EPISODE 1399

[INTRODUCTION]

[00:00:00] KP: As the Internet has grown, increasingly, we are consumers of services provided by corporations, rather than owners and operators of our own systems. To many, this trend towards centralization is antithetical to the spirit of a free and open Internet.

Urbit is a new operating system and peer-to-peer network. There are several layers of novel ideas in this ambitious project. And in this episode, I interview Galen Wolfe-Pauly about Urbit, how he's using it, and how others are as well.

[INTERVIEW]

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

[00:00:45] GWP: Thanks so much for having me.

[00:00:47] KP: So I know you've been on before. But for listeners who aren't necessarily aware of Urbit, I was wondering if you could give us just the high-level. What is it?

[00:00:55] GWP: Sure. So since this is a somewhat technical audience, I'll give you a frame in technical terms and then we can talk about it in different terms if you want to. But the idea is basically people should have a personal server of some kind. Most or many listeners would probably set up their own Unix servers. It's a nightmare. It's super complicated. Unix is an ancient operating system. We can do amazing things with it in the realm of building web services. But when it comes to like doing personal computing, it's really, really hard to basically have a personal computer in the cloud of any kind.

And so Urbit is a new software package that runs on top of any Unix machine with an Internet connection that's designed to be a personal server. So something that an ordinary person can own, and control, and use day-to-day, but also a totally new software stack for people to build on top of where the idea being that you ship software to an individual, they run it themselves,

everybody runs nodes on this totally decentralized network. It's a totally different model of how we might compute in the cloud than using cloud services.

[00:02:03] KP: So when you say personal server, what are some of the services that that encompasses?

[00:02:08] GWP: Yeah, it's a good question. I mean, today we use Urbit basically as a communication tool, so to chat, and share notes, and links, and stuff like that. T think that we have a lot of interest from people in the crypto space broadly because it sort of overlaps with that world. So you could see this being useful for people who are trading, working as syndicates, engaged in sort of personal commerce. So whether that's like as a creator selling things or like a host of a podcast or something. So replacing things like Patreon in that case or tools like most syndicates have some weird mashup of signal telegram and they're using exchanges and maybe they're using dexes. So, basically, social software, right? We have social networks, but we don't really have social software. We have to kind of cobble together SaaS to do that for us, if that makes sense.

[00:03:03] KP: And you also described it as peer-to-peer. Can you expand on that?

[00:03:08] GWP: Sure. Yeah. So Urbit is really two systems technically. It's Urbit ID, which is basically like a PKI that's deployed on Ethereum. So you own a name that's in this name registry on Ethereum with a private key. Those names are these sort of short synthetic names. They're totally pseudonymous. You use that name by registering a key with it to boot a node, an Urbit OS node. Urbit is this totally virtualized piece of software that runs on top of any Unix machine. And when you boot it, you sort of announce yourself to other peer nodes and say, "Hey, like I'm online as this name, and you can contact me at this IP address." So then when you want to communicate with other nodes, let's say you're starting a group to do something, or you're downloading software or whatever, you're always doing that point-to-point or person-to-person, node-to-node.

So while there are – If I host a group and 100 people join my group, I am basically the server for that group. But in general, it's peer-to-peer, meaning people connect directly. It's not like

everyone goes to facebook.com and Facebook has the group basically as an entry in their database, and we all just connect a Facebook server in order to connect with one another.

[00:04:29] KP: And what are some of the advantages of the peer-to-peer model for your use case?

[00:04:34] GWP: I'd say that Urbit broadly is concerned with control. That's sort of the main objective, right? Like when people are in charge of their computers, they can invent new things almost like casually. I feel like this was my experience growing up with PCs in the 90s. Like when your computing is local to you and you have all these different applications that you use on your own, you can sort of invent new ways to do things and to solve problems. And so if you want to do that in the cloud, peer-to-peer networking is almost just a consequence of, "Well, we want to let people, when they share data with each other, do it however they want to." You don't want there to be some intermediary deciding who gets to send what back and forth or how. I mean, I should give that with some caveat, is that all networks do need some form of governance if you look at totally anonymous networks where addresses are free. They're often full of spam and abuse.

And so Urbit IDs are finite. There are only so many of them. And you have to get them basically from someone who already has one. So there are basically 8-bit addresses, that issue 16-bit addresses, that issue 32-bit addresses. And that issuance hierarchy means that while the network is decentralized in terms of when we talk to each other, it's sort of para-decentralized in terms of how it's issued, which keeps everybody accountable. Meaning it costs me something to get an address so I don't want to spam the network and then have people blacklist that address. And if I own a big block of addresses, I'm even more accountable, right? I don't want to be known as someone who's selling to sort of like bots and spammers.

But anyway, sort of popping the stack, the idea is that if you want to give people a tool that they can do whatever they want with and invent new ways to connect, communicate, and so on, peer-to-peer networking, it's just completely obvious. It's the same reason that that network is encrypted. The same reason that you own the thing with a key. There are just sort of natural consequences of trying to make something that you control completely.

[00:06:44] KP: Well, earliest computing days and when modems were just getting people online, we started out in a very decentralized way. There were systems like FidoNet and BBS's that people managed. And it seems that generally we've gotten progressively more and more centralized. A very few people run their own POP3 mail servers anymore. You go to a SaaS solution for your email. Is Urbit like a step in the 180? Is it a step to the left? How do you view it from this trend of centralization?

[00:07:14] GWP: Yeah, it's definitely – It's a 180 many times over. So the way I look at it, like centralization is the easiest way to scale like a young internet company. So most of the companies that we look at is being super centralized and sort of dominating the Internet that we live on today, this kind of Internet of apps and services.

If you just look at the history of what happened there, they were running servers, because it was the only way that they could build something and show it off to people. And people got so excited about the things that they were providing that they had to figure out some way to even more quickly ship updates, be even more reliable, so on and so forth. But the consequence of that process running over now 20, 25 years, is that you just have this insanely industrialized software stack, right? So if I want to start working on a new project, I'm sort of NPM installing my way into something that is super complex, and I can just never understand as an individual engineer. And so I think that that kind of trend of industrial software is what has gotten us away from even the ability to compute directly with one another and like as it was in the area that you described, right? The early days the Internet, like you have these pretty compact and simple systems sitting on people's desks that were cabled together. And it's a much simpler software stack. Now we have this hugely, like this huge industrial scale software stack that provides us amazing quality of service, but it's really not that different than in the 70s, the computer center had a mainframe in it, gigantic industrial piece of equipment, provided amazing compute power, and that early PCs couldn't really compete with that. But the fact that they were on your desk is what made them really exciting.

So I think that there's always going to be something just like distinctly different and unique, and I don't know, like just special about something that's yours and like that you can do whatever you want with. And so those – Yeah, the days of BBS's, have this very unique spirit to them, I think, because people were in charge. The people running the thing, actually, they also owned it. It

belonged to them. And that's sort of where the magic of computing lies, at least to me. And so, yeah, Urbit is definitely – If we live in a very different world than that today and Urbit is trying to solve that problem technically, with a thinking being if you can solve that problem technically, then you can actually deliver a different quality of user experience.

[00:09:54] KP: Well, I don't know if there's any such thing as an Urbit census or whatnot where you could get a measurement of the usership. But I'm curious if you see any patterns or personas emerging. Why are people adopting Urbit?

[00:10:07] GWP: So, yeah, a lot of them – That's a good question. So one thing that's worth quickly clarifying probably is that I think of Urbit is – Think of Urbit like Unix and then we've built this thing we call Landscape. You can think of kind of like a Window manager. Or maybe it's like Urbit is like the Internet and Landscape is like Netflix, or Netscape. Landscape is nothing like Netflix. Meaning, Urbit is a piece of software, is a piece of infrastructure, and then we build an interface for it. Landscape is primarily about, yeah, like I was saying, sort of it's a simple communication tool. So a lot of the patterns I'm going to describe are kind of in that. They live in that world.

Well, about a month ago we shipped a major update to landscape where you can actually distribute software over Urbit that runs inside of landscape. So we haven't yet seen emerge what I think we're most excited about, which is like when people and communities start building their own individual Urbit applications, and tools, and experiments. And I think that will move this in a different direction.

But anyway, for the time being, I mean, Landscape being this kind of – Think of it like a Discord alternative, or like an alternative to sort of like – We built it basically like being a small community in a company that used Google Docs, Arena, or Pinterest, or whatever, and like message boards, as well as a chat app, and trying to combine all those things. So that's kind of what Landscape does.

What you see on there is communities kind of similar to, yeah, what I remember from the earliest days of the Internet, these sort of like either communities that have a shared interest. So there's a bunch of music groups that are really fun and just weird and interesting. There are

people say like really enthusiastic about small computing, like building Raspberry Pi-based home Urbit rigs as an example. And then you have these communities that are more kind of creator-focused. So it's like where a creator has maybe a public Patreon or something, and now you can pay them somehow to get access to a group that they run on Urbit, where everyone you know gets access pseudonymously to kind of connect with them more directly.

As you kind of hinted out, we can't. Since the network is decentralized, we see the people who pass through the public channels that we run, but there's no real telemetry. We can't track who's on the network. So this is sort of anecdotally what I see perusing the network myself. But actually most of what I use it for is to connect with the company and with the community more broadly. And for that purpose, it feels very cozy. It's really nice. It's nice to use something that's not constantly trying to get – There's no like a landscape nitro. There's no advertising. There's no upgrade feels just like a solid sort of – Someone once described Urbit to me as like when someone has like a Tchotchke on their shelf and you pick it up and it's like a lot heavier than you expected it to be. I feel like Urbit sort of has that quality. It's like it just works. It's efficient. It's like sort of stable.

[00:13:06] KP: I think a lot of people are just by popularity or at least they'll be familiar with tools like Slack, and of course email, and Microsoft Teams, and maybe Jira. It's an endless list of things people are using to work together in collaborative fashions. Can you compare and contrast some of those common solutions with what collaboration looks like using Urbit?

[00:13:26] GWP: Sure, yeah. So the most important thing to know is like Urbit – I last talked to you guys in 2015, I think, something like 2016? So for a long time, we were this tiny team. We're about five people who were kind of crazy enough to think, "Okay, maybe we can basically develop a totally different, like almost like lineage of computing. And building a new platform is just like a completely insane thing to do.

And the first task was like, "Okay, can you make this platform even work well enough to build some very, very simple prototype applications?" And I'd say by 2017, the prototypes were pretty nice, and we were using Urbit a good amount ourselves. And so then over the course of 2018 and into 2019, we started to hone in on, "Okay, what did we really want to do with this thing?" And then I think at the end of 2019 we started building what is now Landscape.

So Landscape itself is definitely super immature as a product by comparison to any, yeah, like Microsoft Teams, or Slack, or whatever. We're bigger now. We're about 25 people or so. And that happened in in 2018. So we're like able to build somewhat real software. But of course we also build the entire platform that the thing runs on top of. That's not entirely true. Urbit is open source and there's a community contributing. But just so you have a good baseline of what we're not Microsoft, we're not Slack.

But, yeah, we did build it just to cover our own use case of like how do we keep the team connected. And we're people who really like calm, straightforward, very purpose-built software. So in Landscape, you can create a group, and that group shares channels, and the channel can be either a chat, a notebook, which is just a collection of like basically blog posts with comments. Or it can be a collection of links in the spirit of, say, Arena or Pinterest.

So feature-wise, we're like a little bit different than any other sort of chat-focused communication platforms. And I think that's our kind of main like differentiator that's probably like an advantage. Like most of these things are totally chat-focused. But in terms of just overall maturity, we're still mobile web. There's no mobile app. The onboarding is definitely like pretty tough. It's still rough. That's the area that I think we can improve on a lot, and I think we will. This is not a super mature product that's ready for your giant company. But it is, however, I think pretty good for like your little group of friends that were otherwise spread between whatever, Discord, and Slack, and Notion, and whatever else. So that's more the use case that we see popping up.

This is also somewhat true of – And there are a few kind of crypto focus companies using Urbit. I think because it appeals just in terms of like the level of ownership and control that people have, a lot of these companies are also experimenting with like using Urbit as infrastructure. So building software for Urbit or experimenting with using Urbit as kind of like a backend for moving transactions around and stuff like that.

[00:16:34] KP: So Urbit is a platform. Can you describe some of the opportunities for a developer to maybe start a business or at least some sort of popular project on top of it?

[00:16:44] GWP: Yeah, sure. Yeah. I can talk about actually a few things that people are working on presently that may be a sort of jumping off point. And the other thing that's interesting is that it's a platform where the design of this address space that is finite, the idea there is that, in contrast to the Internet, where an IP address is not a piece of personal property. In this case, address space is property, and the idea is that the value of that property should be used to fund development of the platform. So we also pour a lot of address space as a resource into funding just general platform development. Meaning, we basically pay core developers to help us mature the run time, contribute to the kernel, and so on.

So we'll start with a former case, and I could talk a little bit about kind of like the grants program. One of the things that's been interesting to watch over the last nine months or so is we had a group of employees. So I'm in charge of a company called Tlon. We're sort of like the core developer. But Urbit, of course, yeah, is open source, and it's run by a community. There's even a foundation that supports community development as of about a month ago.

Also, about nine months or almost a year ago, a group of Tlon employees were really, ally excited about payments, d payments over Urbit. And I do think there's some interesting ways in which people can do business over Urbit. Left to start a company that is now working specifically on building payment rails for Urbit. So both building interfaces for people to deliver content to their sort of listeners, or viewers, or readers, and then ways for that audience to pay the creator. Basically to sort of, yeah, their Patreon replacement.

And so in their case, they're, I think, ultimately, going to try and sell sort of subscription-based software. So they ship an app, where you can go and download it from them over Urbit itself and then you pay some feedback to them to actually use their service and to use their payment rails. This is definitely just emerging. But I'm so happy to see it happening, because I've always felt like the interesting thing about Urbit as a platformship software too, in contrast to the Internet, is if I'm a developer of an application on Urbit, I basically need to ship you. It's more similar maybe to like a iOS model or something where I need to have something online where you can get the app itself. And once you've – That's sort of like a single transfer, right? I send you the payload of that application and then you run it yourself on your Urbit node.

But the way that people should be able to make money there is by actually selling that as either a one-off, like you would buy an app, or through some form of subscription. At this point, most people are doing it for fun. But I'd imagine that in the next year or two years, like those models will start to get pioneered. And a lot of people excited about figuring out how that should work are super active like within the Urbit community. And those who are not, I think, specifically interested in building, yeah, at the application layer, yeah, the system itself, being this kind of alternate reality world of computing, is a ton of fun to contribute to if you're like a serious systems person. And we commit quite a lot of address space to funding those people.

So some of that may happen within this sort of little operating environment, which is like a file system, a networking protocol, a build system, an application sandbox, and so on, all of which runs in its own programming language that compiles to its own machine language or its own sort of virtual machine language. So there's a lot of work to do across that stack. And there's also a lot of work to do within the interpreter. So this thing is run by – Has its own runtime and its own little virtual machine that's written in C. And we have apprenticeships, grants, and often accept proposals where we commit address space to people who want to help mature the system in different ways, and even potentially like do research and ways to make it faster or more efficient that are open-ended. We're exploring new things.

So yeah, there's lots of different ways to get involved. And one of the fun things about Landscape itself, and which I always find kind of validating, is that a lot of the discussion around how to do all those things, who to talk to, or what to work on, basically all goes on within Urbit itself these days. So even if you're just kind of curious, you can join the network and check it out.

[00:20:53] KP: You'd describe that if I were an application developer, I could distribute via Urbit and you'd be running my software on your node if we came up with a deal where I was giving you the software. What's the model for distributing updates to software if I've given you the source code and you're running it?

[00:21:11] GWP: Yeah, good question. So we update the entire system including the programming language over the network. And the reason that we can do that is because the virtual machine spec, which we call nock, it basically never changes. It's like 13 opcodes. It's

super concise. It's basically just this little Turing complete definition of computing. So your Urbit at any given point is just a like – It's sort of a pure function of its event log, right? So your Urbit receives events, keyboard messages, network messages, whatever it might be. And then on each event, it computes some new output, some new state. And so some update could be, "Hey, here's an update to the entire operating system. Run that against your existing state." And of course, your existing state contains the current version of the OS, and then output a new state, which is your updated Urbit.

So one of the affordances of the file system is to be able to – It's kind of like reactive Git, right? So instead of saying, "Hey, clone this repository and send me all the diffs. Or, yeah, pull down the difference." You can just say, "Hey, listen to this endpoint that could be somewhere out on the network and send me all the updates as they come in." So if you're a developer, you basically just have a directory where your application is that is the sort of prod version of your application. And assuming you have some that you develop on, you develop updates. When you're ready to do release, just move the code over into that production version. And anybody who subscribes to that, what we call a desk, which is basically a branch, should just automatically get an update over the network.

We regularly do this to update, yeah, like the whole system, as I was saying, which could be all the way down to the language. Updating a language can sometimes be tricky and has to be done carefully. So kernel updates, you have to be a little bit more careful about. Although in general, Urbit is pretty robust. Like the whole thing is basically a database and also acts like a single level store. So we don't really have a distinction between writing to disk or running in-memory. The whole thing runs in-memory.

So yeah, it's a kind of a different model.You're not like – When someone – I guess like the closest thing really is like phone, these things on your phone, right? Like I download an app and I can just say, "Okay, automatically update it." It's probably a lot more like that. We don't have the equivalent of that when it comes to websites, I mean, even with a site. If you're a developer and you've shipped an update, you need to force everybody to refresh if it's like a single-page app. So it's a little bit different than that.

[00:23:37] KP: You'd mention giving out address space as sort of a reward or an incentive in certain cases. Could you elaborate on why people want that? Can they sell it? To whom do they sell it? And how do people leverage the address space?

[00:23:52] GWP: So let's maybe start at the top. There are 256 or like two to the eighth galaxies, each of which issue 256 stars, making for a total of about 65,000, or 2 to the 16th. And then each star issues 65,000 planets, making for a total of about 4 billion. So you can think of the galaxies as basically like governance nodes. So a majority of the galaxies can vote to upgrade those contracts that govern the whole address space. A star is sort of like your local ISP. That's where if you're a planet, you sort of by default get software updates, or like kernel updates from. And you imagine stars could also sell services. So now there are people operating Bitcoin nodes as bolt-on affordance to Urbit itself. And that usually happens at the star level.

And then planets are kind of like individual nodes for personal use. They're like somewhere between – I mean, it's for individuals, right? A planet name is like a domain name and a handle rolled into one, but it's also a network address I can send packets to. Planets also issue moons. You imagine that moons are for devices. So planets and stars can change parents. They don't have to stay fixed in a hierarchy. And then moons cannot be – That's why you don't want that many addresses kind of running around the network 4 billion times. 4 billion is a big number.

So address space is not that big, right? That's about the size of IPV4. People are always concerned that it's not as big as the population of the earth, which is a valid concern in about 10 years or more. The outer space could potentially be expanded by quorum of the galaxies. But for the time being, it's probably roughly actually the number of people who are online today. And the thinking being that, like I was saying before, for any finite asset, a finite asset generally has some value. So ideally, a planet just costs more than you would make spamming the network with it. And the star of a star of course – And so I think that's kind of the baseline value of the address space. Basically, what would someone pay for a planet to use this system? And then, of course, in turn, the question is like why is the system like worth paying for at all? And that I think boils down to its day-to-day usability, which is why we care about making things that people actually use from day-to-day.

So assuming that someone will pay five bucks for a planet, and there are quite a lot of people will pay for a planet at that price, then the network is in turn pretty valuable. And we're not quite there yet. That's an insane valuation. But one of our primary concerns, as Urbit gets more and more useful, is basically how can you distribute address space to people who can make significant contributions to the system itself in a variety of different ways? It could be at the application layer, inventing new stuff to do. It could be at the kernel layer, making things faster and more efficient.

And as you spread address space to them, not only does it have some actual financial value today, because these things are – They're all basically NFTs. I think we were like the fifth person to use the ERC721 standard when we actually did this. But now everyone knows what an NFT is. So they do trade, and they are a live cryptographic asset. But the thing that's exciting to me about it is our ability to basically give developers, people who – And creative people generally. Kind of material stake in a network that they can also contribute back to. So you can leverage some of that value and sell even part of that asset.

As of late, the community developed a way for you to sort of fractionalize these things. So you can even sell a part of it and fund your ability to continue working on or contributing to the network in some material fashion. So there's sort of like a feedback loop of Urbit gets more useful, and therefore it gets more valuable. And the developers can leverage that value to continue making it more useful, which I think is – I mean, ideally, is a virtuous feedback loop. My hope is that people don't simply get lazy and sort of sit on that value. But I'm pretty excited to work on this thing every day. So I think it's working okay so far.

[00:27:53] KP: What are some of the current active projects or recent releases?

[00:27:58] GWP: There's so much going on. It's hard for me to – I realized the other day that I've actually just completely lost track of all the things that people are working on. I mean, we are sort of in this process of continually updating landscape. And then like I mentioned before, the software distribution stuff is probably the most exciting thing we shipped that we worked really hard on last year.

So now I can basically just send you a link within Urbit, and you can click that link, and it will just download a piece of software for me and install it, and you can run it right away. And so people have been experimenting with all kinds of stuff. Almost every day, I see something new that I hadn't seen before. And most of them are total toys. It's like chess – There's a file system explorer that was actually pretty amazing the other day. Oh, someone made an actual like 2FA. So like self-hosted two-factor auth that actually uses your camera to go and scan QR codes and generate 2FA codes, which is really cool. But these are just people sort of like moving quickly experimenting.

So your Urbit has to run somewhere. And we, as Tlon, have worked on a hosting service that we've been improving so quietly over the past year or so. There are now three or four other hosting projects. I think two of which are real actual companies that can actually host you, which is so exciting to see, because that's really important to getting people on the network. There are a few bigger infrastructure projects. I think neither of them – There are two, both of which I think did their entire fundraising, and it brought in their initial teams over Urbit itself. But I don't think either one of them, which I really want to talk about, because I'm excited about them. But I don't think either one of them is public or wants to be publicly known yet. So I can't get into it. But I suppose that's a good teaser for like some of that information is kind of kicking around the network. Those are the things that come to mind.

We hosted a conference about a month ago in Austin, which was pretty impressive, like to me certainly. And so ever since then, I've had this kind of feeling of like I don't even know what's going on anymore, because my feeling at the conference was that people were sort of constantly coming up to me telling me about things that they were working on that were Urbit-centric, and I had never met them before or seen them before. And it felt six, nine months ago, like the community was something I had a good handle on and I kind of knew what was going on. But at this point, I can't really keep up with it.

[00:30:21] KP: You'd mention the address space being roughly about the size of the total population of human beings. Do you have a vision that essentially all people online will eventually have their own personal server? Is that truly the goal of the project?

[00:30:38] GWP: Yeah. Soon the address space is about half the size of the population of the planet. It's about four billion. And I'm pretty sure like seven or eight – Seven change or so. That isn't too difficult to change if we do in fact approach a world in which everybody has a personal server. Yeah, I honestly think that's reasonably inevitable. I think it's very clear that networked computing is like an extension of the way that we think. I think that's the kind of maybe optimistic take on what is exciting about the Internet in general. Ignoring some of the pain points of centralization, it's kind of amazing that the whole world is connected as easily as it is connected. And network computers are this sort of accelerant for how we form communities and think about both ourselves and the world around us.

I don't think that it wouldn't make sense or it would be historically atypical for something that is that important for this number of people to be controlled by a very small population. Like if you look at the way that cities are built or the way that even like empires are distributed across physical territory, there's a lot of distribution of ownership. The people who live in a particular place, and I mean like historically, kind of have a very direct material relationship with the buildings around them, the people who run that city, or that province, or whatever. And the internet, I think, or sort of like whatever the Internet becomes, needs to evolve. Or not even needs to, like will just naturally evolve in that direction. And I don't think there's any way that that's going to happen with the technology that we have. Whether or not that's Urbit, I'm not really sure. But certainly, we build with that future in mind. Like, yeah, when we talk about this thing as an operating system, I mean, look, it's an overlay OS. It doesn't run on metal. But the idea is that, yes, day-to-day people should live inside of their own personal server and be able to sort of compute freely, because I think that's what people want to do.

[00:32:53] KP: And for a developer that's thinking about starting a new project, what are the key features that are going to pull them into Urbit being the platform upon which they want to build?

[00:33:03] GWP: Yeah, great question. This is a question we didn't historically have great answers to. And I think these answers will continue to get better. But they are improving. So if you're a developer today and you want to build an application using the legacy stack, you are in charge of basically DevOps, right? You have to not only make decisions about what stack are you going to use, but you have to keep that entire sort of – I think it really looks like a – It's like a really, really tall tower of wooden blocks, which as I'm imagining it, it's like just scary a little –

Makes you a little uneasy to think about. Because if you know step the wrong way in the room or whatever, the whole thing can come crashing down.

I think most developers are familiar with this. And obviously, like the industrials or software stack has gotten better. But it tends to be that if you want to build a conventional app, you just have to compose a whole lot of software that you didn't write and you don't understand, and it's generally difficult to manage. The Urbit alternative is saying, "Look, the problems of identity and authentication, data storage and networking, how your application business logic is structured and how you do updates, we solve all of that. That's just part of the stack. So we take that entire stack, we just make it into one thing. Everybody runs the same thing. And you should focus on, basically, user experience and sort of like what actual value does your application provide to the user.

So the developer experience of Urbit, I mean, as a designer, like someone who cares about stuff , I at once, I'm like, "This is so immature. This could be so much better. Like we have so much work to do." And then on the other hand I'm like, "This is amazing. This is so much better than the alternative," because it lets people very quickly experiment with making things in a way that you just can't in legacy systems, because you're sort of burdened by the tools and their sort of overall complexity, or just the fact that they're very sort of industrial scale.

So I think that the Urbit model of sort of like you have this very thick stack and you let a developer just focus on the sort of business logic and user experience of what they're trying to provide. And then furthermore, they never run a server. So when you ship that software, I mean, sure, you have to have a node online somewhere, but you're not actually dealing with storing user data, which of course also has its own like liability issues and so on. It's just a much easier experience. I sometimes joke that like your best case outcome of like building a SaaS app is like you get to testify in front of congress. Like that's no longer the case in an Urbit world.

[00:35:39] KP: Can you talk more about the cloud opportunity? I was a little timid about maybe downloading and running this on my own Unix server. Can I get some sort of cloud version of it? And are there any advantages in having that?

[00:35:53] GWP: Yeah, definitely. I mean, Urbit is pretty friendly now. I think it even – I should know this. I'd even have to say, I think like Urbit runs on Windows, Linux, Mac, whatever. Obviously it won't run on iOS because it's a virtual machine. I think our desktop GUI does that too. So just trying Urbit out with what we call a comet, which is like an anonymous identity, is quite easy to do on just about any platform. But they're ephemeral. And, yeah, the whole idea is that you can access your Urbit from anywhere.

So what most people did for a long time was just get yourself a DigitalOcean droplet, or an Amazon box, whatever. You can get a very cheap VPS and just install Urbit in the command line, which is also – I think we've gotten the installation – I had to do it the other day. I think it's like two lines in the command line. Pretty much everything is bundled. If you're not going to build from source, you just curl binary, or whatever, w get a binary, and you're up.

We also though – Yeah, we have been working on hosting infrastructure, which is more of like a turnkey, so I can invite a friend who doesn't care about the command line and have them just get set up in a browser and download an app. Still working on that. You can get on the waitlist at tlon.io. And we're sort of slowly inviting friends in and testing it out.

But yeah, if you do sort of start to go down the rabbit hole, I do recommend having a node in the cloud somewhere. I do use Urbit on my phone pretty regularly. It's nice to be able to access it from anywhere.

[00:37:19] KP: What's the phone experience like? Are you running the actual virtual machine or connecting it through an app? Or how does that work?

[00:37:26] GWP: Right now, I'm day-to-day just doing it mobile web. So just over HTTP. People have somewhat at one point – The thing I loved was they got an E Ink Android phone and actually installed Urbit natively. So you can run it on Android devices. And then we've been working on iOS app, but that's all happening. So that's like Swift, but speaking to Urbit over HTTP.

If you're purely a frontend dev, you can kind of treat Urbit like a personal Firebase plus networking. Like it's just a back end where you're pushing and pulling data from it. And you

could use your own – Whatever it might be, as an interface framework, to just talk to it directly. But anyway, yeah, the phone is just an HTTP client. It's just in a browser. It's easy to see us or like people have experimented with. And I think we'll eventually put out an iOS app. But you could see building interfaces in other like UI frameworks.

[00:38:18] KP: Where do you see the project going in the next five years?

[00:38:23] GWP: I do think it's kind of like time for better communication tooling. So I'm excited about the – I feel like probably the next like year, or 18 months, two years, there's a lot of ways in which sort of landscape groups can get more powerful and more exciting just for I think this pattern of like you have a few friends who are assembled around a particular idea or, I don't know, like a sports team or whatever building computers in a certain way, being outside together, whatever it is. Doing that in a group chat just isn't really exactly quite what people are looking for. And so I'm excited to make Landscape a sort of really great tool for that that feels like it can last a really long time, that it's very sort of stable and secure.

And I think that that's sort of the first step towards what I really hope, which is that Landscape can become sort of like a totally general-purpose operating environment for people in the spirit of the early PCOS's, right? Like what we don't have for – Not only do we not have a platform for personal cloud computing, which maybe Urbit is starting to fill. But what we really don't have is a way that all of our applications can work together, right? So right now I have a bunch of browser tabs. It's pretty rudimentary in terms of user experience.

So if Urbit is mature enough, well, okay, now I can have a bunch of browser tabs that actually share identity, share data, can pass messages back and forth. But what you really want is kind of like a holistic environment where these things can live together. And so I think that's the sort of five-year challenge, is like can you make landscape a fully-featured kind of like user experience where I can run multiple apps together in this very stripped down non-advertising-centric way?

The very loose way I've been describing this is sort of like WeChat if it was like designed by IKEA. You want this sort of like digital basics OS feel. So I think that's where – And I sort of – I like this idea a lot, and I think it has legs. I think it'll take us time to get there, and we'll kind of

get there step-by-step by building individual applications and experiences that we really like and and sort of folding the community into that process.

[00:40:37] KP: Well, where is the best place online that people can follow the project?

[00:40:41] GWP: It's easy to find us at urbit.org. And kind of all the resources are there. We don't really use conventional social media, but we do have a twitter, which is not the most informative thing in the world. But it is a way to get some updates. Either of those are probably pretty good. Urbit.org really contains everything and sort of find the right resource. We do put out – I feel like the newsletter is probably a great thing if you just kind of want to passively follow along. The newsletter that you can sign up for on urbit.org is where we kind of push periodic updates, and it's a good way to sort of just see what we're up to. But if you want to go deeper, yeah, urbit.org also has resources for you to get set up, check things out more deeply, and potentially get involved in grants, whatever.

[00:41:23] KP: Very cool. We'll have a link in the show notes. Galen, thank you so much for coming on Software Engineering Daily.

[00:41:30] GWP: My pleasure. Thanks for having me back.

[END]