Challenges to Implementing DevOps in Highly Regulated Environments

Featuring Hasan Yasar and Jose Morales Interviewed by Suzanne Miller

----

Welcome to the SEI Podcast Series, a production of the Carnegie Mellon University Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the U.S. Department of Defense. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

Suzanne Miller: Hello, my name is Suzanne Miller. I am a principal researcher here at the SEI. Today, I have my colleagues Hasan Yasar and Jose Morales with me. Dr. Yasar has been recently elevated to being the technical director of a new directorate here at the SEI called Continuous Deployment of Capability. Jose and I both worked with him in that directorate. So, congratulations and welcome.

Today, we are here to talk about challenges, which we all are dealing with, in using DevOps in highly regulated environments [HREs]. So, these are environments like nuclear safety, safety, critical systems like airplanes, places like that. Those are some of the areas that the SEI works the most in. So, we are going to talk about sort of what are the challenges there in that. Dr. Morales recently and Hasan and another colleague of ours, Aaron Volkmann, just published a paper about less than a year ago on this very topic. So, we will reference that as we go through things. However, before we start, tell our audience a little bit about how you each ended up here doing this kind of work. Not everybody gets to do DevOps work in highly regulated environments. Why don't we start with you, Jose.

Jose Morales: Well, first, thank you for having me back. It is a pleasure. I first started working in DevOps with Hasan about three years ago. At the time, it was not a very well understood concept. There was still a lot of effort spent in explaining what it is and the advantages of it. As it became accepted as an efficient and effective way to develop software, we started getting more and more clients that live within highly regulated environments, which are environments that have heightened security standards because they deal with projects or information that require this. Ever since then, I have been working almost exclusively in these environments.

There are a lot of challenges when you try to do this sort of thing. That has been going on for about two and a half years now.

Suzanne: Cool. Hasan?

Hasan Yasar: First of all, thank you. Thanks for inviting me. It is a great opportunity to be here. How I started in DevOps? It is a long journey for me. Before starting at the SEI in 2010, I was running a company doing software engineering work where one-person-does-everything type of situation. You know that start up mentality. We were doing a lot of coding, but not really as efficient. So, then I started at the SEI in 2010.

When I started at the SEI, I was dealing with so many projects, like 18 projects. Actually, it was the first time I start using DevOps in managing the projects. It seems awkward, but it really helped me to manage the projects on time in terms of traceability trees, tracing, or looking for any project content. Really, how it helped me as a team, so the team started to really make their work visible to the community with all the team members, all the stakeholders. So, all the work was visible. If the work is visible, that means I am able to manage properly what we are doing and where we are in the process of delivering any tasks. That all started when I saw the benefits of project manager perspective. Then I saw developers taking advantage of being transparent, sharing, and also getting the collection of information amongst each other, which is basically a definition of DevOps and how it started.

We started implementing since that day. Now, within the SEI we were doing more with the web application at the moment. Now, the DoD, essentially what we do at the SEI, it’s more about different systems. I realized that we cannot take really web based way of doing business or web type of application into the same type of mentality into the safety-critical systems or the highly regulated environments. It is not that easy. It is not really getting the continuous testing like fail and then to recover later. You cannot let it fail in production environment, in a safety system. You cannot let something like an aircraft do something crazy things or some other missiles. You cannot let that happen. So, you have to do certain characteristics or certain of DevOps components, in that environment. It is still little DevOps. It will still be applicable, but in a different way of doing the testing, a different way of doing automation and a different way of addressing the security or handling the area of environments. How we can do that area of environments? Usually if you talk about web applications such as Google or Amazon case, and developers are able to go into the production environment. Like they can really entitle or empower the production environment and change the content of applications. When you take that approach, you cannot get into the safety-critical systems because the developers cannot go to the production environment because of the constraints on the compliances and also the other set of rules is required.

Suzanne: What you have just talked about is something that I have heard you talk about is environment parity. Achieving environment parity is one of the ways that you allow that transparency and that you get confidence that what you have done all along the way is actually going to work in the production environment. So, let's talk about that challenge in particular of how do we approach—I'm not going to say achieve—but how do we approach environment parity in these environments where the regulatory framework or the criticality of the hardware that we are using affects our ability to just say, *Well, I want to go off and do this in the production environment, of course I do.* How do we deal with that challenge?

Hasan: I am going to start first and describe what the problems are, why we need that environment parity. It is a foundational concept of DevOps. Why would the developer like to see its result in production environment? Because they would like to see anything that is failing. So, what is really failing in application context is the dependence as an example or the other environment variables are changing. Because typically developers are developing a code in their development environment. They cannot replicate the same as the production environment. That is the reason they would like to see the production environment, take a look at the results and how the system is behaving over there. Then, if you get a parity, that means the same configuration of environment. So, we can give a chance to a developer not only for using their Dev environments, they can use in their staging environment as the same environment as the characteristics of the production environment. It doesn't mean that we have an identical in terms of CPU [Central Processing Unit] and memory and usage. We would like to have the same configuration, the same dependencies through those environments.

Suzanne: Jose, do you want to add something to that?

Jose: I have been thinking about this. I divide environment parity into two areas. There is environment parity amongst developers where there are development environments. Then there is environment parity between the staging environment where developers test their code and the actual production environment where it will eventually live.

I have seen problems in both areas. When it comes to the developers, if they do not have environment parity, then simple things like a dependency that changes a library that is different, a subversion of something will build two pieces of code that are totally different. Thus, this person says, *It is good*, and this person says, *It is bad*. Now you have to go back to figure out that extra time and extra cost that is not required. Also, if they are not storing or sharing their code correctly, and there is this difference, the one that didn't build right or built with a different version of what is expected could end up being deployed out. Therefore, you have a missed configuration. I have seen that cause several issues.

Suzanne: Those are really hard to troubleshoot.

Jose: Those are very hard because there is this minute little detail buried away in a sub-subsystem library. The other issue is when you have your staging environment, which is your internal production environment. It is controlled by developers. It should always be as equivalent to the production environment as possible. Now, in some cases, a production environment is a huge system of which you are developing one subsystem. So, at a minimum, you want to have a staging that has parity with that subsystem, the media environment under which you will be running once you are deployed out. Emulating or testing the connections between that subsystem and other parts of a greater system is a bigger challenge. Sometimes it is unachievable. Sometimes it is [achievable]. When it's not, you are taking a big risk because you are going to test for maybe even the first time, your communication with other parts of the system when you are out in production.

One of the key things that have been adopted today to avoid these sorts of issues and sort of enforce environment parity is the use of containers. Before then was the use of virtualized environments. So well-equipped developer teams have a provisioning server. This server will build you the environment you request. So, everyone agrees upon an environment for development. This provisioning server gives you a container with the environment you are going to work on. So, everyone is working on the same thing, all the developers.

Staging in highly regulated environments can be very difficult because the production environment is either inaccessible, or it is just not accessible now. Sometimes a thrust that has occurred in the community is for some systems upon which you are building, for the manufacturer of that hardware will make available sort of an engineering release of it. In some cases, the developers, the customers, all the stakeholders—true DevOps involves all stakeholders—will come to an agreement that this engineering release, which is publicly available, is good enough for a staging environment.

Suzanne: Or give caveats as to where it can't be.

Jose: Correct. You will identify where the gaps are. You will measure the risk of testing in the environment with those gaps. In most cases, they are acceptable, and everyone will test as a staging the engineering release, because it is the closest you are going to get to the production environment.

Suzanne: This brings up a different challenge, which is a lifecycle challenge, which is if I as a developer, am trying to get this environment parity in a hardware subsystem—just say some kind of a hardware subsystem—I have to coordinate with my systems engineering community and my program management community to make sure that those engineering releases, that hardware, are available in a time frame that I actually can use them. Because one of the things I have seen is where, if that is out of sync, if the hardware people don't know that the software people need an engineering release to work from, then they are moving along, and they are not planning to deliver anything for another six, eight months past when I need to test things. That is one of the challenges that I have seen in moving into that hardware-centric kind of software environment.

Jose: I would like to add to that that one observation I have made out in the field in these environments is when you are applying DevOps a key aspect of it is testing in small chunks in your code. You do unit testing. You do functional testing. You do integration testing. But in most environments that I have been in, the testing does not include an iterative process where you have staging involved in it. In some cases, if there is a big gap between staging and production, production should be involved in it as well. Usually, the testing is unit testing, little chunk unit tests. *Great. Functional tests requirements are met. We are done*. Then we integrate into the larger stream of code with everyone else that is working on this project. At that point, you are now testing in likely a virtualized environment with everybody else's code. But it is not truly staging. So, the staging is lacking. In cases where staging is there, it is a one-shot deal into stage. It is not necessarily iterative because the assumption is once you have met the requirements, you will have to keep doing an iterative testing process, which is completely not the case.

Suzanne: Yes. Actually, one of the things that drives that is the cost, right? The cost of system integration labs, of hardware that I can use as a prototype, that is a scheduled resource, right? It is not something where I can just say, *Hey, I am ready*. If it fails, I can go back and tomorrow I can get in again. There is a queue of people waiting for that scarce equipment because it is expensive in many cases. I mean you don't have nuclear power plant stuff just sitting around, waiting for you to do iterative testing.

Hasan: Yes. That is why you cannot replicate exactly the production environment. Let's go back a little bit, environment parity again that Jose described. We are looking for similar characteristics. That characteristic means sometimes you should have maybe hardware prototype, and maybe having a simulation similar to the environment. Like thinking about, *If I am going to run a system in a bigger picture in a production and what my systems environment it is*. Try to build up the environment from production and try to carry it to the left. We call that a shift-left. We don't want to get shift-left only for the security component. We would like the shift-left on the production parities as well. Build it and get hardware engineers and system engineers in talking to describe what are elements of the configuration of this environment or what are the characteristics of the environment and try the simulated environment first. Simulation is, again going back to your point, when you said cost-related because cost is too much in the hardware. You have to spend a lot of time and money. Why don't we try to think about a simulated environment first as for testing purposes? I know most of the hardware engineers, they are using a simulated environment first before building the hardware. If I know my systems, embedded systems, and I am writing code, then why don't you use my code in that simulated environment to see how it behaves? Forget other components yet. If I am building a skeleton of the house, I would like to see the boundaries first. Make sure that it is going to work, that it is going to communicate well, then I can go deep dive in each iteration. Starting from—in connection with the hardware and software engineers—*Look what we can do in terms of small-testing cases. Where is the environment that we can build it together, which is a simulation environment?*

Suzanne: This kind of relates to the DoD's Digital Engineering strategy is trying to look at improving our overall modeling and simulation capability. This is one of the areas that it can really pay off, is if we start looking at, *How do I simulate the pieces, not just the pieces but the interface in particular, that I need to be able to do iterative testing and get that environment parity*. All right, so, we beat environment parity to death. What are some of the other challenges that that you run into when you are in these highly regulated environments that you want our viewers to pay attention to?

Hasan: I would like to bring the disconnected environments, because you cannot really connect the dots to the system wide. It is disconnected. It is a totally different network, which is an actual production environment running over there, which may be classified. While we are talking about, maybe totally disconnected means that the developer cannot push the code in one button that goes to certain steps and build and deployments starting in the production environment.

Suzanne: There's still Sneakernet involved.

Hasan: It is some sort of communication mechanism. Either Sneakernet or somebody has to do some extra work as a human and move the code from one environment to another environment. It is a big challenge actually.

Jose: I have actually seen this happen where people Sneakernet information from open environments to closed environments, and there are all sorts of problems. In the closed environments, they need updates multiple times a day. In the open environment, where they are going to upload by walking from one place to another, occurs once a day. So that pushes back schedule unbelievably because there is no automation involved. A key part of DevOps is automation. You could have, for example—I will get really technical—a diode that just shoots things one way, and it automates to upload information on more regular basis. But then you get into the business model of things, *Is it worth it to do that? Should we do that*?

There is a big gap in automating movement between open and closed spaces and for obvious reasons. But from open to close should be an easy implementation that can be automated outside of the required authentications. With some proper encryption, it should work fine. If not done, as Hasan was pointing out, it does cause several problems, especially setbacks.

Suzanne: This is the kind of thing where we have a cultural component too. The culture of open-to-close is often, *Manual is better because humans are involved and can check*. What we are saying is there are places where the human actually introduces risk by not allowing us to have the updates as frequently, by not allowing us to have the system available in areas where it is needed. So, we have got to get some cultural kinds of mindsets. We are not trying to say, *Go close-to-open*. Nobody is saying that, but this one direction we should have a little bit easier time.

Hasan: I am just going to add a little bit more. It is not just a human over there. There is a physically disconnected environment also. Totally disconnected means, *There is no way I can connect my current network into that environment*. *There is no way*. You cannot really have the build process on. Now we have to have human elements in the process. Get the system, build it, put it into the other side of the environments or the places.

There are a couple of things we looked at for solutions as advice on what we can do. That goes back to that environment parity. I have been seeing that organizations that are successful, they are keeping environment parity on both sides. So, if I am building an application as a container, that I am sending that container either through the Sneakernet or somebody who carries with a CD or somehow that goes into different environments, make sure that we are eliminating any manual process, such as the build process. Take care of the manual process on different environments, like in the Dev environment, staging environment. Build a package. A container, as an example, can be another binary file that has been analyzed and approved, and it’s good to go since we know the environment parity is the same for both environments. So, my code will run exactly on the other side of the house, like high side...we call as a high side. in DoD context it may be totally disconnected for its physical systems, probably safety system. Get-code pieces have been packaged properly, and we know the dependencies. Since the environment is the similar configuration, same parities, then you can run a code in the high-side environment.

Suzanne: What we are looking for is, I call it one-plus button pushing, right? It’s, *I have got this package that has been established. I really should only have to push one button on the other side to kick off the whole provisioning activity that will result in the same system in both places.* There is the plus in there of, *I have got to get it to the place*, but we still are not looking for completely manual builds. That is out of the question.

Hasan: Right, so it is going to be automated. We are going to automate the build process. We have to build artifacts automatically. We have to create older requirements that the application will run automatically in that environment that we are working on it, and just push when the new build is ready to go. The audience can ask the question why we are not building in the high side or that environment. There are some limitations on this environment. It is a closed environment, which means you may not be able to reach out to internet. You cannot really pull a new set of libraries. Because in this day and age—we talked about this in previous podcasts—we all depend on the Internet right now. Like all the engineers are looking for new libraries and understanding how to write the code and stuff. It is very difficult or time-consuming, building an application in constraints that you cannot go. You cannot look at the notes. You cannot try anything else. So, you need some flexibilities. If you look at DevOps as getting so much attention in the community because it is giving so much flexibilities. Developers can go and pull up ready containers or ready configurations, like chef recipes or cookbooks. It is available somewhere else on the Internet. We don't need to rewrite that configuration element. We don't need to write some images, so we can use that. We can use ready provisioning scripts. We can use ready configurations or deployments. It is available. We just have to go change some of the components. So, this is not available in a closed environment. That is the reason we have to build some things in different environments and push the build into the high-side environment. That is difficult actually.

Suzanne: You and I, actually I think I'm older than both of you, but I grew up in the days when, well, no Internet. Let's just start there. Handcrafting code was the only way code was built. I mean, the number of libraries that were available for different common functions was still kind of limited. People that actually grew up in that environment actually have an advantage in some of these places of they are actually accustomed to handcrafting. Where some of the people that have grown up in more of the Internet space are not as accustomed to this sort of hand-crafting idea. That's just something that sort of came up as we were talking about it. An advantage to being old for once. Yay!

All right, so let's talk a little bit about compliance. One of the things about regulated environments is there are typically compliance issues. Someone has got to prove to someone else that you have complied with safety regulations, security regulations, financial regulations, whatever they are. What are some of the challenges and solutions that DevOps offers to help that side of the highly regulated environment?

Hasan: I am going to talk about the challenges and hand over to Jose for the solutions. What are the challenges that we saw in the compliance perspective? Usually or mainly it is the responsibilities, because we are building a system that has to comply with some privacy. Or, if I am in the financial sectors, I have to become compliant like the SOX [Sarbanes-Oxley Act] and HIPAA of health industries.

All of these compliances have some certain controllers that have to be in it. The controllers are—it depends on which compliances we are working on—it is requiring an approval process. It is requiring some auditor traceabilities. If you look at GDPR as an example, which is more about the Privacy Act. As the developer or system builder, we have to show evidence of how we are handling privacy in our application lifecycles. The same thing for the financial sectors. Some certain things have to be in place, and auditors will look at the code pieces of the system to verify.

So, the human, or an auditor or assessor, has to approve the process. The challenges are it is a manual process, because these people are looking for artifacts. These people are looking for artifacts such as how it is done, how it is implemented. Going back to the DoD, the context is about the ATO [authority to operate] process, what other controllers are in place. As an assessor, the information issued to this person, I need to know how it is implemented, which phase has done proper testing, and evidence. If it is manual process, mean it is manual, somebody has to say, *Yes, I did it. Here's my checklist and checklist and checklist for every component*. It takes time. It is a big barrier because of we cannot really push a version of the application to the production without getting this approved. It takes six months sometimes, and that was a challenge. So, how DevOps is really helping?

Suzanne: How can DevOps help me with that, Jose?

Jose: I have seen this in action out in the field and using DevOps. What I have learned is you can build the compliance into the DevOps process when you are building in little nuggets of your work. There is this one place I worked at where they call them nuggets, but I think it is a generic term. You build a small piece of code. It goes through the regular testing. But within there, there is this one chunk called quality control, quality assurance, compliance control. Once it goes through there, as part of the DevOps process, it goes in and they check that little piece of code that you gave if it is meeting all the compliances. Now within that, there might be some automated testing to which the compliance officer is shown the results. There are still some manual parts, because there is paperwork and other things. But you can build the automated parts in a DevOps way. The manual part you compile. Once you have hit a certain amount that is acceptable to give all the needed documentation, you give that in with the automated results that you had from before. So, it actually does accelerate the process. One major issue that we have is not compliance, but it's ATO.

Suzanne: Which is authority to operate, for our audience.

Jose: ATO, authority to operate. I think you might cover this later on as a good topic for its own podcast, is, that has been built in as well because you can automate a lot of the requirements for hardening, for STIGing [Security Technical Implementation Guide] things, all the compliance needed to show that this is authorized to go out into the production environment and operate there.

Hasan: I am going to open up a little bit the solution side of it. When you look at all these compliances at the beginning, there are certain things required, like such as report, audit, and traceabilities. So what DevOps is bringing that is kind of my journey how I started from SEI, I was able to see every report, every project. It was completely traceable for me. DevOps really brought me a traceability of any given project. I am putting myself in a situation, if I am going to approve a process of the systems, I need to see all the issues that have been created and I will, like, *All the test cases been done*, and I will, like, *There's a test report*. DevOps is bringing all elements automatically. In the DevOps pipeline, it's all connected, my issue tracking system, make sure tracking system is connected to my repository management, then my build server is connected to my repos [repository] and then issue tracking system as well. Then all my testing is also connected as a part of the continuous integration server. So, if the system is automatically creating an end result to me as a report, how it is done, how it is implemented, either doing proper testing, then I am getting the reports. I know exactly, as I'm looking for auditor-perspective, and I can see who touched the code, when they touched the code, and why they changed it. It's one of the requirements in the financial systems. You have to prove why you change that code and for what reason. There is a traceability that goes back to the requirement, which is another, and V&V, verification and validation process requiring to show it is verified and validated. These are the components DevOps out-of-the-box is generating if you implement properly. That is one other thing. I think we discussed in other podcasts, DevOps is not just the CI [continuous integration] server, it is the whole life cycle. We have to have the systems, how we are building from the beginning, which has the planning phases in it and have the case management, having the web page, that always goes back to your production environment. This is the lifecycle they are talking. This is DevOps pipeline they are talking about. If it is properly established, and all of those can collect every information easily from every lifecycle.

Suzanne: One of the things that I've seen you demonstrate in some of the classes that we teach is this idea of embedding documentation that would not normally be considered code documentation. And, I am thinking about compliance type of documentation, using markdown languages and things like that. So that these reports you are talking about, what I have seen you do is to actually generate them from the code base. So, the term that you have used for that is *infrastructure as code*, which I love because it gets you to the planning. The infrastructure is that whole lifecycle infrastructure, not just my configuration management.

That concept is one of the ways for the regulatory environment, for the people working on that side to start thinking about suggesting things. *Why can't we have these kinds of compliances included in the code base?* I think that is a really powerful solution for moving some of the folks that are really overburdened. Most of these authorities, whether it is financial or HIPAA or anything else, they are overburdened with these manual processes. They don't really know what we can do for them, so this is part of the education.

Hasan: The community is building lot of compliances called concepts. It is available in GitHub. There are a lot of folks feeling the same pain, and they are building up libraries. It is like the libraries are a set of code automatically checking the configuration. If I am doing something at the OS level, and there is a FISMA [Federal Information Security Management Act] compliance, it goes to check automatically that my system is compliant or not. It is automatically done. Even though some other configuration elements or the configuration tools have a built-in capability that is looking for the configuration of the systems in that level. So, it is really available to communities to keep producing that. In other things, the developer would like to be connected to the community, because it is available and use that piece of code to check how we are doing in terms of compliances. Like, *Who is accessing some environment*, as one example that is required from the financials. Or, *How I am handling the PII [personably identifiable information] data in my systems* is a possibility.

Suzanne: So, we have hit environment parity, and we have hit compliance, and we've hit full lifecycle, involvement of stakeholders, and we have hit dependencies and interfaces as being big challenges. Is that enough for our viewers for one day or do we have other things that we want to bring in before we close today?

Jose: We can add many more. I think one interesting one that I think we could talk about is authorization to carry out a project when the request comes in from a customer. Often times, this is a part of the SDLC [software development lifecycle] that is not necessarily well defined, because it is viewed to be outside of it. In a way it is, but it impacts. I have seen scenarios where a customer puts in a request to build something. The developers receive it. They have to send it off to management to have them sign off.

In the environments that we work in, management is not the person in the office next door, or down the hall, or in another building, or in another city, or in another state. When you pass that along to them, the amount of time it takes them to sign off can be highly variable. Now for long-term projects, this is no problem. I am talking projects that are months or years, but there are projects where you have weeks or days, in some cases, hours, and you still have to get this authorization. The time it takes to acquire that, which you cannot start until it is in place, digs into your development time and the amount of time allotted to something like this never takes into consideration the time to get it authorized.

One thing that we came up with is to do a fast authorization where there is someone on site with the developers who can authorize it while it goes off in its usual process. The reason why it authorizes on site is because of the short time period. Now, this isn't DevOps directly. But when you are in highly regulated environments, and you are building a DevOps process for their SDLC, you are not going to answer everything with DevOps purely. There are issues in there, and I have dealt with them on the ground. The solution is a non-DevOps solution, but it augments the DevOps, and it enables DevOps to move forward.

Suzanne: But I would argue that even then, if you look at the DevOps principle of collaborate, it is right there, right? We are finding a way to collaborate even if it doesn't involve automation. I would say it counts.

Jose: Why not? It is a good point.

Suzanne: The purple-haired woman said it counts, so that has got to be right.

Jose: Absolutely.

Hasan: The collaboration piece has to be in place. That is a big thing. I looked at the number of the problems or the obstacles we discovered, it was about 12. So, we can cover the rest of study at another time. It's a lot.

Suzanne: Maybe we can give people a break and do some more of these.

Hasan: Yes, I think we covered the fundamental concepts and started from the ground up talking about the environment and talking about the approval process and talking about the compliances, these are the hard-core issues in our solution we discussed. I think we can go to a deeper level and talk about acquisitions, one of things we have to tackle. We have to talk about the people, the skills required, and then also the changing of the people roles, or the people moving from one location to another location, which is retention. These are things we discovered, so probably we can talk about these in the next podcast.

Suzanne: OK. The paper that Hasan is referring to is called *Implementing DevOps Practices in Highly Regulated Environments*. The link to that will be in the podcast transcript, so people will be able to find it. Then they can read about those, so they can get ready for our next podcast on this topic. I want to thank you both for joining me today and having this conversation. You know that I am passionate about bringing these things out into the larger world. I really appreciate you doing this.

For our viewers, as always, if you have any questions, please send them to info@sei.cmu.edu. Also, don't forget, you can find this on our YouTube channel. You can find it on SoundCloud. You can find it pretty much anywhere that you get your podcast today. So, thank you for joining us and go forth and collaborate and have environment parity.

Thanks for joining us. This episode is available where you download podcasts, including SoundCloud, Stitcher, TuneIn Radio, Google Podcasts, and Apple Podcasts. It is also available on the SEI website at sei.cmu.edu/podcasts and the SEI's YouTube channel.