Open Source Week – Day 1

IBM Code Tech Talks

02/26/2018

https://developer.ibm.com/code/techtalks/open-source-week/

>> Welcome, everyone. This is Chris Ferris, I'm engineer and CTO for Open Tech. For Open Source Week, we are going to be sharing with you a number of webinars discussing IBM's approach to open source, and a number of the key initiatives that IBM is investing in from a open source perspective, talk to you about what those projects are, how IBM is helping to innovate in those communities. Hopefully you will get something out of it. I'll kick off with caveat emptor, not all open source is open. I'll give you a perspective on how IBM approaches open source and why we think something called governance is critically important for everybody to understand. Let's get right into it. Everybody is aware of the fact that open source is really eating the software industry. I personally can't think of a single new software capability that has been delivered to the market through, exclusively through the proprietary based solution. About every meaningful development in software over the past three, four years has been a open source project or has been brought to market through collaboration around open source.

This is being reflected in the market not just from a vendor perspective, vendors contributing to open source initiatives, what we are starting to see is consumers of technology increasing both their use and their contribution to open source and many of them see it as a competitive advantage. The important thing to understand, not all open source is built the same. Here I have the graphic of the furniture left by the side of the road.

I think the reality is that I think a lot of open source projects are on GitHub, are basically the moral equivalents of furniture left by the side of the road. Somebody has developed some capability, and they may be in a proprietary way, and then they decide that time for us has moved on, we are going to put this in open source and we can turn our attention to other more important things. And the kind of projects that I'm talking about are projects that are published as open source, they have a nice open source license on them, maybe MIT, and they basically say here you go, it's free, take it. But nobody will be around to sustain the project going forward, nobody is going to be around to monitor issues as they come in or to merge code requests and so forth. It's just out there for anyone to take as they see fit. This is not all open source but there is a lot of that out there. It is important that people be aware of that fact. You have to be careful when you start to choose about an open source component to integrate into your application or into your platform in some way, that you understand where it's coming from, whether or not somebody is going to be around to maintain it going forward, and how, what is the process for continuing to sustain this particular project.

The next open source that is out there is open source that's essentially delivered by either a single individual or primarily, or a single vendor, single company. This is sort of the benevolent dictator for life type approach to software development in open source.

There is good examples of this. Obviously Linux is probably the premier example of a technology that is being developed as open source, but with [inaudible] providing the dictator role. But it is important to understand that Linux Foundation recognizes full well that Linux isn't going to be around forever as much as we might like that to be the case.

So we have to prepare for that inevitability so we have a path going forward even beyond Linus with regard to Linux kernel and its development. They are cultivating basically the set of engineering talent that is going to be able to replace him over time.

They are doing that in a meaningful and well governed way under the aegis of the Linux Foundation. But there are other projects that are essentially single vendor in the sense that nobody else is contributing to those projects. The challenge there and the risk there, right, is that and I should sort of, whenever there is any sort of single point of control, there is some risk. There are examples, and I don't necessarily have to go into all of them, but there are examples of where a company either acquired some open source capability through a merger and acquisition, or developed something in-house, put it out as open source, and then the business decided it is not in our interest to continue to develop this and to maintain and sustain it. So we are going to drop it.

The good thing is it's open source so their code is there. And people can pick it up and take it and they can fork it, they can continue to develop whatever they want to do with it. But that is again part of the risk. If you are using a platform in particular, if you are using a significant piece of technology as a strategic part of your enterprise software portfolio, and all of a sudden the primary support or primary maintainer or primary developer of the technology decides this isn't in our best interest anymore, are you really going to be in a position, are you going to pick up the ball from there and run with it yourself? That is a question that every CTO should ask themselves.

There is a certain amount of risk there when you have a single point of control. But the approach that IBM has taken for some time now is this notion of open governance. By open governance, we mean responsible licensing. There is multiple open source licenses, and this isn't the time or place to get into a lengthy discussion about open source licenses and which ones are good and which ones are bad.

But you want to choose a license that is going to be acceptable by the community that is going to be consuming your technology. For us, that happens to typically fall into some of the more, if you will, corporate friendly licenses like Apache and Eclipse. But there are other licenses equally valuable to the community.

We like projects that have a accessible commit process, by this I mean that they described full well what the process is for contributing to the project. How do I land my first commit. How do I submit issues or propose new features, how do I engage in the whole development process with the maintainers of that project.

We look for projects that have a diverse ecosystem and a diverse community. And by this we mean, you know, again that it's not just a single vendor and a bunch of their employees working on a project, but where you have stakeholders from multiple vendors, from multiple consumers of the technology, for instance, all contributing to and helping to lead the project going forward. This also sort of contributes to the notion of participative meritocracy. By that I mean it's the people that contribute to engage and help move the ball forward in a project, that are the ones that tend to get their voices heard.

Again, I know the word meritocracy comes up poorly seems in open source context, but I think it is important to understand that Brian, a colleague of mine working in Hyperledger will talk about it in a moment, he likes to call it duocracy. Projects in open source are best when people have a scratch to itch get in there and scratch it. Collectively we feel this is the concept of open governance. A lot of the projects that IBM is investing significantly in, we look to bring under a umbrella of open governance. So that typically means that we bring it to one of these three communities, whether it's Apache Software Foundation, the Eclipse Foundation or the Linux Foundation, we found that over time, projects that are developed and sustained under open governance tend to yield results that are significantly greater than you can achieve in other forms of open source projects.

In each of these cases you have a community that is, that is focused on separating apart the business aspects of running a foundation from the technical aspects of developing open source projects and so forth.

We have found that this tends to have the best success in delivering reasonable output and developing communities that offer essentially a safe place to collaborate and to innovate.

You think about Apache, for instance, Apache was developed initially back in the early 2000s, to be the home for the Apache HTP server. Yet what we found is now there are well over 300 top level projects at Apache many of which have nothing to do with HDP. A lot of big data, there is a lot of SOAP and XML and Web services and going forward, now we are starting to see things like serverless with OpenWhisk. The Apache Software Foundation created a community where you can get together even with your fiercest competitors and collaborate on developing software platforms that matter.

I think Eclipse falls into the same category, right? It was created initially as foundation to host the development of the Eclipse IDE framework which was Java IDE framework contributed initially by IBM. But again as with Apache, now Eclipse hosts more than 300 projects itself, many of which have nothing to do with IDEs or Java. Why is that? It's because the community, the development ethos that they develop within Eclipse created a safe place to innovate and collaborate even amongst the fiercest of competitors.

Finally we have the Linux Foundation. Linux was created to host the development of the Linux kernel, but now there is a number of what they call collaborative projects, including some that I'll talk about and that you will hear about later this week, that aren't really related to the Linux kernel, but again the Linux Foundation has developed this approach to open source governance that scales, and that tends to deliver projects that have greater success than you would find elsewhere. IBM, we have had a long history from a open source perspective, reaches back into late 1990s with our early work in support of Linux, even before the Linux Foundation itself.

We were one of the first companies to really legitimize the whole open source movement when we indemnified our clients when they chose to deploy Linux in their enterprises, we would indemnify them from any of the lawsuit silliness that was going on at the time.

Again we were there at the start of Apache, start of Linux Foundation and Eclipse. Then fast forward, right, we were instrumental in helping to stand up the OpenStack foundation, Cloud Foundry foundation, we have been investing significantly in Spark. The Node.js foundation was in large part I think a function of IBM's leadership in the community to bring a set of communities back together after they had suffered a fork. Docker, Kubernetes, native compute foundation, Hyperledger and Blockchain, all these things throughout history IBM has been investing fairly heavily in a number of the key open source projects that probably matter to many of you.

My colleague Todd Moore and I, Todd is the VP for open tech, I'm the CTO for open tech. We collaborated on a white paper that outlined IBM's approach to open tech, and if you have download the deck, you can find the link. But you can also basically type in to your Google or Bing IBM open technology and hilt on feeling lucky and this should pop to the top.

This outlines the theme that I'm presenting to you in this talk but it goes into more depth.

In that paper, I outline IBM's approach to how do we earn our stripes, right? Because while and I'll talk a little about some of IBM's organic open source, projects that come from our research labs or from the development labs, and are open sourced. We also engage in projects that were brought to the open source community by others. How exactly does that happen? One of the first things we do is we get in and we roll up our sleeves and start making small contributions. We get in there and try and fix some bugs. We try to build some expertise around the platform by engaging in consuming it, using it and so forth.

One of the things that we like to do when we first engage in a project that isn't our own is, and this is something that I think is valued, oftentimes, by these communities that we join, we do a lot of the dirty work. We do a lot of the work that isn't necessarily glamorous, and doesn't necessarily get as much attention, but is urgently needed by these projects, so that can be helping to sort of go through and triage the issues list. That could be going through and helping with improving the documentation, some of the samples and just how does one get up and running and start up with a project.

It can also be things like interactualization and globalization, helping to localize the code base and provide some translations to various other languages for the project so that it can be consumed in multiple markets.

We like to help foster these open governance policies in these communities that we join. We are looking to help to grow and build these communities and extend the ecosystem. There again, I think IBM's perspective on open source is that the most successful and the best open source initiatives are those that have multiple ISVs and multiple vendors bringing capability to bear in the market, so the consumers have a choice.

At the end of the day, we are also looking, we are not looking to necessarily just be philanthropists and contributing to development of some of these projects but we want to consume the technology for our own purposes internally, or to deliver offerings to market for our customers.

IBM, we have another theme, we took the trouble to trademark this one, IBM is open by design. I think it's an important characteristic to understand about the approach that IBM takes from a open technology and open source perspective.

We recognize that there is really three different approaches that you can have when it comes to delivering offerings around an open source base. One would be that it's just open source only, right, and then you have the problem of batteries not included and significant assembly is required.

This isn't just an Ikea desk which you are putting together necessarily that's got some pictograms that help you assemble the desk. We are talking about things like hadoop or Hyperledger Fabric or OpenStack or Cloud Foundry or any of the other major platforms, Spark is another example, you know, Kubernetes, where it's not necessarily going to be the case, is you are going to be able to go down to a open source repository and download everything and just push a button and have it up and running in your enterprise. It is almost never that simple.

The other approach would be proprietary, where you are almost assured that there is going to be some vendor lock and this is typically where a vendor has incorporated some open source technology into their offering, but they have done so in a way that you can't access it.

It's sort of hidden behind the proprietary APIs or some other interface. While it's nice that they have an open source foundation, by virtue of the fact that it's proprietary, you are almost assured to get the vendor lock, and that is kind of not why people are doing open source. The reason that we are doing open source is so that we get interoperability and portability workloads and so forth.

The other approach and this is the approach that IBM typically takes is that we like to partner both with the community, right, to help to advance that technology and so we contribute towards the development and continued sustainment, but we also then partner with our clients to help understand what are their requirements, and then working with them and working with the community to provide that sort of bridge, that we also look to make sure that when we are doing this, that we are delivering offerings that fully expose the open source APIs, and ABIs, such that the client actually has and retains the choice that they are expecting from open source initiatives.

Let's take an example and this is a project that I've been working on personally for the past couple of years now, that I'll have a brief story about this, and I think you can from this take away our approach to how we do open source.

In about 2014, IBM began experimenting with Blockchain technology. We started looking at Bitcoin, sort of an early beta stage, and we started trying to understand and explore the underlying technologies. We had people in research doing various things around consensus. Our conclusion was basically that the problem with the technology that was around in 2015 let's say, it lacked a few set of capabilities that our enterprise customers really demanded. I'm not going to get into details here. You can tune into my next talk in the next half hour when I get into details on this.

What we ended up doing was developing our own spoke Blockchain platform that we call the IBM open Blockchain. The team came to me as the CTO for open tech and said we recognize that this isn't going to be successful if it's just an IBM thing. We need to take this to open source and can you partner with us to bring the technology into open source. And I said sure. I contacted my friend Jim at the Linux Foundation and said, I've got this proposal for a Blockchain project, that we put under open governance at the Linux Foundation. He graciously said this is a great idea.

So we got together with about three of our best friends in the industry, whether it's vendors, banks, insurance, healthcare, supply chain and the very early stages of 2016, almost two years ago now last week, we established the Hyperledger project under open governance at the Linux Foundation.

This project again has been one of the real success stories, I think both from an IBM perspective, because it was an opportunity for us to highlight some of the innovation that we were able to develop in our own labs, but then taking that and creating a true open governance project that is starting to really see some, have some legs to it. It has been two years now. We have got over 25,000 commits to the various projects. I'll talk about those in the next half hour.

There is over 200 members including about 30 from China. There is a number of different working groups. We have got, actually it's 88,000 meetup groups worldwide with about 22,000 participants signed up to those meetup groups.

We are producing a number of media hits. There is a awful lot of attention being paid to the work we are doing with Hyperledger I think because of the value it has in going beyond cryptocurrency from the Blockchain perspective.

And I think also, because of the open governance model that we have developed, and again we took this project as originally contributed the Hyperledger Fabric is what we eventually end up rebranding the open Blockchain to, Hyperledger Fabric 1.0 was published last summer in July, very early July. We had gone from a single vendor project essentially, with just IBMers contributing to it, to a total of 27 organizations, 159 developers, only about 40 percent of which were IBMers. We managed to process through about 3500 chain sets, which is incredible when you think about it, and delivering something to market that then IBM has many offerings around.

So again our approach is basically to follow similar things to that whether it's, if it's an IBM organic innovation, then we try and take a path that is similar to what I just described. If it's something like getting involved in cog no compute foundation working on Kubernetes, for instance, we roll up our sleeves and engaged in what I talked about earlier.

Take a look at the IBM GitHub organization. About two years ago this was crickets, there wasn't a lot going on here. I think there was actually at one point, there was a prohibition about putting anything there. We wanted to just sort of prevent anybody from poaching on our own name. But we have turned that actually around. We take a look at IBM and our contributions to open source, not just the major initiative that I talked about, for instance, but just IBM doing things in open source, and there is a huge sprawl if you will across GitHub, that GitHub landscape of IBM projects, either brand specific.

(ringing).

Or just people putting out samples for some of our offerings and so forth. There is over 1200 repositories that we have been able to find, but we are starting to move these all under the umbrella of the IBM org on GitHub. We have got about 1,000, 1100 people in that organization now, and that continues to grow. I think you will see that continue to grow. If you look at IBM.GitHub.I/O we have pulled together that list of 1200 repositories. We have provided links to some of the various key open source links at IBM.

I would be remiss, since this whole series of Open Source Week is being hosted by the IBM code team, that I'm a proud member of, I would be remiss if I didn't provide a link to developer.IBM.com/code, where you can learn more about not just the open source work that we are doing, but also some of the key innovations around some of those technologies that I talked about.

So that's pretty much it. I'll turn it over now to questions for the next couple minutes. Where am I going to find those? Is there a chat, I don't see any questions in the Q&A. Do we

have questions?
>> Chat window.
>> CHRIS FERRIS: I see Q&A, but I don't see a chat window.
>> MARC-ARTHUR PIERRE LOUIS: If there is any question, can you call them out to Chris?
>> Let me look through right now.
>> CHRIS FERRIS: If somebody could read them to me, that would be great.
>> I don't see any questions, Chris.
>> CHRIS FERRIS: Okay. That's it then. I'll thank everybody for attending, and stick around and in the next two to three minutes, we will be kicking off the Hyperledger community update. So thanks again, everyone.
>> Hi, Chris, do you want to start at the half hour? Or go ahead and continue.
>> CHRIS FERRIS: We should, if that is what the schedule is, take a quick break.
>> KATHY GHANEEI: That's fine.
>> Kathy, a brief question. I see a green square on my screen.
>> KATHY GHANEEI: Green square?
>> CHRIS FERRIS: If you don't see it then --
>> KATHY GHANEEI: No, I don't see it.
>> CHRIS FERRIS: Some shadow of I think Power Point.
>> KATHY GHANEEI: You know what, you probably paused it and started sharing something else. That is what it is.
>> CHRIS FERRIS: Am I sharing something else now?
>> KATHY GHANEEI: No. You are just sharing the main --
>> CHRIS FERRIS: Okay. All right. Thanks. Welcome back, everyone. I'm Chris Ferris, IBM CTO for open technology, and if you were around for our previous half hour, this is Open Source Week at IBM, and we are going to cover a number of the technology areas that IBM is investing in in 2018. I'm going to cover the first community update talking about the work that IBM is doing with a lot of its partners in Hyperledger.
So today's agenda, what is Hyperledger and why and how was it created, what are the projects we are working on there, what is next for the community, we will talk about Hyperledger and IBM, how you can get started, offer a call to action, and give you additional resources that you can consume at your leisure.
What is Hyperledger? Hyperledger is a open source organization that we created in the Linux Foundation. Its purpose is basically to develop Blockchain technology for the enterprise.
This is not about creating Bitcoin 3.0 or anything like that. This is really an open source initiative that is focused on delivering enterprise requirements by leveraging Blockchain

technology.

It's hosted by the Linux Foundation. It's a collaborative effort. There is multiple companies involved and we will get into details there. And basically it's got the support of a number of industries. This isn't just about banking or finance or securities or insurance. It's healthcare, it's supply chain, it's you name it. There's just no communities that aren't, and industries that aren't impacted by the work we are doing.

How was it created? Back in 2014, IBM was experimenting with various Blockchain technologies. We started exploring Bitcoin and some of the other cryptocurrencies, we started looking at Ethereum, which I think it became beta in 2015 when we were looking closely at it. We found that it had a number of limitations. One of the, from an enterprise perspective, right, we would talk to our clients about Blockchain technology distributed ledgers and what that particular technology held for them and for their industries.

But we came away with some concerns about the sort of the state of the technology at that time. One of them was limited throughput. Friends at a conference used this phrase, he said I'll give you three numbers, 7, 20 and 2400. You ask yourself, what do those mean? 7 is about the transaction throughput rate for Bitcoin, for instance. Again there is other approaches like Bitcoin cash that have pushed the needle on that a little but not significantly.

20 is about the transaction, throughput rate of Ethereum although again there is a lot of hope hanging on proof of state to start increasing the transaction rates of Ethereum, but again not huge. 2400 that is the number of transactions we process a second. When you think about taking this technology and applying it to a number of enterprise use cases, you are going to need thousands of transactions a second, in many cases, to be able to sustain type of transaction rates that many of our enterprises deal with on a daily basis.

Another concern was the slow confirmations. So one of the things about, let's take Bitcoin as an example, it is you have to wait until 51 percent of the network has committed that transaction that you just executed before it's final, before it's really, can be confirmed that it won't be reversed.

So think about that in the context of retail, for instance, right? You can't go into retail outlet and then pay for something with Bitcoin, what are you going to do, wait in the corner for two hours while the thing settles, well, not two hours but it could be 20 minutes, it could be up to an hour before your transaction settles. That can be problematic if you need to have confirmation and finality in the transactions.

Another concern was anonymous processors, not just, so there is certainly the mining aspect of things and you don't know who the miners are for instance. They are all anonymous. But also they are transactors in the network are typically anonymous. It is just an account number.

That is what Bitcoin and Ethereum and many of the other cryptocurrency platforms, and this is problematic, if you are a banker, securities firm, you have certain customer or money laundering regulations that you have to abide by. So you can't deal with anonymous clients. You have to know who they are.

There is ongoing work to try and provide identity in the context of some of the networks, but for the most part, their core design principle has been to avoid that kind of thing, to work around that. That is a problem, from a enterprise perspective. I talked about settlement finality. Poor governance, right, so part of the problem is, yes, some of the technologies are open source, but the governance that they have and I talked in my previous talk about open governance and how we approach things from an open governance perspective, they don't really have well-defined governance for some of these platforms.

That can be a bit of a problem. Then finally, sort of the lack of privacy, right, is a concern. If you are a securities firm, you are not necessarily going to want to have people know which securities you are trading on, because then they will start to arbitrage on that, and that can be bad.

So there is a number of different enterprise use cases where confidentiality is paramount.

Let's look at the momentum. IBM contributed the work -- oh, let me finish the story. I should get back. We found implementations to be requiring that we rethink how do we take Blockchain as a technology and apply it in a way that can be consumed by enterprise without bringing a lot of the baggage that I just described in terms of throughput rates and cryptocurrencies and lack of privacy and so forth.

We started developing something called the IBM open Blockchain platform, as internal development. We were going to bring it to market and so forth. But the team came to me as CTO for open tech and said hey, we recognize that this isn't going to be a success, if it's just IBM and only IBM can deliver this technology. It needs to be a community asset. It needs to be open source.

They asked me to go and make that so. So I worked with the Linux Foundation, Jim Zemlin, director of the Linux Foundation, I called up and said I have a idea for Blockchain for enterprise project that we would like to host under open governance at the Linux Foundation. So we were able to make it so. We got together initially with about 30 partners, and in the February of 2016, just two years ago now, we launched the Hyperledger project.

That project has been the fastest growing project ever from a Linux Foundation perspective. You have got the most interest of any project. Jim's in-box was inundated in the first couple weeks after our launch with requests to join and so forth.

We processed about 25,000 commits. We have got 9 projects total and I'll go into each one of those in brief in just a moment. We have got two, two of our projects, Hyperledger Fabric and Hyperledger Sawtooth have both been through their initial 1.0 delivery. Fabric is getting ready for its 1.1 release later this month probably. We have got over 200 customers and about 30 or more in China and Far East. We have got twelve active community working groups that are churning away talking about various aspects of Blockchain from an enterprise perspective, identity work group to architecture work group, requirements working group. We have a healthcare working group, a governments, government industry working group. We have got about 88 meetup groups worldwide in cities across the globe, about 22,000 participants registered.

We are getting about 600 or more media hits on a monthly basis, not all of them from coin desk.

Just a brief note on the enterprise Ethereum line. If you are familiar with the Blockchain space, you will note that there is a certain degree of, I think from a media hype perspective, of pitting the Enterprise Ethereum Alliance against the work we are doing in Hyperledger. I want to clear that up. The Enterprise Ethereum Alliance is a group of companies that have gotten together led by Microsoft and a company called ConsenSys with a focus on delivering enterprise requirements into the Ethereum platform. The focus is on taking Ethereum and making it more relevant to enterprise use.

It claims to be the largest Blockchain org, except again I'm not sure exactly how we are measuring, but let's leave it at that. They are focused primarily on the development of standards for Ethereum. If you are familiar with how Ethereum works from a governance perspective there is a set of specifications that are written, many authored by the grandfather if you will of Ethereum and the driving force behind it. Those specifications are implemented in various open source initiatives that implement the, either the Ethereum virtual machine specification or some of the protocol specifications.

But the Enterprise Ethereum Alliance focused on the standard part of it, writing specifications not necessarily on developing open source. There is talk about delivering a reference implementation, but it wasn't expected that that would necessarily be a production ready type implementation. It has been a little more than a year now since it was officially launched. There has been a number of organizational changes. I should have updated it just this past couple of weeks, they have actually hired a Executive Director, and they are starting to move things I think in a little more deliberative manner. And but I want to make it clear that this isn't necessarily a competitor to Hyperledger. If they are developing standards for Ethereum, and I'll talk about how Hyperledger is working with Ethereum, then it's likely if those standards or those specifications become important to the industry that we will implement them in Hyperledger, wherever that is appropriate.

Let's talk about the various projects we have. As I mentioned there is actually nine projects under Hyperledger, there is five what we call distributed ledger platforms, and then there are four tool sets if you will. The first that I'll talk about is the one that IBM initially contributed that was the first project that was incubated in Hyperledger. It was also the first to reach 1.0. It was actually the contribution itself was a collaborative effort between a company called digital assets and IBM, we had been collaborating around the open Blockchain a little earlier on.

So we actually joined forces and we brought together the technologies that each company had been working on, and we established the project as the synergy of those two pieces of work.

Blockchain, Hyperledger Fabric rather, one of the terms of art in Blockchain land is smart contracts. I don't like that term, and in IBM we tend to avoid its use. These aren't smart and they are not contracts. But we call them Chaincode. And Hyperledger Fabric supports Chaincode that is written in Go, and when we released 1.1 in the next couple of weeks we will also have full support for JavaScript. It is available in our alpha release presently.

It runs these Chaincodes in secure Docker containers, these are independent of the actual tier nodes themselves and that gives a certain isolation to prevent various forms of malicious attack.

We provide something we call channels that allow participants involved in a transaction to be the exclusive ones that can see the context of the transaction. We don't, unlike the other public Blockchain that works like Ethereum and Bitcoin and some of the others, we don't disseminate the transaction everywhere. We don't try and just encrypt it but we actually are able to segregate it. So we can create these channels which you think of as a overlay network on top of the overall Blockchain network itself.

As I mentioned it was the first project to be incubated. The first to graduate from incubation and become what we call an active project at Hyperledger and it's the first to reach 1.0. It was a collaborative effort. It started out a pretty much IBM, then it was IBM and digital assets, and it grew to the point where 1.0 we had 27, developers from 27 organizations, 159 developers total, only about 40 percent of those were IBMers. Processing about 3500 chain sets all together to deliver that 1.0 release. It is something I'm really proud of being a part of.

The next project that was incubated at Hyperledger was called Sawtooth. It was originally called Sawtooth Lake. It was contributed initially by Intel. It was the second project incubated and the second to reach active status to graduate from incubation, and it was also the second project just the past couple weeks now to reach 1.0 level of maturity.

We are also very proud of the fact that we now have two of our DLP platform projects out as 1.0. It is really something, and Hyperledger Sawtooth supports both permissioned and permissionless deployments. It is closer to supporting a public network kind of deployment, and that is because unlike fabric, fabric doesn't use proof of work in its consensus. There isn't time to get into technical details. We don't use proof of work. We are using traditional consensus algorithms where there is Byzantine fault tolerance or whether it's crash fault tolerant protocols like raft and Paxos and so forth, but Sawtooth on the other hand has developed something they call proof of elapsed time or poet, and this approach mimics proof of work but without any energy consumption.

Basically it's doing this by leveraging the secure enclave or SGX chip that Intel produces, and it runs the sort of mining operation if you will inside this secure enclave and in doing so, it can replace the computation of a hash that satisfies a certain set of criteria, solving this cryptographic puzzle, with you just compute a random number and wait. You just sleep. Then the first node to wake up wins and gets to mine a block.

It's a novel approach, and obviously there is the hurdle of getting it deployed and getting SGX broadly deployed and so forth. But that aside, it's quite compelling technology.

The other thing that is interesting is Sawtooth was also one of the first projects to start doing the collaborative cross project collaboration with Hyperledger borough which is our first EVM, first Ethereum based projects. Iroha is a Blockchain framework that was contributed by a start up in Japan. Collaboration with Hitachi and NTT data and Colu which is a university in Japan. It's written in C++, the emphasis for that project is on mobile application development and it provides a number of SDKs. It is similar in its architecture to

Hyperledger Fabric. It also does deliver a new consensus algorithm they call (indecipherable).

I'm not going to get into details, it's interesting but not necessarily proven. Hyperledger Indy is a Blockchain framework but its primary purpose is as a identity platform. This is contributed by the sovereign foundation and a company called Everdin. It's focused on identity and providing platform to do verifiable claims. It provides its own underlying Blockchain platform, but there is active work now with Indy and fabric and Sawtooth to find ways of essentially having the Indy identity platform part of it ride on top of the other distributed ledgers.

Finally there is Burrow, and Hyperledger Burrow was our first Ethereum based project and it fully supports the Ethereum virtual machine. It was originally something called ERIS DB, a smart contract engine, I used that term, but a smart contract engine that leveraged the Ethereum machine but it's fully enterprised. It uses tender mint for consensus. There is active work collaborating across the projects between Burrow, Sawtooth and Fabric. We have a running demo that you can find if you chase the links at the end of the deck that shows the integration potential for enabling Ethereum machine based contracts to be run whether in Fabric or Sawtooth, cool stuff there.

Getting into the tools, Hyperledger Composer is one that was also contributed by IBM and a company called Oxchains. It's a suite of tools that allows you to model your Blockchain business network and application, and from that model generate all the different pieces that are necessary to deploy to Hyperledger Fabric. It can essentially take your model and turn that into a JavaScript-based Chaincode, for the one-to-one version of the platform and it also can generate a rest based API for your Blockchain application if you will. Cool stuff there.

It is a complement to the work that is ongoing with Hyperledger Fabric, although again the Composer team is also developing and designing the tool such that you could use it to actually take that model and then generate appropriate smart contracts and other implementation paraphernalia to deploy it to other Blockchain DLTs.

Project Cello is another of the tools contributed by IBM with help from those at Soramitsu, Huawei and Intel. It is used for the DevOps side of things. This is allowing developers to deploy Blockchain networks whether it's Hyperledger Fabric or Iroha and Sawtooth, so it can terraform your environment to stand up some virtual machines. It can provision Kubernetes for instance or Docker swarm and then actually provision your Blockchain network into that continued orchestration

environment. It supports bare metal, virtual machine or Cloud.
   We have for Hyperledger Fabric been able to deploy it on all the major clouds and pretty cool stuff there. I think that going forward, we end up with, I think essentially Kubernetes is becoming the answer, what was the question, when it comes to container orchestration platforms. I think a lot of focus of cello will be on how do we optimize the Kubernetes experience and build off of that.
   Finally Hyperledger Explorer is our visualization into what is going on inside the Blockchain. It allows to you visualize transactions, blocks, network information, transaction rates and so forth, for some of the various platforms. Getting close to the end here. Oh, Quilt, I'm sorry, was the final project we incubated at Hyperledger and Quilt is contributed by NTT data and ripple, it's a Java implementation of the interledger protocol specification being developed at the W3C.
   Basically it's allowing you to swap between ledgers of different ilk. So that is an interesting thing. It is primarily oriented around payments, but again this is one of those things that is the first step in the direction of getting certain degree of interoperability between various ledger platforms.
   I'm running short on time. So just again, we have covered a lot of these things but Hyperledger Fabric is basically the things that make it unique are we have partitioned execution. We talked about the Docker isolation. We talked about the channels and so forth, talked about permissioned membership. This is important for the enterprise, being able to essentially establish a network where the participants are essentially invited to the network. We know who you are. You have a distinct identity. We can check that identity and validate it over time, providing transaction history or a set of tools that allow you to monitor, to log and so forth.
   We are trying to design it such that it's modular. It's something that we are trying to deliver as a, if you will, it's a construction kit for constructing a Blockchain network, so not all of you are going to be the same.
      (phone ringing).
   What is next for the community, the key thing is managing growth. It's growing rapidly. The focus here on fabric which is probably the largest of the projects that we have going on there is how do we get, now we are up to over 30 companies contributing to fabric. I think we are up to, pushing very close to 200 developers total.
   How do you manage that growth, right? We are finding that while we need to slow down a little bit, we need to increase our rate of delivery of releases. I think we will be moving to a

quarterly release model from here on out. Again, we are looking to release more of our projects, as 1.0, the next ones are likely to be Composer and I think Indy is working towards a 1.0 in the near term. There is a lot of work going on around Burrow round turning that into more of a plugable library for Ethereum virtual machine that we can leverage across the landscape of our project.

Here is the roadmap for some of the projects. Not every project has a well thought through roadmap, if you will. But I'll give you a sense here on this chart.

Briefly, IBM's involvement in Hyperledger, we are founding premier member, we have a seat on the governing board, I chair a Technical Steering Committee. I know a colleague of mine is the other elected member from a IBM [inaudible] only two IBMers. We are trying to get this so the projects itself, the community is much bigger than just IBM. That is a critical part of our open governance approach.

We participate across the board in projects and working groups and so forth. Again, our preferred platform from an IBM perspective delivering IBM offerings is going to be based on Hyperledger based projects.

I've got a page here that gets you started, links to the getting started guide for fabric and composer, links to our E-mail, our Rocket Chat which is a Slack alternative, stack overflow Q&A tags, we have, there is tons of work to be done, right, so basically you can if you are looking for things to do in the project. There is Jira with a lot of open issues and many of them are tagged with help wanted if you are interested in getting started.

IBM publishes a number of journeys to get you going to the technology. Get involved. If you have a meetup community near you, join that. Get more involved in the community. We are participating in a number of conferences, and if you are interested, please do encourage your employer to become a member.

Finally, I note here this research paper that we just published that gives a much more in-depth discussion of what Fabric is and what it can accomplish and achieve, including performance numbers that I think will surprise you.

Then from beyond that, from a Hyperledger perspective, there is also a set of resources that can lead you to just about anything you ever want to know about the work we are doing there. With that, I think I'll wrap it up and open up to Q&A.

>> MARC-ARTHUR PIERRE LOUIS: There is one question in there. Is this AT&T or IBM open source code?

>> CHRIS FERRIS: Is it AT&T or IBM?

>> MARC-ARTHUR PIERRE LOUIS: Open source. She is talking

about Hyperledger.
>> CHRIS FERRIS: Hyperledger has there is over 200 companies involved. The code that I discussed from the various vendors, some of it is IBM contributed initially. But IBM is a significant contributor. But there are contributors from IBM, from Intel, Hitachi, Huawei, Secure Key in Canada, State Street Bank, from the London stock exchange, Oracle. I mean a number of different companies are contributing to these projects.
>> MARC-ARTHUR PIERRE LOUIS: Sounds good. I got one of mine before you switch to the other presentation. You said you don't like to use the term smart contract. But that is going the use of that. Can you comment to that quickly?
>> CHRIS FERRIS: Again, smart contracts implies that something is smart and it's a contract. I think it's a euphemism for a piece of software that runs on the Blockchain. We tend to avoid the term simply because it brings with it certain expectations in the market, and our approach is to try and educate people as to why things maybe shouldn't be called smart contracts. There is a certain expectation that leads to people.
>> MARC-ARTHUR PIERRE LOUIS: Thank you. You can take [inaudible]
>> CHRIS FERRIS: One last question I saw here, why is Composer and Fabric more preferred? Again, these are, this is our current choice for delivering our go to market offerings, so the IBM Blockchain platform is currently based on Hyperledger Fabric and Composer, again, going forward, there is going to be increased emphasis on and interest in investment in Hyperledger Indy from an identity perspective. Hyperledger Cello is likely to become the defacto provisioning and orchestration monitoring platform for Blockchain platform. So Cello gets a lot more interest from us as well.
Again, from an Ethereum perspective we are actively looking to integrate the EVM as a library, plugable base code library. So that's basically where we are going with that. I'm at end of job with my presentation. Thanks again for joining in. Up next we are going to have a conversation about Java. So that is going to be led by Dan Bandera, who is in IBM's digital business group, he is a Java community leader and he has been representing IBM in the LSG alliance, on the Board of Directors and serving as its president since 1999. And BJ Hargrave, IBM senior technical staff member, and the digital business group's CTO for Java, and he also serves as CTO of the LSGI alliance and chair of its core platform expert group. Dan and BJ, I'll let you take it from here.
(voice in the background).
>> Thanks, Chris. Thanks for joining today. I'm Dan

Bandera. I'm going to be sharing the presentation with BJ Hargrave. You want to say hi, BJ?

>> BJ HARGRAVE: Hi, I'm BJ Hargrave. Thanks for joining us today. Hopefully you will learn interesting things about Java. Hopefully you are already Java users but if not, maybe we will convince you to think about it.

>> DAN BANDERA: Okay. Let's jump into it. BJ, if you bring up the first slide.

>> BJ HARGRAVE: I shall.

>> DAN BANDERA: We are both in the open technology area as Chris mentioned, at IBM, and work together extensively on Java for a couple of decades now actually. Next slide.

>> BJ HARGRAVE: There we go.

>> DAN BANDERA: Thanks. Today's agenda, you see the structure of this agenda is much like you are going to see in the prior presentation and in future open source presentations. Let's just go through the main points. What's Java, how is it created, where is the community, some technical overview. We have a few slides on that. What is the current status? What is next for the Java community at large? The significance of Java at IBM and how you can get started for call to action for you to join in.

I should mention that BJ and I will not be able to answer your chat questions if there are any right in realtime. But if there is time at the end, we would be glad to address those questions at that time. We will get to that at the end of the presentation. Next slide.

What is Java, write once, run anywhere. I'll go into detail on the next slide, but unlike many of our open source presentations that we are describing in these communities, Java did not start as an open source projects. It started as a proprietary offering from Sun microsystems. What is it? It's a very powerful object oriented programming language, and it runs on virtually everything. It runs on your smart phone, can run on a super computer, and all things in between.

What a lot of people are still surprised about is that it is the single largest community of programming, programmers and programming language in the world. It's number one. There are about 13 million developers that have the skills to program in Java.

As I said, it is extremely powerful and portable, but that is because it's based on virtual technology, virtual machine technology. This isolates it from the specifics of the CPU architecture or the specifics of a given operating system. It allows you to, as said in the subtitle, write once, run anywhere. It is also extremely fast and securely reliable. There are programs that are literally in the millions of lines of code for Java.

Next slide. How was Java created? It was actually created by a team led by James Gosling at Sun Microsystems back in 1995, that is when it was first released to the general public. Obviously they were working on it before that.

Sun Microsystems though realized that as a proprietary offering, they couldn't get the kind of reach and traction that they were looking for, so to expand their community of Java, of programmers adopting the Java language, they created the Java community processes, three years later in 1998, to provide a way of getting community input into the evolution of the Java language and the Java environment, libraries and so on.

This for the time was kind of new and different. It created a community where, nonexistent yet, still not quite the open source community that a lot of our technologies are today, but an expansion of the community as far as gathering input.

So then Sun didn't really introduce a true open source project, the open JDK project in 2006, and put forth the Java platform standard edition out under the public license version 2 with a class path exception that allows you to use Java in ways that the straight GPL wouldn't allow you to do.

Although that project was in 2006, the actual first release that you could get your hands on was in the following year in 2007.

As you see here, we got major participants, Oracle, IBM, Red Hat, Apple and SAP have all participated in open JDK.

Next slide, please. Where is this Java community? Well, as I said before, it's 13 million Java developers, have had over time a lot of interest in getting together. They get together in different ways. One is as I said the Java community process, where people can formally provide feedback into how the language evolved, and also the open source project, open JDK that I mentioned. You can follow these URLs to find out more information.

But it has expanded, Java has expanded over time into other venues and other projects. So open source foundations that we all know and love, the Apache Software Foundation has over 200 projects that use Java. The Eclipse Foundation also has over 200 projects that use Java. We will talk more about Eclipse in a later slide.

There is also a number of projects at the Linux Foundation, that are based on Java. Java is, if you look at the languages used on GitHub, it's in over 4.1 million GitHub repository, the main programming language of a given project. That is a enormous number of repositories, more than any other single language.

There are many conferences every year. The biggest one, being JavaOne, that is typically, well, it is of late been held in San Francisco, and last year had over, about 8,000 people, I should say. But there are other major Java conferences. The Eclipse con series, which includes Eclipse con Europe, Devoxx, which there are multiple but the original is in Belgium, is held in Belgium. Jaxx, J focus and Q con all having elements of Java. Next slide. Please.

Where is the Java community? It's all over the world. But before we had something called meetups, the Java community had things called the Java users groups. These were the meetups before there were meetups. There are at current count 202 of them. Some of the leading ones are the London java community which has over 5,000 members, and Sou Java which is headquartered in Sao Paolo, Brazil, which I believe is the very largest jug in the world with 40,000 numbers and then realizing the Java community, realizing that everyone can get to a meetup, not everybody can get to a face-to-face, created a virtual Jug which is a on-line Jug. You can follow that URL and you can find when their next meeting is.

Also the community is in adopt open JDK which that community is actually providing prebuilt Java binaries that are targeted at various operating systems and hardware architectures, and you can go to that URL and you can select with a menu which of those binaries you want to get for your architecture, your operating system.

Then another place that the community has coalesced around is the Oracle GlassFish open source project which is specifically for the Java enterprise edition, and it is the reference implementation for that platform. You can follow that URL to find it.

But I'll talk a little bit more about that later in the presentation. I'm going to turn this over to BJ now, if you want to pick up on the next slide.

>> BJ HARGRAVE: Sure, thanks, Dan. Let's get into technology that is Java. I hope most all of you heard of Java, maybe you have used Java, maybe you haven't. But one of the things that makes Java important as a language is the virtual machine technology. It's one of the promises that Java makes about being able to write once, run anywhere.

That is done through the use of virtual machines, Java has a abstract machine design, that you compile against, and that can be interpreted or jitted. Over the 20 plus years that Java existed, significant person years of engineering have gone into the virtual machine. We see the hot spot machine that comes from open JDK, and the newly open source J9 virtual machine from IBM but significant work in improving Java performance over the years.

As a virtual machine, it's a managed runtime. The memory is garbage collected. It cures most of the memory issues that plague programming in native languages. There are other layers of the virtual machine that let it connect to various operating systems and process architectures.

Above the virtual machine is the Java runtime environment. These are set of well-known class libraries that are available to developers to use. So you have the libraries that do a different things, there are libraries suitable for user interface design. There are libraries supporting collections, maps, other common needs of programmers. The lower level lets you do networking, URLs, http connections, I/O, reading files and other things.

In order to use all of this stuff at runtime, you need the development kit. There is the JDK which includes the language, the tools to invoke the language, Java to run the runtime, Java C to compile your code into the runtime format cross files as well as tools, documentation and a myriad of other things you need.

Most people if they use Java today, they get Java SE, standard edition which is what this chart talks about. It's Java standard edition is the thing most people download from Oracle or you can get it from JDK. Today Java SC 8 is the most popular current version available. This includes the recent language feature editions of lambdas and streams, which were a big advance in general usability for Java programmers, reduced a lot of boilerplate code in programming. Java SE 9 was recently released. This includes some significant changes, including the Java platform module system.

But that has been a challenge for a number of people, people to begin to use in earnest, mostly because it has a lot of ripple effects through all the existing libraries that exist in open source. There will be some challenges there. The other issue that will probably slow down adoption is that starting after Java 8, not every Java release has long term support. Not until we get to Java 11 will we have another long-term support. But Oracle is planning on releasing Java in six month cycles.

So we should expect Java 10 this March, and Java 11 this September, which will be the next release after 8 with long term support.

Java SE is basically the most common platform people think of when they think of Java. There is enterprise space, there is other things we need to do to build real applications to solve real problems for our company.

There is another layer on top of Java SE which is called Java enterprise edition, Java EE. Java EE adds on a number of libraries and many different areas which are necessary to build enterprise grade applications. It's effectively a super set of Java SE. There are a number of technologies that come in Java EE. There is things such as the context independency injection CD I which is probably the most prominent dependency injection model for Java programming, a lot of other Java EE technologies that use CD Is, their mechanism, there is support in here for microservice programming through Jaxx RS which is the rest services API. There are a number of libraries around JSON processing, JSON P for parsing, JSON B for binding. Many of these APIs are available in a variety of application servers that you can use to build enterprise applications for Java. IBM has made open liberty available, which includes all of, support for all these APIs and a number of other companies in the industry that has their J box server, GlassFish server and also commercial offerings as well.

This is the main APIs you are going to be using if you are running enterprise applications that sit on top of the standard edition of Java.

What is the current status of Java out there? As mentioned, SE 9 is the currently released version, that includes the headline feature of the Java platform module system. If you haven't heard of that, it's basically a means to modularize the Java runtime itself, in the previous slides we saw many different libraries, packages that support certain features. Prior to 9 these all existed in the big monolith. Java SE 9 with JPMS begins to allow the panel to modularize. They have challenges as they try to encapsulate implementation detail away from the user of Java, some APIs are going away, and some have to be replaced with other standard APIs. This is a challenge in the overall Java ecosystem given there are a huge number of existing Java libraries out in the wild but depend on some APIs that weren't necessarily open but were available prior to Java 9. It is going to be a ongoing challenge for the Java ecosystem to begin to take advantage of and work with the new modularity support for Java 9 brands.

But in addition to Java itself, SE, EE, there are many other active projects and foundations that are doing things with Java. One of the other interesting things going on is Eclipse microprofile which is taking a more focused view for microservices programming.

Sometimes you don't need all of the parts of a Java enterprise edition application server. You just need the main key pieces to do microservices programming. This will give you a much leaner faster image for deploying in a containerized environment.

Microprofile is making some important strides in that space. As I mentioned open J9 is another project. IBM has had a J9 virtual machine for many years as part of the IBM SDK for Java. We recently began open sourcing it, the first part was Eclipse LMR which are some of the building block tools for making a virtual machine. Jitting connections, garbage collection pieces, and then on top of that open J9 is a complete Java virtual machine. That paired with class libraries open JDK can be found at the open JDK project, you can get a fully open source implementation of Java SE now. IBM recently open sourced its liberty work. So we have the IBM open liberty project. This is a complete Java enterprise server offering. It supports the full Java EE as well as the Eclipse microprofile supports. It's basically designed to be fit for purpose, so it uses only the particular features you need to deploy your application.

It's quite the lean and mean. We talked about the open JDK project which is where Oracle is developing Java SE, and Oracle GlassFish is their implementation, their reference implementation of Java EE.

One of the other interesting things that happened in the community is Oracle is now in the process of moving the stewardship of the Java enterprise edition specifications and runtimes over to the Eclipse Foundation, so it's more open and it's going to be better for the Java community as a whole.

Then the last piece here is the Java community process which was the original work done by Sun and later Oracle as the steward for the Java specifications.

Next for the community is basically beginning to understand how to take advantage of the later versions of Java SE, and to work with the new cadence, the release cadence that Oracle came up with, 6 month release cycles. We will be getting a new Java version every March and September. There is a lot of work to go on. The movement of the enterprise edition work there, that is certainly a place that you need people to get involved from a Java community, and parallel to that is continuing the Eclipse microprofile work.

We had release 1.3 and it has a number of focus specifications around microservices based programming using Java. Of course many conferences, you can always come and attend one of the key Java conferences, learn about what is going on in the community, talk about what you are doing in the community.

That's sort of what the next thing is. Let me give it back to Dan. He can tell you about what we are doing lately with Java at IBM.

>> DAN BANDERA: Thanks, BJ. I've got a few slides here on Java at IBM.

First, IBM has Java in the Cloud, as you expect from any major Cloud vendor provider. We have a couple of different ways, couple different examples that I've got URLs here for, which show you Java applications in the Cloud.

Also, Java at IBM, one thing that's very important is if you remember early in the presentation, I said Sun released Java in 1995. IBM has been contributing since 1997, even before the JCP was set up in 1998.

We are a very early adopter of the technology and supporter of it. As BJ mentioned a few projects, we are also founding member of the major open source projects related to Java technology. J9, open J9, OMR and microprofile. The open liberty project is really, if you want to see an amazingly large piece of code, it's over 3 million lines of code on GitHub. It is actually modularized with OCI technology and one of the biggest examples out there of a well written OSGI Java application. It is as BJ said a full Java enterprise edition.

We also of course work in other areas, we are significant contributor to many Apache projects that use Java. And Sam Ruby, the current president of AFS is a IBMer.

Next slide, please. How significant is the Java at IBM? Well, we have over 3,000 offerings that embed or bundle Java. And over 200 IBM products embed traditional WebSphere application server and another 200 that incorporate the liberty technology, which is a reimplementation, as I said, and open liberty is a subset of the Liberty offering, the commercial Liberty offering.

Java and WebSphere, WebSphere being our brand name for our Java enterprise edition, are key to many of our offerings in the hybrid Cloud. Watson embeds Java and Liberty at its core. There are several other offerings listed here, care manager, explorer, marketing insights, analytics, all of these and more.

Many Cloud services as well, mobile first platform, application security manager, mobile first foundation, voice gateway, these are examples of all the places we are using WebSphere or Liberty technology.

Next slide, please. So, BJ talked about the SE platform earlier. But just to give you a graphic of how those things all go together, is the OMR project at Eclipse as BJ had mentioned has some virtual machine technology that are applicable to multiple different languages.

As you can see as you go around the clock, around the OMR, you can see the different languages that you could use those core technologies in. That then goes into open J9, a separate project that wrappers all the Java specific portions to that VM. And then when you combine that with open JDK's class libraries, which are available on adopt open JDK site, then you end up with a full offering. As we move to the right, you see that IBM also adds some value in our own SDK, and these are very specific editions that are needed by our customers for security and hardware optimizations for IBM platforms like our zSeries and Power.

Also, we have support tools that are available in our commercial offering.

Next slide, please. How do we get started? There is a number of different ways for you to get started in this Java community, if you are not already there. Follow the blogs. Join mailing lists. We have a number of URLs here that point you at places that you can go. Contribute to a projects. We have mentioned several that are important, open J9, microprofile, OMR, open liberty. Get on board. Also you can explore at IBM code, IBM code patterns for Java. The general category is under that first top bullet, but also provided a couple of examples here that I think are good ones to start with.

Next slide, please. So, please get involved. Join a project. Go to users group. Go to a conference. BJ and I are at many of these conferences. We hope we see you there. So thank you for your time. We really appreciate your involvement. And if we have a minute, take a look at the chat, or we will take questions.

>> There might be a couple questions.

>> There is a question around test IT and [inaudible] emulator.

>> BJ, you want to take that?

>> BJ HARGRAVE: Yeah. I'm looking at the question. I'm not really sure, are you talking about integration test when you mean IT? Like any environment, you can construct testing scenarios, in the Java programming world, Maven is among the more popular tools for developing Java based applications as a build and testing tool.

Maven can be used as a means to build your code and drive unit test as well as integration tests via plug-ins that are available. There is others as well, a rich variety of tools available in the Java ecosystem for development.

>> MARC-ARTHUR PIERRE LOUIS: BJ, I have a quick question for you. We see that open source is changing the landscape by getting ideas. One great idea that I find myself is the ability to try and [inaudible] if you go to Python or JavaScript, you have the ability to test those functions without writing a program. This is coming into Java, I hear, is that true?

>> BJ HARGRAVE: Yeah. I mean again, Java has been around a long time. It has been mentioned, we, Dan and I have been at this for over two decades. So there is a massive amounts of free and open source software that is in Java. You can design the most impressive systems out of software that is available

for free. As we mentioned, this past year, you can get a complete open source Java runtime environment and application server on top of it. So you basically, what you had to pay for years ago is now available for free.

You can clearly try before you buy, or when you are ready and you want support, there are numerous companies including IBM that are ready to assist you and provide services and support.

>> MARC-ARTHUR PIERRE LOUIS: Yeah, couple more. At the console, in JavaScript, you can see those functions and invoke them without writing a full program.

>> BJ HARGRAVE: Right.

>> MARC-ARTHUR PIERRE LOUIS: [inaudible] Java today?

>> BJ HARGRAVE: Java 9 introduced ripple (overlapping speakers) read evaluate, I can't remember what it is, but basically yes, a new feature added to Java 9 so you can fire up the Repel in Java 9 and play with it, type in lines and explore the results of the variables and method calls, and tool around a bit without having to actually go compile code in a formal structure.

>> DAN BANDERA: That will be available going forward. One thing we didn't mention is that backwards compatibility in Java is extremely important concept, so as we go forward that Repel will be available in the future as well.

>> MARC-ARTHUR PIERRE LOUIS: Great, thanks.

>> CHRIS FERRIS: Thanks, Dan and BJ. Great job. That concludes our introduction to Java and the work that IBM and others are doing in Java and open source.

Thanks, everyone for coming. In a couple of seconds we will kick off the next talk on Open API.

>> KATHY GHANEEI: Jeff, can you share your slides?

>> Welcome back, everyone. I'm Chris Ferris, IBM's DE and CTO for open technology. This is the fourth installment of almost a week's worth of open source at IBM, and I'm happy to introduce to you Jeff Borek. Jeff has been active in a number of the open communities, strategic to IBM, including the Open Container Initiative, Cloud native compute foundation, he is on the board at the Open API initiative and also leads the IBM team that manages our open source clearance process, and Erin McKean, developer advocate in San Francisco for IBM, and she is the founder of Wordnik and has been a supporter of Swagger and Open API initiative since day one. They are going to tell you about getting involved with the Open API initiative. Jeff and Erin, take it off.

>> JEFF BOREK: Great. Thanks, Chris and thanks for the kind introduction. I'm very pleased to be here today with you and with Erin to give you [inaudible] Open API initiative, and as we start to move through the set of slides, we will take a look at

today's agenda.

We are going to follow a similar format with a few differences, and I'll take the first half and then we will transition over to Erin to cover the second half of today's content.

But just to set up today's dialogue, if you think about it, doing APIs is something that's been a hot topic around IT ecosystems, even predating open source, but certainly in the last decade, it's becoming increasingly apparent that doing open APIs and doing them properly can be challenging. It's hard from both consumer perspective, because if you are trying to work with a program or interface to an application, you have got to read the documentation and hope the documentation is good. That is a bare minimum.

Ideally there is a decent SDK and I'll put air quotations around decent. Sometimes that can be hit-or-miss. How do you figure out what the important parameters are? What about the payload, and even doing it from a provider perspective is challenging, because you have spent the time creating the code, writing the application, and now you have got to make sure that you have got to provide good documentation or you have got to try and do the right thing from an SDK perspective. And let's face it. Supporting end users can be hard, and even just doing what some folks think of as the boilerplate code can be challenging as well.

So with that as a setting for the -- pardon me one second. As that for a setting for this, the Open API initiative was founded in November of 2015. I was happy to be there representing IBM as part of the very early discussions. This happened down in California, back actually in the first months of January of that year. I'll reiterate that when it comes to doing these Open APIs, people have tried to solve this problem in the past, and it's been very challenging. A lot of times these types of issues where you try to get large parts of the ecosystem together to take on standardizing APIs, past efforts to do this, corbel and others to some degree met some level of success, but they also essentially, many of them collapse under their own weight, because they try to be all things to all people, and even that Internet itself can be difficult.

With this Open API initiative, what we were able to do was focus on a very successful open source project, and let me transition to talk about that. You may know it from the name Swagger. So it was started by Tony Tam back in 2010. It was something that Tony did to, like to say open source community, scratched his own itch. He was working on a start-up and having to repeatedly provide connectivity to other third party partners, and he found that effort to be very burdensome and repetitive and frankly boring.

So he tried to come up with a way to do this with REST APIs in a consistent way, that also gave it structure and gave it base documentation capabilities.

That is when I actually happened to meet with Tony the first time, also SmartBear and folks from apigee, IBM and others started to come together to take a look at this and provide some guidance and direction. And Tony, as the project founder, did good due diligence. He talked to a number of different foundations and options and ultimately what they felt comfortable doing and what the community came together to initially support was the announcement of the sharing of the open API specification. The specification was donated by Tony and SmartBear but the code itself and the Swagger brand remain under SmartBear control.

This initiative took off organically. It gained traction with a number of companies. You can see the link to the main GitHub repo. There is the link to the specification itself. But most importantly, I think, is the indication of community adoption. You can see from the graphic from stack overflow, far and away compared to some of the other options, Swagger has just enjoyed tremendous amount of adoption as well as adoption from major companies and entities around the globe.

One of the things that is compelling about it is its independent of language framework deployment technology and it also supports both an API first approach as well as a code first approach to defining, building and documenting APIs.

A brief update from 2017. A major event, the first release 3 of the Open API specification, done under the guidance of the Linux Foundation and the working group. They have released the specification. As you see from the graphic on the right of the screen a lot was done to simplify and consolidate the structure and make the framework a lot more consumable.

A number of other additional technical innovations have been added to this spec, and folks are now in the process of adopting and coding to that specification. I know IBM is certainly one of the leading companies doing that. With that, I'm going to transition it over to my colleague Erin to take you through the second half of today's presentation.

>> ERIN McKEAN: The current structure of the OAI is straightforward. There is basically three groups that handle the governance. There is the business governance board which handles trademarks and certification and budget for things that the OAI wants to do, to both expand and promote the use of the spec. What conferences can OAI go to, what kind of things can we do with the website. There is Technical Oversight Board which is the tiebreaker board, for managing conflicts when

people have different ideas of the direction [inaudible] handling any violations of procedures or guidelines, and any issues that can't be resolved in the TDC.

[inaudible] technical development community manages the Open API specification development. That is open to everybody. There is a weekly call. It's a fairly long call but it goes quickly, and if you are interested at all in the inner workings, I highly recommend sitting on one of the calls. They are open to everyone. It's easy to get started. The people on the calls are a fun and generous group of technologists who are always welcoming [inaudible]

Next slide. (overlapping speakers) in the community, e-bay has joined which is great. More people involved in the spec is always helpful. A little bit of sad news, in Tony Tam left SmartBear to pursue new interests. [inaudible] he left behind some really capable leaders, especially Ron Ratovsky, who also worked on Open API stuff. He has institutional memory and is generous about sharing it.

Next slide. There is lots of things that IBM is doing, both with and about the spec. Probably the easiest one to get started with a loop back which is a open source framework that lets you create APIs quickly with node. You can both generate a spec from your LoopBack application and you can import a spec to generate a LoopBack application. There is lots of content around those particular techniques for getting a API running quickly. Then of course API designers and API connect both leverage using the spec. There are links there to tell you more about that.

It is very easy to get started to get involved with OAI community. They are a welcoming group of folks. There are people involved in the community all over the world. You can start just by sitting at home in your pajamas reading a spec, it's up on GitHub. It's straightforward. The Open API spec is active to Twitter. There is a IRC channel. There is more documentation apart from the actual spec up on GitHub and there is a active mailing list.

That last link is easy to get started. Next one. So, if you haven't beat the get involved drum loud enough or long enough yet, here are more ways to get involved with the spec. There are meetups. The meetup in Boston where SmartBear is based is active. Also San Francisco bay area. There is a conference, API, strat will be in the fall this year. I can't tell you where it is going to be yet, but it will be somewhere fun. If you are interested in the technical side of API, business side, outreach side of API, it's a conference you would like to be at. It's a small conference. It is not overwhelming. There are lots of opportunities to make connections with other people who

are excited about APIs.
That is where you can find the cutting edge information on APIs, and yes, there were presentations about [inaudible] it's not just RESTful APIs. I highly recommend following news about API strat. Also if you are interested in becoming a member of the Open API initiative, membership is on the sliding scale. If you think, my company is tiny, we can't be involved in OAI, that is not true.

There are people, there are members who are just, who are companies with a few people, and of course there are members such as IBM which has hundreds of thousands of people involved. Don't let that stop you from getting more involved with the Open API.

We do have some time to answer questions. I see that there is a question in the chat about where the OAI community is. So I can just answer that quickly. You can find pretty much all the information you ever wanted to know at Open API.org, probably even more than you wanted to know.

So, as Chris said when he introduced us, I have been a proponent of the Open API specification since day one. Tony was my cofounder at Wordnik, and I joke that I asked so many questions about API parameters that Tony [inaudible] and a developer advocate and software engineer, there is no kind of interaction where your API, either at a technical level or business level or outreach level that isn't improved by having an Open API specification. It makes APIs easier to develop with. It makes APIs easier to sell to [inaudible] might want to use your API. It makes APIs easier to manage.

I really recommend that people pick up the spec, and it's not a dead document. There are discussions going on right now about how to expand the spec for the next point releases to make it more extensible, to make it more modular, to make it easier to use, and making a recommendation for a change in the spec is as easy as opening a pull request on GitHub. You do not have to plug hours of your life pitching it. Make a request and the technical community will take a look.

>> JEFF BOREK: Now is a good time to make that beep sound.

>> ERIN McKEAN: Next slide, Q&A.

>> JEFF BOREK: You have a great history with the project. Can you share with the audience how Tony came up with [inaudible]

>> ERIN McKEAN: It was another developer, front end developer actually who said that we should call it Swagger because why waddle, which is a competing spec, when you can Swagger. And it stuck.

>> JEFF BOREK: Interesting case study, if you think about it, Chris earlier in our two-hour session talked about the concept

of BDFL or benevolent dictator for life. Tony was certainly a great example of a positive BDFL for the early stage of the project, because I alluded to it early on but many attempts to solve this problem in the past ended up getting crushed under their own weight as everyone threw everything into it from the kitchen sink to the latest technology. One of the things Tony was effective in doing is try and strike the right balance of keeping the project focused and keeping the scope somewhat narrow, so that it really served the needs of the general developer that was looking for a good basic tool to help them do APIs and document APIs. (overlapping speakers).

>> CHRIS FERRIS: Community behind it, keeping things going, even though he stepped aside.

>> ERIN McKEAN: There are thousands use cases for APIs and a lot of them are very narrow. I think that the PBC has been good about keeping specs from rabbit holing on one of the use cases. We have been talking lately about security and APIs, and I believe that they are setting up a separate working group to look into that. It may not be part of the spec. It may be a parallel spec that will work interoperably. But I think it is focused on keeping the Open API specification something that provides the greatest good for the greatest number, and that keeps the interest in general developer in the forefront.

>> CHRIS FERRIS: Thanks, guys. Thanks, Jeff and Erin, great talk.

So this concludes today's session on open source at IBM and Open Source Week, and we are going to pick it up again at 2:00 p.m. Eastern tomorrow. And Todd Moore, my partner in crime here is going to keep us informed on what is going on in container land, and we will hear about the Cloud native compute foundation, about Kubernetes, Docker and the Open Container Initiative.

So come on back tomorrow. I'm sure it will be some great stuff again. Thank you, everyone. We will see you again soon.

(end of call at 1:53 p.m. CST)