EPISODE 1240

[INTRODUCTION]

[00:00:00] JM: IT infrastructure provides the components required to operate environments like networks, virtual machines, or containers, an operating system, hardware, data storage. As companies build out different deployment environments with infrastructure configurations, they must maintain the different environments, replicate them and update them. The management of infrastructure often automated to some extent is referred to as infrastructure as code.

Gold Fig Labs helps growing companies better understand their deployed infrastructure beyond the basic infrastructure as code principles. Gold Figs Labs developed two main tools to deliver the clearest view of infrastructure security and compliance. Checkup provides periodic security and best practice reports for AWS accounts. The report details a specific actionable and relevant advice to improve security posture. Their other tool, Introspector, is a unique security and auditing tool that provides in-depth analysis of larger cloud deployments with complex regulatory requirements and custom internal policies.

In this episode we have Vikram Nijjar and Greg Soltis. Vikram is co-founder and CEO, and Greg is the CTO. We talk about infrastructure as code, the complexity of cloud infrastructure security and regulatory compliance.

[INTERVIEW]

[00:01:14] JM: Guys, welcome to the show.

[00:01:16] GS: Thanks for having us here.

[00:01:17] VN: Yeah, we're excited to be here.

[00:01:18] JM: You both have experience working on large-scale infrastructure companies. I'd like to just start off with getting your respective experiences in working deep in infrastructure and how that shaped your perspective at Gold Fig.

[00:01:34] VN: Sure. I'm Vikram. I was an early employee, actually the first employee at a startup called Firebase back in 2011. There I led the SRE and operations side, and my co-founder Greg led the engineering side of things. In 2014 we were acquired by Google, and during the intervening time the product shifted from the real-time database, hosting and authentication to Google's entire mobile platform. So pre-acquisition we had a lot of challenges in selling a cloud infrastructure offering. So this was both around performance, product differentiation and cost. So we have a lot of experience in not only building that type of infrastructure, but consuming other underlining pieces for utilization in our offering.

[00:02:34] GS: Yeah, actually one of the first projects when we joined Google was migrating all of our infrastructure to Google Cloud, which was a new experience for us. We were actually still on bare metal at the time. So we had to quickly ramp up on using one of these giant cloud providers and porting our infrastructure over and that kind of gave us a view into the complexity of one of these providers.

[00:03:02] VN: Yeah. And so at that time this is 2011, 2012, we were you know deep into Chef and some of the proto IAC tools, but that's our background in a nutshell.

[00:03:14] JM: And are there some lessons there that you derived into thinking about product development? Thinking about how some of these problems that you might have encountered could be turned into a product?

[00:03:28] GS: Sure. I think one of the biggest product insights that came from that time is around ease of getting started. So this is back in that time like MongoDB was big and they – I don't know if you've ever done their getting started, but it was essentially download MongoDB and start running it and okay you're good to go. And Firebase had a JavaScript include, and it's back in the stone ages, where if you want to use a JavaScript library you added a script tag to your website, but that was it. You added the script tag to your website and you could start using it. And that meant that it was very accessible to people in ways that more complicated products weren't. Like there are examples, plenty of examples of possibly technically superior products that were just so much harder to use that they didn't last. They couldn't build up a sufficient user base to get momentum going just because you could start using something else in five minutes.

[00:04:32] VN: Yeah, and I think you know further along that point, that aspect of it directly influences the product decisions because the time to get started directly influences engineering decisions you're making around the complexity for the zero state, right? Getting a productive session earlier in the life cycle influence is adoption.

[00:04:56] JM: So is your vision to improve the time to market for having a secure system?

[00:05:05] VN: A little bit, but I think it's also influencing how we deliver our own product right now. So we currently have an open source project that is based around Docker because it was our observation that like, “Okay. We have this Python code, but if you've ever tried to replicate your Python setup on somebody else's machine, it's a huge disaster.” So what actually we do right now is we wrap it in Docker and then have a Go binary that you can just download and run and as long as you have Docker installed it'll manage the Docker container for you to run our open source project, and it's very much just like download it and go. And that I think comes directly from our experience building Firebase.

[00:05:51] GS: Yeah, and answering your question, currently one of our offerings is Gold Fig Checkup which we target startups who are maybe at the precipice of product market fit, but the stakeholders have concerns around security. But they've accumulated rightfully so technical debt around whether it's scaling, compliance or security in service of their product, and that's a totally normal and totally legitimate way to iterate your product. That is one of the tradeoffs you make, which is accumulating technical debt. And security is one flavor of that technical debt. However, once you come out of that fog, you might not know what questions to ask around your infrastructure or you might not remember where all the bones are in the settings and configurations that have brought you to the present. So we have this white-glove service which we’ll work with you to understand your team, your product and where you are in the life cycle of your company and provide a targeted report so that you can quickly get all the security low-hanging fruit and easy wins out of the way. But a lot of it is, most folks early on, they don't have a dedicated security person. They don't have the cycles to dedicate to that modulo, any product that is strictly focused on security. But for the most part iterating on product market fit. A lot of these types of concerns fall by the wayside.

[00:07:26] JM: Got it. So the checkup system that you've built, it has some periodic system that you can just run to verify your infrastructure against certain best practices and reportings. Can you give me like a sample of what are some of the best practices that Checkup system tries to check against? Like what are you trying to verify that is secure on the consumer's infrastructure?

[00:07:52] VN: Sure. So one of the early things that people do is, as you onboard folks, you might not have the two-factor on all of your principles. So the folks that are logging in, there is an inconsistent application of let's say two-factor. Another place is your storage buckets. You might have iterated settings to get your application working, but it might be – Whether it's a public bucket or a bucket that allows any principal to read it, that might have been prudent in iterating the product to get it functional. But going back and cleaning these things up across your infrastructure are often just overlooked. And as you have the product team moving and pushing things forward, those things, like I mentioned, falls by the wayside. But it's things that people know about. Like once we resurface them, it's easy for folks to remediate these things, but these are a couple of the really low-hanging fruit which we see across the board.

[00:08:56] GS: Yeah, actually one common scenario we see is engineers are trying to get two pieces of infrastructure to talk to each other, and for whatever reason it's difficult, there's a complicated configuration language. And so they start using a bigger and bigger hammer to try and open the hole so they can talk to each other. This might be like a Lambda talking to a database or an S3 bucket or something like that, and they'll reach the point where they get it working and then they'll move on. They'll move to the next part of their product, and then months later, if nobody's checking, they've left this gaping hole in their security. And it's the sort of thing where if you could see it you would fix it, but you don't necessarily know to look. And that's part of what we provide is this look at all of these things. So you can see, “Oh! We actually did not mean to leave that open.”

[00:09:49] JM: Okay. And do you think over time you can improve the Checkup product by adding additional points of checkup?

[00:09:56] VN: Yeah, absolutely. That's actually one of the longer term visions that we have for the open core product, Introspector, where we'd like to have your source control infrastructure

settings in there as well. We've been burned by if a repo had previously turned on code reviews and for some reason someone turned it off. Was it ever turned back on? So staying on top of settings like that. Who are the collaborators? Who are the owners? Who are the admins around your source control? So we definitely envision not only Checkup, but Introspector being able to get the expressive nature of queries across all of your infrastructure providers.

[00:10:39] JM: Okay. So talk a little bit more about the Introspector product. So what does that do in relation to the Checkup product?

[00:10:46] VN: Sure. So we were actually inspired by Osquery. Osquery was an open source tool out of Facebook which they categorize as infrastructure analytics. It basically provided a SQL interface across initially Linux, the proc file system, but now it supports Mac and Windows, but it gave a uniform and consistent view into what previously was wired together by Bash scripts or Python scripts. So we wanted to bring that same level of uniform query ability to our infrastructure settings. And historically, actually Netflix had a product called Edda. Lyft has a similar product, which slurps in all of your config and settings plane into a uniform place. And we had a lot of experience with whether a Chef or IAC, like Terraform or CloudFormation, and that gets you quite a way around the intent of what your infrastructure should look like. However, querying the infrastructure as deployed, there's always a gap there. We're always chasing you know little one-off Python scripts to query Boto or bash scripts to get the metadata service to answer a question. And we really saw the power of having all of that type of information easily accessible and expressible was kind of the motivation for Introspector.

[00:12:12] GS: Yeah. To that end, Introspector is mostly a snapshotting tool. It will take the configuration from your cloud infrastructure and format it so it can be read by a SQL database and you can write SQL queries against it, which really lets you ask arbitrary questions in a way that's difficult with, as Vikram, said, just a bash script.

[00:12:35] JM: Who would want to use Introspector?

[00:12:39] VN: Yeah. So we actually we're working with one customer who they already have a dedicated security team, right? So the security team, they have their SIEM and they have their security tooling. They have their log alerting. They also have a compliance team, right? So they

have folks that are managing the SOC 2 and the ISO compliance and all of the evidence, gathering and snapshotting that is mandated by compliance. However, the infrastructure team doesn't have the tools which give them the productivity around asking questions about their settings, and this takes its form as Python scripts that are making AWS CLI calls or Boto calls.

So we see Introspector around teams that want to express any internal policies that they might have already, express them in a uniform SQL interface.

[00:13:34] GS: Yes. A lot of times, security and infrastructure teams, they obviously have to conform with the policies for SOC 2 compliance, but they might have their own policies that they also want to mandate. So there are tools that will cover the compliance aspect, but they stop there. They don't go on to just arbitrary policy about your infrastructure. And similarly there are linting tools for your IAC, right? They'll look at your Terraform files. They'll look at your TF state and tell you whether or not something conforms, but it doesn't touch what is actually in your account what is actually deployed. And so if you wanted to mandate some policy across your company across multiple AWS accounts, multiple AWS organizations, and just verify that this is always the case. This is a hundred percent of the case whether somebody set it up via the CLI or the console. That's the sort of thing that you can do with Introspector. And we've seen a number of companies have these sorts of policies. Like they treat the compliance stuff as a baseline and then build on their own policies on top of that.

[00:14:42] VN: Just elaborating there. In the modern layout for a product, we see obviously all regions, but now we're seeing more and more dozens of accounts, and the churn around whether it's Kubernetes workloads or Docker workloads that churn quickly through infrastructure and getting a grip on how those settings, whether they're consistent or not, and then along the lines of the source control side at Firebase even at Gold Fig. We're a small team. We have less than 50 repos. However, we're working with some folks that have on the order of 500 to 1000, and in some extreme cases excess of 15,000 GitHub repos. And so what does it look like for a small team to be able to manage that? Are they writing custom GitHub tools? Are they clicking around in the console? And we've all gone through the experience where you need to validate something and you click around a console. And so we want to bring leverage to getting views around tools which might not have been made for that use case and, frankly, slurping them in

through a tool set which gives you the ability to find that needle in a haystack. We find a lot of value in that.

[00:16:00] GS: There are actually a bunch of one-off tools like this out there already. So there are various like S3 scanners or things that will check specifically your resource policies on your Lambda functions, and each of those is kind of like its own little island, and having one of them doesn't really help you onboard the next one. And what we're seeing with Introspector is once you get the snapshot there, once you can write your first query, writing the second query is much easier. You already have all the tooling there to do it. And so maybe you look at one of these S3 scanners and say, “Oh, I like the policies that that one's enforcing. Okay. I’m going to write that query.” And then you look at the one that checks Lambda resource policies. That's just another query to add rather than a whole another tool to onboard.

[00:16:52] JM: So the security checks that you do, are they at run time or are you doing more static analysis over the code in the repos?

[00:17:03] GS: So it's not over the code in the repos. It's over what's actually deployed. So we take a snapshot of the infrastructure, and I guess it's kind of static analysis over the snapshot, but it's specifically not over things like your Terraform code, your HCL, because we want to look at what's actually out there, whether it's controlled by Terraform or not.

[00:17:26] JM: Got it. So can you help me understand like the state of the security? Like one of the areas of software engineering that really is a blind spot to me is the security product landscape. So can you tell me what are the other products? What are the other solutions that people are using to have this kind of security? Or are there other products?

[00:17:45] GS: Sure. I mean the security landscape is pretty wide, and there absolutely are static analysis tools that will look at your cloud formation templates or your Terraform files, and those are great. Those are super helpful, and that will help prevent you from introducing errors into your environment. Similarly, there are things like real-time attack detection tools that will kind of scan your logs as they're coming in and look for anomalies. And those are also super helpful, but there's kind of a gap in between of what is the configuration of what's actually there. And there's some kind of light stuff around there, like AWS has some built-in things to check for

specific misconfigurations of their service, and that's actually what we see is most of the service providers will provide a kind of light version of this for their specific service. It's a lot tougher to go across services. And that's where third-party tooling really comes in, is you might have something that – Let's say you have an IAM group and you want to tie that to your org chart. Like that starts to get a little bit more difficult. You might have to go up to an SSO provider to do something like that, which can be a big jump for some companies.

[00:19:07] VN: Yeah, and like Greg mentioned, security is really fractal in nature. You can go extremely broad. You can go extremely deep. Our belief is the vigilance around the basics is where you get the most ROI. And I kind of alluded to earlier, this is like making sure your database instances aren't exposed to the public internet. Making sure your cues don't have star principles. Making sure everyone has MFA turned on. Making sure all of your S3 buckets are private and secure. We're in that camp, and we slot alongside other tools, whether it's exotic pen testing or red teams and blue teams and where they fight in between. Security is, like I said, a type of technical debt, and staying on top of the basics is where Gold Fig slots in.

[00:19:58] GS: Yeah. And in particular, this is a periodic thing to do. It's not on every commit. Like some of the static analysis tools, it's not a real-time tool like the log analysis. This is a periodic scan of your configuration. In that sense it's similar to some of the compliance checks, but you go beyond what is mandated by compliance. You get more customized policies. You can express your own. As I said, a lot of companies have either their own policies that they try to enforce or would like to enforce. So this fits in kind of that cadence.

[00:20:37] JM: Once I get some advice or some error or problem in my infrastructure that needs to be remediated, who is responsible for fixing it or what's the – In your system, like who gets notified? Who gets delegated to working on something?

[00:20:53] VN: Yeah. We leave that to the stakeholder and prioritizing that engineering work in many circumstances. We'll work with teams to help them remediate it and give them the guidance on what that remediation should look like. But typically it is the engineering team that is then updating their Terraform code to make changes and get them reviewed and deployed according to their normal release cycles.

[00:21:20] GS: Yeah, some of these things, they're not always drop everything and fix it now. It might be an extended project to fix it. And so they need to be prioritized alongside other engineering work. So we really have to involve the engineering managers and the engineers themselves to make sure that we have a shared understanding of priority, but also so that they can fit it into their schedule.

[00:21:48] VN: One specific scenario we worked through previously was locking down a customer's security groups, and our visibility into that, we have a read-only credential that accesses the config plane. But us remediating public access might have implications for their internal private network's traffic. And so working with them to make sure it gets rolled out in a piecemeal fashion so that there isn't an outage for instance that's caused by closing a security group.

[00:22:22] JM: Tell me about the difficult technical challenges in building the different tools that you've built. I guess maybe you could tell me the architecture of Checkup or Introspector.

[00:22:32] GS: Sure, yeah. Probably the largest challenge is that the cloud infrastructure out there is not made for this. It's not built consistently, even something as simple as like on AWS, most resources can have tags. Well, in the API, the tags can be expressed in probably a dozen different ways. So there's just a lot of ground to cover. And to that end we kind of took a phased approach on how we import things. So this is all part of the open source Introspector project. We start by just taking whatever the cloud provider has. If they're going to tell us, let's say an S3 bucket for example, like here's the configuration of an S3 bucket and it's just a raw JSON payload. It might not look anything like the payload for an EC2 VM. Even if they have some things in common like maybe when they were created, it's just however the provider decides to give it to us. That is fine. We'll just take that and save that. So you have that data, and that's great for the tools that are custom built against just like one kind of resource.

The next thing we do is we break all of those down into a notion of resources, attributes and relations between them. And actually the relations can have attributes as well. So that's kind of a foundational set of tables, and you can actually use this to do graph kinds of queries. Because the relations and the resources are kind of generic, you can follow them in a pattern however far you want to go. So one example of where you might do that is looking at principles. So user's

roles and groups and the permissions they have. So you could have permissions attached directly to a user. You could have permissions attached to a group that a user is a member of. Or a user might have permission to assume a role that also has its own permissions. But finding all of the permissions assigned to a user then becomes a graph problem, right? You start with the user principle at the center and crawl out those different groups and roles to find all the permissions that that user has access to. And that's this kind of generic resource and relation layer is pretty helpful for that sort of query. But it's not necessarily the most helpful for writing a SQL query because it is so generic, right? If you want to write an intuitive SQL query about users, you probably want to have a user's table. So that's the next thing we do, is for each resource we create its own table with actual attributes that look like what the provider has for that resource. Users have usernames. They have metadata about their credentials. They have are they allowed to log in? Things like that are all columns on a table. And so this is the third phase of all these resource-specific tables. And then it behaves a lot like a SQL database that was designed for this sort of thing. There're foreign keys between tables where appropriate. There're many too many join tables where appropriate. You can start asking things like, “Oh, which EC2 VMs belong to which security groups?” And you can join between them the way you would if the database had really just been constructed to model that.

But having these three layers kind of gives you escape hatches so that if you need to do some of that graph and relation calling, you can actually join between the two different layers that do that. You could say like, “Okay, I need to find a user based on when they last logged in.” That's a great query for the user’s table, but then you want to query the set of permissions for maybe the most recent logins. Well, that's one of the graph style queries. And so you can join against the graph layer of the data after starting with the user layer of the data.

[00:26:35] VN: One of the challenges that Greg alluded to is just the scope of config plane data that real infrastructure has. There's a litany of tools that try to visualize your infrastructure, but it quickly becomes intractable just based on the sheer number of nodes and clicking around in places. And so I think the next challenges for Introspector will be not only managing the scale of large infrastructure deployments, but what do the semantics look like when it bridges over to your source control settings? And then further as we pull in things like your SSO, whether it's like G Suite or Octa, those wind up being different tables in Introspector. And how do we as Gold Fig make it easy for our users to express queries across different domains?

[00:27:26] JM: How do you test that your product is working in the way that you think it is?

[00:27:33] GS: We rely a lot on infrastructure as code. We have a bunch of templates to spin up infrastructure, and we run it against that. And then we have a whole bunch of queries that we use to validate that our snapshotting is still working properly. And then probably the other piece that helps a lot is just via our customers, when we go and snapshot their accounts, we're essentially running QA on that as well. Every time we take one of these snapshots, we go through it. A human looks at the report and understands like, “Hey, does this actually make sense?” We talk with the customers about what specific services they're interested in and have concerns around and dive deep on those and make sure we are supporting everything that they need to get some confidence around those. And we've added services as a result of some of those things. Some of the networking layer stuff was added as the result of one customer saying, “Hey, the thing I'm really interested in is making sure that I have my networking and routing properly set up.” And so that was an impetus for us to go and really dig in on that piece and make sure it was well supported.

But, really, I think the biggest benefit just comes from seeing a wide variety of accounts. Just running this tool over and over and making sure it can handle weird edge cases, because the configuration space is so large and so many people are doing so many different things that it's important to just throw a lot of variety at it.

[00:29:11] VN: Yeah. Just a quick shout out to a few projects. Salesforce recently released a tool called Endgame, which attacked resource policies. In that repo they had Terraform configs to actually spin up improperly configured resources. There's another one called Sadcloud. And the third which we use called TerraGoat. And so these are all Terraform files for creating intentionally misconfigured and insecure infrastructure.

[00:29:39] JM: Cool are there any very common misconfigurations that you find your customers or test clients having? Like what's the most common form of misconfiguration?

[00:29:52] VN: When we see a bunch, is actually database instances open to the public internet? And this isn't the credentials are weak or anything, but just merely having a database

whether it's Postgres or any of the cloud offerings. Security groups wind up being configured in such a way that that port is open to the public Internet, and we see that one a lot.

[00:30:17] GS: Yeah. I think I'm going to add two. The first one is multi-factor auth. We see a lot of teams that will have maybe eighty percent coverage on that, but they'll have an admin who might be the CEO who originally set it up and has just been resisting multi-factor auth. And that's frankly a weak spot in their security posture. We see that one pretty frequently, and that's one of the ones that we try to prioritize very highly, is all of your admins and absolutely the root account should have multi-factor auth. Ideally everyone, but usually we can convince people that all of the admins should be using multi-factor auth.

And then, let's see –

[00:31:02] VN: Stale credential.

[00:31:03] GS: Oh, stale credentials .Yeah. Yeah. We see a lot of, “Oh! We onboarded this third-party vendor for a trial and then forgot to remove it.” And so you have some stale access from somebody you're no longer doing business with, but they technically have access to your account.

[00:31:21] VN: Yeah. And then a final one that we see a bunch is – S3 has this setting which if you use policies to lock down your bucket, you get one universe of outcomes. But if you use ACLs to let the objects be readable, it also allows the enumeration of the bucket. And our guidance usually is there are many situations where you do want a bucket to be public. But if you do not need it to be innumerable, you should turn that off. There are obviously enumeration attacks that you can run against buckets. But if least privilege buckets can be configured to exclude enumeration, we try to do that, but it's tricky because if you use one set of AWS console settings, you get one set of effects. But if you use it in a different set, you get a different set.

[00:32:12] JM: Let's talk a little bit about the business. Go-to-market is always hard with these kinds of products. Can you tell me about your go-to-market strategy for the Checkup and Introspector or whatever your plan is of productizing them and selling them?

[00:32:28] VN: Sure. Our open core is not only our participation in open source and giving back to the community, but it's also to, frankly, engender trust with the security community. And in the coming year we're going to be releasing more tools built on Introspector. One we have right now, it's called RP Checkup, which checks all your resource policies. That's one prong of it. The second prong is actually just reaching out to folks that through one mean or another we've identified as switching from ignoring security or ignoring this type of technical debt and getting them on boarded. I think most of our early customers have been inbound from word of mouth and some of our internal marketing efforts, but we're still in the early days.

[00:33:20] GS: Yeah. As Vikram touched on with the RP checkup, this is actually an open source tool that we released that was built on top of Introspector, and this is going back to that easy to get started. It was a binary that you download and run and it spits out a report about the resource policies in your AWS account. We have, as a result of working with a number of customers, built up a bunch of these reports and we are in the process of turning them into these tools that can be just run one off. And they're all built on Introspector. So it's actually fairly simple to construct one because it's really just packaging a query along with Introspector. And so we're hopeful that those will lead to more interest in Introspector as an open source project. And we'll kick off a cycle of somebody sees this query, says, “Okay. This is pretty useful, but in my company we actually want this specific policy,” and can start building up a library and a community of people who are interested in expressing these policies via SQL.

[00:34:30] VN: Yeah. And just to further elaborate there, the tools that we're putting out, I think the community is rightfully skeptical of open source tools and companies that join them. Our view is we want to put out tools that deliver value. They're not stunted. They're fully encapsulated for you to run across everything that we run against. One of our north stars is to give back to the community in a genuine and holistic way where people can pick up these tools piecemeal. But if you want our white-glove service, that's something that we can work with you on. However, the open source tools run in isolation and completely without us as well.

[00:35:15] JM: Do you expect it to be difficult to convey the value of Gold Fig or to communicate what it does relative to other security products just because the security market is so crowded?

[00:35:27] VN: Right. Yeah. Not only do we expect it to be difficult. We are finding it to be difficult. And we've kind of alluded to the profile of company that we are currently going after. It is extremely narrow. That buyer window for us is very narrow. However, Greg and I, we've been doing this since 2019, and I think we have extremely strong founder market fit. Like we love doing cloud infrastructure. Like that is our – It's in our blood. We resonate a lot with that narrow market. We've had those experiences. We've had those pains. And so we can speak to those as well. We are finding it to be difficult, but we have a laser focus on where those pains are. And providing that thing that niche loves and gets utility out of, we find that to be you know our guiding principle.

[00:36:19] GS: Yeah, it's been a little bit – For the people who it resonates with, it really resonates. This is the like, “Oh, we just did a lot of experimentation to get where we are, and like I think we've got something solid now, but we've left a mess in our wake.” And there're a fair number of those. And if you have that in your mind, then I think the idea behind Checkup really sounds appealing. Like, “Hey, we need some help to just identify what all the stuff we've left behind is so we can slot those into our engineering prioritization.” Getting that message out, getting that as like this is who this Checkup product is really helpful for is one of the things that we're focusing on.

[00:37:02] VN: Yeah. Checkup is that narrow piece, but we are hopeful, and there's promise around teams that do already have a security team, that do already have a compliance team. But Introspector is slotting in between those. We're still early in exploring those use cases. However, there's definitely promise there where on the security side they might be using eventing, or logging, or SIEM on the compliance side. They're using whatever tool they have there to deliver evidence and create snapshots. However, infrastructure and the managers and engineers responsible for the configs of the infrastructure, they're still a huge universe of questions that fit between security and compliance.

[00:37:47] JM: When I try to understand what you guys are doing, one of the valuable points of messaging I've turned to is the blog post that you've written about infrastructure as deployed. Can you, I guess, define this term infrastructure as deployed relative to what people know as infrastructure as code and just explain how – Maybe if somebody listening to this is somewhat

curious about Gold Fig, but still not exactly sure if it applies to them, maybe drive home this meme of infrastructure as deployed.

[00:38:18] GS: Yeah, absolutely. Infrastructure as deployed is what is actually out there. Like what is actually sitting in your account? What repositories you have? What VMs you have? What buckets you have? Whereas infrastructure as code is a means of expressing what you would like. And infrastructure as code is great. It's a great way to try and get a handle on this complexity, but also it doesn't know what it doesn't know. If you tell Terraform like, “Go and create this S3 bucket for me.” It'll do it and it'll keep that S3 bucket in sync with exactly how you want it, but it will have no idea that you have 10 other S3 buckets. Infrastructure as deployed is saying, “Okay. Forget how it got there,” whether it was Terraform or something else, “we're looking at what's actually there. What's really been deployed.” It'll find those 10 other S3 buckets and you'll be able to say, “Hey, do these all look as if they're configured properly? It's sort of like you're familiar with the saying like the map is not the territory. Infrastructure as code is the map, and maps are obviously helpful. We're concerned with the actual territory. Like what is really there.

[00:39:32] VN: Yeah. And on a long enough asymptote, IAC does get you pretty close. If infrastructure as deployed helps you discover those 10 other buckets and you fold those in, now you have pulled that in with line with the rest of your infrastructure. However, if those are now existing between different repos or their different settings or the matrix between how you intend them to be configured, isn't immediately clear whether it's in ETL files or elsewhere, then infrastructure as deployed gives you the ability to get a view of how all of those things are actually configured.

[00:40:19] GS: And for security and compliance, what you actually have deployed is what matters. If somebody is trying to hack into your AWS account, they don't care whether that bucket got there via Terraform or not. They just care that they can find it. And so for a certain set of concerns, you really have to know what is actually there, not just what did you intend to be there.

[00:40:44] JM: So I'd like to close off by just getting your vision for how the company is going to unfold in the coming years and what challenges you expect to encounter.

[00:40:55] VN: Yeah. The overarching vision, we want to have other concerns folded into Gold Fig. I've alluded to whether it's source control or your SSO, but there's no reason why other config plane concerns couldn't have similar visibility. Think you're Zapier zaps. You have dozens and dozens of those. What do those configs look like? How does that unlock the ability to make sure those are uniform? What does your HRIS look like, vis-a-vis your ability to query a security group? And that ties back to your GitHub repo. And that ties back to your HRIS. Our long-term vision is to provide a single pane of glass around all settings and configs pertaining to your infrastructure.

[00:41:50] GS: I think there's a lot of power that can come from joining across data sources. Yeah, if you have your org chart and can join that against who are the code owners in a repo and join that against where the code from that repo gets deployed. Maybe that goes into a Kubernetes cluster. Okay. You have the configuration for the Kubernetes cluster and can join that against the permissions that are assigned to that cluster in your AWS account. You start being able to draw this line from like, “Okay. This human, this employee has these capabilities over far down the path via this source code that they're allowed to check in that gets deployed over here that then has these permissions.” And you can start to really express things that are difficult to do today. You can maybe attack like one piece of it or you have to write a custom tool if you want to go across all of that. At bigger orgs we've seen you can pinpoint a single piece of infrastructure and know all the provenance to it. That is not a common thing that exists in public infrastructure. That is only afforded to big orgs like Google or Facebook. And unlocking the ability to join across these disparate SaaS providers, SaaS and infrastructure providers, will give managers and team owners the ability to like pinpoint provenance across all these concerns.

[00:43:17] JM: You guys both worked on Firebase, and I see firebase as – At least when it came out it was super transformative, and I still think of it as transformative in the sense that it's for many people the best way to deploy infrastructure. Like for many hackers that are kind of new to deploying infrastructure. Personally, I think in many cases it's going to be better than Heroku. It's going to be better than Netlify. It's going to be better than Vercel. It's going to be the easiest way to deploy infrastructure. I think Firebase is just an incredible product. So given that you guys have had that vision into the future of infrastructure from your standpoints at Firebase, what's your point of view today? What's the future of cloud services and infrastructure?

[00:44:01] GS: It's kind of not a secret right now that I think the biggest spends are around just VMs. There’re a lot of just people moving from on-premise data centers to pick it up and move it into a VM. There're a lot of – Hype is probably the wrong word, but excitement around containerized workloads and serverless. Actually I think that's where things are going. Like even though the bulk of it, if you're looking at the numbers, it’s still just like raw compute. The serverless model is I think undeniably more scalable and has more concerns removed from the developer in a way that I think is beneficial. And what I mean by that is they're not really losing anything that they really needed by having those concerns removed. It's going to take a while for the ecosystem to catch up. Libraries that are out there will probably have to be rewritten to fit those constraints, but it's happening, and I think that that's going to continue to happen. Whether it's Lambda or a framework built on top of Kubernetes, or whatever comes after Kubernetes, I think definitely the serverless model is not going to go away. I think that's only going to continue to grow.

[00:45:22] VN: Yeah. And Gold Fig is powered by Firebase. It's powered by AWS. It's powered by a litany of services. And as the interop between them gets better, the ability for, like you said, the hackers that are getting spun up on a weekend project versus an extremely mature product, they'll begin to converge. There's still a lot ahead. I think as technologists, we are focused on the cutting edge, but just last year I was at an office in the Bay and I saw a closet full of computers. And so when we think about moving to the cloud, there's still a lot to be had here.

[00:46:07] JM: Okay. Guys, well, thanks for coming on the show. It's been a real pleasure talking to you.

[00:46:10] VN: Likewise.

[00:46:10] GS: Thanks for having us.

[00:46:11] VN: Thank you.

[END]