Future of APIs

Evolve from pipes to data-aware APIs that automatically keep data consistent, correct and up to date.

Vendia

Table of contents

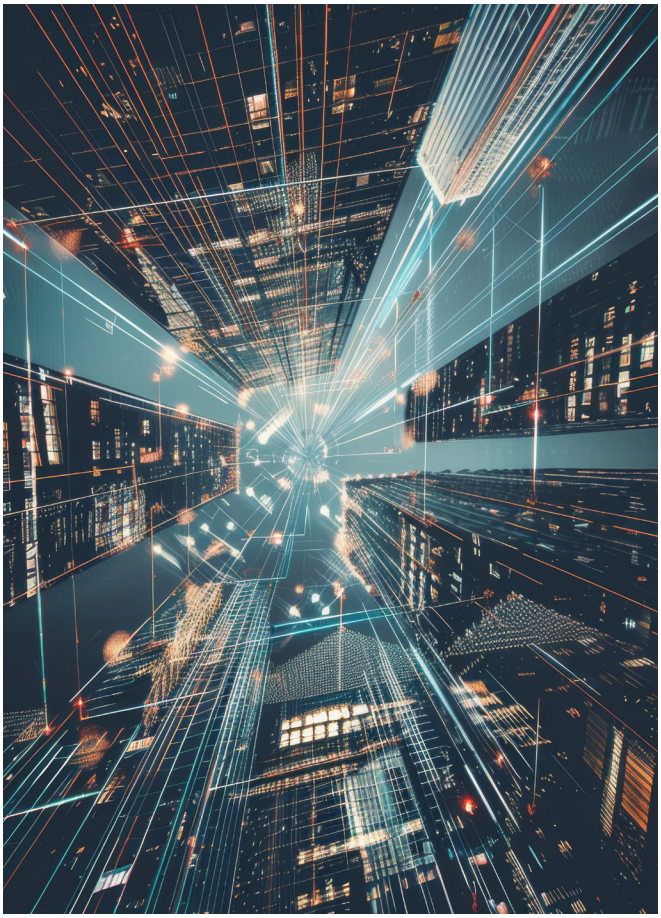
Executive Summary

APIs are the workhorses of modern IT solutions and applications. As businesses increasingly move to SaaS models of consumption and production, APIs are also becoming the “public surface area” of every company in the financial, technology, commerce, media, logistics, and other industries.

Understanding the challenge of building modern, secure, and scalable APIs and how to address the needs of API consumers, especially the challenging nature of data API consistency and fidelity, is essential for CIOs and cloud architects looking to build modern, robust business solutions. Managed solutions as well as powerful platforms for data APIs like Vendia and Fivetran help address these challenges and offer “out-of-the-box” capabilities that can greatly reduce the cost, risk, and staffing conventionally needed to create world-class business APIs.

Introduction

APIs are ubiquitous in our industry. From payment APIs like Stripe and PayPal to communication services like Twilio to the ever-growing list of AWS services, it seems that every company and concept has an associated API. And yet, despite their popularity and the abundance of tools and services to create, operate, and monitor them, developers and businesses still struggle with the high cost of creating APIs. In this article we take a deeper look at what an API really is, why they exist, why carrying business data between companies remains challenging and expensive in 2022, and what business and technical leaders can do to overcome those challenges.



What is an API, really?

APIs are abstraction boundaries between disparate systems – two modules in a software architecture, two regions in a multi-region application deployment, or two companies that need to share business data. Over the years they've grown from a simple way to isolate the details between two systems to being full-fledged security firewalls, and more. APIs, especially "public" APIs that form a company's externally visible surface area, are usually responsible for many jobs:

- **Expressing the API owner's data model**
  The "shape" of information that an API can accept and/or deliver
- **Expressing the API owner's workflow model**
  What has to be done, in what order, for jobs or steps to be accomplished successfully when a sequence of API calls is required over time
- **Data model and workflow evolution**
  As business needs evolve, handling the modeling of those inevitable changes by versioning the API (and potentially its clients)
- **Security, encryption, identity, authentication, and authorization**
  Given that anyone could attempt to use the API, which people or machines are actually permitted to use it, and, when their permissions are not homogeneous, deciding who gets what level of privilege (and how you know they are who they say they are)
- **Operational firewalls**
  Defending against DDOS and other forms of attacks, applying (potentially client-specific) throttles and quotas
- **Monitoring and analytics**
  Using calls to the API (and the identity of the callers) for both operational integrity and business analytics, so that the API owner has a clear picture of who's doing what on their systems

That's a lot of work for any portion of a company's architecture to perform, but we're still not done: APIs also have to worry about the content of what they carry. Let's peer into an API's payload next to see why this can be an even bigger challenge than the list above.

API payloads

While aspects like security are obviously critical, APIs are only useful to the degree that they can carry information between two or more parties. Let's examine the ways in which APIs are used as a way to understand why the content carried by the API can be even more challenging than developing and operating the API itself.

While there are as many uses for APIs as there are companies and developers, we can generally group them into four categories:

1

Control APIs

Control APIs are "verbs"; they allow one party to tell another to "do something". Control APIs can be chained together into more complex workflows, for example creating a shopping cart, filling it with items, and then eventually checking out to purchase the content of the cart. Webhooks and other "callback" mechanisms allow the control relationship to be reversed, so that an API owner can also tell a client to perform some action.

2

Data APIs

Data APIs are the workhorses of the API universe. Their job is to get two or more parties to achieve a shared understanding of a set of facts. For example, if you're an ISV or SaaS vendor, you may use Stripe's APIs to align your internal metering systems with the charges and subscriptions Stripe uses to bill your customers. Stripe's APIs (and your webhook API that Stripe calls) are continuously trying to ensure that your database of usage and Stripe's database of customer charges are in agreement by shipping data back and forth.

3

Metadata APIs

These are APIs that ask or answer questions about other APIs. They may be used to learn information about a supported schema, retrieve documentation, gather usage data or other analytics, add or alter identity or permission information, etc.

4

Special purpose APIs

Uncommon APIs that handle high bandwidth media, process high frequency trades over optic fiber, implement low-level 5G protocols over proprietary networks, etc. These APIs are far less frequently encountered by IT departments and most developers and are outside the scope of this article.

API payloads (cont.)

Of the previous categories, data APIs tend to be the hardest to build and maintain, for several reasons:

- **High volumes / low latency** – Data APIs payloads tend to be larger (because they model complex business or application data) and more frequent than either control or metadata API call rates. This means data APIs need to support higher throughput rates and typically need to be lower latency (so they don't fall behind a steady stream of inbound requests). Control and metadata APIs usually have more relaxed requirements and lower demands on the systems on both sides of the API.
- **Consistency and fidelity** – For a control API, the API call itself often is the "system of record". This makes tracking and auditing relatively simple and straightforward. Metadata APIs tend to carry information that either changes rarely or that changes frequently but is advisory in nature, such as getting a "mostly correct" summary of recent usage over the last hour. But data APIs have to be right. For example, if you're an airline sharing passenger list information with a partner airline on which apassenger is continuing, you can't get the name of the passenger wrong, or accidentally include a passenger from a previous flight, or share private information about other passengers that shouldn't be conveyed. Any kind of error, including under- or over-sharing, could be devastating to a business.
- **Data drift** – Control APIs are usually called whenever the client has a need for something to happen. Metadata APIs are usually advisory in nature. But data APIs carry payloads that are both business critical and which can become out of date as soon as they "hit the wire". That makes one-way APIs especially brittle: clients need to repeatedly, and quickly, poll them to catch data that may be drifting unpredictably at any moment, and which could affect mission-critical outcomes.

Data APIs are a public reflection of a company's internal database, without the benefits of a database's capabilities. Often, it's not even just one database, but several, that need to be reflected in the APIs payloads. And to make the challenge even greater, it's usually more than just a read-only relationship with the API's clients: Often the clients themselves are implicitly adding or changing data in those database(s). In the Stripe example above, the data APIs have the difficult job of trying to keep two databases – Stripe's internal payment representation and the client's metering representation – in perfect alignment, without the opportunity to ever see or edit each other's content directly. Data APIs have all the problems of a database, without the benefit of any of its traditional solutions (ACID transactions, write ahead logs, etc.)

Given that data APIs are the most common and pervasive type of APIs, that they have the largest and more frequent payloads among APIs, and the many difficulties imposed by trying to keep those payloads consistent, correct, and up to date among API owners and their clients, it's not surprising that companies of all shapes and sizes wrestle with the challenge of developing, operating, and maintaining APIs that meet both their needs and the needs of their customers and partners. In the next section we take a look at tools and services that can help reduce this burden.

Managing payloads

APIs are some of the most critically important – and challenging to build – elements of a modern IT stack or application. Fortunately, great tools and services exist to help with some of the undifferentiated heavy lifting.

Managed API services, including Amazon API Gateway, Azure API Management, and Google Apigee provide hosted solutions that handle many of the basic challenges, including implementing APIs from data models, handling infrastructure deployments and scaling, dealing with basic forms of authentication, and some of the governance and lifecycle aspects of API evolution. Outside of API owners with very specialized needs – such as building a cross-continent real-time streaming media service or a high-speed stock trading network – these managed solutions remove many of the burdens of self-deployed solutions. By offering many built-in capabilities needed for production use, such as encryption on the wire and compliance, they make it easy for even resource strapped teams to create world-class API hosting outcomes with relatively little time or ongoing oversight.

While they help significantly with the undifferentiated elements of hosting and operating the basics of an API, managed API services don't do anything to assist with the payloads themselves. For data APIs, this can be a significant limitation, as the challenge of keeping data among multiple parties consistent, complete, and correct at all times is arguably an even bigger burden than deploying and managing API-related infrastructure.

Companies like Vendia and Fivetran focus on going further than conventional managed API services, by addressing the challenges of payload management directly:

For real-time operational data

For real-time operational data, Vendia offers schema-driven solutions to create data APIs that act like databases, providing not only the benefits of managed API services but also handling the larger challenge of keeping data correct, complete, and up to date among all parties with which it's shared.

Vendia also provides built-in compliance, privacy, and governance mechanisms that make it easy to decide (and then audit) who shared what with whom, and when. Vendia is a SaaS offering, so API owners don't need to worry about deploying, scaling, or managing the underlying cloud infrastructure powering their APIs.

For analytics and BI data

For analytics and BI data, Fivetran offers an easily configurable, fully managed ETL solution for getting data from popular SaaS platforms, such as Salesforce, as well as operational data stores into analytics solutions such as Snowflake. Fivetran eliminates the need to create "intermediary" APIs manually, moving data directly from systems of record into a company's chosen analytics or BI platform without the conventional need to deploy or manage ETL infrastructure.

Evolving to smart APIs

With the rise of SaaS companies, APIs are the business, and neither business owners nor developers can skip the hard job of ensuring that they're secure, compliant, and —especially when they carry mission-critical business data— correct. As described above, the mechanical task of deploying and hosting APIs has gotten easier over time with the rise of cloud-based API services, but the job of "wiring up" API payloads to their underlying systems of record to create a shared source of truth remains a difficult task for most companies and developers. Platforms like Fivetran, for analytics ETL, and Vendia, for real-time operational data, help tackle this challenge and lower the TCO of keeping critical business data consistent across companies, clouds, regions, and tech stacks.

But what about the "correctness" of data. In the next section we take a look at a different, smarter approach to APIs, where evolve from "dumb pipes" to APIs that automatically keep data consistent, correct, and up to date.

APIs: a tradition of "dumb pipes"

APIs are critical to virtually all modern applications. From internal APIs shared between teams or departments to the surface area of global enterprises, APIs represent the public "contracts" of modern software systems and IT solutions. API protocols have also matured over the years, from Electronic Data Interchange (EDI) to HTTP to GraphQL, as both businesses and developers have evolved their demands on the data that APIs need to carry. The infrastructure for implementing APIs has advanced as well, from "rolling your own" to deploying open source libraries (such as NGINX), to fully managed cloud services such as Amazon API Gateway and Azure API Management. But for all these advancements, the conceptual role of an API – that of a "dumb pipe" that doesn't understand, control, or mediate its content – has remained largely unchanged: APIs are all "syntax", no "semantics". To put it another way, an API doesn't care if the data it carries (and carrying data is largely what APIs do) is right or wrong, consistent or inconsistent, complete or partial, up to date or radically out of date. And even if the data you drop into one side is perfect, by the time it reaches someone's system on the other side it may no longer be.

As a result, a ton of developer time (and company dollars) go into trying to close what we like to call the API data gap. And that gap is huge: Sharing data, and keeping it consistent across companies, clouds, geographies (aka regions), accounts, and technology stacks is arguably the single hardest challenge in modern day applications. Most IT challenges ultimately come down to solving this problem – storing (your own) data isn't particularly difficult in 2022, and neither is defining or standing up an API. But ensuring that the parties that API connects are seeing the right data at the right times (and not seeing data they shouldn't see)? That remains a hard problem, one that consumes untold amounts of people, distributed systems infrastructure, and monetary resources, even with all the advances that have been made over the last 20 years in API technology.

Can we do better?

Is this a fundamental problem, forced on us by the "physics" of software? Or is it an accident of design, reflecting an outdated methodology that hasn't caught up with better alternatives? To explore the answer, it's useful to look at a different technology, one not normally associated with APIs: blockchains and other types of distributed ledgers.

Why blockchains? Because, leaving the details of API syntax aside, they are all about moving data around while maintaining a uniformly consistent, correct, and up-to-date representation among all the various parties...even if those parties are mutually distrustful of each other. (If crypto-style trust issues don't sound relevant, just substitute "operationally and security isolated" or "employing enterprise-grade zero-trust architecture" for the term "mutual distrust" in the preceding sentence.) If you strip away topics like staking and on-chain incentive mechanisms and just focus on the distributed ledger aspects of blockchains, they appear to offer something that APIs are missing: A simple way to move data across these gaps (companies, clouds, organizations, regions, etc.) while keeping it consistent everywhere it lives. In other words, they're smart about the data they carry, unlike an API. Could distributed ledgers be the answer to filling the API data gap?

Because we get carried away thinking the problem is solved, let's acknowledge that blockchains, at least conventional ones, don't solve the kinds of challenges modern API solutions handle: There's no GraphQL (or HTTP) definition mechanisms, no fully managed or SaaS-style hosting offering at the scale of an AWS, Azure, or GCP service, no fault-tolerant and automatic scaling mechanism, no easy cloud or event integration pattern, and on and on and on. You'd certainly be forgiven for thinking that – data consistency aside – blockchains are very far from being a solution to building a smart API.

But what if we could put these two technologies together? Imagine a system that combined the very best of modern API hosting solutions with all the power of a distributed ledger to keep data securely consistent, correct, and up to date, regardless of where or how it was shared and with whom? Such a system could close the API data gap, but to do so it would need to meet a lot of demanding requirements:

1. High-speed, cross-cloud / cross-region / cross-account data sharing, with automatic cost and latency routing optimization ... all with zero application coding or operational cost to span any of these elements
2. Full ACID semantics and support for user-specified transactions (groupings of updates into atomic changes)
3. Automated batching and concurrency optimizations that require no application coding
4. Automatic caching and storage optimization with intelligent tiering and heat management
5. Decentralized implementation, where each party (or operationally and security-isolated region) has its own, synchronized copy of the data without the need for a single, centralized agency holding all the data in escrow

Can we do better? (cont.)

6. Built-in access controls and ownership models, so that the owner of a piece of data, no matter how large or small, can control (and audit) who sees it and when
7. Full support for versioning and lineage of all data that also obviates the need for separate application logging by providing a fully queryable audit trail for all changes
8. Support for a broad variety of data types, from simple key/value pairs to highly structured business data to multi-terabyte files
9. Easy, config-driven integration with other cloud services, including high-speed streaming, queueing, event hubs, cloud functions, etc.
10. GraphQL-based APIs, generated automatically from a rich, standards-based data model, such as a JSON schema and supporting dynamic introspection for clients
11. Built-in support for easy data model evolution as business needs change, with static (compiler-provided) checks for backwards compatibility to guarantee clients won't break across model updates
12. Robust role-based access control and authentication/authorization model that provide each participant with the ability to model its local administrator and user profiles and control client access to any portion of the data model, without reliance on (or visibility into) the configuration of any other participant or need for any "global settings"
13. SaaS-style (i.e. fully managed), cloud-based deployment model that requires no third party code deployments or server/container/Kubernetes management.
14. "Serverless"-style, meaning scaling and fault tolerance are handled automatically by the system rather than through operational burdens on the application or its developers/operators
15. Next-level fault tolerance, including the ability to survive cloud service provider outages, even for entire regions (such as the dreaded AWS "us-east-1 is done" scenarios)
16. Ability to add participants in minutes without needing to write code, deploy software, or perform operational incantations

The list above might sound like a pipe dream (pun very much intended), but it truly reflects what Vendia has to offer. We incorporated all our experience building managed API solutions in the cloud at AWS and all our insights gleaned from Coinbase and other distributed blockchain companies to create a solution that offers the best of all worlds: a truly smart API. Unlike existing blockchain solutions, Vendia has all the power of modern, managed GraphQL APIs. Unlike existing API solutions, it's not "dumb" about the data it carries – it automatically keeps that data consistent and correct, without any effort on the part of the developer.

How Vendia implements smart APIs

One of the key realizations that led to the patented technologies underpinning Vendia is that the work of data distribution over APIs – while technically challenging – is utterly undifferentiated. Unless you're Netflix or YouTube, 99.9% of the data shared internally or externally through APIs looks like either "database values" or "files". A solution that embraces both (and solves all the hard challenges in the list above) can thus offer a broad, industry- and application-agnostic API solution for individual developers looking for an easy way to stand up an API all the way to a globe-spanning Fortune 10 company that needs the highest level of enterprise-grade API and data platforming.

The "API part" is built using cloud-grade components that scale automatically with request traffic. This serverless design helps ensure two things: Tight cost enveloping (because the marginal cost of owning and operating the system is zero when its not in use, whether that's for a second or a year) as well as the ability to handle massive amounts of traffic (through implementations that are highly multi-tenanted at the hardware level). We call this architectural pattern STAMTI: Single-tenanted accounts over multi-tenanted infrastructure.

On top of this, Vendia layers a revolutionary cloud-based consensus architecture capable of keeping data fully consistent across different clouds, companies, regions, and technology stacks. Because it understands the data, the consensus algorithm can make intelligent decisions about everything from how to route traffic between clouds to minimize cost implementations to how to batch data without violating ordering guarantees or consistency semantics. While those algorithms are quite complex (among other things, they involve carrying out on-the-fly commutativity and associativity proofs), they are agnostic to the specifics of the data, meaning that they can work equally well for two airlines sharing routing information for a traveler to two different departments sharing employee account data to a media company sharing an updated version of a movie with a theater.

To make the system easy to use, Vendia includes a custom compiler that turns data models (e.g., in the form of a JSON schema) into public cloud deployments automatically. Developers don't need to understand the dozens of AWS, Azure, and GCP services required to make all this work … they only need to know their data model and then connect to modern, easy-to-use GraphQL APIs over HTTPS. The compiler (and SaaS-style deployment infrastructure) handle the hard job of converting data models to database and API configurations, including complex tasks like managing indices and access controls.

Sharing data is a challenge, but ensuring that the right data is shared with the right people is even more critical. Laid directly into the consensus and replication layers of the platform is full support for security, on-the-fly and at-rest encryption, and a powerful access control and rights-based identity management solution that ensures that every bit of data can be secured and access controlled at all times. The platform can even model changing access over time, to express concepts like sharing version 4 and 6 of a file with a third party while not providing them access to version 5.

Under the covers, Vendia provides an enterprise-grade self-monitoring implementation wired directly into its consensus engine. This way, if a cloud service provider has an outage, such a failure of the AWS us-east-1 region, other parties can continue sharing data and the system can "self heal" any copies in us-east-1 when the region recovers. Users can even invite new business partners or add other clouds or regions at any time … without writing any code. Because it understands the data, and the data's lineage, Vendia can automatically deploy new regions, backfill their local storage, and add them to the consistency maintenance fabric automatically. Development and deployment challenges like going multi-region or multi-cloud, which can take hundreds of developers and multiple years to accomplish using Kubernetes or other low-levelapproaches, can now be reduced to a couple lines of configuration for Vendia-based applications.

What can you build with Vendia?

The short answer is, “If you can build an API for it today, you can probably build it simpler, easier, and with far lower operating and staffing costs using Vendia.” We give you a simple way to model the bread and butter of applications – so-called CRUD APIs – that handle the process of creating, reading, updating, and deleting both scalar and file data. Like a cloud-based database, it can handle high speed storage of scalar items, varying levels of read consistency retrievals, and offers the full power of ACID transaction grouping despite supporting multi-party and multi-cloud consensus. (In other words, you don’t have to give up all the benefits of strong consistency and transactions when you need to share data across regions, clouds, or with other parties, keeping your application easy to create and prove correct.)

Because Vendia uses GraphQL and provides built-in schema evolution as your data model needs change, you’re never “stuck”. Vendia isn’t a one-time deployment tool or a facade on top of someone else’s technology: It’s an enterprise-grade platform capable of efficiently sharing a single string or hundreds of thousands of transactions a second. So it can span a single developer looking for an easy way to create an app all the way to an enterprise looking for a multi-cloud operational data sharing solution. And Vendia’s data replication is just the core of a suite of capabilities that includes elements like smart contracts that make it easy to create stateful, transactional data triggers and workflows in any language…you’re never locked into a single cloud provider, a proprietary data format, or a hard-to-learn language like Solidity.

Customers have used Vendia to build a variety of solutions. Enterprises with complex supply and logistics chains rely on Vendia to create a “single source of truth” across supplies, manufacturing plants, and distributors to tackle some of the most challenging delivery problems facing the world today. Pet care providers have used Vendia to create a universal registry of lost and abandoned animals with a goal of getting them to safe homes. Financial service companies can easily “connect the dots” between consumers and vendors to ensure that everyone sees the same, auditable information about balances, financial ledgers, and accounts at all times. Travel companies and providers use Vendia to create a single, seamless view of everything from tickets to traveler information to payment settlement. Hotels use Vendia to efficiently sell rooms across multiple web and mobile platforms without risk of over- or under-selling. Startups use Vendia to model their entire backend system, enabling them to concentrate on the user experience and product market fit instead of building large distributed systems engineering teams just to get started. These are just a few of the many ways in which Vendia can accelerate time to market and lower the ongoing cost of API ownership.

These use cases span many industries, sectors, and sizes…from two-person startups to massive Fortune 100 enterprises. But they have one thing in common … they all asked the question: If we could remove 95% of the effort required to build and connect APIs that carry our most precious assets with partners, vendors, and users, what could we innovate and deliver with those saved resources? One of our key realizations is that everyone needs the same thing: From a single developer who wants to create a low-cost hobby project to a mid-market company looking to build cloud-based workflows without staffing up a huge engineering team to an enterprise division strapped for time and cash and looking to get projects to market faster, everyone needs APIs that do more for them, with less work and lower cost.

What about performance?

If you've ever looked into blockchains, especially public chains like Ethereum, you'd be right to be worried: At $25-50 per write, 14 TPS in worldwide capacity, and needing up to 15 minutes of settle time (aka latency), a lot of existing distributed ledger technology would be a poor match to high-speed, highly available API needs for conventional applications. Fortunately, Smart APIs don't rely on legacy blockchain protocols. They're engineered from the ground up, using cloud native technology, massively scalable serverless consensus engines, and are designed to tightly envelope costs, so that things "turn off" when there's no work being done. Smart APIs perform like best-of-breed public cloud services because they're built using the same techniques, such as multi-tenanting for economics, fault tolerance, and low-latency capacity allocation.

Getting started with Vendia

Because it's schema-driven and SaaS-based, getting started with Vendia couldn't be easier: If you can model your data using JSON Schema and list the clouds and regions where you'd like APIs to be present, Vendia can craft a secure, scalable, production-grade deployment in a few minutes and then maintain it automatically. If your data modeling needs change, you can easily evolve the schema in a self-service fashion.

And we offer both a free tier to experiment without cost as well as a monthly pay-as-you-go subscription that provides full flexibility to allocate your subscription towards whatever your application needs, whether that's file storage, scalar reads, or GraphQL subscription deliveries. Mobile, web, cloud, or on-prem applications … whatever you're building, you can get started in minutes and grow without limits using Vendia as your API and data platform.

Tired of building with "dumb pipes"?

Join the Smart API crowd – in under 5 minutes you can turn a JSON Schema into a production-ready, GraphQL-enabled Smart API for free and see how it works for yourself. To learn more and get started, visit www.vendia.net/developers.

About the author

Dr. Tim Wagner is the co-founder of Vendia and serves as CEO. Tim is the inventor of AWS Lambda and a former general manager of AWS Lambda and Amazon API Gateway services.

He has also served as VP of Engineering at Coinbase. Tim is the driving force behind the Lean Apps Movement, holds dozens of patents in serverless, cloud compute, and database technologies, and is a frequent keynote speaker at software conferences.

linkedin.com/in/timawagner

@timallenwagner

About Vendia

Vendia is the future of collective data intelligence, combining smart APIs, databases, and distributed ledger technology inside a single platform. Vendia's data automation cloud makes it easy to share data inside and outside of the organization in real time and with full visibility, governance, and control. Companies such as BMW, Delta Airlines, Resolution Life Insurance, and Fannie Mae use Vendia to automate contextual and compliant data flows between any-to-any systems for a harmonized, accurate view of data that unlocks speed, innovation, and cost savings. Learn more about us at Vendia.com and #UnchainYourData with Vendia.