

How Startup Pipe Went From Prototype to Production In 11 Days With Hasura

\$600k USD

Saved Annually: Using Hasura, Pipe needed 50% fewer developers than they anticipated

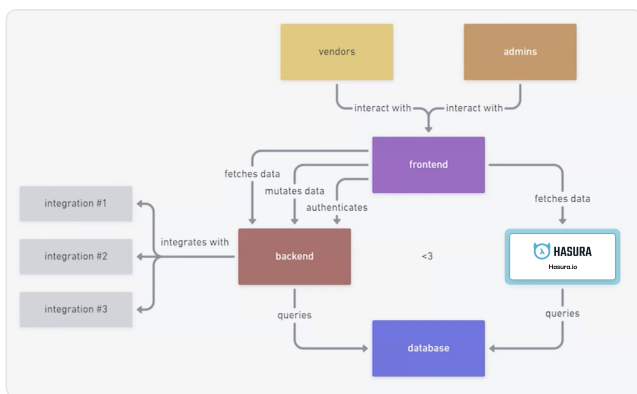
11 days

The time it took to go from prototype to production with Hasura

Pipe is a rapidly growing fintech startup that's transforming how companies fund their growth by unlocking their biggest asset - revenue. Through Pipe's platform, companies can trade their monthly or quarterly contracts for their annual value upfront, instead of taking on highly dilutive equity and restrictive debt.

In just one year Pipe has experienced amazing success, gaining over 3,500 customers and \$1 billion in tradable annual recurring revenue - accomplishing this with just 11 engineers.

Helping fuel this efficiency was Pipe's decision early on to make Hasura an essential part of their technical architecture, which has provided tangible benefits to their dev processes and their bottom line.



Pipe's technical architecture



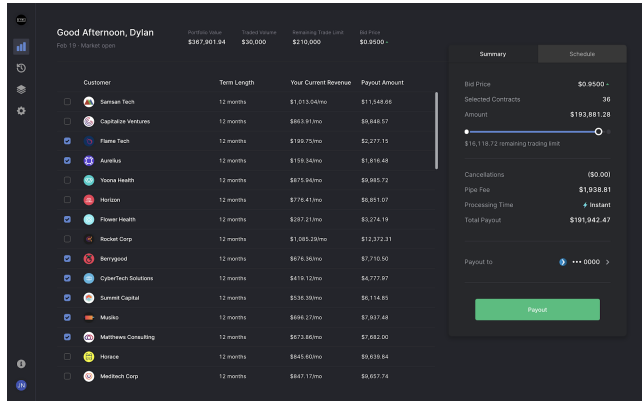
Hasura helps us remove the strict boundaries that traditionally separate front-end and back-end engineers, and makes it easier for any developer to make changes.

Other advantages include:

- ▶ 90% reduction in time for new feature development
- ▶ Simplified GraphQL data fetching architecture makes it easy to add new features
- ▶ Authorization and permissions model secures sensitive financial data
- ▶ Incremental migration to GraphQL rather than monolithic rewrite
- ▶ Breaking down barriers between frontend and backend engineering
- ▶ Consistency in production deployment and configuration utilizing Hasura metadata

Pipe's Challenge

Pipe makes it easy for companies with recurring revenue streams and subscription-based business models to convert that into capital to scale and quickly scale and grow their businesses.



The screenshot shows the Pipe dashboard. On the left, a table lists customers with columns for Customer, Term Length, Total Current Revenue, and Payout Amount. On the right, a 'Summary' panel shows a 'Total Payout' of \$191,942.47.

Customer	Term Length	Total Current Revenue	Payout Amount
Sansan Tech	12 months	\$1,013.04mo	\$1,348.68
Capitalis Ventures	12 months	\$683.91mo	\$8,648.57
Flare Tech	12 months	\$189.75mo	\$2,277.15
Aurelia	12 months	\$119.34mo	\$1,818.48
Nova Health	12 months	\$875.84mo	\$8,885.72
Horizon	12 months	\$776.41mo	\$8,891.07
Flower Health	12 months	\$287.21mo	\$3,274.19
Rocket Corp	12 months	\$1,085.28mo	\$12,372.31
Berrygood	12 months	\$676.86mo	\$7,710.50
CyberTech Solutions	12 months	\$419.12mo	\$4,777.97
Sunset Capital	12 months	\$536.33mo	\$6,114.85
Maple	12 months	\$686.27mo	\$7,937.48
Mathews Consulting	12 months	\$572.85mo	\$7,282.00
Nice	12 months	\$645.60mo	\$8,639.84
Meditech Corp	12 months	\$847.17mo	\$9,657.74

Pipe helps businesses convert subscriptions and recurring contracts into up-front capital

Pipe's development team needed to thread the needle balancing the immediate requirements of the business while also building for the future. Their technical architecture started with a Postgres database, Go as their backend language, REST APIs, and React for their web front-end running on Google Cloud. As they rapidly grew, they needed to consider additional concerns:

► Fast Time to Market & Feature Innovation:

Pipe's team needed to quickly create or change features based to gain new customers and respond to user feedback

► Flexible Architecture:

One of Pipe's architectural goals was to keep the architecture as agile as possible, where changes to the underlying database and frontend UX could happen rapidly.

► Small Development Team:

Pipe's 11 engineers needed to stay agile and flexible - understanding all layers of their system - database, backend, integrations, and the frontend web UI

► Flexible Web APIs:

One of their early challenges was inflexibility of utilizing REST APIs, where they encountered multiple variations of similar request types, but each requiring its own REST endpoint because of differing requirements.

► Performance:

Pipe's website needed to run quickly to provide a great user experience, and scale to meet the traffic of their rapidly growing customer base.

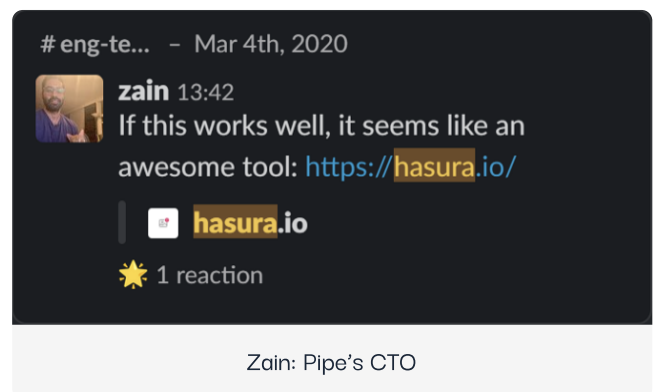
► Security & Authorization:

Pipe's system stores sensitive financial and customer data of their customers and they needed to make sure that the data was secure and that users could only view data they were authorized to see

Why Pipe Chose Hasura

As Pipe started investigating solutions to these challenges, they discovered Hasura and went into production with it within 11 days.

"We needed 50% fewer frontend developers than we thought we needed. Hasura and GraphQL reduced the toil to build and iterate on the frontend."



#eng-te... - Mar 4th, 2020

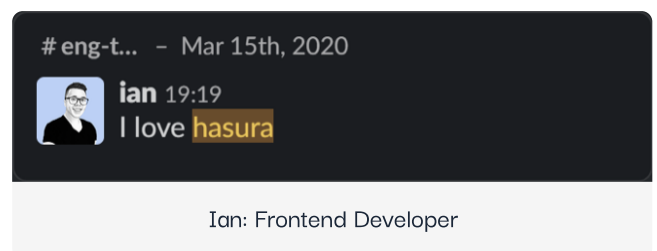
zain 13:42

If this works well, it seems like an awesome tool: <https://hasura.io/>

hasura.io

★ 1 reaction

Zain: Pipe's CTO



#eng-t... - Mar 15th, 2020

ian 19:19

I love **hasura**

Ian: Frontend Developer

Pipe put Hasura into production within 11 days of discovering it, helping them develop new features rapidly and breaking down barriers between backend and frontend development.





Hasura is a fantastic way to create a data fetching layer to our database. It's ultra-stable and often better at planning queries than ones we were writing ourselves.

Ease of Use

Hasura's intuitive tools and documentation made it easy to set up and connect to their Postgres database, instantly [auto-generating](#) GraphQL schemas and resolvers based on the tables and views in their database. This saved them development and maintenance time not having to author any more of their data access APIs themselves and modifying the code every time they made a database change.

GraphQL-based Data Access

Moving to GraphQL from REST also enabled them to unify and simplify their data access APIs. The flexibility allows the same endpoint to return only the requested fields and child (tree) data structures - making the queries more efficient and eliminating the need to create a different API for each unique request.

As Pipe made changes in their database, they didn't have to rewrite their APIs - they just had to adjust requests made from their React frontend to access the new data.

Security and Authorization

Another benefit Hasura provided is its built-in security and [authorization](#) capabilities, which can limit what data is returned down to the individual column and row level. This capability saved on development time implementing it themselves and ensured that customers could only view their own data.

Development Team & Processes

Utilizing Hasura made it easier for every engineer to understand the flow of data from the database to their frontend, making it possible for backend or frontend developers to make changes and add features quickly.

Operationally, their deployment processes are simplified by utilizing Hasura's [metadata](#) capabilities to make sure configuration changes made in development and staging are consistently deployed to production.

What's Next for Pipe

Pipe is currently using Hasura to query data to present in their React web frontend. In the future they plan on exploring additional Hasura features such as mutations - utilizing their existing Hasura GraphQL APIs to perform data inserts and updates -- and remote schemas to fetch data from 3rd party services.

They're also looking closely at GraphQL [subscriptions](#) to add real-time updates and notifications to the user experience. As their site traffic increases they plan to use Hasura's database read-replica support to balance queries and subscriptions across database replicas.

Finally, webhooks - initiated by Hasura [Event Triggers](#) when database changes occur - may also provide a layer of real-time integration to their other microservices.

Thank you to our friends at [Pipe](#), and [Peter Downs](#) - Director of Engineering, for sharing their story with us!

The on-demand recording of this webinar is available [here](#).

About Hasura.io

Hasura makes developers superhuman and simplifies app development with its open-source, real-time GraphQL API engine to instantly create reusable, GraphQL and REST APIs from your new and existing data. Power modern apps and complex data integrations while radically reducing your development time by using a unified interface in the cloud or on-prem. Hasura provides the tools, scale, and granular security to power the most mission-critical workloads. Learn more by visiting <https://hasura.io>