see, while a backend developer can work on the underlying system, or what users and customers do not see.
- *Reducing operating costs:* Another reason developers use APIs is to reduce operating costs. APIs can help perform a number of functions that may have been done by humans from pulling reports to sending emails to abstracting data from one system to share with another system. APIs can help reduce operating costs in many, many other ways. Examples include, automatically starting up and shutting down manufacturing systems, creating schedules for workers, or even reducing the number of people who need software licenses.

- *Reducing software development costs:* One of the biggest ways APIs can reduce software development costs is by allowing developers to build reusable components. For example, a backend developer can create a system that serves up information about customers, including their names, email addresses, recent product purchases, etc. Then other developers across the organization can use APIs to grab that information and track payments for finance and accounts payable, help customer service resolve problems faster, or even create recommendations for marketing campaigns. APIs can also help reduce software development costs in a number of other ways such as reducing architectural complexity and reducing the effort required to make changes to systems.
- *Improving software and system testing:* APIs can help improve software and system testing by allowing quality engineering teams to separate tests for frontend components, the parts of software that users see, from tests for backend components, the parts of software that users don't see. API health, quality, and performance can also be checked using automated testing, and API testing can be integrated into the CI / CD pipeline.
- *Improving organizational security and governance:* APIs can be used to improve organizational security. For example, APIs are often used to power Single Sign-On, which is the ability for users to use one username and password to login into multiple systems. This helps avoid the dreaded pile of sticky notes with usernames and passwords, which can be a big security risk. APIs are also often a big part of corporate governance, too. APIs can be used to enforce and automate corporate rules and policies like requiring approval before expenses are paid to employees.
- *Enabling mobile applications:* A lot of mobile applications rely on APIs to deliver important information to mobile users. For example, if you use your mobile phone to check in for a flight and select your seat, APIs can communicate the seat you selected so that flight attendants know where you are seated when you get onboard.
- *Reducing outages and non-performing systems:* Finally, APIs can help reduce outages and non-performing systems. For example, a company might use an API to quickly identify a specific problem with a manufacturing line, and even recommend a fix, which helps maintenance staff fix the system and get back online faster.

## The different kinds of APIs

There are many different types of APIs, and many different ways to categorize them. Here are some of the most common.

#### Internal vs. External vs. Partner APIs

One way to categorize APIs is by who has access to them:

- Internal APIs are APIs that are private and only used by your team, department, company, or organization.
- External APIs, also known as public APIs or open APIs (which is not to be confused with OpenAPI), are publicly available APIs that are available for anyone to use.
- Partner APIs are private and shared only with specific, integration partners outside of your organization.

**API architectural styles**

When it comes to API architecture, there are a number of styles—some newer, some older—and all have a place in the API ecosystem. Defining “architectural styles” broadly, here is a list of the most popular styles listed in order of how frequently they’re used:

- *REST API:* REST is an acronym for REpresentational State Transfer. REST APIs rely on a few guiding principles such as a client-server structure, simple, uniform interfaces to communicate across systems, stateless operations, and more.
- *Webhooks:* Webhooks are event-based, and simply put, are automated messages sent from one system to another system anytime an event occurs. Webhooks are even referred to ‘reverse APIs’ as a concept to check for changes in data.
- *SOAP API*: SOAP is an acronym for Simple Object Access Protocol. SOAP APIs are more structured and formalized than other APIs, they are reliable and trusted, but can be slower than other APIs. SOAP APIs uses an XML-based messaging protocol which includes the Envelope, Header, and Body tags as required by the endpoint.
- *GraphQL API:* GraphQL is an acronym for Graph Query Language. The Graph Query Language defines how one API asks another API for information and instead of relying on how the server defines the endpoint, a GraphQL query can ask for a specific piece of information. GraphQL was originally created by Facebook as an internal tool in 2012 but they publically released it in 2015 as an open-source language for APIs.
- *WebSocket API:* WebSocket APIs rely on the WebSocket computer communications protocol, which is a full-duplex communication channel over a single TCP connection. Compare WebSocket protocol to HTTP (HyperText Transfer Protocol), which is a half-duplex communication. WebSocket APIs provide a standard way for servers to send information and data to clients, even when the client is not requesting data. WebSocket APIs also allow data to be communicated between clients and servers, while keeping connections open.
- *gRPC API:* The RPC in gRPC stands for Remote Procedure Call; gRPC APIs were originated by Google. In gRPC, a client can call on a server just like it is a local object, making it easier for distributed applications and systems to communicate with one another.
- *Server-sent event API:* Server-Sent Events, also known as SSE, is a technology that relies on data being pushed from the server. This allows a client to receive updates automatically via a HTTP connection.
- *AMQP API:* AMQP is an abbreviation for Advanced Message Queuing Protocol. AMQP is a protocol that follows open standards, and works at the application layer. AMQP is best suited for message-oriented middleware, and like other protocols, AMQP dictates how messaging providers and clients communicate with each other. (Hint: message is in the name!) There are a few features that distinguish AMQP including its ability to queue and route messages, which support the reliability and security of AMQP.
- *MQTT APIs:* MQTT is an abbreviation for Message Queuing Telemetry Transport. The MQTT messaging protocol is defined by Organization for the Advancement of Structured Information Standards, better known as OASIS. MQTT is well-suited for the Internet of Things (IoT), in part because it is extremely lightweight. MQTT allows devices to publish and/or subscribe to messages.

- *EDI:* EDI is an abbreviation for Electronic Data Interchange, and it's been around for a long time, since the '70s! The idea behind EDI is to allow businesses to communicate electronically with each, typically transmitting information that was written on paper, like receipts or invoices that an accounts payable might send out, or order information such as purchase orders.

## Challenges of producing APIs and consuming APIs

When it comes to building APIs, there can be a number of challenges and obstacles that developers and teams face. Here, too, we sought insights from thousands of developers and professionals across the API industry to see what they think about the most common challenges:

- *Lack of time:* The number one obstacle for API producers, as well as an obstacle for many API consumers, is lack of time. Building and consuming APIs can be a time-intensive effort depending on what you want APIs to do or how you use them.
- *Lack of documentation:* The number one obstacle for API consumers is documentation. API documentation refers to the information API providers provide to API consumers to help them understand how to use APIs.
- *Lack of knowledge:* Another obstacle for many producers and consumers is lack of knowledge. In some cases individuals working with APIs need deeper technical knowledge, while in other cases individuals are technically adept, but lack knowledge about the features of specific APIs.
- *Lack of people:* Lack of time is closely related to lack of time and knowledge. Many teams find it challenging to produce and consume APIs with their existing team infrastructure, and need additional people on their team to both expand their technical knowledge base as well as expand the amount of time individuals can spend on API efforts.
- *Complexity:* While the concept of APIs is relatively simple, the implementation of APIs can be extremely complex—many individuals who both consume and product APIs find complexity to be a major obstacle.
- *Stakeholder prioritization:* Stakeholders, or individuals who have a "stake" in projects and efforts, are often the ones who control the resources, staff, and funds that are allocated to projects. If stakeholders do not prioritize API projects, that can be an obstacle to consumers and producers alike.
- *Lack of budget:* For most teams, it takes a budget to produce and consume APIs. Sometimes that budget is spent on staff and training, sometimes on tooling, and some APIs require funds to use the API on a regular basis.
- *Stakeholder expectations (unrealistic/unclear):* We just mentioned stakeholders, or individuals who have a "stake" in projects and efforts. Often these stakeholders don't have technical backgrounds, and are not clear about what they're trying to achieve with APIs, or even worse, unrealistic about what it takes!
- *Leadership buy-in:* It can be really hard for teams to work on something that organization leadership does believe in, or "buy-in" to. When leaders don't buy-in to APIs, projects don't get prioritized and they fall by the wayside.

- *Lack of tools:* Very few developers open up a blank notepad and begin coding. There are a myriad of tools available to developers to help them code smarter, faster, and better. But, not every developer has access to the tools they need, so it becomes an obstacle.
- *Team buy-in:* Like leadership buy-in, it can be hard to move projects forward if the teams themselves—the ones actually building and using APIs—don't believe in the project, making forward progress unlikely, or even impossible.

## Real-world examples of APIs

Looking for real-world examples of APIs? Go no further than our directory of APIs, better known as the Postman API Network. The Postman API Network provides a central place for both API consumers and API producers to easily discover, explore, and share APIs.

You can find APIs from many popular providers in the Postman API Network, including the following:

- Twitter APIs:  Twitter APIs allow you to do many different things like looking up specific users, looking up specific tweets, searching for tweets, filtering real-time streams of tweets and much more.
- Imgur APIs: Imgur APIs expose the entire Imgur infrastructure. In fact, using Imgur's API, you can do just about anything you can do on imgur.com, like finding and sharing the funniest, most informative and inspiring images, memes, GIFs.
- Okta APIs: Okta offers a wide range of APIs, including endpoints to authenticate your users, challenge for factors, and recover passwords. Okta also offers ndpoints to configure resources such as users, apps, sessions, and factors.
- SurveyMonkey APIs: SurveyMonkey APIs make it possible to explore SurveyMonkey data, including survey questions, survey responses, contacts, benchmark, errors and more.
- Yelp APIs: Yelp's Fusion API allows you to get the best local business information and user reviews of more than a million businesses in 32 countries.

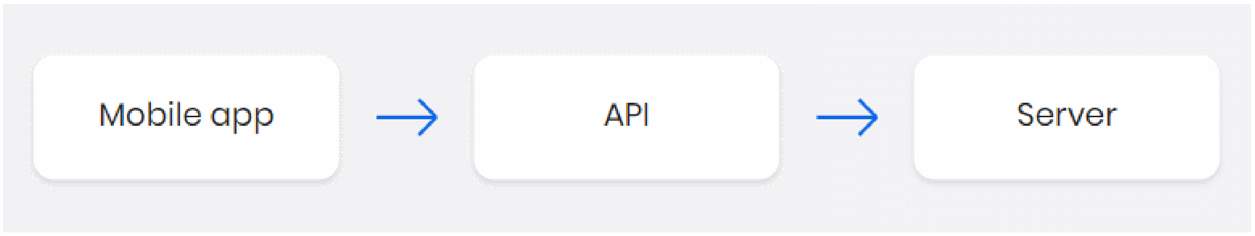
# How do APIs work?

Andrew Park  Tray.io
API Integration

## What are APIs and how do they work?

Anyone who works with business software has heard terms like "API" and "webhooks" thrown around. In this article, we'll not only cover what APIs are, but we'll also answer the question: "How do APIs work?" Understanding the function of APIs is the key to understanding how they can help business users in any role accomplish more, faster - without necessarily having to learn coding.

API tools have fundamentally transformed how developers write applications. They have introduced an entirely new vertical of "platform as a service" software companies. API-based tools are the reason why data integrations between essential business software are possible. In fact, API integrations have become essential to operations and revenue professionals.



API stands for "application programming interface." An API is essentially a set of rules that dictate how two machines talk to each other. Some examples of API-based interactions include a cloud application communicating with a server, servers pinging each other, or applications interacting with an operating system. Whenever you use an app on your phone or computer or log onto Twitter or Facebook, you're interacting with several different APIs behind the scenes. Nearly all businesses that use any kind of modern technology use APIs at some level to retrieve data or interact with a database for customers to use.

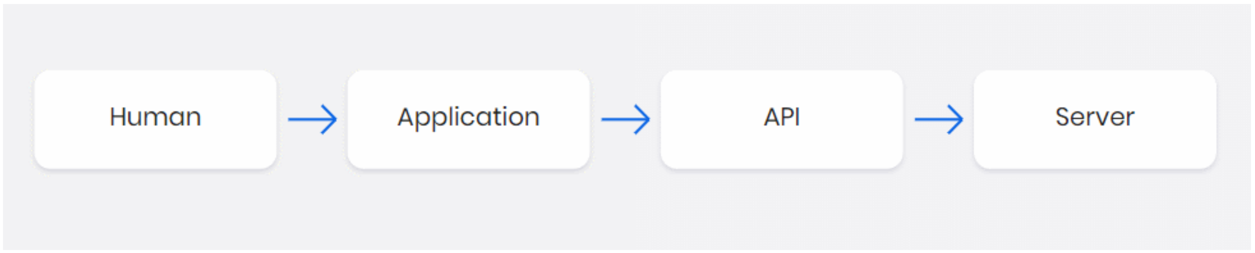
An API's defined communication protocol is what enables developers to build, connect, and integrate applications quickly and at scale. Consider, as an example, Jeff Bezos' famously-issued 2002 mandate. Amazon's change of direction shows how APIs helped it move faster than its competitors, and is reportedly the reason Amazon is so successful. Bezos ordered all of his teams to communicate and expose data and functionality through service interfaces, that is, APIs. Once the APIs and infrastructure were in place, Amazon's teams were able to operate much more efficiently. Launching this new infrastructure enabled the creation of Amazon Web Services, which has since become Amazon's largest revenue driver.

Some businesses don't just employ internal APIs, which their engineers use to build features for their consumers. Many companies also use external APIs, which the developer community uses to launch products. Some examples include Twilio (communications API), Stripe (payments API), and Sendgrid (email API), which offer a "Platform as a Service" (PaaS) model. Such companies enable developers to build applications on their platform, which might perform functions such as hosting web servers or communication applications. There are businesses whose main value comes from connecting different APIs and web services, which are categorized as "Integration Platform as a Service." IPaaS companies let users connect disparate web services and tools, most notably to route data or automate workflows. Both of these verticals have grown tremendously, fueled by the extensibility and ease of use of APIs.

APIs have played a critical role in shaping modern business by enabling automation. But how do APIs work? And more importantly, how can you use them to connect the services that you rely on to route data or automate critical workflows?

## How do APIs work? And how do they help you do more every day?

An application programming interface is a set of rules that define how computers, applications, or machines can talk to each other. You can think of it this way: the typical user interface is intended for use by a human being, while APIs are intended for use by an application or computer.



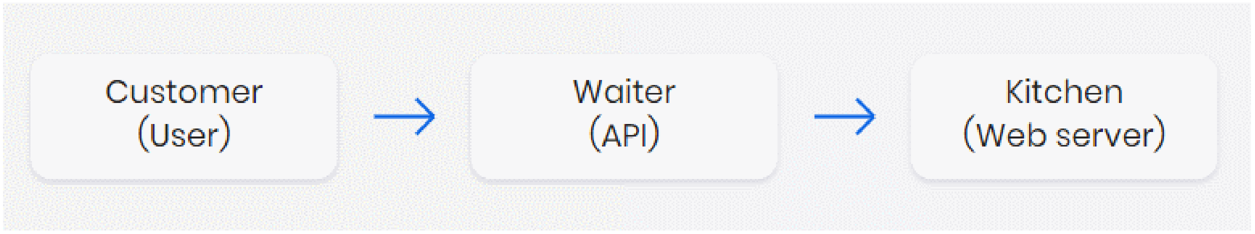
Most web APIs sit between the application and the web server. The user initiates an API call that tells the application to do something, then the application will use an API to ask the web server to do something. The API is the middleman between the application and the web server, and the API call is the request. And every time you use software to communicate with other software or online web servers, you're using APIs to request the information you need.

It's important to note that while web APIs are the most common, APIs aren't limited to the web. There are APIs for virtually every machine or system that expects to interact with other machines or systems.

## How do APIs work? An abstraction

Let's use a metaphor to explain how an API works.

Imagine you're a customer at a restaurant. The waiter (the API) functions as an intermediary between customers like you (the user) and the kitchen (web server). You tell the waiter your order (API call), and the waiter requests it from the kitchen. Finally, the waiter will provide you with what you ordered.



The waiter is the intermediary between you and the kitchen. In this metaphor, the waiter is effectively an **abstraction** of the API. In software engineering, an abstraction is a fundamental concept that aims to simplify a complex mechanism by only focusing on the details of higher importance. In this example, as a customer, you don't need to know how the kitchen or the restaurant operates in order to get what you want: the food. You just need to know how to order it.

An API is also an abstraction of the web server. The application (such as a website or a mobile app) will make an API call for a set of data to display for the end user to consume. The request is made via the API that accesses the web server to retrieve the requested data, which is populated in the user interface.

Note how abstractions are evident at every "level" of the web application. The application doesn't need to know how the web server works, just how to use the APIs to get the data it needs to display. The end user doesn't need to know how the APIs work, just how to navigate the user interface to perform the tasks she needs to perform.

In most API requests, there are often a few key pieces of information:

- Type of request
- Authorization credentials (to prevent mimicked API calls from any attacks on the server)

## Types of APIs

The most common discussion you'll hear about APIs tends to focus on web technologies, but APIs aren't limited to web services. Here's a roundup of common (and not so common) APIs.

***REST APIs*** If you've heard people talk about JSON (javascript object notation), chances are they're talking about REST APIs. Over 70% of all public APIs use REST, because of its fast performance, reliability, and ability to scale by reusing modular components without affecting the system as a whole.

***How do REST APIs work?*** REST, or "representational state transfer," is a type of software design that gives access to data (aka "web resources") by using a uniform and predefined set of operations. The **payload** - the data to be delivered - defined in the request itself, will be formatted in a language such as HTML, JSON, or XML. The set of operations are the methods available to HTTP, which is the underlying protocol for how browsers retrieve websites from servers. These methods include GET, POST, PUT, DELETE, and others.

There are four parts of a REST API request:

- URI (uniform resource identifier, which is the URL address, also known as an "endpoint")
- HTTP method (most commonly either GET or POST)
- Headers (which include authentication tokens, define the data format of the response, impose rate limits, and perform other administrative tasks)

- Body (the actual part of the request)

***SOAP APIs*** SOAP, or “Simple Object Access Protocol,” is a bit more complex than REST because it requires more information upfront about security and how it sends messages. These additional standards require more overhead, and as a result, SOAP - an API standard that has been around since the late 1990s - tends to lack the lightweight portability and flexibility of REST.

***Browser APIs*** A browser is capable of creating a wide variety of user experiences, such as playing music, displaying intricate animations, and reacting to mouse or keyboard input. A browser gives control of these experiences to web developers via browser APIs using javascript to manipulate the HTML or create unique experiences within a browser instance.

***iOS/Android APIs*** Similar to browser APIs, each mobile platform has its own set of APIs that provides developers the tools to build experiences for their end users. App developers can use these APIs to transmit data to the device’s hardware, use a sensor from the hardware such as a camera, play music or video, or perform many other capabilities.

### What is the API economy?

Since APIs act as an abstraction for developers, they play a crucial role in the scalable nature of software development. Any developer who knows how to access a REST API, for example, is now immediately capable of integrating payments (Stripe), SMS/VoIP (Twilio), and email (Sendgrid) into their applications. APIs turn complex processes, such as payments, into a few lines of code, fundamentally amplifying developer productivity. What once took developers a few weeks to build can now be done within a few minutes thanks to APIs.

The **API economy** is driven by a new class of businesses who provide public access to their APIs (also known colloquially as “devtools,” shorthand for developer tools), such as Twilio, Sendgrid, and Stripe. These devtool API providers aim to drive developer productivity by tying together complex functionalities with a REST API. Tying together functionalities with APIs has componentized and modularized most common operations you’d expect from web or mobile apps. This means that thanks to APIs, developers can easily build formerly complicated operations such as payments, mapping, transportation, and ride-sharing directly into their apps. As more API providers enter the marketplace, the API economy is trending towards applications that are built mostly with APIs.

## Single-handedly driving modern business innovation: the webhook

Due to the proliferation of APIs, modern businesses now rely heavily on them for data and web automation. For example, many revenue and customer-facing teams use tools (which are built on APIs) for lead-to-account matching between a CRM like Salesforce and a marketing automation tool like Marketo. Savvy SaaS vendors, aware of their responsibility within the bigger picture of their customers’ tech stacks, enable data exports via an API endpoint to help their customers run analysis across all of their data sources to get a complete and holistic view of their customer journeys.

Prior to the API economy, it was common for businesses to stick to a single vendor's product suite for a CRM, marketing automation, and other apps on the assumption that it would be easier to pass data among them. Thanks to APIs, teams can instead use a "best-of-breed" approach to selecting vendors that empower their teams. By taking advantage of the way APIs enable software apps to talk to other software apps, teams can use the best CRM suited to their needs, the best marketing automation platform for their needs, and so on.

But how does the data sync between tools work? If we peel back the technologies in an integration between cloud tools, we'll find that they're all built upon another REST API concept: the webhook. It's a simple yet effective method to deliver or "push" data in real-time.

In fact, when it comes to accessing data between tools, in the absence of a dedicated integration, most vendors will offer a webhook. Consider this scenario: your team wants custom real-time status notifications on performance data from a different tool or simply wants to export data out of that tool (usually one row or record per webhook). You can enable the tool to deliver an API call (usually a JSON payload) to a URL endpoint that you provide.

In order to parse the inbound JSON payload, your endpoint should be on a server that you have access to or another tool that has specifically built a webhook ingestion endpoint. Once you have received the webhook, you can transform this data and save it to a data repository of your choosing (a popular choice of data repository is a data warehouse like Redshift for analysis), or if the webhook is a notification, you can programmatically trigger any action you desire: such as sending an email or a text or creating a calendar event. The flexibility and simplicity of webhooks has helped them become the basis of modern web automation. Learn more about webhooks and real-world automation examples.

## What are the benefits of APIs?

APIs are all over the web, and are therefore common in modern business. Due to their ease of use, there's been a huge increase in API usage among platform and infrastructure businesses. On top of that, APIs enable users to integrate apps like Salesforce, Eloqua, and Marketo to better integrate their lead routing. It's important for revenue teams to flow lead data between their marketing platform and CRM, and for customer support teams to flow data between their helpdesk and payment processing system to manage renewals, upsells, and churn.

The best platforms for working with APIs make use of their power and flexibility by letting users port over data from custom fields, which is a real pain point, even for software with out-of-the-box integrations. For example, Marketo and Salesforce offer native integrations, but since every sales organization designates their CRM fields differently, this integration typically isn't up to the task of keeping up with every custom field. Modern API-based tools, such as the Tray Platform, give users the ability and flexibility to map custom fields between different apps, and even flow data directly between them via API calls. Providing this power saves a ton of time and prevents manual errors, while freeing up users to focus on more-important strategic concerns, such as how to grow customer engagement or win more sales deals.

## Takeaways

APIs have been a game-changer for modern software. The rise of the API economy not only enables software companies to rapidly build in key functionality that might previously have taken months or years of coding to implement, but it has also enabled end users to connect their best-of-breed apps and flow data freely among them via API calls.

Modern companies make the most of their cloud apps' APIs to rapidly and automatically deploy mission-critical data across their tech stack, even from custom fields that don't work with out-of-the-box integrations. FICO used API integrations and automation to grow engagement for its marketing campaigns by double digits. AdRoll used automated API calls among its revenue stack to increase sales meetings 13%.