EPISODE 377

[INTRODUCTION]

[0:00:00.4] JM: In its most basic definition, machine learning is a tool that takes a dataset, finds a correlation in that dataset and uses that correlation to improve a system. Any complex system with a well-defined set of behaviors and a clean dataset can be improved with machine learning.

Several precipitating forces have caused machine learning to become widely used, there's more data, there's cheaper storage and there's better tooling. Two pieces of tooling that have been open sourced from Google help tremendously; Kubernetes and TensorFlow. Kubernetes is not a tool for machine learning, but it simplifies distributed systems operations unlocking more time for engineers to focus on things that are not as commodifiable, like tweaking machine leaning parameters.

TensorFlow is a framework for setting up machine learning systems. Machine learning should affect every aspect of our lives including tuxedo fitting. Generation Tux is a company that allows customers to rent apparel that historically has required in-person fitting. Using machine learning, they have developed a system that allows customers to get fit for an outfit without entering a brick and mortar store.

In this episode Colan Connon and Thomas Bell from Generation Tux join to explain how Generation Tux adopted Kubernetes and TensorFlow and how the company's infrastructure and machine learning pipelines work.

[SPONSOR MESSAGE]

[0:01:42.3] JM: To build the kind of things developers want to build today, they need better tools, super tools, like a database that will grow as your business grows and is easy to manage. That's why Amazon Web Services built Amazon Aurora; a relational database engine that's compatible with MySQL or PostgreSQL and provides up to five times the performance of standard MySQL on the same hardware.

Amazon Aurora from AWS can scale up to millions of transactions per minute, automatically grow your storage up to 64 TB if need be and replicate six copies of your data to three different availability zones. Amazon Aurora tolerates failures and even automatically fixes them and continually backs up your data to Amazon S3, and Amazon RDS fully manages it all so you don't have to.

If you're already using Amazon RDS for MySQL, you can migrate to Amazon Aurora with just a few clicks. What you're getting here is up to five times better performance than MySQL with the security, availability and reliability of the commercial database all at a 10th of the cost, no upfront charges, no commitments and you only pay for what you use.

Check out Aurora.AWS and start imagining what you can build with Amazon Aurora from AWS. That's Aurora.AWS.

[INTERVIEW]

[0:03:21.5] JM: Colan Connon and Thomas Bell work at Generation Tux. Guys, welcome to Software Engineering Daily.

[0:03:27.7] CC: Good to be here.

[0:03:28.7] TB: Yeah, thanks for having us, Jeff.

[0:03:30.0] JM: Some of the shows that we do are about a deep dive into specific technology building blocks and other shows that we do are more like case studies in how certain companies are building their infrastructure. This episode is going to be more of a case study. You guys work at Generation Tux, which we'll explain what it does and then we'll talk about the infrastructure that you're using and how you're using it.

Let's start with the company and the business model. Explain what Generation Tux does.

[0:04:04.1] CC: All right. Yeah, sure. Generation Tux, we're a tuxedo rental company and sort of the selling point that makes us different from other companies, the big ones like Men's

Wearhouse that rent tuxedos for weddings is that we're entirely online. We really have no physical presence anywhere except for our warehouse operations and where we actually our development.

The way it works is you come on our website for an event, like a wedding. Obviously, that's a big part of our business, and you book your event and you put in your event date and then you invite all your wedding party, like your groomsmen and you create an outfit for them and yourself and then you go on and you pay for it and we ship the tuxedos to everybody in the wedding party based on wherever they put their shipping address to be a week at least before their event. Often times, we get it there 10 to 14 days before their event which gives them a chance to try everything on, make sure it all fits and if there are any issues we can either — They can either ship them back to us or we can help them find someone to do tailoring if it's just like a little sleeve length change or something like that.

Basically, the big value proposition is it's a much smoother process for getting a tuxedo particularly for a wedding where, historically, you'd have to make three trips to a brick and mortar place; one, to pick up the tuxedo an to get measured; one, to pick up the tuxedo; and then one to return the tuxedo after the event. We've basically made that very seamless and there's no trips to any store. Just go online, book your event, and get your tuxedo.

[0:05:40.8] JM: It sounds like you'd get economies of scale too because if you're making a large single order for everybody in the wedding party who needs a tuxedo, you can get a much better deal per tuxedo than if you were to have each person individually go and shop around for their tuxedo.

[0:06:03.7] CC: It is certainly true that because we don't have the brick and mortar that our prices are definitely way more competitive than like Men's Wearhouse. You could get a complete tuxedo outfit, like really high quality tuxedo and with a shirt and a tie and shoes and a belt and all that stuff for $189 dollars. At Men's Wearhouse that could cost as much as $300. Certainly, very competitive on price and that's largely due because we don't have the brick and mortar overhead.

[0:06:31.6] JM: What are the different components of the brick and mortar experience that need to be ported online that are more difficult to port online? Because, obviously, the things like shipping or a checkout process, that's pretty well standardized at this point. That's not going to be revolutionary, but there are certain elements of going to get a tuxedo. People who are listening who have gotten to go get a suit before, probably most people have, or nice outfit. There's nothing to be said when you walk into a nice clothing store and you have somebody helping you and tailoring your stuff, because there's all these finer points and you got to get it fitted really correctly. What are those different elements that are hard to port online?

[0:07:18.5] CC: really, the biggest thing, the difficult thing to port online, at least in my view, is a trust aspect. The thing that makes this business sort of unique and a challenge is that you don't get more than one bite at the apple. The person has a wedding, and for most people that's going to be the last wedding that they have themselves. They'll probably attend other weddings, but you have to build a lot of trust with the client because this is their big day and it really has to go perfectly.

Still, one of the reasons this is such a great opportunity and growth opportunity is because the big challenge is getting the bride, in particular, to trust that we're not going to screw up their wedding. That they're going to get the right garment. That it's going to arrive on time, and if there are issues we'll be able to solve those for them before the event. That's the big sort of soft challenge, if you will. It's the human challenge that comes with online.

Then the sort of physical challenges are they don't have a professional men's wear person in the room with you to make sure that you are going to fit in the garment. That's the other big challenge is getting people the right people. Getting their measurements and their body types and making sure that they're going to get the right suit, jacket, etc. That's our second challenge that we'd taken a lot of technological steps to help resolve, in particular, with our new sort of piece on our website, we call it e-tailor, which is we'd use a machine learning to take the process of getting measured. Historically, we would actually just send you a tape measure and you'd have a friend measure your body, and that's very error prone even for a professional tailor, that's error prone. Then when you have, basically, amateurs for lack of a better expression measuring each other and those measurements have problems. In addition, it adds in those step in the process.

What we've been able to condense it down to is — What is it? Five questions on our website; your height, weight, shoe size, age and whatever your gene waist is and then we're able to predict your measurements actually just as well as — We've done some testing with this just as well as like a professional tailor measuring —

[0:09:26.3] JM: Really?

[0:09:26.9] CC: Yeah.

[0:09:27.2] JM: Wow. Okay. Let's dive right into that. I know you guys use TensorFlow and we'll talk about that. More broadly, the machine learning aspect of it — In any machine learning system, you have training data. You have to train a model and then you continue to run training processes overtime as people are trying out that model. They're using the model and feeding actual real-world examples into it and then they could say whether or not it was a success. Describe what the process of training and using that machine learning model for the tailoring processes.

For people who are less familiar with this process, if you go into a real-world brick and mortar tailor to get a suit made, they measure your arms, they do all these measurements and stuff and it takes like 30 minutes or 40 minutes or 15 minutes depending on how good the tailor is. It's kind of annoying process but it's something that I accepted as — I just assumed you had to do that to get a suit. I guess explain what the machine learning algorithm looks like at a high level.

[0:09:27.2] CC: When we started this business, we weren't doing any machine learning at all. Our training data actually came from customers measuring themselves. Based on that, we had feedback of these customers were getting reships and these customers weren't. To initially train our model, we basically gathered all of our measurements that weren't reshipped and basically we fed that all into just a neural network and TensorFlow and basically we're just using like a regression based on all the questions, the height, weight, and pant waist. Based on that we just output their measurements.

From there, we actually have another piece. Once you actually have some measurements, we'd actually fit in our garments, because there might be some differences in our jacket size versus measurements. We actually have another machine learning model that takes the measurements and it translates it into our actual garment sizes.

[0:11:47.7] JM: How do you know when the dataset that you've gathered is big enough to develop an accurate model?

[0:11:56.3] CC: That was something that we actually tested a lot. We started probably back last October and our data quality wasn't as good so we had to work on cleaning that up. Also, we just made sure that we had all the pieces in place to collect all the data that we needed from customers that we were getting and we just kept testing and testing and testing until we got enough data that we felt confident deploying in.

[0:12:20.9] JM: Can you describe the process of taking data that was not originally gathered to be used in machine learning and putting that into a machine learning system? Describe the process of taking — It's basically a company that was not built with machine learning in mind. It sounds like machine learning was adopted as a tool to improve the processes at Generation Tux. Describe that process of onboarding with machine learning.

[0:12:51.8] CC: Right. Basically, we had to get all the systems in place to make sure the data we were collecting that was attributed to customers correctly and the quality was — There was no missing data. Basically, we had sure up all of our customer data collection to make sure we were getting the data that we needed and we could trace it, measure it in the correct way.

Then it was a lot of trial and error and manually cleaning data. There were a lot of outliers, so basically we had to do some statistical analysis on some of the outliers and get those out of there to make sure that everything was clean. Then we could actually feed it into the algorithm. That was actually the majority of the time spent developing this was actually getting the right data. The easy part was actually just feeding in the machine learning algorithm.

[0:13:44.6] JM: That data cleaning process is what I've heard takes up a large portion of the time that a company typically allocates to "machine learning". Explain the data cleaning a little bit more. You mentioned removing outliers. Why would you want to remove outliers from the dataset? Tell me other aspects of the data cleaning process.

[0:14:08.3] CC: Well, it's not that we were specifically removing outliers. It was just they were completely bad data points. Somebody that was six foot tall and they said that their arm length was 39. Obviously, that's probably not possible, so we just removed all of those out of there.

Basically, we established a range of somebody this tall and this height and weight is 99% of the time going to fit within these garments. Then, basically, we just fed it into Python and Pandas, and Pandas was a really good tool for doing the statistical analysis. It gives you some good helper functions and then allows you to really manage the data and drop the bad data in an easy way. From that, we would just output it to a CSV, and then from there we could actually import it to the TensorFlow file and actually train it that way.

[0:15:07.6] TB: Yeah, one of the other things that we had is a lot of domain expertise. The company itself is founded by George Zimmer who's the guy that originally founded Men's Wearhouse. He has 50 years of experience in men's clothing and then we have several other people who have decades of experience in men's clothing. We really lean on them a lot to help us look at this data, in particular, the ones that we're kind of on the edge and they give us a lot of insight in what was legit and what wasn't. That helped a lot as well.

[SPONSOR MESSAGE]

[0:15:48.5] JM: At Software Engineering Daily, we need to keep our metrics reliable. If a botnet started listening to all of our episodes and we had nothing to stop it, our statistics would be corrupted. We would have no way to know whether a listen came from a bot, or from a real user. That's why we use Encapsula to stop attackers and improve performance.

When a listener makes a request to play an episode of Software Engineering Daily, Encapsula checks that request before it reaches our servers and filters bot traffic preventing it from ever reaching us. Botnets and DDoS are not just a threat to podcasts. They can impact your application too. Encapsula can protect your API servers and your microservices from responding to unwanted requests.

To try Encapsula for yourself, got to encapsula.com/sedaily and get a month of Encapsula for free. Encapsula's API gives you control over the security and performance of your application. Whether you have a complex microservices architecture, or a WordPress site, like Software Engineering Daily.

Encapsula has a global network of over 30 data centers that optimize routing and cacher content. The same network of data centers that is filtering your content for attackers is operating as a CDN and speeding up your application.

To try Encapsula today, go to encapsula.com/sedaily and check it out. Thanks again Encapsula.

[INTERVIEW CONTINUED]

[0:17:33.5] JM: If you have the model over fitting in certain cases and there are real life outlier cases where people have certain shoe size, pant size, waist size that actually ends up not being so normal, and so they actually need something more custom. Do you have some sort of way to deal with edge cases, like, "Oh, yeah. If all else fails, Generation Tux will pay to have a tailor come by your house and measure you in real-life." What's the fallback mechanism?

[0:18:09.6] CC: When we ship your garment, like you said, we ship it usually 14 days in advance. You have plenty of time to try it on. We actually have our personal concierge team so you can call in and then we'll walk you through how the garment is supposed to fit. If you are concerned about something, say, you want a jacket size that's bigger, then we'll just reship it to you for free within two or three days.

[0:18:35.3] JM: Okay. Interesting. You got the data. You had a dataset that you felt was comfortable. How do you start to build a TensorFlow model around that? I think there's lot of people , including myself, who have heard about TensorFlow. They understand how it works in the abstract. They've never worked with it. What's the process of onboarding with TensorFlow.

[0:19:01.6] CC: I would say, honestly, we didn't start out with TensorFlow. We started out with scikit-learn, and that's what I'd recommend somebody that's getting into machine learning start out with. Basically, we started out with scikit-learn and we tried a bunch of different machine learning models and we just weren't getting as good enough results there. We decided to pick up TensorFlow, because scikit-learn, you can't really build the neural networks as custom as TensorFlow. Basically, we went on TensorFlow's website, read the docs and basically just experiment with all the different network architectures, like how many layers we have, how many neurons we have, what activation functions and what training algorithm we use, so like gradient descent and different stuff like that. Basically, it was just a huge experimental process just to see what results we could get.

To make that experiment — So we had confidence in it, we had to really split our data into training and test sets and really have confidence in the data we gather was going to be good enough to journalize in the real world.

[0:20:14.2] JM: You mentioned two terms I would like you to disambiguate. You mentioned neurons and activation function. Describe what those two things mean.

[0:20:25.3] CC: Okay. A neuron is basically the structure of your neural network. It basically takes input as like a matrices of weights and it uses an activation function which is just a different algorithm to determine the output based on the weight. It's input in the neuron.

[0:20:46.3] JM: Okay. Wait. Sorry. Did you say activation function?

[0:20:49.4] CC: Yes. The activation function is something that actually lives within the neuron.

[0:20:54.3] JM: Okay. How does this relate to the number of layers?

[0:20:58.0] CC: Each layer will have a certain amount of neurons and in each one of those neurons has zero activation function. The number of layers was just an experiment for us. We started with two and then we just experimented to see what would work for us.

[0:21:15.9] JM: I did some shows recently about image recognition, and with image recognition the different layers in the convolutional neural network are — It's easy for me to understand why you're using layering. I'd like to think of that as an example, because when you're looking at image recognition you start with the lowest level of the pixel and you're just doing some sort of low level analysis on the pixel and what sorts of edges the pixels aggregate to. Every image has edges. Different colors make up different contrasting, and the contrast defines the edges. Then you can abstract up further form that and you can say, "Okay, what does a collection of edges aggregate to? What sort of image does it aggregate to?" You kind of look at that image and then you build up higher and higher levels of abstraction. You say, "Okay, this collection of symbols aggregates into something that looks like a bowl of cereal," and you just abstract higher and higher and higher.

Those layers make sense to me from an image recognition standpoint. What are the different layers of abstraction that you're building there with — At the lowest level you've got these different data points around what is your shoe size and what your pant waist size. What are the different layers that build on top of that as you're trying to get to the highest level which is what size of tuxedo does this person want to wear?

[0:22:56.9] CC: We start with the first layer. It's just one to one mapping of all the inputs that we get. Then from there we condense the layers down to — We'll have a layer that takes the inputs from your height, weight, and it will have an output. Then there's layers that take input from your pant waist and your — I forgot the specific details. Basically, we just decrease the layers at each point until it maps out into a single output for what specifically it's trying to predict.

[0:23:32.7] JM: Okay. Can you talk about that in more detail? Because I know part of what you're doing here is you have these different variables and you're trying to assign weights to those variables. If you're looking at pant waist size and show size, if those were just the two variables, you would have a weight on either one. Basically, the weight is saying, "How important is pant waist size when you're trying to determine the tuxedo size to recommend to somebody?" Then the weight for the shoe size is, "How important is the shoe size when you're trying to predict the right tuxedo size for somebody?"

For each of these little data points, you have a weight. Explain how those weights get established and how they get trained and improved?

[0:24:18.9] CC: Actually, TensorFlow does a lot of that work for you. When you feed in your training data and you're going to the training process and you have your labeled data, TensorFlow will use back propagation to update your weights your neural network to make sure that it's predicting the output.

Honestly, that is not something that, as a programmer, I really have to worry about because TensorFlow is so powerful, providing the different algorithms and updating all that; the weights and everything for me. Basically, with that, it was just an experiment on choosing the different algorithms in TensorFlow for using the back propagation to update the weights. That was the only real work that it required from us.

[0:25:06.9] JM: That's fantastic. I would much rather here you don't have to know how to do this thing than a deeply technical explanation for how back propagation works, because every time I've had somebody explain how back propagation works to me, I've walked away a little confused in terms how to implement it, but I understand what it does in the abstract. Like, "Okay, back propagation, when you have a training example and that training example gets confirmed as having a certain outcome or having a certain level of accuracy, you can feedback that information into the neural network in order to train the neural network further.

But if you don't actually have to know how to do that mathematically, you just have to say, "Okay, this is something that I want to be improved overtime." That's a lot more easier. It'd like when I use Ruby on Rails for the first time, I was like, "Well, this is my first time writing a web application. Oh, wow! That was really easy." I don't have to know how IP addresses are mapped to domain names or something like that where it would have been in the earlier days before Ruby on Rails had to have more lower level implementation. Just takes care of it for you.

[0:26:19.2] CC: Yeah, that's the beauty of a lot of the machine learning libraries that are coming out right now, is it's honestly like the first web or first web frameworks coming out. You don't really have to all that theoretical knowledge. It's really nice to know what's going on in the hood and you probably will need to know what different activation functions do, but you can really get a lot done just with putting different pieces together and doing some experimentation.

[0:26:48.9] TB: I think I want to add to that is that we're probably really are about to see a huge explosion in artificial intelligence or machine learning or whatever and it's not because we have more people that are theoretically capable at implementing those things because we have all these pieces that you can kind of put together, which is one of the great thing in my opinion about software engineering in particular that you're able to build on what other people have done and you don't need to have all that math knowledge upfront and you really kind of like learn it over time.

As you start building things, you start to sort of gain that theoretical, understanding as you start using the applied components. That's really exciting for us as a company, but just in general it's really exciting because all kinds of companies, small and large, are really going to start to be able to leverage this stack to make their products better, and I think that's really exciting. We really are on the cusp, I think, of complete revolutionizing of how we manage data and build applications.

[0:27:53.3] JM: I definitely think so too. Do you guys think that this is something that you can explain to a lay person that's not a computer scientist, or an engineer, because essentially what you have to describe to people to explain machine learning to them accurately is you've got training examples that are examples of how certain variables have correlated with an outcome and you are using that to frame reality in a certain way so that when new examples come in, you just map them to this correlative graph that you've developed and that's all that machine learning is, and you just update this overtime.

It seems like explaining that to people is really more a process of explaining — This is really just the scientific method. It's just what data have we seen and what are we looking at going into the future.

[0:28:54.1] TB: Yeah. I think it's interesting because I believe anybody with really basic math understanding or basic statistical inference understanding, you can explain what machine learning is too with the caveat that as long as you do it in a way that doesn't overwhelm them. Kind of honestly, my personal sort of criticism of the machine learning space, or AI in particular, is there's a lot of sort of hype around it and this sort of magic that people talk about. They don't really — I don't know what it is. It's almost like some people just like to sound intelligent, so they talk about things like machine learning, like it's Skynet or something, and it's really, like you said, at the end of the day it's the scientific method. It's statistical inference at a massive scale and that's all it is really.

In terms of explaining it to people, I think if whoever is explaining it is able to sort of check their ego, for lack of a better expression, and say, "Really, what we're doing here is we're not building a human mind here." Even the concept of a neural network is sort of overwhelming to think of it that way. Really, at the end of the day it's just a way to horizontally scale statistical inference.

[0:30:16.5] JM: As you're improving the machine learning model overtime, have you started to dive deeper into how these things work? Because a lot of it under the hood is complicated, but you just made a great point which is that it's probably not good to emphasize these complications because you might scare people off, and there's a lot of low-hanging fruit that people could be attacking if they understood how easy machine learning was to work with, but they get intimated by, for example, matrix multiplication, linear algebra. Is it helpful to know that stuff? Especially as you have to improve and scale the model or just improve it overtime, have you had to learn more about how machine learning actually works? What exactly are you doing on a day-to-day basis to improve the model?

[0:31:06.7] CC: Definitely is helpful to have a good intuition of how the algorithms work, because honestly you can reduce a lot of the experiments instead of just trying everything. If you know the certain activation function, for instance, performs well on this type of data, then you don't even have to try to other one. That's one of the benefits of having the good domain knowledge in machine learning is, "Okay, I can eliminate. I don't need to try these different methods because I just know that it doesn't perform well on this type of data."

Honestly, it's good to know, have the deep understanding of it because if something does go wrong, say, a specific prediction is not performing well. It really helps to debug why. For example, this activation function or this structure might be performing weirdly in this case.

[0:32:06.7] JM: On the infrastructure side, you guys are using Kubernetes. Give me story of how you started working with Kubernetes, how you deployed it.

[0:32:16.8] TB: Yeah, that's a great story if I may say so. When we started working here at Generation Tux, or at least when I started, one of the core sort of principles that we wanted to follow was — One; we wanted to use Docker because I've used it in other places. It was really when it was just starting that it really exploded. It gives you so much ability in terms of leveraging continuous deployment and making sure that the code works from your local machine all the way to production.

As we started to look into how do we deploy that into a production environment in a reliable way, we looked at a couple of alternatives. We looked Docker Swarm and we looked at just literally having a Docker host, maybe MVP, we just put the Docker containers up there and managed them ourselves. We quickly found out that to manage it yourself, meaning you just have a Docker host that you deploy Docker containers wasn't going to work for production, because containers by definition are very ephemeral and you kind of almost have to count on them eventually dying which is not really acceptable if you have just one container running for a production environment.

As we sort of explored the options, Docker Swarm and Kubernetes were kind of the big ones at the time. This was in 2015. Basically, Kubernetes was pretty easy to get started with, in particular, on the Google Container Engine which is where we started. We were able to get that up and running pretty quickly on Google Cloud Infrastructure and we were able to get off and that served us well for several months.

The big reason behind Kubernetes was because we were using containers and because we wanted to do a micro-service architecture and because we wanted to do continuous deployment, so we had to come up with a solution. Basically, Kubernetes was oddly enough, because of Google was the least barred entry because Google managed it for us.

What we found was, at least at the time after several months, there were certain things about the Google Cloud Infrastructure that made it difficult to manage stability. For instance, Google would update the Kubernetes master nodes, update the versions. We had a couple of times where that resulted in backwards compatibility issues with our version of Docker, etc.

[0:34:34.5] JM: Oh no.

[0:34:35.2] TB: Yeah. Basically, if we're going to use Kubernetes, we needed to have control over the master nodes and when it got upgraded and all that sort of thing.

We since moved to AWS and we just host Kubernetes on our ECT nodes building it out with Terraform. Nowadays, there're dozen of open source things that are all about getting you up and running with Kubernetes quickly. At the time, when we did it, this was a couple of years ago, there were not. We sort of implemented our own roll your own Terraform sort of implementation. The thing is once you build out that infrastructure, if you do it right as like an infrastructure's code with Terraform and some provisionners, it really is not that much to get it up and running and it's been very very stable for us since then.

Kubernetes is basically is taking over the container orchestration space. I haven't really actually seen anybody using Mesosphere in production. I'm sure it's out there, but if you want containers running in production and you're not running Hadoop jobs or something like that, then it really is a great solution. With the tools today, it can be off the shelf and up and running with not that much effort.

[SPONSOR MESSAGE]

[0:36:02.8] JM: Simplify continuous delivery GoCD, the on-premise open-source continuous delivery tool by ThoughtWorks. With GoCD, you can easily model complex deployment workflows using pipelines and you can visualize them end-to-end with its value stream map. You get complete visibility into and control of your company's deployments.

At gocd.io/sedaily, you can find out how to bring continuous delivery to your teams. Say goodbye to deployment panic and hello to consistent, predictable deliveries. Visit gocd.io/sedaily to learn more about GoCD. Commercial support and enterprise add-ons, including disaster recovery, are available.

Thank you to GoCD and thank you to ThoughtWorks. I'm a huge fan of ThoughtWorks and their products including GoCD, and we're fans of continuous delivery. Check out gocd.io/sedaily.

[INTERVIEW CONTINUE]

[0:37:04.7] JM: That is quite an interesting history. Just to clarify, the container engine, Google Container Engine, this is the platform as a service for Kubernetes. It's surprising to me that it was not as performant as you would have liked and you ended up porting your Kubernetes infrastructure to a self-managed AWS cluster. When was this? This must have been early days in the container — I feel like the container engine has stabilized since then.

[0:37:35.3] CC: Yes, it has. I think our first deployment of Kubernetes might have been like 1.0 or I'm not exactly sure what it was, but it was basically towards the end of 2015 and — In terms of performance, it was fine. It was updating the master nodes without us really being able to even know what happened. That was the problem.

Basically, with our own infrastructure, it just gives us the ability to sort of upgrade Kubernetes at our own pace because we have other fish to fry as it were sometimes. We tried to keep up to the latest version as possible, but at the time it just wasn't acceptable for us the way they managed that. Honestly, it may have improved dramatically and maybe there was something we've missed about that. I don't want to speak ill Google Container Engine because I'm sure it's — It served us well for several months and it may have improved. As you know, the Kubernetes container orchestration space is moving so fast. It's really mind-blowing.

[0:38:37.0] JM: Talk more about why Kubernetes was so useful and why you wanted to get on a — What is the motivating factor behind getting a container orchestration system?

[0:38:50.2] CC: It's really — Like we said, just using Docker and just hosting a single Docker image in production is just not — You don't have the reliability. It doesn't have the scheduler in Kubernetes, for example, keep all the containers alive and it will perform the health checks that's needed to make sure that that container is running and then clients can actually reach that code. Whereas with Docker, just a single Docker container, you'd pretty much have to roll that all on your own.

Also, Kubernetes just provides a lot of easy ways, deploy your containers to different environments and continuous deploy, and it gives you a lot of high-level primitives. For example, our machine learning service gives you the ability to wait until all of your data is loaded into memory before it all starts switching over traffic and doing a rolling deployment to the new service, which is really helpful for us.

[0:39:49.3] JM: Okay. You talked a little bit there about the interaction between your machine learning code and the Kubernetes orchestration system. Explain more what those synergies are how TensorFlow is interacting with Kubernetes, or I guess how your machine learning models are interacting with Kubernetes.

[0:40:07.7] CC: Our machine learning models are in a Docker container. Basically, we have all of our basic files that we have to load into memory. Our machine learning models are persistent to files that need to be loaded into memory. To do that on deployment, if you would just deploy it and not have anything in place to make sure that it was actually ready, the clients that were just hitting those endpoints, it would just timeout because the data isn't loaded into memory because it takes a little bit of time.

With Kubernetes, it allows you to define a timeout period to say, "Oh, I need this to wait 60 seconds to load into memory, and then hit this endpoint to make sure that everything is okay. If everything is okay, then start switching the deployment." Those primitives in Kubernetes really made a really hard problem super easy.

[0:41:00.9] JM: Okay. The dataset gets processed through TensorFlow. Your machine learning model gets output from TensorFlow in the form of vials. Describe what those output files are after you run a machine learning job through TensorFlow, because for people who don't know, with machine learning you train a model and then you deploy the model and then the model serves new requests because you're basically — A new request is asking the question, "How does this dataset that I've just received," like if you're got a user somewhere who's submitting their pant size and their shoe size and so on, they're asking the question, "What size tux do I need?" That's getting submitted to the machine learning model that's already been output from TensorFlow. Describe what that output from TensorFlow looks like. What kind of file format it's in and also how it gets deployed to one of these containers.

[0:42:04.9] CC: We use the H5 file format. Basically, I think it's just a binary data file format. They don't have any sort of human readability to them. It's just binary data that the TensorFlow knows how to read back into memory later on at a certain point.

To get those on our container, basically we will check those files into our git repository. We try storing them in S3, but we came with some latency of downloading the files every time we wanted to build the container. We decided it's just best to check your files into the git repository. Basically, when you build your Docker image, all your files are contained, self-contained in the Docker image along with your code. Then once you deploy the container to Kubernetes, basically, we have a script on startup to just start loading those files into memory.

[0:43:07.6] JM: Is there a terminology for this? Is it TensorFlow serving? Is that the serving process when you're actually serving the machine learning model requests?

[0:43:16.3] CC: That is one of the ways that you can deploy to production, but we actually had some trouble with getting that to work and we really didn't have the resources to completely go all in on that. We decided to kind of use the file format instead of using the TensorFlow serving.

[0:43:34.4] JM: Okay. I remember there were some period of time where a lot of the shows that I did around machine learning, this is before TensorFlow had really hit a lot of market penetration. People were talking about the difficulty of getting machine learning models into production. There would be some sort of mismatch between the training process and the deployment process. Do either of you have an understanding of why that was a problem and what TensorFlow did to solve it?

[0:44:07.8] CC: Honestly, one of the things that we saw was we needed a verification step to make sure that once we trained a new model and we were loading it on to the Docker container that everything was still functioning correctly. In order to solve that problem, we actually have something like API test that basically have a few data points to make sure that everything is returning correctly to make sure nothing was messed up when loading that model into memory. That's another place where Docker really helps because we could just spin up that Docker container in a non-production environment and it gives us the reproducibility to make sure if it works in that non-production environment against our test dataset, that we can just push that into production with confidence and make sure that nothing is — The predictions aren't blowing up.

[0:44:59.5] JM: I'd like to talk a little bit about how engineering works at Generation Tux. You've got machine learning. You've got data collection. You've also got a website that you need to keep up and running that serves user requests. Can you break down how the different teams are interacting? Because you obviously need the data from the users to feed into the machine learning model, so there's an integration point there. You need the machine learning models to be served by Kubernetes, so there's some integration there. Can you describe how the different teams work together?

[0:45:36.0] TB: That's a great question. We have sort of a team division along the ecom, which is the main site as you say that supports user experience, and marketing, and etc. Then we have sort of an operations team and a dev ops team. We do divide that way. The operations team basically supports a lot of the backend stuff and then we also take on the role of sort of manipulating data and training the data models.

One of the things that we really emphasized at Generation Tux as a team is that there really aren't silos. Specifically, from the time we built this team, attempted to make everyone sort of capable at least in one area of expertise so that our teams kind of rotate almost organically as when new priorities come up.

One of our big priorities last year in machine learning was the fit, that sort of thing, and trying to figure out how to mitigate that friction about the tape measure for our customers. That sort of took a high priority and we worked together as a team to do the data training, whatever changes we would have to do on the site. Someone mentioned recently that on our team our bus factor is very low, which is any one person could get hit by a bus, God forbid, and we'd still be able to continue to achieve all of our objectives.

As an organization, we emphasize less technical expertise, although we are very technically capable and more about whether people are capable of learning new things, whether they want to learn new things and whether they're capable of working with other people, because, to meat least, that's probably the most important thing is being able to collaborate and argue and have discussions about things and then to be able to deliver at the end of the day.

That's the important thing, is that we're basically a team of teams and we've very cross-functional on every single team. The ecom guys are the guys that work on the frontend could easily jump on the backend and even work their way through the TensorFlow stuff if need be because one of the other things that we do on a regular basis is we have innovation talks where people that are interested in things, like for instance, we've been doing a lot of GraphQL for many of the new APIs we've implemented and that was just an initiative of one of the other engineers that gave an innovation talk about GraphQL. We have innovation talks about functional programming, etc., and that's kind of how we share the knowledge and then challenge each other's ideas and then we all become capable in basically every element of the stack.

[0:48:15.8] JM: There seems to be some correlation between that improvement in the bus factor a lot of the different companies I talked to and the micro-services movement, because I guess there's just fewer people who are building these tightly coupled monoliths. I just remember seeing these large Java monoliths earlier on in my career where you have to just tightly coupled, and it's hard to even — If you draw it out on a whiteboard, even then, it's too complicated to understand. When you draw out these micro-services architectures on a whiteboard these days, even if it's a complex company, it's easier to understand how the information flows through the business system from my point of view.

[0:48:59.5] TB: Yeah, absolutely. The other advantage of that, like you say, is just to be able to treat the services like black boxes which is — Like you said, for onboarding new people, that's great, because if they're going to be working on the UI or whatever, I can just say, "This is the thing that manages our accounts. This is the thing that manages our shipments. Don't worry about that right now, but just know that it takes this input and gets this output."

Even the fact of getting them up and running to start developing with a micro-service is a lot because they just get the UI running and they don't have to worry about all these other things with a monolith and be like, "Well, you got to make sure you have Mongo installed and you got to make sure you have MySQL installed," and it's like, "Oh, well I just wanted to change this header tag on this marketing page." That's been a huge benefit.

Then the other pieces, very important, at least form my perspective, is our continuous deployment, because there's an interesting thing that happens with developers when they know that when their code gets merged, it goes to production. The interesting thing that happens is, suddenly, tests matter, and suddenly making sure that what I just worked on actually works matters. Versus knowing it's going to the QA environment. It suddenly becomes easier to merge on a Friday and be like, "Oh, it's probably okay."

[0:50:25.7] JM: Okay. I want to begin to wrap up and just talk about the business and the plans for the future. Your founder is this guy; George Zimmer, people who have Men's Wearhouse in their country, they've probably seen the commercials where the guy who says, "You're going to like the way you look. I guarantee it." That's George Zimmer. I recognized him when I was looking at the Generation Tux website. He kind of looks like the most interesting man in the world.

This is a cross section with the question of Amazon, because retail is getting really threatened by Amazon. It's having to reinvent itself and it seems like Generation Tux is a pretty strong reinvention of the men's wear area and it's a domain-specific area with a lot of money coming through it and it's probably something that Amazon is not going to replicate anytime soon. Can you describe how George Zimmer thinks about retail and how that's led to the engineering organization at Generation Tux?

[0:51:39.3] TB: Yeah. The men's fashion industry is pretty important to him because he was such an instrumental part in building the Men's Wearhouse brand which he's no longer associated with Men's Wearhouse. He had a bit of a fallen out with the board there.

One of the important parts of that whole business; Men's Wearhouse in particular, was tuxedo rental. We talked about it as a company a lot that only is tuxedo rental sort of being very disrupted, but the whole men's fashion industry is sort of in disarray at the moment. For instance, Men's Wearhouse's one example is they were worth just a few years ago $2 billion. As of today, they're worth $500 million. They're struggling a lot. There's a lot of other — Macy's is struggling. A lot of the brands that traditionally served for men's clothing have gone by the wayside. As you say, there are some picking up of the slack there with Amazon, but for us as a company, our big goal is to be dominant in the tuxedo rental space and really change the way that looks for people and to really not only add value by reducing costs, but also to add value by making the process simpler.

Then beyond that, really, I believe that our big vision is this sort of change the whole men's wear space starting with Tuxedo rentals. Like you said, it's not something that Amazon could really pick up right away. I'm not sure if they're even working on it, but because this business of tuxedo rental is so oriented around weddings, that it takes a special kind of sort of engagement. The buzzword is high touch, but it really is true in our business because you really have to engage with the bride and the wedding party because you can't mess it up. There's no do-overs with weddings, and that's one of the things that George Zimmer really is passionate about and brings to the table because he has that experience and he's been doing weddings and tuxedo rental for so long. It's about building that trust that we talked about earlier.

[0:53:53.5] JM: All right. It's been really interesting talking to you guys. I'm glad that we connected. This episode was sort of reminiscent of — Not completely, but somewhat reminiscent of a show I did about this company; Golf Now that, migrated to Kubernetes and just saw a lot of advantages. They migrated much older legacy infrastructure. You guys are certainly newer, fewer rings around the tree, but nonetheless it's clear that there's a lot of advantage to this Google infrastructure for everyone, whether it's Kubernetes or TensorFlow. It's just interesting seeing case studies in people who have benefited from these new technologies for, widely, varying reasons. Clearly, this stuff is really having a lot of impact. Thanks for coming on the show you guys.

[0:54:48.1] TB: Yeah, thank you very much, Jeff.

[0:54:49.5] CC: Thanks for having us, Jeff.

[0:54:50.9] JM: Cool. All right, thanks Colan and Thomas.

[END OF INTERVIEW]

[0:54:56.0] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. That's S-Y-M-P-H-O-N-O.com/sedaily.

Thanks again to Symphono for being a sponsor of Software Engineering Daily for almost a year now. Your continued support allows us to deliver this content to the listeners on a regular basis.

[END]