EPISODE 1474

[INTRODUCTION]

[00:00:00] SPEAKER: Developers looking for read or write access to Ethereum, Polygon, IPFS or other Web3 networks in order to get their idea in the hands of users need reliable RPC endpoints they can count on, whether they are working on a hackathon project, or running hybrid architecture for a production application.

Infura provides a platform for developers to get access to those endpoints, as well as tooling to help them build on Web3. With its origin being Ethereum, Infura has expanded to both EVM compatible and non-EVM compatible blockchains as well as decentralized storage. From wallets, DeFi, gaming, Metaverse, or NFT projects, Infura is relied upon by some of the most used and popular projects in Web3. With billions of daily requests handled, the team has built knowledge on how to scale with reliability. Co-founder and general manager of Infura E.G. Galano joins the show to talk about blockchain infrastructure and protocols.

[INTERVIEW]

[00:01:16] JM: Eleazar, welcome to the show.

[00:01:18] EGG: Yeah, thank you.

[00:01:19] JM: You run Infura which is a blockchain development suite mostly known for spinning up nodes. When you spin up a node on Ethereum, what are you actually spinning up? What does that node do?

[00:01:35] EGG: So, to be able to participate in the blockchain networks or Web3, whatever people decide to call it right now. It's a peer to peer network. So, just like with previous peer to peer networks, like BitTorrent, where to participate on that network in either share your data or get data from the network, you have to run this piece of software, which is called a client and that's what allows you to connect to the peer to peer network. So, the Ethereum node is one example of a peer to peer client, and by running that software, there's a database component that's part of it and there's a peer to peer connection.

So, it connects you to this greater network and it needs to download all of the latest information. And once you have that latest information, you can transact and interact with the network. So, Infura provides an API that looks like, talks like, and feels like your own personal Ethereum node, but scalable beyond what just a simple node running on your laptop can do.

[00:02:38] JM: So, what are the purposes of a client? What does a client actually do for you?

[00:02:45] EGG: By having the client running, that is your gateway to the network. Without having that you wouldn't be able to send transactions. You wouldn't be able to trade NFTs. Whether you're running that node yourself, or you're using an API like our to act as that gateway to the network for you, you need to have that to be able to interact with any of these applications that are being deployed in Web3, whether that's a decentralized finance trading application, an NFT marketplace, a NFT based ticketing portal, anything like that.

[00:03:22] JM: Okay. If I spin up this node, can you give me an example of API calls to the node that I might be making?

[00:03:32] EGG: Yeah, so when the node’s in sync, so they'll talk about clients being out of sync or in sync, and that means is your database up to date? Because you can't send transactions, which is like the primary right action on this global database, you can't do that until you have the latest state. So, once the client is synced, you can do things like get the latest block information and this is all very low level. These are things that definitely a consumer typically doesn't think about and developers are having it abstracted away through tooling, whether that's a library or some other SDK or something that they're using. But it's getting the latest block information. So, it's aware of what the current timestamp is on the network, what the latest tip of the chain is, what the latest transactions are on the network

So, from that block, you can kind of expand out to get all of the information that's interesting to you. What were the latest transactions for this contract? Or what were the latest transactions for my address? What's the current balance of my address? What were the recent events that happened on chain related to this application? So, if we're talking about an NFT marketplace, you can say, what were the latest NFT's that were created or sold or traded.

[00:04:54] JM: So, if a transaction gets performed on that client node, how does it get propagated to the rest of the Ethereum network?

[00:05:06] EGG: So, when people create – we'll say, an Ethereum transaction, but really any blockchain transaction, there's this process of signing the transaction. People will sometimes refer to this as their private key. Sometimes they'll refer to it as their wallet. MetaMask is the most popular wallet on Ethereum that people use, or not just Ethereum, but Ethereum compatible EVM based networks. They'll use a tool like MetaMask to hold this private key, which is used for their identity that's on chain, and this means that they have an address, and that address is their identity on chain, and when they need to send a transaction or assign a message for the network, it's signed locally, meaning nobody else on the network should be able to sign a transaction impersonating this person.

So, if it's my wallet, I'm going to sign a transaction locally, that key material lives on my device locally here in my house, when I sign that transaction, and it gets broadcast to the network. If I'm running my own node, it's going to hit that node through a local RPC interface, and then because that node is connected to the peer to peer network, it gets gossiped across the peer to peer network. So, it will first go out to the 64 or 128 peers that I may be connected to, and then it'll give out to their peers until it gets spread throughout the peer to peer network. Everyone should be aware that this transaction is out there, until it gets picked up by a miner right now and eventually a validator on Ethereum, that will then go through the process of including it in a final block that gets added to the blockchain.

[00:06:54] JM: There's obviously the question of gas fees. So, I have to pay something to run a transaction and to have it be executed by all the other nodes. What is the association between a transaction on the Ethereum network and the wallet that is paying for the transaction? Does the payment for the transaction, does that get included in the information that's being – that's included in the API call? Or I guess, walk me through the payment component of a transaction?

[00:07:31] EGG: Yeah, the transaction fee is an auction. You're in control as the user of setting how much you're willing to pay to get the transaction included in one of the next blocks on the network, and it's going to vary depending on the current state of the network. How congested is the network? How much throughput is there currently? During low periods of activity, you could probably get a transaction through at 10% of the cost, and it's up to you to decide how important is it that this transaction gets included quickly. So, if for instance, I wanted to pay you $10 worth of Ether or some other token, I can say, “Well, this needs to get to you within the next 24 hours and I can set that fee lower.”

This used to be a lot more complex. I needed to understand exactly what that number should be. But there's now smarter interfaces like in MetaMask, you can just set a low, medium or high type of setting and say, what's the priority of this? Well, it's a low priority transaction and it says, okay, so this is what we'll set the fee to in the transaction. And to answer your previous question, that is part of the payload that you sign.

So, in a transaction, you're sending the message data, in this case, it's the transaction itself that you're signing. But there's also the parameters to that transaction, which is, how much gas does it take on the network, which is paying the network itself for the resource utilization of this transaction, because when you send this transaction, all of those peers on the peer to peer network, the miners that included in the block, have to commit a small amount of resources to including this transaction or verifying this transaction. That gas was the concept for Ethereum of how you capture that in the user experience of the transaction fee. So, how much gas is the transaction take and then there's, what's the price of that gas? And it's denominated in way, which is the unit that the designers of Ethereum decided to use for that, but all of the interfaces that you'll deal with will show it in like a dollar or Fiat amount so that it's understandable to you. But you sign all of that. You get to decide what you're willing to pay. It gets signed locally and then sent to the network and sometimes you can get included within 15 seconds or less. Sometimes it might take hours depending on how much you are willing to pay and the congestion on the network.

[00:10:10] JM: So, the people who are spinning up nodes, for example, Uniswap. Uniswap is one of the featured clients on your website. That's a decentralized exchange. So, if I'm uniswap, why am I using Infura, as opposed to just spinning up a node on AWS for example? Can you talk through like the benefit of using a domain specific provider as opposed to just spinning up your own node via AWS?

[00:10:50] EGG: Yeah. So, it's actually a misconception that Infura runs tens of thousands of nodes for all of our customers. That would be extremely expensive and inefficient, and to an extent, that's kind of how we started. When we started Infura, we didn't know how much adoption and traction we would get, how useful this tool would be for developers. And so, we started with spinning up a handful of nodes, putting a reverse proxy in front of it, and load balancing requests across them. But the issues that you run into there are that there's limitations to what a single Ethereum node can do, their state and consistency issues, because each one of those nodes that you're horizontally scaling, have their own copy of a database, the Ethereum state database. And so, they'll all get information at slightly different times over the peer to peer network, and then you have this coordination problem that you have to solve.

So, we ended up solving that issue by decomposing the Ethereum client into smaller, more focused services that can fill the role of specific components of what's comprising an Ethereum node or a blockchain client. Something that's syncing the database, something that's indexing a certain portion of the database, something that's responsible for a resilient peer to peer connection, something that's broadcasting the transactions and making sure that they're getting rebroadcast if necessary.

That's really the value that Infura provides is that the ethos of the early blockchain community was that everybody should run their own node, because it's don't trust, verify. You need to have control of that data, to be able to say, like, "Yes, this is not being tampered with, we can take a look at this database ourselves, and we just need to take on the burden than the cost of running this piece of infrastructure ourselves." Our goal with the early Infura team was to help foster the adoption of like Web3 and Ethereum, and make it as easy as possible for people to explore, and it's really hard to sell people on, you should try experimenting in this space, you should try building something cool on Ethereum and Web3, if a lot of them were running into this initial hurdle of running this infrastructure.

It might sound easy to you, it might sound hard, it depends on your technical background. Some people were intimidated by it or ran into a lot of simple issues, technically. But that was enough to kind of stop them from experimenting more. So, we said, here's an API, that's exactly what you would get once you set up your Ethereum node, and you can just immediately start using it. Initially, there was no registration sign up, it was completely free, and we've always kept an element of that in the service that we offer, so that somebody can just take a URL, drop it into their application and get started.

So, applications like Uniswap, started using Infura at hackathons and things where it was at the ETHDenver Hackathon a few years ago, where Uniswap was initially created, and they were able to take the Infura URL, drop it into their application, and immediately show it to the hundreds or thousands of hackers that were at that event, and that's extremely valuable for mainstream adoption. Because if your goal is to get to onboard the mainstream, you can't say, “Well, you should believe in this like ethos of decentralization, and everybody should run their own node, you're going to limit the adoption of the technology.” And maybe the primary benefit to them was the ability to transact on this network, like the ability to use a very cool product like Uniswap, and they're not as concerned about running the infrastructure themselves. There are other ways that you can try to ensure that the trust is not being violated in that relationship, but people or development teams like Uniswap choose to use Infura because it simplifies that for not just their operations team, but also for their users.

[00:15:17] JM: The architecture, as you said, was originally this, basically like load balancing API calls across nodes on, I assume a cloud. Where you're using AWS?

[00:15:31] EGG: Yeah. Actually, we use a little bit of Azure and AWS in the beginning, because my experience was very heavy AWS, and a lot of the tools that we were using, were working a lot better at the time on AWS, for example. Terraform is the infrastructure as code tool that we were using for deploying some of the core pieces of our infrastructure, their support for Azure, this was back in 2016, wasn't as good as it was or as it is today. So, we migrated to the Amazon cloud primarily for our infrastructure, because it was easier for us to maintain with the tools that we were comfortable with.

[00:16:12] JM: Right. So, as it stands today, can you give me an overview of the architecture?

[00:16:16] EGG: Yeah, so it's fairly straightforward, honestly. When traffic hits the edge of our network, there's a network ingress point. So, cloud load balancer behind which we have our reverse proxy system, that reverse proxy has customizations in there that allow us to route traffic internally, to the subsystems that serve specific types of traffic. When you're dealing with blockchain API calls, they're not all simple reads, that's why it's an RPC. There's actually like a compute element to it as well, because there's the Ethereum virtual machine or other blockchain virtual machines that are executing the bits of code that might be part of the RPC call.

So, if it's something like give me this account balance, that's a pretty straightforward database lookup, as far as the blockchain API call is concerned. If it's interact with this smart contract, so something like you mentioned Uniswap, something like doing a trade or swap on Uniswap is not a simple API call. It's an actual like compute execution, where there's this payload that's part of the API query that needs to get submitted to a blockchain node that has this Ethereum virtual machine, and it's going to execute that, and then the resulting output of that is kind of the response that gets returned and then used in the developer's application.

We have a system that scales the compute aspect of it, and this is something that allows us to scale those like compute executions beyond just route everything to a blockchain node. There's also, I mentioned events previously, which is, when a transaction happens on the network, there are different side effects that can happen, and one of them is that a smart contract can emit events, and that's one of the ways that smart contracts build off of each other, is they'll listen to other events or look out for other events coming from the smart contracts that they're interacting with. So, I can say my smart contract should do this based on like the events coming from another smart contract and that means that there's a lot of queries that end up looking for events on certain contracts.

The way that Ethereum stores data, or blockchain clients in general, the way that they store data is optimized for very, very compact storage and quick verification of the data, which is the Patricia Merkle tree style, data structure of storage that's used in these blockchain clients. The problem with that is it's poorly indexed for queries of this fashion. And so, we had to optimize it because we were basically getting denial of service by simple queries to look up event contracts. The API that blockchain clients have is, I'll say, not optimized. Like, for example, you can say give me all of the events from this smart contract from the beginning of time, like the launch of Ethereum, and it might return gigabytes of data, and there's no pagination. There's no way to like really filter that aside from a block number. Between this block number and that block number.

So, when those types of queries would hit our back end, it would just take down the back end without there being like a caching or like performing indexing layer. So, we built this indexing layer for events, that it increases the amount of like on disk storage that you need, obviously, because you have to build additional indexes of data tailored towards your query pattern. But once you have that, these queries that were timing out, essentially, when they would hit a normal blockchain node, now can return in tens or hundreds of milliseconds.

[00:20:42] JM: Thanks for that background. So, if I deploy a smart contract, can you walk me through the transaction of the deployment?

[00:20:53] EGG: Oh, yeah. So, whenever you're deploying a smart contract, you're really doing the same thing as sending a transaction. There's one type of write operation that can happen on the blockchain network and that's a transaction. It's just there's different types of transactions. Some of them are sending value from one address to another. Some of them are modifying state, meaning like specific variables of storage inside of a smart contract, and then others are deploying code itself into a part of the blockchain, which is that smart contract deployment. So, the most popular tool for like helping people get started with developing a smart contract is Truffle. Truffle is a tool that has this like default integration with us, with Infura, so that there's like a blockchain connection that's already part of that.

You go through your process of developing your HelloWorld smart contract. Once you have that saved locally, you can run a Truffle command, Truffle migrate, or Truffle deploy, to be able to deploy that to the network. Initially, there's like a test environment that truffle includes that allows you to do that locally without having to pay any like real dollar tokens to deploy to the public network. But once you get to the point of deploying, it's going to cost you money, because you're sending a transaction, that whole transaction fee stuff I went over earlier, and then you have to pay the network for the actual storage of that contract on the network, which is going to be significant depending on how large that contract is.

But the actual action is still a transaction. It's just a very large transaction. So, when the tool that you're using to deploy it, so Truffle, like I said, there's one of them, runs that action for you. Behind the scenes, it's still going to be the actual name of the method is eth_sendrawtransaction. So, from Infura's perspective, we're going to receive the same type of method payload, as we would if you were to just transferring a token, but the payload contents are going to be different. It's going to be the code of the smart contract that you're deploying.

[00:23:13] JM: Got you. So, the integration with Truffle, can you talk more about what truffle is and what it actually does for Infura?

[00:23:26] EGG: Yeah, I can talk about what it is. What it does for Infura, it's more likely what Infura does for truffle in sort of like the relationship between our two products. But yeah, I can get into it. So, Truffle is like a development suite. There's not just like the tool that I was mentioning, that allows you to manage your code. That's one of the important things that it does, helps you organize your code, manage updates to the code. So, the migrations that I was talking about as you're iterating on your smart contract development, there's the piece that I've used most with Truffle, because of my experience with Infura, is a tool called Ganache, which is a local test environment.

So, imagine like you can have your own little like private blockchain, to be able to test some of the interactions of what you're developing in a much faster iterative fashion than what you would need to do if you have to go through the process of deploying to a public network, even if it's one of like the free test networks that are out there. There are public test networks that you don't have to like pay real money to be able to send a transaction. But even on those networks, it's more of a pain. It's closer to production like conditions, but it's more of a pain to develop on these test networks.

So, the migration or the development path typically goes you start developing locally with Ganache, and then, once you iterate and you get to the point that you're ready to start testing on the public network, you pick a public test network. Girly is when you might hear about Rinkeby and Ropsten are probably the most popular ones, and this is all outlined on not just Infura's documentation, but also on Truffle's documentation. There are examples to kind of help you walk through, like, what should I do now, because it can be pretty intimidating?

And then once you actually get to the point of needing to deploy, whether that's to the public test network, or the public main network, there's the connection to Infura. That means that you don't have to say, “Okay, well, now I'm going to download that Ethereum client or Polygon client or Arbitron client, or whatever network, you're targeting to deploy to, wait for that whole process to go through, syncing the database that might take anything from hours to days, to be able to deploy.” It's just instantly available, always on, and then you can just click the button or type the command and deploy your smart contract.

[00:26:06] JM: If I'm executing a transaction on Infura, what are some of the failure modes that can occur to cause that transaction to not go through.

[00:26:19] EGG: There are two that I can think of off the top of my head. One is that your gas price is too low, so that could be because of the state of the network, like I was talking about. If everybody on the network is paying that unit, again, is GWEI. Say they're paying 10 GWEI to transact on the network. For you to be able to get your transaction included, you need to be pretty close to what everybody else is paying. If your transaction is mistakenly set, much lower than that, say one GWEI, it's an error, but you may not know it. Because when you send a transaction, it's an eventually consistent system, right? Like you send this transaction out, but it's not immediately going to get included. You're waiting for the writers on the network, the miners, or the validators on the network to pick up that transaction, because it's profitable for them. They'll say, “Oh, this is one of the most profitable transactions that's asking to be included in the network. I'm going to take this and put it in that next block.” Because you set it at one GWEI, and not close to 10 GWEI, maybe it needed to be 12, maybe it needed to be eight. Again, it depends on the network conditions.

But because you said it so far below that your transactions just going to be like sitting there in that pool indefinitely until the network or the blockchain clients based on how they're configured say, “Yeah, I need to reclaim this memory space here. I'm going to drop this transaction because it's been sitting here for 12 hours, or it's being priced out by a bunch of these other transactions that are now willing to pay 15 and 20.”

So, as a user, that becomes a bad experience, because you don't actually get – or as a developer, you don't actually get a notification that your transaction was dropped from the network. So, it's not in the protocol. Tools like Infura, tools, like Etherscan, will kind of monitor the state of the pending transactions that are out there on the network, and let you know through a user interface or something that your transaction was dropped, and you need to rebroadcast it or resend it. That's one of them.

The other one is probably out of gas is the other error that we'll see a lot of, and that's a specific transaction needs a certain amount of gas. So, when I'm talking about deploying a smart contract, the Ethereum virtual machine assigns a cost to the opcodes of the virtual machine. And it might be to create a section of storage of data section of storage, update a section of storage, any type of interaction of that opcode level has a cost, and there's a way to estimate how much a transaction needs in terms of gas. So, there's an API call that's like estimate gas, and that tells you how much gas this transaction that you're about to send should take. For the most part, that should give you an accurate number of this is how much gas this transaction should take, I'm going to set the parameters for that. Now, I'm going to send the transaction. But because, again, you're not dealing with a static system, you're dealing with a dynamic like ever changing network, maybe what took, I don't know 25,000 gas a minute ago, needed to be more like 26,000 gas.

So, if somebody is too precise with how they set the gas of their transaction, it might try to execute the transaction and then run out of gas, and then the error that gets returned is out of gas, which is a very confusing, very low-level error. Again, that's why now in 2022, compared to 2016, or even 2018, I hope a lot of users don't run into that error anymore, because application developers, tooling developers have gotten a lot better in dealing with that, both in terms of estimating gas properly, accounting for the variance in terms of how they set the gas price and abstracting that all from the user. Typically, when I send a transaction, and I'm doing something on my MetaMask, I'm not setting that gas amount, like I was back in the day. It's setting it for me and it's a much, I guess, better user experience now than years ago.

[00:31:05] JM: I'd like to know a little bit more about the operational day to day of running Infura. So, I assume you have some kind of continuous delivery system, you've got a DevOps team, you've got probably still some kind of infrastructures code setup. Can you walk me through the day to day operations and infrastructure?

[00:31:30] EGG: Yeah. And I'll focus on our engineering team. How large are we now? Probably 35 to 40, I'd say, is our current engineering team size. So, my background is a lot of back end development primarily focused on like DevOps, internal tooling, cloud scalability, site reliability types of stuff, more so than, like, forward or user facing application development. When we started building Infura, there was a lot of stuff that we did that was very heavily like backend infrastructure oriented. So, right from the beginning, we had infrastructure as code. We used Terraform. We've since started using Pulumi as another tool that had more flexibility, and the developers that are on our team, were really liking what Pulumi was allowing them to do in a less restrictive way than what Terraform was doing for us.

We don't have like traditional configuration management. So, back in the day, I used to use like SaltStack as our configuration management. But with Docker, and like that whole model, transitioning to Docker style deployments, Salt really didn't make much sense when we started building out the Infura infrastructure in 2016, and everything was treated as immutable deployments. That was really important for us, especially being in the crypto space we are concerned about, like security of the systems is always a concern. But security of systems that have like a financial value impact are always like, next level stressful. So, immutable deployments were something from the beginning that we wanted to have. So, if there was ever any concern about the integrity of a system, like there shouldn't be, it's because we're rerolling it constantly and we don't have these, like long lived pet systems.

Docker, Terraform, tools like that, putting it through CI/CD. Initially, we had Jenkins as our CI/CD system, and then transition to more GitHub actions stuff over time, trying to do more like Slack based ops, so that our team is able to not bounce from tool to tool. It's not something that we've implemented currently. But it's something that we want to get to, as there's more Slack based ops for that CI/CD pipeline. We have a separation of our team of, we have a dedicated reliability team just focused on performance, optimization, observability, things like that. But as a culture, we don't want that team to be seen as this is the on-call production team, and they're the ones that deal with incidents.

I worked at companies in the past where there really was – some of them were great. Some of them had like a throat over the wall type of culture of, "Oh, that team is going to handle the incident. We're just going to focus on our application development, the more fun stuff and somebody else has to deal with the mess." As much as possible, we didn't want to have that culture on our engineering team. We wanted everybody to feel like they were helping when there might be an incident or feeling invested in the quality of service of our production network and not saying that's somebody else's responsibility. So, everybody on our team, all engineers are in the on-call rotation in some fashion.

Back in the day, it used to be that you were on call for everything. But as the complexity of our systems grew, things needed to be a little bit more focused, so that somebody wasn't overwhelmed when they're on call and it's not a perfect process right now, there's still a lot for our engineers that are on call to keep in their heads. But we've tried our best as we've had to scale the team from just me and my co-founders to now about, 70 people to share that.

So, dedicated reliability team, now we have a DevOps function within our engineering org focused on more like internal tooling improvements, and then we have a split on our team of how we handle things that are kind of closer to the blockchain client, and more traditional, I guess, we would say more traditional, like non-blockchain specific back end engineering. So, the team currently, we just call it our Ethereum team, even though they now deal with many other protocols besides just Ethereum. They're the ones that have to understand, what's this network upgrade that's happening on the Polygon network? Or what's the major database change that's happening to this Ethereum clients? Or if you've been following the Ethereum ecosystem, this whole Ethereum to Ethereum merge that's happening in the next few months, what's the impact that that's going to have on our infrastructure? Because that's one of the things that's interesting about being an infrastructure provider in this space, is, it's like having your product built on top of an open source ecosystem that has a roadmap that could significantly impact your product, right?

So, the Ethereum merge is an example of that, where significant changes are happening to the Ethereum client, and the network as a whole. And our product needs to be constantly iterated on to maintain compatibility with that and ensure that we're still building the right things, that our systems are still going to operate as they should, and there's a dedicated team within Infura just focused on that. And then we have, like more of an internal platform team is kind of what we call it, focused on like our core billing systems, our data pipelines that feed the usage statistics for our developers to see on the dashboard, things like that.

[00:37:41] JM: Well, to wrap up, you already mentioned the Ethereum update as being something that could impact your business. Can you talk more about the future and how changes in crypto will change your product development?

[00:37:58] EGG: Feeling pretty well prepared for Ethereum emerge. The ecosystem, the community as a whole has tried to make that extremely seamless, because you can't say, "Okay, at this point in time, every single application on our network needs to run a database migration of some kind", or, "Oh, at this point in time, every single user needs to throw away these keys they're using and these new types of keys, and use this new method of signing", like that's outside of our control in Infura. That's a community wide effort to make sure that this merge that needs to happen is done in a way where existing applications are not impacted negatively.

So, the way that we're preparing is more about making Infura more dynamic and multichain ready. At the time that we were building Infura, it was really just Ethereum that was what people were deploying to when they were talking about smart contracts. If you were saying, "Oh, I'm going to build like a smart contract enabled application doing X or Y", you are doing that on Ethereum. That's not the case anymore and it's not because Ethereum can't do it. It's just there's like usage optimized chains. There are chains like Palm NFT that just focus on this is the right chain for you to use, if you're going to be deploying an NFT. I think of recent ones like a Batman movie NFT. This is the chain for that. Before it was everything went on Ethereum. And so, our infrastructure was just optimized for this is the network that we run.

Over the last year we've had to significantly retool our infrastructure to say like, "We don't know what the next great network is going to be that developers are going to want to use." That might be one of the ones that we're already aware of like Solana or MIR or StartNet is one of the other ones that developers are asking for. Or it might be something completely new and one we haven't heard of yet. So, we've been preparing our team, both organizationally and at a technical level, for being able to meet the demands of whatever the new networks are that developers are going to want to deploy to and use, because it's not just going to be one. We see the pattern that applications are now spanning multiple networks, or want to be deployed across multiple networks.

[00:40:33] JM: Cool. Well, thank you for taking some time to give us an overview of Infura, and I'm sure it's useful to anybody out there building crypto infrastructure. Thank you.

[00:40:43] EGG: Sure. Thanks. Appreciate it.

[END]