EPISODE 1357

[INTRODUCTION]

[00:00:00] KP: Infrastructure as code is an approach to machine provisioning and setup in which a programmer describes the underlying services they need for their project. However, this infrastructure code doesn't compile to a binary artifact like traditional source code. The successful completion of running the code signals that the servers and other components described in the configuration file have been created automatically by the tools being used.

In this episode I speak with Christian Tragesser, DevOps consultant with World Wide Technology. We discussed the way modern development groups are using infrastructure as code and other tools to advance their products and solutions.

[INTERVIEW]

[00:00:42] KP: Christian, welcome to Software Engineering Daily.

[00:00:46] CT: Thanks, Kyle. Great to be here.

[00:00:48] KP: Can you tell me a little bit about your background as a technologist?

[00:00:53] CT: Sure. So I've been in it for – Coming up on 20 years. Now have traversed the roles of basic desktop support, worked my way up through systems administration, systems engineering, networks, virtualization. Worked for an MSP for a small amount of time and then kind of got into helping the infrastructure aspects of cloud.

Not too long after that, started working for a great company, a software development company, where I was actually able to sit and work directly with development teams. And from there I learned a ton. I've been pretty much doing that since then. The business more or less is traversed outside of our custom development teams to actually helping clients of our company. And I've learned a lot, been having a great time. And yeah, that's pretty much where I'm at from now.

[00:01:47] KP: Well, technology has certainly changed quite a bit, and it's a rather a treadmill to keep up with at times. You'd mention cloud as being one of those major innovations. Do you have any perception on the degree to which we're having success with cloud adoption? Are most people who should be on cloud currently there?

[00:02:05] CT: You know, that's a good question. I think that cloud has certainly become mainstream. I think that that's good for a lot of organizations. I think it's a challenge that many organizations are learning about the differences between more or less running a data center, running virtualization, and more or less using APIs to provision aspects that you're now managing or using.

[00:02:32] KP: And have you seen any ways in which, I guess, maybe the titling or the professional services have evolved in this time period. Certainly, where you might just have been a software engineer or a programmer at some point, you're probably specialized now. Do you have any thoughts or opinions on the general structure of an effective team?

[00:02:51] CT: So there are certain aspects. I think why is this challenging is because it traverses so many aspects of what it means to run solutions, run technical solutions. Typically, we would have the siloing of responsibilities that would come together for certain efforts, projects and more or less contribute together to provide that solution, right? And with the way that the technologies exist today, there is a ton of enablement that is available to any one organization that's trying to do this in the cloud.

I'm sorry. One of the advantages in my opinion is the advent of infrastructure as code, right? Infrastructure as code allows us to treat our infrastructure as software. And that's a very empowering thing if you know how to work with software, how to create software. It's a different domain, I think, from the traditional sysadmin or infrastructure roles. While you do have knowledge in those domains of what you're accomplishing with your code, if you're not experienced in writing code, and this is not only just typing in your IDE to do something automatically, but the way that you manage that code, the way that you validate or qualify that code, that's a new thing to learn.

I personally don't think I would have stumbled upon it had I not had the experiences that I have of working with development teams who utilize certain characteristics of their processes, like TDD. Those were enlightening experiences for me. And I found that it greatly benefits the amount of success that I have with those tools. And so if you're not familiar with those aspects and you're kind of just getting introduced to cloud through the tooling, that's where I think it becomes pretty challenging or people start to kind of struggle, organizations start to struggle.

[00:04:43] KP: When I'm working on personal projects, I love infrastructure as code because I can stand everything up, work on it for an afternoon and get rid of it. I'm very cost conscious in that regard. What are some of the other use cases or ways organizations leverage infrastructure as code?

[00:04:59] CT: So I love the same kind of ephemeral characteristics of being able to prove out proof of concepts. It's great to kind of work on something, build something, throw it away. Cool. We have kind of a successful thing that we can do whenever we want. But I think that there are wonderful aspects to infrastructure as code in terms of the longevity of projects or the longevity of infrastructure that really allows us to capture a point in time or as change happens throughout the life cycle of a solution. Those are very enabling characteristics of infrastructure that allow you to test your code, to audit your code, to review your code. These aspects of I think what are now becoming mainstream characteristics of developing software can now be applied to our infrastructure. And it also provides another medium of collaboration in that of the code repo. So no longer – I shouldn't say no longer. But we can reduce the amount of time that we spend having face-to-face discussions or other types of what used I think maybe require meetings at the top. We can more efficiently communicate through code to accomplish these strategies.

[00:06:17] KP: Do you have a vision for how infrastructure as code should be managed from a software release cycle point of view?

[00:06:25] CT: Do I have an opinion about it?

[00:06:27] KP: Yeah.

[00:06:28] CT: Absolutely. I think I've been told at times I'm pretty passionate about these aspects. And really, it's not that what I think is canon, or law, or maybe even the best way to do something. Another one of the aspects of kind of a systems thinking of, I think, really the true benefits of what DevOps in that we all have ideas of how to do things as engineers. And being in a cross-functional team, being able to have these conversations across many different domains, getting many different perspectives, a diversity of thought to contribute to the solutions is really one of the more appealing aspects of my job currently.

I love talking with other roles, other perspectives of the solution that we're working towards. And so to get that feedback or consider those perspectives when we're working towards the solution I think does enable that role who's ever responsible for maybe the infrastructure or cloud or whatnot to understand really what's more valuable for the team, for the organization. And so it's, again, more about collaboration and finding different ways to come together as a team and work towards a common goal.

[00:07:43] KP: Can you tell me a bit about your role at World Wide Technology and what they do?

[00:07:47] CT: Sure. Like I said, I was able to get hired by this wonderful company to work with development teams directly. And it was another organization before. We were purchased by World Wide Technology. But I was brought on to kind of help with the delivery aspects of our common or our custom software solutions. And so I, like I said, learned a lot from working with developers where I never worked for developers before. I was really just back on the infrastructure as code building of infrastructure and this organization allowed me to work side by side with these developers. So I started to get this perspective and just absolutely loved it.

We do custom software development as consultants. We also help out organizations with their DevOps efforts. So the organization started to see the benefit for our development teams to have this infrastructure knowledge on their teams as they progress throughout the different aspects of creating software, right? So there's management of the assets that we're working with, the technical assets. There's the automated testing of the software that

we're creating. There's the delivery of these solutions. And as we got more and more information about those types of capabilities for teams, we also started getting requests from some of our clients, “Hey, could you help out with some of our efforts, our DevOps, or transitional efforts?”

And so I’ve gone from kind of being in the systems engineering role when I was first hired with this company called The Synchrony. We were purchased by World Wide Technology. I think it was 2015. And since then we've been able to do external projects, right? Working with our clients to help them through their transformational or transitional projects or overall improvement efforts.

And so from my perspective, World Wide is a very big company. We do a lot of things. But from my perspective, it's custom software development and these assistants in DevOps in one shape one way or another.

[00:10:01] KP: What sorts of transformations do these companies want to go through?

[00:10:06] CT: Well, I think we have a traditional agile transformations. I don't want to say traditional, but agile transformations are some aspects that I’ve helped out with. We certainly do have services for those. Others are maybe tooling implementations. So there is an organization that's interested in a specific tooling. They might need some help to understand how to best use that or utilize that tool. And then there are process improvements. It’s kind of how I think. Process improvement, performance enablement capabilities, and we assist along with that as well.

[00:10:39] KP: Do you see a lot of multi-cloud or combination cloud on-prem solutions that you have to work with?

[00:10:45] CT: Yeah. There are a lot of projects where the client is essentially in transition. That could be from a data center to a specific cloud or from one cloud to another. Yeah, I think that cloud is being – The public clouds that we have now are really kind of falling or getting favor for certain services that they have, right? So we're going for something like we want to build our own infrastructure in a cloud. I think that the all of the technologies in

AWS are kind of favored there. We're talking about AI and those types of aspects looking more towards Google Cloud.

I've had the fortunate ability to work with Azure dealing with a lot of enterprise organizations. And I've not had to take care of an active directory server in many years, because that's more of a service now that's just kind of available. Very easy to integrate on-prem. So yeah, I think we may be in a transition all the way to cloud. We might just stay with this hybrid thing. I'm not really sure. But the integrations are certainly there from what I'm seeing.

[00:11:58] KP: Well, there are a lot of trends in software. Your comment that active directory, I guess, you could still choose to manage it if you wanted to. But I don't know why anyone would. Just get the managed service. We're seeing database as a service and ascension of these higher level things. Do you have a vision for what a typical technology group looks like in the future? Are they just gluing together all these services or is there really core IP that companies will own?

[00:12:24] CT: I don't know about IP. I think that there is a certain level of effort to create things and own them, like from a custom perspective. And so there's always that question of do we buy or do we build things. And so as certain clouds get known for services, I think there's really just a focus on what is the business objective. What are we working towards? How do we make money? How do we provide value? And if you are able to take on certain responsibilities, own certain services without too much of a deviation from what provides your organization value, I would say just kind of continue to pursue that. But to go back to the active directory thing, if I can pay somebody to kind of take care of that for me as our business pursues, what it is that provides values? Then I think that that makes sense.

So it's a bit of give and take. And if you can get into a service, I think that's popular. You're going to make a lot of money. But again, always staying business focused. What's providing us value versus the operational overhead that's required to achieve that goal.

[00:13:32] KP: Do you get into many conversations about technical debt and planning in that regard? It's sometimes hard to know when to invest in just refactoring technology and

things like that. Do you have any general thoughts or are you asked by clients to make considerations about how much they should invest future looking versus maintenance?

[00:13:50] CT: In my experience, I've seen organizations being pretty eager to make, I guess, a capital expenditure. I think that there is quite a bit of purchasing that's done at times with a lot of hope up on a technology. And there's really an interesting realization, maybe an enlightenment when new technology is kind of first brought in and people try it out, they kick the tires on it. And I think there's maybe an experience there that's not quite expected.

And so understanding that you operating in the cloud is not going to be like operating on a virtualization platform in your data center. Those are interesting considerations to kind of see happen and understand. But I'm not sure. It's really up to the organization.

Ultimately, we get down into certain aspects of in order to become more performant, we need to change in the processes in which we're working. Not so much those individual tools. And I would say that given the patterns that are starting to erupt to form across the individual vendors of all of the different technologies, I'm not sure that the tooling is the most important aspect of the these efforts. But that's usually where we start off.

[00:15:18] KP: Makes sense. When I think about infrastructure as code, I've done some projects with Amazon Cloud Formation. I think Azure and GCP may have equivalent tools, or perhaps they don't, because there's Terraform that I think has an approach to multi-clouds through different connectors it plugs in. Are there any other major tools I should be thinking about if I want to get interested in infrastructure as code?

[00:15:43] CT: I don't want to keep hitting the independence drum, but it depends, right? So I think Terraform is being adopted very well, and it's very common. And I work with it a lot across individual projects. But again, I think that a lot of organizations might default to that in certain aspects. Where it might not be best to do Terraform is if you had a cross-functional team and the application itself is maybe written in an interpreted language or aesthetically typed language. It might be better to go with a tool like Pulumi where we can have all of not only the application code, but the infrastructure code be written in the same

language. And the same test framework being used all the way across that stack I think is a very powerful kind of concept in order for the team to kind of reduce the cognitive load to use the tooling.

That being said, being able to have a solution that's created with Terraform probably increases your chances of being able to hire a skill set to come and join you or maybe excel in the role that you're hiring for. So there's something to be said about commonality. But again, in that context of what's the easiest for your systems, for your teams to work in is certainly a consideration to have. And so I don't always go to the same tool. It really doesn't matter to me which tool as long as you understand those patterns and what you would try to achieve with a given tool.

[00:17:18] KP: Yeah, good points. I've had a just a personal affinity for Terraform. And I've noticed many developers find it pretty easy to pick up for whatever reason, the ergonomics, or whatever, are very suited and people get the use cases. And then you just start working in it. It's easy to spin up and write yourself a YAML file or whatever it is. But it's also then easy to get the equivalent of spaghetti code going in your configuration perhaps. Do you have any best practices or advice for how you should structure or properly structure a Terraform solution?

[00:17:53] CT: Yeah. I think a primary consideration there is as there's change going throughout your solution, we're going to be starting all the way from some type of representation of your environment via VPC or it's a virtualization platform or whatnot. But there's going to be a lower level of configuration that's hopefully happening by Terraform, right? And that level of code, if you will, that level of the codified thing is probably not going to change as often as the technology or the solution that's sitting on top of it be like an application. So you'll want to split up those aspects of Terraform to make sure that you're not running through, say, your VPC configuration every time you want to make an application change, right?

So being able to level or slice those implementations of that code be it by importing state or however you'd like to slice that up, you're reducing the blast radius of any mistake that might happen. And so I think dividing up the solution by that and of course writing reusable

code as many kind of modular type of things that people can reuse or you can use for replication is a key aspect. And that's typically I think a skill that's just kind of learned. There're certainly patterns and you can watch in YouTube videos or read some blog posts. But I find that those patterns really aren't kind of realized until you made a mistake and you learn from them.

[00:19:31] KP: Sure, yeah. Well, there are certain patterns. I'm thinking of like the mono repo approach that if you take on a new client, I imagine you just go with the flow. That's how their style. They do that, makes sense. Are there any patterns, or I guess we call them anti-patterns, that you see that you have to really take a step back and convince people they need to change their ways on.

[00:19:53] CT: Oh, yeah. The first thing I'm looking for usually when I'm working with a client is any type of tests around the infrastructure, maybe even the validation stages, processes that happen within this code. So again, writing code and representing your infrastructure as code is a very powerful thing. But at the end of the day, we're all humans, we're going to make mistakes. And so if you're using this tooling and you're not providing processes, which enable failure. Meaning, if we have a failure, it's not catastrophic, but we get to kind of necessarily learn. Understanding that we're human, we're going to make mistakes. Being able to catch those and learn from them is a key characteristic that I'm always looking for within our projects and make suggestions in in terms of, "Okay. So we have a problem? Can we learn from this problem that we just experienced? And can we improve our qualitative or validation steps to help catch this before it goes through?" And us being human, we don't have to be perfect. Certainly no one person causes a failure. But can we construct or can we question the systems that we're working in to improve those processes and implement with confidence? And I think that's ultimately what's pursued with continuous delivery, not just the compilation and integration between different tools, but the processes that you build to implement and change and work with the life cycle of that solution throughout its existence.

[00:21:33] KP: When it comes to simple unit tests and following test-driven development, I feel like every language and framework has a good story. I know how to do that. The story

for testing my infrastructure is not quite as clear. What are some approaches for getting started?

[00:21:50] CT: Yeah. So this of course depends on which tool it is that you're working with. I don't know that you want to go maybe all the way down to certain what I would consider unit testing aspects because we're basically testing the functionality of the tool that we're using. But at a more maybe integration level or getting a picture of the world, one of the ways that I kind of like to go about thinking about what type of tests I would use or write for infrastructure is let's say I'm provisioning something in AWS via VPC, EKS, anything in AWS. When I when I do my provisioning, if I find myself going into the web console and checking certain things at certain places. I want to go to EC2 and see if the VMs are there. I have the appropriate security groups. Is the load balancer there? Do I have the correct DNS records, right?

This web front end is really also – These capabilities are also accessible through an API. And so rather than really kind of visualizing those checks every time, codify those aspects the same kind of checks that you're doing. And this applies not only to like a cloud level, but even down into an application testing aspect. If we are changing code, bringing up a web console to look at a message, can we automate that, that repeatable activity so that we can really use testing creativity? Let the humans kind of do the creativity work, let the robots, let the computers do the repeated work.

And so however you're working through your solution, if you can automate those same type of validated or testing patterns that you find yourself doing as you're writing the code, those lend to good practices for how you understand to test those aspects that you are coding. And that's going to traverse everything down to configuration management that you're using to build a server, to provisioning a solution up in the cloud, to running things on Kubernetes, right? Those aspects with your understanding of how you're implementing something, what the end result should be? Automating those testing efforts is really my best suggestion, I guess.

[00:24:12] KP: Makes sense. What are the advantages that companies realize if they take continuous delivery seriously? Why make the investment?

[00:24:22] CT: Well, performance is usually the biggest motivator, right? They're the aspects of we want to be quicker to market. We want to have technical agility. We want a better quality. And continuous delivery is commonly brought up in those discussions. And again, I think that organizations tend to go to the tooling that's most commonly associated with those methods. But I think that not too long after, you adopt those tools.

There is a realization maybe that we may now be working harder than we were before and not perhaps seeing the expected benefits of what we were pursuing. And I think it's that technical aspect that gets everybody kind of in. And soon people start to realize the cultural aspects, which is why culture is generally so talked about in in the DevOps world. And what does it mean to kind of have good culture in order for us to kind of get continuous delivery, get those performance aspects? We need to then focus ultimately on the processes, on the social systems that exist within an organization, because they're typically the constraint in the pursuit.

And so getting all of these tools that help enable continuous delivery is kind of like having a – You're going from riding a bike, right? Riding a bike, doing everything kind of manual, and you traverse to a tooling that is essentially kind of a jet, right? But if you're not enabled from a cultural aspect, if you're not enabled to move as fast as the tools allow you, then you essentially have this jet that you can't really take off the lot. You can't go any farther than what you're responsible for. And this eventually kind of just ultimately breaks down into lean and theory of constraints and an emphasis on quality, efficiency.

When I first heard about lean through **[inaudible 00:26:32**], let's say. It's an acronym kind of five pillars of DevOps. About lean, I'm thinking, "Oh, do less. Get more. I'm kind of all about that." But it's not really about a pursuit of leisure or kind of being able to do less, but more about influencing pride and work, understanding and enjoying your purpose in the organization. And those are the type of problems that I help work through most commonly and that are really a requirement for true continuous delivery for these performance and kind of quality enhancing characteristics that I think that organizations are usually pursuing.

[00:27:19] KP: Could you expand on some of those organizational challenges? I guess, maybe to start with, when an organization is struggling, they've upgraded and they don't know how to use the rocket, are they aware of the problem? Or do they need someone to help diagnose?

[00:27:33] CT: Some do. And I think that some are not maybe enlightened quite yet. And we typically start off with technical projects that I guess evolve or maybe pivot into different aspects once realized. And so I think the solutions to maybe some of these difficulties are a bit a different dimension maybe than was kind of anticipated by the organizations. And in that, I mean, the dimensions essentially of culture. And there are characteristics I think that should be pursued versus technologies when attempting to make these types of performance-enhancing improvements. That's a rough way to basically talk about what I think is essentially explained by Conway's Law. But an organization is going to perform. It's going to create what the culture is.

And what I mean by that is, essentially, if engineers don't have a good understanding of their purpose in a cultural aspect what they're kind of being relied on for, understanding their purpose, the value that they bring to a solution, then it's difficult to understand completely how to best do something or how to really give their best effort into something that they're working on, that they're responsible for, right? So I think it comes along with incentivizing people in a certain way. Aligning goals, communicating context, sharing as much as possible and working towards a common goal than rather representing a domain or being siloed off. From an organization, that's ultimately trying to achieve an aligned goal.

[00:29:29] KP: Gotcha. Yeah, totally agree on those points about an engineer knowing their function and role. I don't think anyone necessarily intends to hide that from them. Maybe if they're working on some top secret project or something. But in general, I think everyone's fully behind the vision you laid out. I expect so anyways. What prevents people from naturally being in that state? What gets them out of alignment?

[00:29:55] CT: So this is going to be an opinionated take here. But I think there are three primary objectives to accomplish before you can kind of take on all these tools and start seeing how fast you can go. And those are really more cultural. The first one is that of

vulnerability. So I wrote a blog post recently that kind of distills all this information from my perspective. But in my experiences throughout my career, I used to come from a very prideful aspect of my work. And that I found out was not the best for me not only from the perspective of personal relationships with the people that I work with, but in the technical aspects, and the skills, and the learning um about these technologies that I've been so fortunate to work with in the past.

And so coming into a solution, or a project, or whatever responsibility that you have without an aspect of vulnerability I've learned is a hampering for one's career. And so when we are a coach – I talk about diversity in terms of thought for any project or team that you're a part of. Without listening to others, without taking in different perspectives for even solutions that maybe you've done many times before, I think that that is a restrictor. And if you're not working with a team who is open to that type of thought, then it becomes very tough to influence or help them in any one type of way. They might be perceived as being threatened with their role or even maybe their status in the organization. That's a tough thing to work through. And ultimately, I think if you break down the barriers of maybe pride or whatnot and let them know that we're here to collaborate, we're here to all succeed together and work towards an objective, we'll all be better off if we can just kind of trade our ideas and have discussions about them.

The second aspect is to be able to question. This is known as building a learning organization. Being able to go through your experiences, reflect and question on the processes in which you're using allows you to continuously learn about things. And this is more along the lines of learning through experience. If you're in an endeavor and for some reason that fails, you don't essentially scrap all the lessons that are available to learn from there, but iterate and improve on them. That then is going to allow you to alter or change things but in a motivation of improvement, right? We're always trying to continuously improve. And through our failures we should learn more and be able to kind of pivot or change and understand how we can maybe not repeat that mistake or experience.

Finally, with those two avenues available in communication and just kind of being able to talk about things in real world terms, then we can focus on quality and everything that we do we can improve on in our processes, understanding that we are vulnerable.

Understanding that we have the ability to change and improve with a focus on quality, we can improve our systems. We can improve in the way that we collaborate. And generally when we're making those types of improvements to accomplishments that we need to professionally accomplish or our roles or responsible for, that tends to know have a good influence on your everyday life, on your everyday at work. And so you're getting quality of life through work. And that's a very hard thing to change. There's no way that that can probably be done in a couple of months. But that's the higher overarching goal, I think, essentially, of a lot of the projects that we do participate in. So that's a lot there, but that's why it's not as direct as just kind of getting Kubernetes installing your stuff and you're off and running.

[00:34:13] KP: Yeah. Great vision though, for sure. Well to wind up, are there any DevOps trends or emerging technologies that you're excited about?

[00:34:23] CT: Yeah. So in a general sense, I think that we are starting to kind of get out of this technology association. I think, certainly, there are a lot of organizations who've been successful in in working with individual tools, selling individual tools. But I do think that there is a bigger picture that's starting to be realized through systems thinking. More along these lines of lean and theories of constraints and just kind of getting over that technical aspect and solving the harder problems, but really the more beneficial problems to solve.

And so tooling is all over the place with the CNCF tooling landscape. I'm happy to see all of those tools. I love playing around with them. But I think for organizations to really keep success with their projects and their endeavors, this aspect of the cultural phenomena of collaboration, these things that I kind of talked about, vulnerability and quality, those are going to be trending up more I think.

[00:35:28] KP: Well, certainly will be interesting to follow. Christian, where can people follow your blog?

[00:35:33] CT: So I don't have a specific blog. World Wide Technology has a digital platform. I invite you to check that out. I have a blog post up there entitled Small Batches Glimpsed Into Systems Thinking. If you just Google WWT in small batches, I think that'll

take you there. I myself am on LinkedIn, Christian Tragesser. And that's about the extent of my online stuff.

[00:36:02] KP: Well, Christian, thank you so much for taking the time to come on Software Engineering Daily.

[00:36:06] CT: Thanks, Kyle.

[END]