Tyler Kern (00:03):
Welcome to the Sunrise Podcast, powered by Sunrise Labs.

Tyler Kern (00:12):
Hello and welcome to Making Bright Ideas Work, a podcast from Sunrise Labs. I'm your host today, Tyler Kern. And today we're going to be answering four very important questions about continuous integration and automated testing. We're going to answer the what, the why, the when, and the how. And so joining me today are our two subject matter experts. They're going to answer these questions for us and provide a lot more information and insight. First, we have Jim Turner, he's the director of software engineering at Sunrise Labs. Jim,, thank you so much for being here.

Jim Turner (00:40):
Thank you, Tyler. Really happy to be here.

Tyler Kern (00:43):
Absolutely. We are happy to have you, Jim, and I'm looking forward to getting to talk further here on the podcast today. And also joining us today is Mike Goulet. He is a program manager and principal software engineer at Sunrise Labs. Mike, thank you for being here as well.

Mike Goulet (00:57):
Great. Thanks, Tyler.

Tyler Kern (00:58):
Well, Mike and Jim, we are thrilled to have you on the podcast today, and just to get things started, we're going to answer that what question? Give us the details on what continuous integration and automated testing is.

Jim Turner (01:09):
I'll start with that answer. Continuous integration, automated testing, the most basic form for continuous integration is actually doing repeated builds on your local machine. And there's this continuous spectrum of continuation to automated testing. Automated testing is actually using software to test software. And that spectrum begins with just a build in your machine, being able to run some type of test, like a static analysis tool, all the way to having all of the software completely tested using unit tests, integration tests, system software tests, and actually system tests. There's this dynamic range of being able to build. And we can go into more details of that a little bit later on. Mike, do you have anything you want to add to that?

Mike Goulet (01:59):
Yeah, I think that kind of hits the nail, Jim. In its most basic form, integration is if you're a programmer is hitting the compile button. You have successfully integrated a bunch of lines of code into a program and the key to continuous integration and there's a lot of benefits to doing this, is doing that frequently and in a way that has lots of visibility to the team and to the people that consume the software. I think what is continuous integration is obviously like you said, on a spectrum, but at its core, it's building a software product in an automated fashion in sort of a central location.

Jim Turner (02:53):
And I'm going to add to that, which is the shared repository as well as shared targets. And what I mean by shared repository is being able to use the same revision control system to get other people's work and bring it into your work on a continuous basis. Having 10, 20 people working on one product, the number of changes that can happen in an hour are enormous. If you can build at a frequent pace, not just once a day, with everybody's changes at the same time, you're going to be able to bring in all those other changes, and simply by compiling with them and running your code with them, you're going to find bugs sooner than if you had to integrate at a much lower frequency.

Jim Turner (03:43):
And then adding the shared test environments or test targets, you're going to target up to four, five different areas. You have the area of where personal testing is happening for an individual software. Then you might have a branch within your continuous integration, which instead of just running the software on your laptop or on your individual workstation, it may be running on a centralized location that integrates a certain feature. And then you have all the way up to production code where before you run it on production code, you're actually going to run it on a simulated production environment so you can actually see and ensure that your code is working in a production environment before you actually deploy it. And so that's a continuous integration and a much bigger view of where you would like to get. And each one depends on the product and in our case, the customer that we want to work with.

Tyler Kern (04:53):
Well, and Jim, as you were explaining some of the benefits there and Mike as well, one of the things that started to become apparent were some of the benefits. That takes us into the why? Why is continuous integration and automated testing important when it comes to medical device software development? Talk us through some of those benefits and why this is an important thing to discuss.

Mike Goulet (05:15):
I'll jump in on that one. I'm glad you brought up the medical device part of it. One benefit and one real challenge in medical device development is, and anybody who's been in this industry will know this, is controlling the tools that are used to develop the software. On a given project, one of the projects I was on recently, I have one in mind. And if he were on this call, he would be laughing with me. I think there must be 20, 30, 40, maybe 50, if you add it all up, so many tools involved in that build of software between compilers, handwritten tools, off-the-shelf products, repositories, testing tools, you name it. And when you've got 20 developers and you say, "This is how you're supposed to develop the software and these are the tools and the configuration you're supposed to use," it's really, really important that you do it in a very specific way because we have to control the inputs and the outputs, especially in a medical device regulated industry.

Mike Goulet (06:29):
And so one of the almost obvious, but so subtle and important things about CI in medical is that those tools are located in one place. And so it's easy to go to the central continuous integration environment, configure those tools, write it down, it's this exact version of this exact compiler. We obtained it from this exact location. Control all those inputs to the build process. And then it becomes far less important that the 20 or 30 other engineers on the team have the exact same tools or the exact same configuration of those tools. And then when you go to submit your product to and submit your claims and your documentation to FDA, it's really easy to explain how you built that software.

Jim Turner (07:29):
Thanks, Mike. If I could jump in, one of the things that I'd like to expand upon in the why, and I briefly discussed it in the what is the simply compiling your code continuously. It reduces the number of bugs. The other thing you have with the ability to have a continuous deployment with the automated testing is that too will reduce the number of bugs. And what I mean by that is if you have SQA and or QA within your organization, being able to take the output of one of your builds in an integration area, they can get a headstart with their test procedures and their protocols and their test plans and make sure that the code that we are developing, matches what the expectations are with the requirements. I briefly spoke before about the testing being at the unit level, being at the integration level, the system software level, these are areas within IEC 62304.

Jim Turner (08:27):
And you can see that if I have this continuous delivery happening, that the unit test occurs and they're

testing each individual module and the integration testing is occurring automatically because we created these automated test vectors in order to test it. That may or may not find all the bugs, but the next level up is actually to have real individual users using the software. And with this continuous delivery, what it enables is those individual users to be able to test at the system level. And so we're building our confidence up as we're building the software so it's the continuous integration, automated testing with this continuous delivery.

Tyler Kern (09:22):
Do you guys have any examples you could share of a time when you didn't use continuous integration and automated testing and what was the outcome of that example?

Jim Turner (09:34):
Go ahead, Mike.

Mike Goulet (09:36):
Yeah, I definitely have examples. We worked on a project, not that long ago, and we chose for at the beginning in order to try and save on costs, to not put the engineering time to build up that continuous integration system. And it was a learning experience for me where I knew that in my experience, in my career, automation is almost always worth it. And in this case, we didn't do it. And it was a very complicated system with an operating system that had to be configured in a certain and very specific way and we paid for that. We had to write work instructions to explain in very detailed stuff. Now we would have had to write those work instructions anyway, but the work instructions were very manual, configuring the operating system, setting up the software on it, configuring the drivers, making an image, and then building the program, deploying it.

Mike Goulet (10:53):
There were so many steps. And in the end, before we were done, we were down to basically kind of a half a day to put together a release. And it probably was even more than that. And we had multiple releases and what it did was two things. First of all, it made our releases more expensive for us, for our client. It was just a lot more time-consuming to implement them. And the other thing was that it made us afraid to do releases. We didn't want to do a release because we knew how much effort it was going to be. And it caused us to go longer with the bugs in the current release. We just didn't want to get the fixes in because of the cost of getting a release out. And so in that specific case, that was sort of, after that, I've sort of come to the conclusion that I'm going to do a continuous integration. It's almost a necessity at this point to have that automated building and testing.

Jim Turner (12:01):
Tyler, I have a couple of examples of how we haven't done continuous integration and it's simply caused issues around versions of tools. And you can easily see that if you have one developer has one version of a tool and they pulled it down from the internet and somebody else has gotten a different version of the tool, that there could be some subtle bugs between the difference of the two developers. And that in its most simplistic form is what you're trying to avoid, which is essentially bugs introduced by different versions. It also makes it difficult for your developer, your testers, if they don't have the same version. I've seen all of those. But one of the things I'd like to do is speak a little bit about GUIs in which continuous automation is not always or automated testing is not always the best way to go.

Jim Turner (13:01):
I worked on a project that had 400 screens and the reality is manually testing it was better because we actually developed what the test cases were. And the other thing that occurred is as we were developing and people were finding bugs, there were changes as well as expectations for what the requirements were. There were changes while we were developing. There are advantages not to doing automated testing right away. A human-looking at a screen can actually find pixels that are off,

very, very quickly. The best software that would compare to images on the screen, may or may not find that subtle difference. And so the eye is very good at doing that. We haven't quite got based on cost, gotten to a low-cost way to determine how to get errors out of small issues within the screen layout. There are times when you manually do the testing and then you add in the automated testing.

Jim Turner (14:09):
Once you've gotten confirmation that this is the screens and these are the requirements you want and you get that buy-in from the client, then you cannot automated testing. I want to add one other thing as to why you do this. For any product that's going to be in the field long term, testing is expensive and if you can automate it and let's say this product, a lot of medical products are out in the field for five years. The first year, let's say the first six months, it's probably cheaper to manually test it, but if it's going to be around for five years and you need three people to get through that test, you can see that those costs go up substantially once you add time to the factor of do you do automated testing or not? I hope that expands on the why of doing it.

Mike Goulet (15:03):
Yeah. And I'll jump in a little bit on that. Jim, I think you're right. I think we've had clients where it would be too expensive to automate testing a GUI, especially if that GUI maybe is an alpha version GUI and it's probably going to change in the next six months or a year. It takes a lot of effort to automate testing a GUI. And we do have experience doing that. But to your point and so then it's better. It may be better in these instances to do the testing manually, have a protocol, follow that protocol, it's click the button, expect the results. And a lot of people do that because it just is easier.

Mike Goulet (15:44):
But then down the road, I think this is what you were saying, down the road, if that test is going to get run again and then you're going to do V2 and V3 and verification, it's an ordeal, especially on a medical device. it's time and it's a costly exercise to go through. And so if you can automate tests and demonstrate that the software is correct, it can be a huge cost savings and we definitely have done a lot of that kind of automation at Sunrise.

Jim Turner (16:18):
Yeah. And where the V2, V3, V4 regression testing is very important, especially if you are going to re-architect a certain area of the software, then having those regression tests will allow you to see that you have not corrupted another piece of the software. The architect should be set up that you don't do that anyway but we work on projects where we have not done the complete architecture and so there are times when we will find that segregation of certain areas of the code are not as clean as we would like it to be.

Jim Turner (16:56):
I do want to go back to what Mike was saying and just to reiterate that simply having automated unit tests and some integration tests as low-level regression testing that you can use as you build up the product, are often useful. Even if you can't get the full GUI to be tested.

Mike Goulet (17:19):
Jim, what you were just saying, it prompted me to think about something. One of the benefits of doing the unit testing in this continuous integration environment is we are trying at Sunrise more to do what we call test-driven development, where I write a test while I'm writing that item of software, that module, that unit. And if we have a good continuous integration environment with good testing frameworks, such as VectorCAST cast or Google Test or Unity or Seedling or whatever, I can add my tests, check them in with the code and show my development team, my peers, that not look at my code, it looks good. But look, you can see it actually works. And then you were talking about regression testing in V2 and V3, I commit those tests and they're memorialized. They're always going

to be run till the end of the project. And so if somebody comes along and breaks that line of code that I wrote two years from now, they'll know about it.

Jim Turner (18:27):
Absolutely. And while you were talking, Mike, one of the things that I was thinking about is the way we develop software, creating stories for the new features that we're adding and adding our definition of done or in this case, we also talk about it as our objective criteria for completion. This is it enhances what we're talking about here, because if you have the integration tests that we can document and we automate any of them, then you also have a really good story recorded within those Jira stories of demonstrating the completion of that individual story, which also adds to the integration piece. And our method, what I'm really getting at is our methodology follows our guidelines for, if we have a story for doing test-driven development, then it will enhance the testing that we're doing and the development of the code.

Tyler Kern (19:33):
Absolutely. I think we're ready to move onto when is the appropriate time to utilize continuous integration and automated testing? How would you guys answer that when question?

Mike Goulet (19:42):
Now.

Jim Turner (19:43):
Right away. Go ahead, Mike.

Mike Goulet (19:46):
Now is the time. Do it early. On day one.

Jim Turner (19:49):
I said right away.

Mike Goulet (19:51):
The first thing you should do is get your DevOps person and have them start building the infrastructure or using the infrastructure that you've already built on the last project. And that's the short answer, Tyler.

Jim Turner (20:06):
The longer answer is, as soon as you start choosing your tools, you're going to want to put them in a place that is common and especially to be able to add additional developers so that they can pull those tools down to their machine laptops and be able to work in the same version of software that you're working on. That's the first thing, find a common location to store your tools. And then set up, as Mike said, DevOps, your revision control system so as soon as you're done writing that first line of code or that first line of test, you can check that in, and other people can pull that down onto their machines and start integrating as they develop. Mike, do you want to add more?

Mike Goulet (20:54):
No, I think that's pretty much it. Just to reiterate, at the beginning of a project, I try to get that kind of stuff set up and sort of one of my things as a project manager is I really push the team to get through the hurdles of the tools, get the tools running and it really slows you down at the beginning, but boy is it worth it later on once you get over it in the beginning. It's get it done early and you will be glad you did it later on.

Jim Turner (21:32):
And one of the things I've seen Mike do using the agile development process is one of his first stories

that he creates when he's doing backlog refinement is to have a story of, get the build going, get the tools loaded, expand on the get the list. All about putting together the infrastructure so everybody is aware and has the same project structure so when you add a new team member, it's not as hard as that first team member that you bring on. That's very, very important as well as to when.

Jim Turner (22:16):
The other piece of this, which I want to add to the when is thinking about testing. Mike and I have been talking about this a lot and bringing on software QA, as soon as possible is critical. They need to know what the requirements are upfront. They need to know what your design is and they need to know where they can get the latest version of code that's even if it's a preliminary that they can start using to formulate their testing and you can define in the software verification and validation plan, what your plan is going to be for unit testing, for integration testing. And it's so as you plan it out and bring in the software QA, they're on the same page as the rest of the team and they're helping us by pointing out deficiencies in our requirements where there can be greater clarity to help not only the developer but also them writing their test cases. If you think about the entire development process with continuous integration and automated testing, you bring in the quality piece of this early on.

Jim Turner (23:28):
I'm going to also add that you bring in your client early on. I am working with a client right now in which we defined units differently and it may sound well why is that so hard? Well, because different software engineers have different ideas about what a unit is. Get those definitions down early, which will help with continuous integration and automated testing.

Tyler Kern (23:56):
I really appreciated the emphatic answer right there off the bat on that question. That was fantastic. I think you guys made it quite clear, the answer to the when question. Now let's answer the final question, which is how. How is this deployed? And what are the tools and methods that you use to accomplish this?

Mike Goulet (24:15):
A lot of the how, so specific tools. Sunrise, we have ways that we do things on projects, where we run them. We also work with a lot of clients that do things using their systems and we either take part in their systems or consult on how to build their systems. But the how, so one of the key elements of CI these days is in your source code repository, your git repository typically. You will set it up so that when the build, when a commit is made to a branch or when what's called a pull request is merged to a branch, the git, that's the place where the source code is stored, that repository can trigger an external system to build.

Mike Goulet (25:10):
And so what we've done is we've configured our git repository, we use Atlassian tools, and we add triggers to those builds to cause a program, a server called Jenkins, that's the technology Sunrise would use internally on our projects if we were building it. And our Jenkins server which we have, we probably would create a new instance for each client of ours, would automatically kick-off and run the build. And then from there, the how, the build, it depends on the project. If it's a C project, we'll be compiling using whatever the C compiler is or if it's Python, we'd be assembling the Python project into a final executable or Linux, we might be building the OS using the Yocto BitBake tools. It really depends on the project, the technology that we're using.

Mike Goulet (26:15):
In other places, we've had experience using other tools. We've used CircleCI, that's a cloud service as well, to implement continuous integration. That one's pretty cool. And then we've had some experience with TeamCity. A lot of folks are using that. That's a JetBrains tool and it's similar to Jenkins. It listens to trigger events off of the git repository and then it runs the build and we're getting

more and more into using artifact managers. And so tools like Artifactory or Nexus. These are tools that can collect the inputs to a build such as an apt repository for Linux packages or a pip repository for Python packages. And we do the build and then we would output the product of that build back into that same artifact management tool so that the QA teams can go get them from that repository or that artifact management tool. That's sort of a lot of the what of how or I guess how we would do it. And Jim, by all means, jump in there.

Jim Turner (27:42):
Let me add a couple of things about the medical device space, where we have to keep track of packages. And so when you download a Linux system or you download a node, you have hundreds, if not thousands of packages that you have to manage because in the FDA world, you can't or I shouldn't say the FDA, in a regulated environment, you have to know what you're putting into your product. You can't assume that the latest on the internet is going to be compatible with the work that you're doing. And even worse, they will change things and your code will no longer work. And so your delivery time now just got delayed. By controlling this and then ultimately what we do is we do a validation of that software to make sure that it meets our intended needs, we are controlling the software, the inputs, and then the output so we can rebuild that software for infinitum, for the next five, 10 years.

Jim Turner (28:49):
Couple of tools that I also wanted to talk about. We're using AWS to build some of our Linux systems. And the reason why we're doing that is it cuts build times from a day and a half to two hours because we can get 24 core machines to be able to build this software in a much faster manner. There are some others, we talked about static analysis. Lint is one that we use. There are others that we have used in the past. Sometimes with the compilers, these are all by setting up your tools with Lint, you are able to on a regular basis, make sure that the code is compiling to the rules that you expect. You can actually do some reformatting of code so that when Mike types it in his editor with his two tabs, it reformatted because we've agreed that three tabs are what we want across the company. It's little things like that.

Jim Turner (29:47):
We can also do checks of the code for proper headers that we can spit out such as Doxygen so detailed designs that are in the code can be improved. A lot of tools that you can do for an automated, that's all part of your continuous integration. And it's one of the things which Mike and I have touched on, but I'll mention it now is if the code doesn't compile, the offender usually gets an email in their mailbox and nobody wants that coming into their mailbox, that they caused a build failure. It's a very negative way to have corrections on people, but it's useful, so we also use email, is the point that I'm making in order to correct bad habits.

Mike Goulet (30:47):
Yeah. Getting back to your thing about the artifacts, Jim. There is this one famous example on the NPM, node package manager, repository on the internets. There's this tiny little package with, I don't know what it was. It was 10 lines of JavaScript code that it turns out like every single library in NPM and by extension, thousands of websites on Planet Earth depended on this and this guy for various political and I don't know what reasons, you can read about it. I think it's called Kik, K-I-K. Anyway, he removed that package and so many websites broke because their continuous integration systems were pulling from the internet. We actually, in our CI environments, we are pushing to control those inputs and not just let them be subject to the whims of the folks on the internet. We try to bring them in-house and control them. And that works nicely for the regulated industry as well.

Jim Turner (31:51):
And I have another example of that, iOS 13 came out and one of our clients was using. We're working with them and they had a specific GUI representation they wanted and the code was deprecated. We

actually had to fix the code before the next release happened. This goes into the whole concept of continuous integration. You upgrade the whole OS, Android and iOS do continuously and some of the code no longer exists that you've been relying on. You find that early, you fix it, and then you can redeploy with your fix and then do your targeted regression testing if you do it manually or you can do it automated and check that code out and then fixed it and then start putting in at the branches so software QA can look at it before you release. And we were talking about storing code, but this is also important forever with Artifactory, but there are also versions of test code that you may change. The version may change. And so we have to keep on top of that.

Tyler Kern (33:08):
Well, Mike and Jim, we have I think, thoroughly answered those four questions. And I think you've provided some incredibly fantastic information here on the topic of continuous integration and automated testing. Is there anything that we haven't covered yet or any summary statements you'd like to make before we sign off for this episode?

Mike Goulet (33:28):
No, I think we kind of went over it from a pretty high level. It isn't really that complicated of a concept. And I think this conversation was good and reflective of that. And no, I don't have anything else to add. I think we did a reasonable job just explaining what it is and why we would do it and when we would do it and some of the technology about how. And no, I would encourage others to go do some research into it. If you're not doing continuous integration, go online and go learn about Jenkins and TeamCity, and CircleCI and see if those tools are right for you and your teams. And then of course, by all means, reach out to Sunrise Labs and we can help you with that as well.

Jim Turner (34:19):
Yeah. And what I'd like to add is Mike is correct. The concept is not too hard. The entropy's at work and you get developers, you get five people together and you don't have control over what they're going to do and you'll have a mess on your hands very shortly. Think about the universe wanting to break apart, well, so does your code ultimately, you got to put the processes in place to bring it back.

Mike Goulet (34:48):
I like that.

Jim Turner (34:48):
And make sure everybody's using. What's that?

Mike Goulet (34:50):
I said I like that.

Tyler Kern (34:56):
Well, Jim and Mike, thank you again so much for joining us for this episode of the show. We appreciate your insights and your expertise in this area quite a bit. Jim and Mike, thanks for joining us here on Making Bright Ideas Work.

Jim Turner (35:07):
Thanks, Tyler.

Mike Goulet (35:08):
Thank you, Tyler.

Tyler Kern (35:09):
Absolutely. And to our listeners out there, thank you for joining us for this episode of the show. Reminder to go to sunriselabs.com for more information and stay tuned for more episodes of the

podcast. If you're not already subscribed on Apple Podcasts or Spotify, make sure you're subscribed there to stay up to date with the latest from Sunrise Labs. And of course, we'll be back soon with some new episodes, but until then, I've been your host today, Tyler Kern. Thanks for listening.