SLW26 Agentless with Jordan Borean

Red Hat engineer Jordan Borean discusses his journey from his early interests in IT to his role at Red Hat. He talks about how scripting and automation made programming easier and how he became involved in Ansible for server management.

Artist: Thorsten Butz
Album: Sliding Windows
Year: 2026
URL: **http://www.slidingwindows.de**

Automatic Shownotes

Chapters

0:30 Roots in Computers
2:20 From Bank Sysadmin to Ansible
6:35 PowerShell Enters the Picture
8:59 Windows, Linux, and Mac
13:11 Fedora, Desktop, and Choice
17:44 Linux Fragmentation Debate
22:34 Editors, VS Code, and AI
25:42 PowerShell on Linux
28:50 Curl, JQ, and AI Help
31:38 Linux Dominates Infrastructure
35:47 Red Hat and Open Source
39:24 Open Source License Views
41:09 Linux Beats BSD?
43:01 How Ansible Began
47:26 What Ansible Actually Does

53:24 Ansible Versus Windows DSC
1:04:06 Future of Ansible
1:05:56 Developer Experience Matters
1:07:44 Idempotence in Practice
1:12:42 Why .NET Matters
1:16:52 Curiosity and Learning

Long Summary

Jordan Borean is introduced as a Red Hat software engineer in Brisbane who works on Ansible core, and the conversation frames his career as unusually “agentless” in the sense that he moved into the role by following practical problems rather than a fixed plan. He recalls being interested in computers from childhood, studying IT at university, disliking programming at first, and then taking a sysadmin-style internship at an Australian bank working with Windows Server, MS SQL Server, Java, and Groovy. A major part of the interview covers how Ansible entered his work. Jordan explains that the bank wanted to move from hand-managed servers to cloud instances and more automated deployment. Ansible was attractive because it was free and community-driven, but Windows support was immature in that environment, with issues around WinRM and security settings. To make the project work, he started learning Python, contributing pull requests, and eventually moved into a Red Hat role focused on the Windows side of Ansible. The discussion then turns to his operating system background. Jordan says he grew up entirely on Windows, later experimented with Apple devices, and only learned Linux through Ansible work. He now uses Linux as his main development environment and has tried several distributions, including Ubuntu, Arch, Fedora, and openSUSE. More recently he switched to macOS for the hardware, especially battery life, but still runs a Fedora VM for development because he prefers Linux tools and customization. Jordan also talks about Linux desktop choices, window managers, and package systems. He says Fedora fits well as a middle ground between long-term-stable distributions and rolling releases. He prefers tiling window managers and says too many customization options can create confusion. On package management, he values the trust and vetting of system packages, but also sees strengths in the diversity of Linux packaging approaches. The conversation moves to PowerShell, Bash, REST tooling, and automation. Jordan says he never became comfortable with Bash, while PowerShell’s object pipeline and interactive discoverability made sense to him. For REST work, he says AI tools now make curl and jq more convenient than before, reducing some of PowerShell’s advantage for quick command-line tasks. A large section focuses on Ansible itself. Jordan describes it as an orchestration tool built around YAML playbooks and modules, with an agentless model that talks to systems over SSH, WinRM, and other transports. He contrasts it with Chef, Puppet, and DSC, saying Ansible’s Windows story is helped by its ability to call DSC resources, but that DSC’s tooling and performance are limited. He also says Ansible’s idempotency depends on the modules used, and that the agentless model is sometimes less suitable for endpoint-style or pull-based scenarios. Near the end, Jordan discusses .NET and C#. He says he likes C# a lot, values the performance and developer experience of .NET, and wishes he could use it more professionally. He also notes that .NET’s open-source, cross-platform direction matters, but that languages like Rust and Go have strong momentum. His current work includes improving developer experience, debugging support through the Debug Adapter Protocol, PowerShell debugging

inside Ansible, and making Windows integrations easier, while also continuing to push performance improvements. Jordan closes by saying his main advice is to stay curious, question details, and try different tools and languages rather than following guides blindly.

Brief Summary

Jordan Borean describes an “agentless” career path into Red Hat, starting from childhood interest in computers, an early dislike of programming, and a sysadmin internship at an Australian bank before moving into Ansible work. He explains how Ansible became central to his job as the team tried to automate Windows-based deployments, and how he learned Python, contributed fixes, and eventually focused on Windows support at Red Hat. The conversation then covers his operating system preferences and tooling. Jordan talks about growing up on Windows, learning Linux later through Ansible, and now using macOS hardware with a Fedora virtual machine for development, while also discussing Fedora, tiling window managers, package management, PowerShell, Bash, and command-line REST tools. A large part of the interview focuses on Ansible’s design, including its YAML playbooks, agentless model, WinRM and SSH transports, Windows integration through DSC, and the limits of idempotency and pull-based use cases. He also discusses C# and .NET, his interest in developer experience, debugging support, and performance work, and closes with advice to stay curious and keep trying different tools and languages.

Tags

Jordan Borean, Ansible, agentless, Windows support, Red Hat, Python, YAML playbooks, WinRM, Fedora, developer experience

---

Transcript

Thorsten:
[0:24] Welcome to a new episode of the Sliding Windows. My guest today is Jordan Borean.

Roots in Computers

Thorsten:
[0:31] Jordan comes from a land down under, but thankfully that doesn't stop him from popping over to Europe every now and then. As I'm not particularly keen on remote interviews, I took the opportunity to interview him in Wiesbaden during the annual PowerShell Conference Europe, as I've done so many times before. Jordan works as a software engineer at Red Hat in Brisbane. playing a key role in developing the Ansible core. It would have been easy to simply title this episode Ansible. However, I opted for the somewhat ambiguous term agentless, as this not only describes a key feature of the Ansible software, but also serves as a metaphor for my guest's unique career. But let's hear from Jordan himself. I hope you enjoy our conversation.

Jordan:
[1:22] I had a fairly good idea what I wanted to do growing up. So I remember when I was maybe four or five years old, we had an old Windows 95 computer that was breaking down all the time. And there was a technician that would come in, he would open up the side of the computer, fiddle with all the components and motherboards and stuff. And I feel like that maybe put an impression on me. I really liked computers. I was interested in it. Went to school, did all the sort of normal stuff. And then I decided I wanted to study IT at university because I was interested in that. And then I wasn't sure what side of IT. IT is so large, like you've got programming, networking, all that type of stuff. And I was just sort of working around and seeing what was involved there. And funnily enough, while I'm a software engineer now, I find that I actually hated programming back in university. It was something that didn't sort of come naturally. I didn't really like the subjects that we did on it. I didn't find them good

Jordan:
[2:17] and educational, so they just didn't adapt. And then graduated, I got an internship for a bank in Brisbane, where I'm from, and it was more of a sysadmin type of work. So we mostly deal with Windows boxes, a lot of sort of the Microsoft stack, so like MS SQL Server.

From Bank Sysadmin to Ansible

Jordan:
[2:37] Windows Server, all that type of stuff and i can't say that we were very interested in powershell at that time so we had a, a solid background in the team that i was with around java and we had a programming language called groovy which is sort of a java like dialect um and we use that for a lot of the sort of scripting automation and some of the tools that we had in the actual team itself and And I feel like that gives me more of, a real-world application around programming and how that works in an actual environment for business rather than sort of the theoretical... Use cases that you get in the university and over time like I got better and we learned a whole bunch of different things like CICD um, programming type of elements like sanity tests and test-driven development and all that type of stuff and I got more and more comfortable, I enjoyed it and had some very good mentors as well um in the team that sort of helped foster my knowledge and sort of helped me along because, when you're new you're new you don't really know too much um.

Thorsten:
[3:42] So you're a Red Hat employee for quite a long time

Jordan:
[3:45] Yeah well this is even before Red Hat so um but i've been with Red Hat for about i think eight eight nine years now so it's it's a fairly decent time yeah um and then at that previous job we, decided okay let's try something different we were a bank so a lot of vms physical servers were treating them sort of like pets where we're hand crafting them making sure they're staying alive and stuff like that, we had a project coming on where we're trying to change things up. We wanted to spin up instances in the cloud, treat them more like cattle, where we can sort of blow them away, reconfigure very easily. And that got us onto Ansible. So it was being used in our company for things like developing, deploying web servers and other things like that. And because it was

essentially a free product, so a community driven one, it was very easy to use, and sort of sell to the people saying, hey, we don't have to pay money for this one here.

Jordan:
[4:40] We found that because we were a Windows side, but we wanted to use a traditionally Linux tool, they only sort of was very new to the area on the Windows adoption, so a lot of the things didn't work in our environments. We had a higher security tolerance around WinRM settings, they only allowed basic auth, which wasn't allowed in our environment, and a whole bunch of issues that came up because just no one had done it before. So because of that, and I wanted this project to succeed, I'm like, okay, how can I make this better? How can I improve this tool to make my job easier? And I started getting into Python, which is what Ansible is written in and all the dependencies and started to get more and more into it, more into the programming side, submitting pull requests to GitHub, which was a first for me. I hadn't used it before and sort of interacting more in that open source world and the people in the actual Ansible team.

Jordan:
[5:33] From there, they saw the work that I was doing, they seemed happy with it and they decided, they asked me if I was interested in a potential future role, I said sure why not, like it would be fantastic to try it out, and then a job eventually opened up around Ansible and the Windows side, so essentially what I was doing on the side from my previous job to start getting paid for it, I'm like sweet I'd love to do that but it'll be awesome.

Jordan:
[5:58] I started that one there and it's been yeah eight or nine years and while I'm still in the same role some things have changed a bit but, I really like Red Hat and the role that I do right now especially because they give me a lot of leeway to deal with not just what I'm currently working on but I have the opportunity to try different things and, things that may not be necessarily 100% related to what I'm working on so I can work on various protocols and sort of figure out okay how can I re-implement them in Python and,

Jordan:
[6:29] sort of just learn different topics and use what I learned on Ansible and deal with that side there. And that's essentially what got me also into PowerShell. So because we weren't using PowerShell on the other side, Ansible uses PowerShell for Windows management. So dealing with that type of stuff. And I started getting, well, I had to use it because that's what we're using on the remote side. And then eventually I met a few people through Ansible around that's, in the PowerShell community, more on the the American PowerShell side and then they introduced me to the Discord group so the PowerShell Discord that's on there and then more and more people and just interacting with that has been.

PowerShell Enters the Picture

Jordan:
[7:11] A massive boon to my knowledge so they're not just like issues that I've had but also seeing what other people go through and what are their issues

and sort of just problems that they come up with that sort of had that little touch in the back of my head saying oh that seems fun let's try that out see what's actually involved and, I've just been using that ever since and I've learned more and more while I probably say I do have a fairly decent knowledge there's always there's more out there and always more problems that I've never touched. So I'd say that that's essentially been my journey towards now. And I started PS Conference, EU and Antwerp two years ago, and I haven't been looking back ever since.

Thorsten:
[7:50] Yeah, you mentioned PS Conf EU because we're just here in Wiesbaden this year. We celebrate the 10th anniversary of this flavor of PowerShell conference. And we're happy to have you with us. And I very often take the chance to record interviews with guys like you. So, great. And it's funny you mentioned it, that you're being a Linux professional. That's not the thing that people expect, and that makes it even better. If we compare the things that you just said, we can compare PowerShell and Python. We can compare Red Hat with Windows. We can compare Ansible with Group Policy, DSC, whatever you like, if there is a comparison. Let's go through it all. You mentioned Windows 95 as a child or a young guy. Have you ever been a user of a specific platform in your private life? Was that, first of all, Windows? Or did you find out Linux is much greater or even Mac OS? How was that before you started working as a professional?

Windows, Linux, and Mac

Jordan:
[9:00] So growing up, I was Windows only. So we grew up with Windows XP 7, Vista 7, all the ones in there. And when I was in university, so after high school, I would probably say I was a pretty strong Microsoft supporter. So I liked the Microsoft products, Windows. I even have a Windows phone at some point in time. I was one of the 12 people in the world that did. But no, like I was a very strong Windows supporter and I sort of just liked it. But I think maybe that was more just I wanted to be part of a camp type of thing. So I wanted to have a tribe to be part of.

Jordan:
[9:35] Starting my job outside of university, I was still on the Windows side, but I sort of started to dip into the Apple ecosystem. So I got an iPhone for the first time, which shocked my family because I was always anti-Apple and it was the worst thing in the world. And I sort of grew out of that type of phase. And then as I started my job with Ansible, I got more and more experience with Linux. So before I started there, I hadn't had any experience at all. I was brand new, I was all Windows, and I learned sort of a bit of the command line things actually run Ansible itself, but I was still fairly green. It wasn't too much on there. From there, I started to get more into the general ecosystem. So you're dealing with Microsoft, a closed source product like with Windows, and then you have Linux, which is a bit more open source, all the various distros that are out there and all the custom distros and what you can actually do with that one. That sort of opened up my mind a bit. I wanted to try different things out and see what was possible. So I branched more into Linux in that side. One, because it was part of my job. Two, I just wanted something different. so give that a try, i've been running linux as sort of my main dev machine since then so for like eight years now i lean more towards the red hat products because that's what i know that's that's what i work for but i tried a whole bunch of them so i've tried ubuntu, arch fedora um open sus open sus i'm not sure what it says they call it in germany.

Thorsten:
[11:04] Um because that is the name because but that is a distribution, but we know what you mean. Yeah.

Jordan:
[11:11] So I've tried them all and tried the different ones. I have recently moved back to mac os well not moved back this is my first time using an apple hardware and mac os the reason why i did that was around the hardware so the really good battery life, um i found that sharing my screen with linux and some of the multimedia aspects were a bit limiting like i could get at work but there was just so many workarounds and different things that i had to do so let's try apple and, i love the hardware fantastic hardware i think they've done an excellent job with that i'm not as much of a fan of the software i don't like the lack of, like freedom that i get around customization so i still have a vm that i run on my box which is where i do all my dev work and that that's currently fedora right now but i'm sort of used to the, gpl like tools like gcc and bash and all that type of stuff where you might be able to get an equivalent on macOS, but it's just not the same.

Thorsten:
[12:14] Yeah, it's a little bit strange, and that's true. In my age, Apple always were the good guys, and we suffered from the fact that they almost disappeared. And then, the rest is legion, and you don't need to tell the story again and again, Steve The job came back, the iMacs were invented, the iPods were published, and then the iPhone, which was yet another miracle. The fun thing is that, if you think about it,

Thorsten:
[12:50] The time when PowerShell was invented was almost at the same time when the first iPhone was published. So that, to me, around 2006, 2007, that was a great time, right? So many things that really changed my personal life and of course also my professional life. Smartphones are more than a tool.

Fedora, Desktop, and Choice

Thorsten:
[13:11] The interesting thing is, I never liked that split into Fedora and Red Hat, being Red Hat. So that leads to the question, I know a bunch of guys who use Fedora on the desktop to be more current, because stability might not be as important as on the service side when you want to have more fancy stuff. Would Fedora for you be the better option if your company says, say you have to use our own product more or less would you would you prefer fedora on the desktop side

Jordan:
[13:47] I i think so so it's linux that there's so many different options and they scale from like you have debian and rel which are super stable um, very long life um packages that don't really get updated except bug and security fixes versus on the other side we have arch linux which like a rolling distribution, people do claim that it's pretty stable in terms of like things don't break but, it's broken a few times for me so i can't fully claim that but from a, developer perspective i like fedora because it sort of straddles those lines where it has sort of more up-to-date packages so newer linux kernel versions for newer features,

packages that might be a bit further up to date compared to what's in the past, but still not bleeding edge, so like you still get some time for them to vet some of those changes and updates that happen on there. I believe, I can't remember if it's every six months or every year that they push out a new release, but I've just found that it's hit the sweet spot, so that's why I really like it.

Jordan:
[14:53] I would recommend it for people wanting to use Linux if they're wanting something a bit more stable. So if they're not doing development-like work and they just want something that they can browse the internet, maybe send some emails or maybe some word processing if they're using something like the LibreOffice. Potentially they might be better off with a more stable one like Debian or RHEL if they can, well maybe not for home users, but of course... Essentially because they don't need some of the more bleeding-end features. They probably just want something that works and don't care too much, or I need to get version 6.1 of the kernel or something like that for some tweak type thing. So I think it really just depends on who's using it and what their purpose is for. For me, I just think that Fedora fits very nicely between the two different cams.

Thorsten:
[15:44] Yeah. And if you see what your colleagues are doing, what is most popular, or is it anything that can be?

Jordan:
[15:52] Yeah, I think with my colleagues, because Red Hat, they offer our laptops and we have to use a distribution, they say it's essentially between RHEL, Fedora, or MacAWITs. I would probably say nobody really uses RHEL itself as part of their dev machine, even though they would use it as part of maybe VMs for testing and other stuff like that. I would say it's a pretty even split between macOS and Fedora. So it really just comes down to whether you prefer the Apple hardware or whether maybe you're a bit more of a diehard open source Linux fan and you don't want to do that. But I think, yeah, 50-50 would be the split between those two.

Thorsten:
[16:31] Yeah. There's always a tiny, let me say, religious approach in the Microsoft, Linux, Apple world, right? It says you belong to a certain tribe and ignore all the other guys. But the thing in Linux is you're split into different favors. Let's say there are some guys that heavily focus on G-N-O-M-E. How do you pronounce that in Australia?

Jordan:
[17:01] I call it GNOME, like the fairy tale.

Thorsten:
[17:04] I've heard everything, right? That is, by the way, if I got it right, this one is written by Miguel de Icaza. By the way, he was for a very long time a Microsoft employee. Many people are not aware of that. And on the other side, at least back in the old days, you have KDE, which is a more or less German. There are always a lot of people involved, but the influence came here, heavily influenced by SUSE Linux. What's your spin on that? What do you prefer? And you can also talk about the underlying technology, Wayland X server. What's your religious approach in things?

Linux Fragmentation Debate

Jordan:
[17:44] From my approach, I feel like, oh, this will be sacrilegious, but I feel like maybe some of the fragmentation of Linux potentially is what held it back from the desktop market. There's yeah choice is great there's so many choices and it can fit someone but if you're on a computer you don't care about what it looks like you sort of just want whatever is the defaults and you want it to work type of things that's what i think apple and microsoft in a way has sort of done a really good job on historically, um from my perspective, i am a fan of tiling windows managers so there's like i3 which is maybe the the older um x11 based ones but then there's sway and hyperland and i find that that's given me a lot of, ability speed so like i have all the sorts of keyboard shortcuts and and things like that um now that i moved to mac os i don't, sort of wade into that debate too much but there are a lot of people that really dislike with Gnome 3, because they feel like maybe it's a bit dumbed down from what they used to with Gnome 2. I didn't hate it. It was fine. So I will admit that maybe they have pulled back some of the customization options that you can do with it, but from my mind I find that when products have too many options, it's too much confusion for people. So yes, it's great you can do it, but.

Jordan:
[19:09] I think maybe sometimes if you're just trying to be good, you want to try and make sure that your defaults are good versus making all these different options, which can break and interact with each other and lack of testing and stuff like that.

Thorsten:
[19:22] Yeah, that's the face to the customer, right? When you start with it, you see the GUI, and then, yeah, it makes you scared if you don't recognize what you're searching all day long because you were used to another window manager. That's a little bit awkward. There's another thing that... This fragmentation drives. We have that Reddit Package Manager as one of the very, very old tools, a great tool. But to me, my personal favorite would be the Demian Package Manager that was introduced in the late 1990s. And I'm always making fun in my workshops and my speaking opportunities and talk to people that Microsoft never could make it. And right now we have Winget, at least we have Winget, but the Linux world is so mature. But again, the problem, we are so split up into multiple solutions because that's not the whole story. There were more. Would it be nice if all the distributors would focus on a single solution, or is that fragmentation finally something worth because it drives progress?

Jordan:
[20:42] I think, personally, I don't mind the fragmentation around system packages. Yes, the CLI tools might have different interfaces and commands to use. That can be a bit annoying. But one of the big benefits that I've found with system packages is the fact that they're vetted by maybe more of a trusted authority. So it's essentially vetted by the people that's running the kernel and that type of side. So you get more trust with that one there. that side channel attacks is a really big thing these days. They're trusted, they won't break your system, and they align with what your distro is trying to achieve around is it stable, is it bleeding edge type of thing. There's been various attempts around sort of trying to unify that so flat pack um app armor um and then debbie not debian ubuntu's got snap, i think in that side that would be nice that that was a bit more consolidated because their aim is to be sort of a bit more universal um it seems like flat pack may have the edge but maybe i'm saying the wrong thing and i've made so many people angry by saying that, but I actually really

do like the diverse set of package managers in Linux. I think that is one of the key strengths and I know that's probably hypocritical with some of the other things that I said, but I think it does a lot of things right there and there's always things that go wrong. So the whole systemd versus.

Jordan:
[22:10] Initd and all the sort of the init systems that are on there i think that's a bit of a full of hot air with some of the viewpoints yeah um and i think that, yes the fact that linux has now made that a bit more, uh common between the various ones has been good for some of the packages that are out there but i know some people are very just have very strong opinions

Jordan:
[22:33] around that one absolutely.

Editors, VS Code, and AI

Thorsten:
[22:35] If you go back and just talking about text editors right the the one and only them, we are, struggling with the strange Emacs users, you know, but what I see now is that the younger generations, they don't care because they all seem to use Visual Studio code and its forks. That's highly, people love it. And so they don't care about that. But still, you see, one of the first choices you can make in an antitypical editor is that, would you like the iKeybindings or the Emacs Keybindings? Yeah. The younger generation, what's with that?

Jordan:
[23:17] Yeah, yeah.

Thorsten:
[23:18] So I tried that, for example, over the last month. I'm really looking forward to it. I don't know. Have you tried that, the editor?

Jordan:
[23:29] I haven't. No, there's a few people that have mentioned it to me, and I really want to have a look at it.

Thorsten:
[23:33] I have high hopes, let me say it like that. These are the people that The listeners of my podcast, I'm a history guy, I always tell the same stories, but they're all interconnected. And Visual Studio Code is sourced in Atom. And Atom is one of the foundational technologies of GitHub. Sounds a little bit strange, but read blog posts about that. And some people simply were able to speed up that Atom editor a lot. And that is what you today know as Visual Studio Code. But the main problem is, to me, it's written in TypeScript, which is more than ludicrous for

Thorsten:
[24:22] A couple of things that we would like to do. For example, use a shell. So there are some people who say that's not the right approach. We have to use Go or Rust or something like that. So that from the interface looks more or less familiar. It's just like Wizard Studio Code or any other of the siblings, but it's written in Rust, I guess, so not in TypeScript. So it's a completely different performance, and I do have high hopes. And the best thing that can happen is that they are simply interchangeable. If you see, today you have so many forks of Wizard Studio Code, I even can't count them. I tried to create a PowerPoint slide for one of my last talks, where I tried to explain a little bit, so don't take too much care about anti-gravity and windsurf and cursor and so on, because actually it's all the same. It's just different vendors, and they combine it with some AI extension typically, and a subscription model, and choose whatever fits your needs. But when you can handle the one, you can use any of the others, it's more or less the same. So that's the good thing, so that's actually standard space now today.

PowerShell on Linux

Thorsten:
[25:42] You talk about Python.

Thorsten:
[25:47] I always tell people in my professional life, I'm totally fine with the non-existence of PowerShell. If Microsoft chose 10, 15 years ago another technology, maybe Python, I would do Python trainings, Python consulting today, and I would be using this. But, of course, I'm kidding a little bit because we all love PowerShell, you love PowerShell, I love PowerShell, because for me, it's more or less the only semantic shell, this object orientation, focusing on rich objects. That is something that Python does not have that way, because Python is more a programming language and less a scripting language. It's got both flavors, but I think it's more influenced by programmers and it's meant to be a proper programming language. And of course, PowerShell can also be a real programming language, if you come to PSConfEU, you will say, but it's primarily meant for admins. To summarize that all, if you talk to customers that actually don't get it and they ask, are you using Linux and PowerShell? Does that make even sense? Yeah. What would you answer?

Jordan:
[27:13] So I find I've never been a fan of Bash. When I was moving to the Linux world, I just couldn't get my mind around things like sed and orc and the different syntax that's on there and dealing with all that. And I had dabbled in PowerShell a bit at that point in time. And I sort of knew a basic view of how it worked and I really liked one yes the objects that are in the actual pipeline, but I think the key thing that I think that PowerShell really shines is around the the REPL so what PS3 line offers around tab completion, some of the new history stuff maybe have some rough edges but there's some really good ideas and and if you configure it a bit, I think it works well for what I'm using for. So the discoverability aspect around just using it as a command line interactively is fantastic. I think that's one of the better ones out there. People have said that maybe fish, sort of another sh-based one, it has some really good ideas behind it. I probably need to try it out to see what it looks like, but then I also to give up on the pipelining aspects. So dealing with objects, things produced, it's a lot easier to filter them and select things and then work around that. So I think from that aspect, that's where I really think PowerShell's shine-ins.

Jordan:
[28:35] I don't necessarily think that it's winning the battle on converting Linux users. So typically the only people that would use PowerShell on Linux are people that have won, use PowerShell from Windows and want to have like a similar terminal experience on there.

Curl, JQ, and AI Help

Thorsten:
[28:51] I think the perfect example to make people aware of the options that you have is simply curl and jQuery. So many people that I see, they use these tools and a bunch of others to fiddle around with REST APIs, because REST APIs are very important for all of us, no matter which operating system you use. And yes, curl is a great tool, yes I know, and jQuery is mighty, but actually, for the first two hours, you are solving problems that you don't have with InvokeWebRequest and InvokeRestMessage, right? Is that the way that people should understand, I'm way more productive using PowerShell, for example, for REST API, for this kind of talk? Is that true?

Jordan:
[29:38] If you asked me that maybe two or three years ago, I probably would have said yes. So dealing with JSON and it would be a lot easier in PowerShell. Yeah. Now with AI tools, I think that it makes it so much more convenient to just ask a simple query, give me the curl command and the jq syntax, because I don't recall what it is off the top of my head, but it does. It's very good at sort of coming up with those snippets and the curl executable and all the functionality that's in there is very powerful. There's a lot of things that it can do, and while it might be hidden behind documentation with AI, it knows that already for you and it can figure it out. Yes, I think today I don't see that advantage too much in PowerShell. It still is nice to use, and it's good if you're used to that environment, but I wouldn't agree. I think with AI, it's tipped it in the favor of CURL and JQ and some of the other command line utilities.

Thorsten:
[30:38] But mind the gap for those who listen now, some people might tend to have a world view where the old legacy stuff is number one. And they don't want to use AI, so keep in mind that Jordan just said the combination makes all the difference. Yeah, you're true. KURL is super mature, super fast maybe in your environment. And if you tell the AI, Robert, write me his code snippet and explain that in detail, everybody can use KURL, everybody can use FFMPEG, which is a ludicrous complicated utility. But with the help of these utilities, you can fiddle around. But when PowerShell started, it was just focused on human beings. And without AI, I definitely prefer PowerShell for that. Yeah, that's an interesting thing. Use the tool that solves problems, right?

Linux Dominates Infrastructure

Thorsten:
[31:39] If we go one step further, Linux is the dominating operating system. And it's just like, if you go to Saudi Arabia, 99% of the people are Muslims. If you go to Australia, there are also a lot of Muslims, but maybe 80% are Christians or some other religions. It always depends on where you live, right? So what I

find out is that many Microsoft Focussed customers, they still think Linux is a kind of hobby project from a Finnish student. And they don't get that everything that we need today is built on Linux. Amazon wouldn't be there without Linux. Google wouldn't be there without Linux and so on.

Jordan:
[32:25] The thing in their pocket probably wouldn't work because it's running Android, which is Linux.

Thorsten:
[32:28] Yeah, maybe, yeah. Not in my case. I have the other one, you know, that with the Apple. Yeah. Which is not Linux. But what's your point of view? How dominant is Linux in our professional IT world? And how does it split with all the other operating systems?

Jordan:
[32:49] I think, yes, it is definitely dominant. So the sheer amount of Linux servers just powering the web is absolutely crazy. There's really cornered that market. And I don't think there's anything that Microsoft would be able to do to try and either claw that back. It's just crazy how deep they're in there. From, as you're saying, it does depend on the area that you're in. So like some industries might be more Linux focused, some industries might be more Microsoft, as well as the country itself. So Australia, I think probably has maybe, well, at least in the industries that I've worked in, so the banking and finance sector. Yes, we had some sort of mainframe systems and Solaris, AX type of things. But a lot of the stuff that I was working on was Microsoft. So .NET, IIS, MS SQL Server. But I'd say maybe more when you come to Europe, maybe some people use Microsoft, but there'll be other people that uses Linux. So I believe that both have their advantages and disadvantages. It really depends on the use case. I do have the personal opinion that Linux does have the edge in probably a lot more. And especially in today's world where there's less focus on sort of that click ops type of thing, which Microsoft has been more dominant in. And also moving to the cloud where Active Directory and a lot of that on-prem integrations have been less important.

Jordan:
[34:12] I think that the use case from Microsoft has definitely lessened compared to using Linux, which, yes, you might pay for support, but, something being free if you wanted to use those other distros is a very compelling argument for some companies.

Thorsten:
[34:29] I'm living in a country that does a lot in the Linux environment. We mentioned SUSE, which I guess was founded here, I don't know, but it was very influential here. KDE, the desktop environment, a very European project. And there are many more examples, but actually in many cases we couldn't make it that the industry embraces it. And then there are also political activists. Germany is also the country of the famous CCC, the Chaos Computer Club. You might have heard of that. Great guys, great environment. There is only one company that really understood that many, many, many companies don't want to have a completely open, free operating system, which might make you wonder. They might prefer a company where they want to have a contract and kick some ass if things do not work the way they expect. Just like back in those days with Solaris and Sun.

Red Hat and Open Source

Thorsten:
[35:47] Am I right that Red Hat is the only company who understood that approach, that a bank might prefer to pay for something that they could also get for free

Jordan:
[35:56] I i think so i think um red hat was probably one of the few companies at that time to sort of think about that and sort of take advantage of that situation because as you're saying, businesses don't care that something is free they want something that works and if it doesn't they want someone to fix it and they want to not pay so much money to fix it so i think being able to take advantage of that has been very successful for Red Hat and spawning the ecosystem that it currently has. Fundamentally, companies don't care that it is free. It's sort of... They're looking for something that works and microsoft has also been able to take advantage of that because yes they're a support company and, yes they may not be as prevalent if you go percentage-wise but there's still a fairly big player in the industry and they do that because one their products typically do work out of the box and they offer things that people are looking for, um type of thing yeah.

Thorsten:
[36:49] You mentioned that you live in brisbane brisbane and we can hear that but you're from Australia, a great rugby company, rugby nation. I like that a lot. From your point of view, you might not know that, but from your point of view, how is that in public services in Australia, maybe compared to Europe, if you can, if you have an opinion on that, is that public services focusing on open source software to avoid those US companies for software energy?

Jordan:
[37:24] They potentially might be. I haven't seen any sort of big pushes, but I am also not sure how much of the push away from Microsoft is either generated by the media to sort of sell clicks or whether how much of it is an actual groundswell. I assume it is real and that there is work to sort of move away from Europe, but I haven't seen that same sentiment in as much sort of ferocity compared to what seems to be coming out of Europe in the data sovereignty and other stuff like that. I don't think Australia has the... Technical ability to do what potentially europe could do so we don't have that historical aspect we don't have a distro that's based in australia i think, one of the major tech companies is at lacy and that's was base was set up in sydney but is a massive multinational company so if we're wanting to do that we're just trading the us for another country so in our mind i think it seems like the U.S. Is better the devil you know type of thing.

Thorsten:
[38:22] Yeah, that's a good point, yeah. And Red Hat, what is the headquarter where it's at? That's also in America, right?

Jordan:
[38:29] Yes, so you're asking where it's called headquarters? Yeah, Red Hat. Red Hat's headquarters is in Raleigh in North Carolina. Yeah, that's also in the United States. But then, I think maybe six, seven years ago, now they're bought by IBM. So an even bigger fish has gulped Red Hat, which I thought was big, but obviously not us. you can certainly hear the companies.

Thorsten:
[38:50] Yeah, so if you want to avoid USA-based companies, Reddit is not the company you're looking for, but it's still open source, so you have way more control over what you're doing there. Of course you have. But within these days, Microsoft is publishing almost anything that's new within that same approach with MIT license, so that argument is getting less and less important, right? Microsoft has so many efforts to pushing open source, It's simply because they want the people to use Azure services. That's it.

Open Source License Views

Jordan:
[39:25] You say that, but then that also gets into that sort of philosophical debate around open-source. What is open-source? Is it just the code is available for you to use? Is it what sort of the GPL-based licensing does where you have to ensure that you contribute back changes that you made?

Thorsten:
[39:43] What's your approach on that? You know, you raised the question.

Jordan:
[39:49] I think personally, I don't mind MIT-based licenses. Like a lot of the stuff that I write that's maybe not work-related is probably MIT. I write the code. I don't really care who uses it. I don't care if people don't contribute back to it. But I can see the benefits that GPL has brought to the world. So that's true. Without that, I don't think that it would be as open source or as prevalent as it would be because there wouldn't be the driver towards it. So I think some of the more stronger factions on that side of the debate, while I may not agree with them, I think they've done some decent work in producing the way that the world is and that sort of open source type of community that we have right now.

Thorsten:
[40:33] I don't know if that saying exists in English, but I'd like to translate something that we would say in Germany. The revolution eats up their children. And that means when I started in IT, I was also a young guy. I was a Windows user because that was the, or a DOS user even earlier. That was the only thing that was available to me. I couldn't afford to buy a Mac. I like Linux a little bit. We observed that. But back in those days,

Thorsten:
[41:06] when I started in IT, I worked with real Unix systems. And I always wondered why the FreeBSD, OpenBSD, NetBSD, why they struggled so much, why they lost the competition, to say so, against Linux. Because FreeBSD and all the flavors, they have lots of advantages over Linux. But the revolution eats up the children means We had that court trials, all the legal affairs, and they struggled upon all these things. And maybe Linux was the laughing winner out of that. Is that true from your point of view?

Linux Beats BSD?

Jordan:
[41:52] It's not something that I've really thought about too much. Like, why has Linux succeeded versus FreeBSD and some of the other Unix distros? If I was to guess, I would have thought maybe the fact that they have focused maybe a bit more on that desktop experience. So getting it out for individual developers. Like I remember buying computer magazines when I was younger, you'd get the disc on the front, which would have like a copy of Ubuntu or some other Linux distro. So making it easier for the developers, like the Microsoft mantra, developers, developers, developers has helped sell that. And then those developers bring that into their company. And maybe that's why Linux has been more successful. But I honestly don't know why BSD, the BSDs have struggled to gain traction. Obviously, they are popular. I know Netflix uses a BSD variant for their servers.

Thorsten:
[42:44] And some might say, no, they didn't struggle. Yeah, you're a macOS user. Yeah. Using BSD? Yeah. Darwin, Colin? Yeah, it's interesting. I would like to focus on yet another technology that I know that you are working in that area.

How Ansible Began

Thorsten:
[43:02] Let's talk a little bit about Ansible and if I have the chance to do some deep dive into it, I will gladly do this. Maybe we can start at the very beginning. Ansible is today, let's say, supported by Red Hat. Many people think it's a Red Hat product, but in the beginning it was just open source project. Can you tell a little bit about the beginnings or at least about your beginnings with Ansible and yeah maybe you can explain then after that what it does?

Jordan:
[43:39] Yeah so I wasn't part of Ansible from the beginning so what I know is essentially either what I've been told or learned about. So as you're saying Ansible started as an open source project so the author was michael dehan who's a us-based um developer um, i believe he worked at red hat and he was working on similar products i can't remember what they're called but like similar things to puppet and configuration management and he was looking for ways of creating a tool that was agentless that was easy to use and just sort of simple type of tasks that you actually define that a normal person can actually do and, some more he seemed to have struck gold where he created this product and it's sort of built up a bit of groundswell. We've always had a fairly strong open source community behind it. So when I started, like all the project was on GitHub. Our conversations with the community was an IRC, which is sort of an older school compared to Discord. And we've now moved to Matrix for those discussions and sort of other type of forums, which are more on that open source. But we still have a very strong community behind that and we get community contributions. But, as with all companies, Red Hat eventually thought, okay, we can find a use for this actual tool. So, they eventually bought Ansible, I believe, maybe 2014 or so. So, a few years in its development, and they sort of continued shepherding it on.

Jordan:
[45:06] It wouldn't really be a thing without red hat because they pay my salary to work on that actual product and that product while yes we have paid offerings um as well like support agreements that people can actually buy from red hat, they're free free they're free to use ansible as it is they can look at the code they can make changes to it they can pull it down without giving us a sense and like they can even ask questions on the community and there's there's a pretty vibrant bunch out there that would be able to help them.

Thorsten:
[45:32] So uh to keep that um still there is one ansible yes that's it yep you all use the same

Jordan:
[45:39] Yes so there's been no forks as of the many years that it's been out um there's been a few projects that sort of utilize ansible in maybe a different way um and, believe maybe they might still be going but i'm not too sure how they are but ansible itself is the one tool used both in the community side as well as internally as part of the products that Red Hat supports and offers.

Thorsten:
[46:03] You mentioned it's written in Python? Yes. So is it compiled for performance reasons, I assume?

Jordan:
[46:11] No. So it was written in Python. I believe that's just because what the original author liked and used. So Python, while they might have some just-in-time type of compilation. Maybe I'm saying it wrong and it actually isn't it. I don't know. Just Python. Yeah, just Python. So there might be some C extensions that dependencies use to speed things up, but Ansible itself is pure Python. There's no other language that is involved on the engine side.

Thorsten:
[46:42] I do raise that question because I think it's so interesting that if you have a look at Python, you find out, okay, it's not very fast because it's interpreted, But then you find out, okay, you can compile it, because it's more or less the underlying thing is C, C++, so it can be very fast, so it wouldn't make sense to think about it, but maybe it's simply not necessary. Ansible is a simple tool, right? And you can also run it on Windows, at least using WSL, see some colleagues do this. And can you explain what it actually does for someone who might have heard the

Thorsten:
[47:23] term, but actually does not know what it really does?

What Ansible Actually Does

Jordan:
[47:26] Yeah. So like the way that I would describe it is an orchestration tool. So essentially people would define steps that they want to do. So you could call it declarative languages where you say, I want this file to be present. I want to make sure this package is installed or uninstalled or other things like that. So we try and, provide a way that people can provide that configuration as part of a easier language for humans so in this case we use yaml um another markup language.

Thorsten:
[47:57] If that is easier to read

Jordan:
[48:00] There's very strong opinions on that one there very strong opinion yeah it it's a fun way, but essentially it's it's a way that humans can sort of easily see what's happening so like we provide a way of saying hey i give a description of this task and then like in in the case of windows we might have, um win service so that would configure a windows service with the name the stage executable and also it's.

Thorsten:
[48:25] A modular system

Jordan:
[48:27] Yeah so users can provide their own plugins if they want to but typically they just use what is available um but i think while yes the configuration is a pretty big selling point, um there those tools that do that are a dynadolson so there's dsc there's puppet um there's chef um salt sack there's a whole bunch of other ones out there. I think where Ansible's abilities shine is the fact that we can talk to so many different hosts out there. So traditionally it was through SSH, but we support other transports like WinRM or older Windows servers, and we don't need an agent on those actual hosts. So the server is ready as long as the firewall is allowing the information or the ports and connections to come through. We can set that up without any prerequisite software on there. It sort of is ready to go out of the box and dealing with things on Windows especially like rebooting when we've done steps. If you're trying to do that in PowerShell trying to get the script to start back up after the reboot is a real mess of pain so getting that working with Ansible makes it a whole lot easier and farming it out to many hosts if you do want to. There are some drawbacks if you go with that agentless approach, but I think by the most part, and I think also history has shown that people think that's a better option compared to some of the other ones out there. So yes, Puppet and Chef are still things, but I think Ansible seems to have captured the market share mostly for that configuration management type tool.

Thorsten:
[49:54] In a nutshell, from if you want to be as the neutral as possible? What's the major difference between the three of them?

Jordan:
[50:06] I think mostly that agentless approach.

Thorsten:
[50:08] So Pappen and Sheff have agents?

Jordan:
[50:12] I believe they did originally. I think maybe when Ansible started getting more popular, they started introducing an agentless approach. But I haven't used those tools too much. But I think, yes, the syntax might be different. There might be different plugins, but the same fundamental nature of what they're doing is fairly similar um, so i think the fact that was it was agentless and we did support a whole bunch of different transports is what allowed it to get more popular, theoretically maybe the fact that red hat also owned it and provided more enterprise support allowed it to have more adoption inside organizations but i know at least when i started using Ansel. So before I was made an employee of using it, it was the fact that it was free and we could use it already versus Chef and Puppet, where we would have to get approval. We'd have to spin up services and get approval to actually install that on service and file rules and all that type of stuff. So I think that was one of the key differentiators. It could be in a whole mixture of factors. I'm not sure.

Thorsten:
[51:17] Yeah, I will look it up, have a look at the show notes so that we don't get it wrong. But more or less, it's the same approach, and one more time, it's just the usual thing in Linux. You don't have one solution, you have multiple. And then you choose, which can I use? And sometimes the first step is the hardest one. Because if you would have worked 10 years with Ansible, 10 years with Puppet, 10 years with Chef, then it would be easy. That's not reality. The reality is you have not worked with one or the other. And the first step is to choose what do I focus on. It's amazing. If we compare that, if it can be compared, it's an orchestration tool, as you said. So somehow it reminds me a little bit of group pharmacy. Group policy is agent-based, but the good thing for the customer, the agent is part of the operating system, so it's just a service. And there is one more thing that you can compare.

Thorsten:
[52:19] Group policy consists of multiple, let's say, extensions, to describe it in a neutral way, that can focus on the one or the other technology. So, for example, to make a registry key change, you have the user ENV, DLL, and group policies. And that is more or less the same approach that also Ansible has. You can extend the system. You can easily write extensions. You even did a speech about that, a talk about that. I remember we have it on the PS Confidue channel. And so it can be easily extended. Why not use Ansible more in Windows environments because many people complain group policy is no longer developed, there's nothing going on, Microsoft forces us to use in-tune policies.

Thorsten:
[53:20] So how does Ansible fit in this as an alternate approach?

Ansible Versus Windows DSC

Jordan:
[53:25] So I think this is probably one of the things that Ansible probably isn't as strong in. So the fact that it is agentless and you're pushing configuration

from a remote source is great from some aspects, but in this case here where you want to avoid things like configuration drift, having that sort of pull enforcement-based one may have the edge over Ansible. So I think that's potentially why some people may not have done it.

Jordan:
[53:50] I also think that on the Windows side, well yes, we have people using it, and it is a very important and sort of big facet of Ansible. I find that a lot of people that use Windows are in a Microsoft shop. So the question is more is, why should we not use the tool that Microsoft offers over another tool and have to pay them money as well as what we're already paying Microsoft type of thing? So it's trying to convert that type of argument into saying, yes, Ansible is good and can work, but sort of trying to convince them. So similar to like, why should we use Slack when we can get Teams as part of our 365 license? Type of thing. It's a hard argument to sell inside companies. And I think we do have a very good selling point in the fact that we work with Linux. So we work, if you have a Linux fleet or a bunch of Linux servers in your company, which most people do these days, it's easy to say, yes, well, yeah, you might be paying Microsoft their license and support. We now have one tool that works across all of them so that's been a fairly good selling point uh yeah.

Thorsten:
[54:57] Yeah the uh which you just mentioned is that this agentless approach makes it very useful obviously by nature in on-prem environments because you can figure out here are my resources but that contradicts a little bit with the approach of having a cloud environment where it's not so easy to push software to a random clients somewhere on the planet without an agent.

Thorsten:
[55:20] That's definitely true. But I want to focus on that a little bit more because there are some things that are actually quite half-bacon, I would say, in the Windows environment. One of our favorite topics is design stage configuration. I know so many people who invest in this and love it. It's a huge topic in my environment and I'm always making fun of it because Microsoft really does not love it and does not really support it. And then you have people that say we need that config management and they invest a lot of time so that they spend a lot of money to have consultants or employees work on that desired state configuration. And now if we compare that with Ansible you find out you have these playbooks You said it's a YAML-based config. And you check out this extension thing and you easily find out, okay, that is more or less the same idea. It's config management. And then you find out there is also a DSC extension. And now it gets weird. From my perspective, a config manager tool calls another config manager engine.

Thorsten:
[56:38] It doesn't make sense at first sight. Can you talk a little bit about that weird relationship between the DSC folks, the DSC extension, and to be honest, the Ansible world is stepping away from DSC, is that correct? So you have a lot of time to explain this in all the detail. Yeah. You would like to.

Jordan:
[56:57] That's definitely a fun topic, and I think this is where Ansible has shined over DSC. So the fact that Ansible can actually call DSC sort of makes Ansible more of a superset of what it's offering. Of course, that's not 100% true. There are things that DSC might do that Ansible can't. But essentially, one of the key benefits that I find that Ansible has over DSC is that it takes a step away from that Windows source where, instead of defining, or at least with DSC,

you have to configure a completely different tool to then push config or set up an agent with MOF on the old DSC version 1. I'm not sure what DSC 3 has, if it even has a configuration element like that that's equivalent. But essentially, Answell can now talk to as many servers that you want. So you have the resources on Windows. We have a whole bunch of them. DSC has a whole bunch of them as well. But if we're missing something, we can just use whatever DSC offers. So that's great because we don't have the full coverage of what people are wanting to use, whereas DSC might have the equivalent where there's already existing resources to call from it. So it's great to be able to piggyback between them. I've used DSC a few times. I've never been a fan of the MOF-based approach. So I find that syntax, yes, you're sort of giving me a smile about YAML. I think MOF is a whole thing entirely.

Thorsten:
[58:20] Yeah, the difference is to be honest here. You don't write the MOF. You let it create, but you write the YAML. Yes.

Jordan:
[58:28] So I find, yes, so the DSC one, the tooling around that really was non-existent. So yes you could write it in PowerShell and invoke it that way but like the configuration around it, it just wasn't there it seemed like it was a an MVP product from Microsoft and they never really took it forward type of thing and that sort of was the case because it's now no longer really a thing aside from a legacy type product, um DSC 3 I'll ignore 2 because that's gone gone now um DSC 3 seems to be another attempt from Microsoft and they seem to have made some good strides around the metadata and creating a good, tooling base around it. But I fear it's falling into the same problem with DSC where, yes, we've got that MVP offering, but it hasn't had that step further to make it useful.

Thorsten:
[59:22] I will talk about that in just a second, but keep on explaining how that works. You have extensions and just WinDSC is yet another extension, so you can directly call the DSC resources.

Jordan:
[59:36] Yes.

Thorsten:
[59:37] Yeah, that's like this domino thing, where one brick lets the next brick fall, right? And if you miss one thing, then the complete chain is broken. You know what I mean? That doesn't seem to be, for me, that's not stable enough. And the code is, from my perspective, pretty hard. Once again, we could say now in the age of AI, I don't need to write it myself. I have a proper description and let the AI create it. That might be an interesting approach. But what we will find out, yeah, we need something on the operating system that does the job. So we don't have an agent. And the problem is we had an agent in DSC, the LCM. But Microsoft removes that from the, more or less, from the operating. It doesn't really remove it. It's deprecated. So that is Windows PowerShell. Yeah, it's paused forever. it works somehow, but it will not mature. I never liked this approach, right? So anyhow, I don't see that this solves any problem. And do you see, from my point of view, despite the fact that I know and like the author, Steve Lee, more or less,

Thorsten:
[1:00:51] DC3 is a copycat. If you want to learn Ansible, just check out DC3. And now my question to you. What about someone at Microsoft thinking about it like this? We don't have a proper orchestration tool.

Thorsten:
[1:01:08] Why don't we invest in Ansible directly? There is no one that disallows the creation of an agent. So on the long run if they want to connect people within tune and they see a necessity it is absolutely possible that you connect the dots so my approach would be Instead of creating yet another version of DSC that nobody's interested in, why don't invest in Ansible? Would you like some Microsoft employees, devs that work on that and strengthen the Windows side of the tool?

Jordan:
[1:01:45] It would be great to see. I would love that. But I think maybe the fundamental reason why that hasn't happened is that Ansible doesn't run natively on Windows. So you were saying before where you've seen colleagues run...

Thorsten:
[1:01:56] WSL, yeah.

Jordan:
[1:01:57] Through fwcell you can't mandate that on, customers endpoints so you can't say that for our configuration management tool you need wcell to run this tool on top so i think that's probably one of the main reasons why, well i don't know what they've talked about internally but i think that probably is a key thing that would take it out of consideration for that type of um tool.

Thorsten:
[1:02:19] So if i imagine i would have a linux without python then endsible wouldn't

Jordan:
[1:02:23] Work yes it.

Thorsten:
[1:02:24] Needs it on the

Jordan:
[1:02:25] Yes, so it would need Linux on there, but Python is pretty much a given on most Linux distros, and even if it isn't, it's one package away to try and install type of thing.

Thorsten:
[1:02:37] If I don't want Python on my box, then I can't use Ansible, right? That's the reason why I ask. I do ask because the main difference between the operating systems and the philosophy behind that, we all know Windows is API-driven. So back in those days when they invented something like the registry, which seems a little bit awkward from today's perspective, where anything is yet again a text file, as in the 1970s. The main idea was that you simply have an API that you can ring the bell and tell the operating system, do that change. So that, again, is completely agentless because you simply have an API. And that is the origin why Microsoft invented the registry over the earlier approach of having any files. Well, it's exactly what Linux does until today, just have some random text files. But now we're moving away from that because if you watch all the new products, we have more and more that text-based approach. It mustn't be YAML. Very often it's JSON or something else. So again, we combine that. And the improvement over what we had before is that JSON is

Thorsten:
[1:03:55] Is actually nothing else than a representation, a serialization of an object.

Thorsten:
[1:04:00] So I think these things are more comparable than split apart, right? So again, what would you wish for the future of Ansible and the Windows world? Did you have a vision? What would be the perfect thing for your customers, for your work? What would you like?

Future of Ansible

Jordan:
[1:04:20] For me, I would just love it to... so one of the key things that we're hoping to improve on is around performance so it is a bit slow um compared to like just running a script natively so i want to get ansible to a point where one customers can, on especially on the windows side they don't have to deal with the cli they can create their playbooks they can more easily interact with it through like lsp so um visual studio code and other things like that and make it easier to author this type of stuff.

Jordan:
[1:04:52] In terms of overall vision, it would be great to see more people use it and be able to sort of contribute more things back to it, which then is sort of a good cycle. So a virtuous cycle where the more people use it, the more people's added to it, then the more people we get in and so on. But from a fundamental vision, I think... We just need to improve the product more, add more features that are on there, and then start looking in some of those edge cases where around, okay, how can we enable more, agent-like approaches where it's sort of more pulling in that configuration and reach those devices that it probably isn't great right now. So those client devices that may not be connected to the internet, or at least to a secure network type of thing. So that's one of the areas that we really need to have a better think about and see what would be possible and how we can sort of get into that type of side. Because yes, Ansible can do it, but should it do it right now is another question.

Thorsten:
[1:05:50] Is that your primary topic that you focus on in Red Hat, or do you have any

Thorsten:
[1:05:54] other projects that are also very important?

Developer Experience Matters

Jordan:
[1:05:57] There's a few other projects. So one of the key things that I've been hoping to do or that I've been working on in the past few years is around just the developer experience. So I've created a project that runs on the debug adapter protocol. So that's something Microsoft has introduced to make debugging implementation sort of generic so it works across one editor to another. So a debugging protocol for Ansible itself, so you can put breakpoints on the YAML tasks, step through it and see the results in a more programmatic fashion.

Jordan:
[1:06:30] Um also things around debugging powershell code that's actually running in ansible itself so trying to improve that story and making the developer experience easier because essentially they're the ones that will be using the tool and if they start using it yes there might be some pressure from above to continue, to continue doing it, if they keep on going back saying no that's not good or they talk bad about it to their colleagues or other people that they know, it gets into that mindset where ansible isn't a great tool to use so that goes back to what i think microsoft didn't very well in the 2000s and maybe even 2010s is they really focus on that developer mindset so make the tools easy to use and that that's one of the aspects that i'm working on, the other side is yes let's just start adding more features as we go on to the windows thing so enable things like dsc3 so, you can while yes it's not, perfect it's it opens up the field to what people can use and WinGet integrations and other things like that.

Thorsten:
[1:07:31] Hearing that, I would like to focus on two final aspects to wrap that all up. We can't finish that chat recording, that talk, without...

Idempotence in Practice

Thorsten:
[1:07:45] Speaking for a second about that, idempotency in DSC, that's always an argument, that get set test thingy, that you can run DSC code forever, and we simply check, is that set? Yes or no? So that is what we describe in the PowerShell world, as idempotency, you can run it, it doesn't really matter in what state the machine is. It will do the same job and guarantee if everything is fine that the outcome is just what you desire that's the reason why they call it desired state configuration yeah how good is that in ansible if that is my argument this is way better because of get set test what could you answer

Jordan:
[1:08:27] Essentially ansible does the same thing so we we say that we are idempotent and yes we sort of are but it all fundamentally comes down to the code or the what you would call extensions we call modules that you call, essentially the modules are the ones that implement that logic and i'd say the far majority especially the ones that we ship are item potent to a certain aspect so similar to dsc you can create a dsc resource that isn't item potent, um so it's really, up to the resources that you use and yes our resources are pretty much item potent or at least that's the ideal uh, result that we're trying to aim for when we're creating these modules.

Thorsten:
[1:09:07] If i to really nail it it would say okay i i need to and i'm a team lead i need to invest in either dsc or ansible what would you recommend what do i lose if i lose track of dsc and focus entirely on on ansible because there might be some niche aspects that the one is better than the

Jordan:
[1:09:27] Other yeah.

Thorsten:
[1:09:27] But i can only lived my life once, so I have constrained resources. Is there one thing that you would say, okay, be careful using Ansible, that is where DSC shines? Yeah.

Jordan:
[1:09:41] I'd say probably the only thing that DSC may do better than Ansible, and while I've used it, Roughly, I haven't really used it too much, so I can't say for sure, but potentially the performance aspect. So because this configuration is already on the machine itself, and you're saying, hey, apply this, it doesn't have to deal with starting up the connections, sending the data, waiting for the results to come back, and so on. So it definitely has that edge in there. But Ansible can also do just that. So with that DSC module, you can still use Ansible on top to take advantage of the work that's already done in DSC, saying hey here's my dsc configuration let's push it so, i think that's sort of complementary um in terms of like ansible can use dsc dc can use ansible as a way of pushing that information out there but also on the same side ansible can then also not use dsc.

Jordan:
[1:10:35] And to do a similar type of aspect whereas dsc needs a completely different set of tools so whether that's something from microsoft or something else something homegrown to be able to deal with hey how can i push this configuration how can i apply it and invoke that type of side so, i think we all i know i'm biased but i personally think that we have a good balance between the two where you can, pick and choose the components that you want to do and it's the same thing where people saying oh why should i use ansible when i can use terraform there's things that terraform that does well and then there's things that Ansible does well it's it's don't try and fit a square peg in a round hole choose the thing that works for your organization and I just think that we work in more situations than DSC doesn't.

Thorsten:
[1:11:20] Very interesting. There is a chicken-egg problem, to be honest, when I dove into the Microsoft world. It's simply because of just the customers. And I always wanted to focus a little bit more on stuff like that, stuff like Red Hat. I worked for a company back in those days, there were Red Hat partners, and a couple of colleagues who were engaged in that area. And I couldn't make it to that team because they didn't need anyone else. If there would have been a lack of resources, maybe I would have started in the Ratchet environment and doing this ever since. But now, back then, this is 25 years ago, I started more taking care of Microsoft technology because simply the customers use it. And that leads to the chicken-egg problem. If I would love to do more with Ansible, but I don't have a single customer because I have that much soft label and you make a living, right? You have to make a living for that. But I think it's very, very interesting because that simply seems to solve problems that are otherwise solved in a more complicated way. And I like easy solutions. And the final aspect, we touched that a little bit.

Why .NET Matters

Thorsten:
[1:12:42] I would like to talk for a moment about .NET. The underlying framework of Bauschelle is .NET, and we just mentioned Miguel de Icaza, and the reason why he was, is or was, I think, was a Microsoft for a lot of years, is that he created that .NET runtime for Linux, and so they bought in, said, now you're a Microsoft employee, do what you did, and they bought the product.

Thorsten:
[1:13:11] I do mention this because .NET from the very beginning was meant to be platform agnostic. So in the very beginning they couldn't make it, so it was towards Windows. And then they shifted a little bit, just as PowerShell shifted to PowerShell 7. Back in those days it was PowerShell 6. And today we have .NET, .NET Core, how it was called in the early years, that is available for Linux, for macOS, for Windows. How important is .NET in your life? And is that a technology that customers can use as technology to connect dots? So we said Python is so influential, but why not redevelop certain solutions based on .NET to make them even better? What's your take on .NET? Is there a connection between all these topics?

Jordan:
[1:14:08] Yeah. So with .NET and my own personal usage, I love C Sharp. I think it's a fantastic language. And the runtime that Microsoft has built there around just performance and the sheer depth that you can go. So you can use C Sharp or even F Sharp for very high-level stuff, but it enables you to go even deeper if you want to, if you want to get at that performance in a more compiled language. And I know I was talking about before about the developer experience. I think debugging and even sort of just analyzing managed C-sharp code is so much easier than unmanaged C-code and all the various C-debuggers that are out there.

Thorsten:
[1:14:46] So you finally found a language that you like. Yes. You remember the start of the conversation.

Jordan:
[1:14:53] So yeah, I really love C-sharp. I wish I could use it more in my job, but it's more of a hobby language right now for myself. So it's sort of just what I do on the side when I have some free time versus what I use as part of Ansible.

Jordan:
[1:15:08] In terms of where it sort of fits, I think it has some really good opportunities, especially in the AI side, which can try and smooth out some of those initial rough edges. So learning out around the tooling, how do I build this code? How do I compile it? Which has always been sort of a barrier for entry for people, especially when Python is you just need the executable and run this. You don't need to compile the code. The issues that i probably see that it has is dealing with languages like rust and go and other languages that sort of seem to have that existing mindset especially in the linux side um they seem to, have an easier time especially because it's not a microsoft product to sort of being sold and use in this ai space i've spoken to mark andre from devolutions and he loves rust um he uses it for a whole bunch of things and seems to be one of the main products that he's using today. Looking online, you see so many different Rust rewrites from different tools. I think that's probably one of the things that it may need to watch out for. Otherwise, it's going to be relegated to what .NET Framework is, which is purely a Microsoft world, which would be a shame because it's a wonderful product. The things that they've done, especially just making it open source and compatible on there and performance is crazy. It's a fantastic tool.

Jordan:
[1:16:31] So, like, I wish it would be better, but I think it does have a pretty hard fight right now to get the mindset and continue forward and maybe AI will blow over, maybe it will become more dominant. I don't know what the future is, but it's definitely here right now and, like, it's going to be interesting to see what it turns out and what it's going to be like.

Curiosity and Learning

Thorsten:
[1:16:53] Did we miss anything that you want to talk about or maybe give a tip something that people that listen to this we should have a look at yeah

Jordan:
[1:17:03] Um i don't think we missed anything i can't think of anything but in terms of tips i think one of the key things especially in my career getting to where i am is, just be curious and willing to try something so, don't just blindly follow guides online um if you see something and sort of you're thinking why drill into that so go into the the finer details and yes maybe it will be a useless exercise and you've wasted a day, the next time that you come across a problem that it could be really influential and sort of important for your knowledge and, digging into that has allowed me to work on so many different things and try different languages so like i started with java in my previous job i don't do that today but if i have I'd love to try it again to see what it's like. So be open-minded, be willing to try different things and also be willing to dig into something and see what's happening. Don't just blindly read something. It is as it is because you learn things that way and you learn different aspects and different things that people are using.

Thorsten:
[1:18:07] Yeah, that's almost a perfect ending. I would love to just mention something that you told me before. We talked about AI, and what a useful tool it is, and just because it's such a useful tool, think about translation of human languages. You told me that you always wanted to learn Italian a little bit better than you already know. And I always tell people, just because of the translators that are so great today, that is just another reason to learn it on your own. Right? Because that human spirit that we have, we want to be unemployed because some robots do all the things. There will always be a deeper dive into culture, in understanding people that will never be provided by an automatic translator. That is true for music, that is true for human language, and this is just a great ending if we think about your experience with computer languages. You will understand, as a real programmer that learned, let's say, C-sharp, you will understand Rust or Go, and you will even better understand and handle what the AI helps to create.

Jordan:
[1:19:23] Yes.

Thorsten:
[1:19:24] So, Jordan, it was fantastic to have you here. Thanks for your time.

Jordan:
[1:19:29] Thank you very much for coming out. It was fantastic talking about this. I really enjoy it. It's fantastic.