Deloitte.

Architecting the Cloud, part of the On Cloud Podcast

Mike Kavis, Managing Director, Deloitte Consulting LLP

Title: **In-production testing: the future of software testing**

Description: Traditionally, the software testing has been firmly planted in the staging process, so as not to disrupt in-production systems and functionality. However, by testing in staging only, developers may miss issues that only occur with live systems. In this episode, Mike Kavis and Split.io's Talia Nassi discuss how savvy software companies are breaking tradition and moving their testing process from staging to production. According to Talia, in-production testing is the future because it's where users experience the application, so it's more effective if done safely. Mike and Talia also discuss what's required for safe in-production testing, and they debunk some production-testing myths and discuss how the emerging discipline of chaos engineering is related to production testing.

Duration: **00:20:22**

Operator

This podcast is produced by Deloitte. The views and opinions expressed by podcast speakers and guests are solely their own and do not reflect the opinions of Deloitte. This podcast provides general information only and is not intended to constitute advice or services of any kind. For additional information about Deloitte, go to Deloitte.com/about. Welcome to Architecting the Cloud, part of the On Cloud Podcast, where we get real about Cloud Technology what works, what doesn't and why. Now here is your host Mike Kavis.

Mike Kavis:

Hey, everyone. Welcome back to the Architecting the Cloud Podcast, where we get real about cloud technology. We discuss all the hot topics around cloud computing, but most importantly with the people in the field who do the work every day. And speaking of that, today we have with us Talia Nassi, developer advocate at Split.io. Welcome to the show. Tell us a little bit about your background and your journey to where you are today.

Talia Nassi:

Yeah, thanks for having me. So, I'm Talia. I'm a Dev Advocate at Split, and my focus has been testing in production. And the way this all kind of started is I was working as an automation engineer at WeWork, where I was implementing testing in prod from start to finish and I was using Split. So, now I work at

Split and I help other companies do it, and I create content around testing in production. And I'm very passionate about the topic, so I'm so happy to talk to you about it today.

Mike Kavis:
Yeah, so I saw one of your keynote presentations recently and what was interesting is you talked about how you got introduced to the topic. I thought it was a good story, so I figured why don't you share that with us here.

Talia Nassi:
Yeah, absolutely. So, I was working as a test engineer for a few years at the time, and it's something that I did right out of college, too. I knew that I was passionate about testing so I went straight into it. And the norm of testing, of just like the regular SDLC process that I had always been used to is like you test in staging and then you deploy to production, and if there's problems, that's not your problem. It's the ops team or it's someone else's deal, because as a QA engineer or as a test engineer you don't have access to production in some cases.

So, I was always used to the norm of like testing in staging, and a couple years ago I was interviewing for this company and I asked the interviewer about their staging environment and about their testing environment. And I asked how often do you dump data and who's responsible for maintaining staging and how many staging environments do you have, just all these testing questions. And they said that they don't have any testing environments, that they test in production, and that was the first time that I had ever heard of the topic. And at the time I was just so confused. I had never heard of this. Is this a trick question? Am I being punked? What is going on?

Mike Kavis:
[Laughter] They're trolling you in your interview.

Talia Nassi:
Yeah. And the way that he explained it to me made so much sense and there was just a lightbulb moment in that interview where he asked me to do something. He said, "How do you know that your features are working in production right now?" And I couldn't give him a straight answer. I don't know. Unless you're running tests there, how do you know? And since then I was just like, "I can never go back to testing in staging. Staging is dead. Goodbye." I wrote a blog post a couple months ago on – it was a breakup letter to staging. And I basically – I was like, "I'm done with you staging. Goodbye."

Mike Kavis:
Oh, we're going to have to include that in the show notes so make sure you send that to us. That's going to be a good one. So, did you get the job?

Talia Nassi:
Yeah, I did. *[Laughter]*

Mike Kavis:
Good. They must've been excited about your interest. So, that leads into my first question, because usually when you say testing in production to people who've never tested into production, after they get over the shock there's a lot of myths about testing in production, both the magical myths of the plusses and then all the scary myths of the minuses. So, why don't you go over some of the myths about why people think testing in production is bad and then some of the myths, why people think it's the magic bullet.

Talia Nassi:
Yeah. So, I think some of the myths around it, like when you first say testing in production – and this happened to me, too. People are just afraid. They're so scared of what it means and how to implement it, they think it's impossible and you can't do it safely and this is why we have staging. And it's just the way that people have done software testing for so long is this old process of using staging. But as technologies evolve and as our code processes evolve the way that we do things is going to evolve with that.

So, we don't have to use a staging environment anymore. We don't have to keep going with the norm of how things have been. There's always going to be those people who say, "It's never going to work. Testing in production will just never work. Just use staging." And honestly I don't care about those people. If this is something you really just can't wrap your head around and you can't see the light at the end of the tunnel with setting up test in production safely, then the process just isn't for you. But if you are willing to do it safely and set it up using the right tools and using feature flags, then I would say absolutely give it a try.

Mike Kavis:
So it's not the end-all be-all as well. There may be some myths that you just do that and you're good. There's still testing that's going on before production, so what is the ultimate difference when you get to production?

Talia Nassi:
Yeah, and some of those myths are that testing in production when you do that, those are the only tests that you run, and that's absolutely a myth. That's not true. So, there are tests that are not run in production, and those are due to data issues or privacy issues or there's GDPR and – what's the other one, the –

Mike Kavis:
There's a lot of compliance. There's PII. There's PCII. I mean –

Talia Nassi:
Yeah. So, there are tests that are not run in production, and it's based on the scope of your product. So, I would say every team is going to have end-to-end tests that are going to be running in staging because of these other dependencies. But your main user flow, those should be done in production.

Mike Kavis:

Testing in production probably requires a certain level of application and architecture maturity before you go there, right?

Talia Nassi:
Yeah.

Mike Kavis:
Maybe around the areas of maybe automated provisioning, CI/CD pipeline. But before you go there, what are some of the things that both technically and culturally have to be in place before you go there?

Talia Nassi:
Okay, so technically I think there's two things. So, first is an automation framework. You need to have automation in place, and you need to have a testing framework that's already in place. So, my favorite testing framework for automation is Robot Framework. It's a keyword-driven library that I've used for automation for years and it's my absolute favorite, very easy to read and easy to write. And when a test fails it's easy to see what exactly happened. When a test fails in production you want to know exactly what happened, what's going on. You want to be able to analyze it really fast, so you want a tool that does all of that and has really good reporting. So, I go through a few tools in my presentation, but my favorite is Robot. So, the first thing you really need to have in place is an automation framework, and you need to have good test reporting with that framework.

And then the next thing you need is a feature-flagging framework, and feature-flagging is kind of the key to testing in production. It's a way to separate who sees your features at which point of the process so you can, kind of like dogfood, in the beginning and say "I only want my internal teammates to see these features so that I can test in production with my team, and then once it's ready I'm going to slowly release it to everyone else." So, I would say the two fundamental technologies that you need are an automation framework and feature flagging. And then with that comes your alerting tools and your job scheduler that can be integrated with your tests. But those are the two technologies.

And then in terms of culture, I think having an open mind is very important, because generally the people who want to test in production are the people like me who, years ago, just had so much trouble testing in staging. It's the worst feeling when you spend so much time testing a feature in staging. You make sure it's perfect. You spend I don't know how many days testing end to end, all the different flows. You know, you spend a lot of time writing automation. And then as soon as you push this code to production it breaks because of some other thing that was going on in production. And it's horrible as a test engineer when you have that feeling because a lot of times you're seen as the gatekeeper and you're seen as the person who's supposed to be in charge of all of the code quality.

But when you're testing in production, really the whole team owns product quality and code quality. So, I think in terms of culture, the more experience you have with how bad testing in a staging environment is, the better. I think it's really rare for companies to go from testing with nothing, to testing in production. I think generally they go from testing in staging, to testing in production because they know of– they know why they want to do it and they understand why staging is just not the place to be testing, because at the end of the day I don't care if my features work in staging. That's not where my users are going to go to experience my application. They're going to go to production. So, when you're doing it safely, there really is no other choice.

Mike Kavis:
Yeah, and staging environments usually aren't a replica of production, right? Especially in a large-scale system you can't afford to have a duplicate. Usually you have to scrub data so the data's not the same. There are all these differences, so getting it working in staging is nice, but when you roll to production, you're crossing your fingers, hoping for the best.

So, that's one thing. I wanted to ask a little bit about the canary or blue-green releases. So, I grew up in the point of sale world and we used to do something similar, very manual. But we would roll to five stores and let the store system software run a while, because we didn't want to take down 30,000 stores, right? So, it's the same concept but it's an old manual thing, and that's kind of what canary, blue-green, those types of things are. Let me try it with a subset. But does that require a certain change in architecture to the applications themselves? I mean, is it as easy as turning on tooling on the backend, or does the application have to be built to support that?

Talia Nassi:
So, when you're performing a canary release, I would say the easiest way to do it without having to change much of your architecture is by using a feature flag management tool, because, within the tool you can implement a canary release, and then you can correlate the different experiences with the different data that's coming in. And I think that's the most important thing with the canary release, is you want to compare what you said, those five stores that you initially release some sort of different experience to – you want to experience that to what's existing. And you want to be able to see that data side by side before you increase the allocation to another five percent.

So, I think visualizing and seeing that data and being able to compare the different experiences – and you can also import your KPIs and see how is this experience affecting my KPIs, and how is the existing experience affecting my KPIs. So, I think having that visualization of how the different experiences are impacting your metrics is the most important thing with experimentation and canary releases, because if something is impacting your feature or your experience positively, you want to be able to confidently allocate more traffic to that, knowing you're benefiting your customers. But if something is dramatically negatively impacting your user base, you also want to be able to roll that back really easily, and using a feature flag management tool just makes that so much easier.

Mike Kavis:
Does it require, or is it helpful if the application does some level of instrumentation to provide you with additional metrics? Or, I'm just trying to figure out, can we just do this or is there some input we need from development, from applications, from microservices or – this tightly-coupled app that doesn't work because they're hardwired to an IP or to a server? I'm just trying to understand from a development side what do we need to do as developers to assist in this?

Talia Nassi:

So, from a development side there's not much that you have to do. With Split the data will come in and you can view it. Usually, generally the product person or the businessperson will go in and look at the metrics and figure out how the different releases are going. So, as long as you have the feature flag set up correctly you don't have to worry about any other specific architecture.

Mike Kavis:
Does it help, though, to have developers build for testing in production, meaning implement different things they put in the log or some level of instrumentation or metrics or something that helps you gather information on the back end while you're testing? Or you just basically turn it on and go?

Talia Nassi:
With my experience I feel like the development team spends more time with creating features and creating new improvements, and all the metrics and the data and the backend stuff is automatically done with Split. So, they don't really have to worry about how their changes are impacting stuff because the product person will go in and say, "Oh, we're going to release this to more people or less people." Generally, as a developer you focus on creating more features, and with testing in production especially, you'll spend more time creating features and less time fixing bugs and defects, because you'll have already tested those in production. So, the process changes a little bit, but generally the developers will spend their time with features.

Mike Kavis:
Well, that's good thing. *[Laughter]* So, if I'm a tester and I'm in a company and I'm trying to bring this concept forward, which I'm sure you've had to do, what advice do you give? Because I'm sure the compliance and security folks are going to have a fit with this, as well as maybe even the engineers. But what's good advice for trying to bring this into an organization?

Talia Nassi:
Do it. *[Laughter]*

Mike Kavis:
Just do it.

Talia Nassi:
So, the biggest thing is I would say use your experiences from the past. Many times, like I said, engineers have this experience – and not just test engineers, but developers, too – where – I worked at a company where I didn't have access to production, because it was a payment processing company, and I couldn't access production because there was real money being processed, and I was just a test engineer, so why did I need access to other people's money? So, a lot of times you have the experience of, I've spent so much time with this in staging and it gets pushed to production and then it fails. Why even bother testing in staging? So, I would say use your experiences from the past to propose a path forward. Something that you can do is Split has a free version that you can just start testing with, testing in production, like using feature flags. So, you can go ahead and implement that for free and see if it's something that you like.

I would also think of why the pros outweigh the cons for your company. So, think of your staging environment. Is your staging environment reliable? And I know for a fact that it's not because I've never seen a staging environment that is an exact replica of production. It just doesn't exist. And think about if there's frequently issues that you think could've been caught if you were testing in production, which of course there are. When you do this process correctly and safely, it just makes such a big difference in your company culture that it would just speak for itself. Your developer velocity is going to go up. Your user experience is going to go up. Your bugs and defects are going to go down. Your productivity is going to go up. So, that's why I would say.

Mike Kavis:
Cool. So, next question is chaos engineering. So, we're just talking about testing features and functions in production, making sure what we put in our sprints are being delivered. How does this tie into chaos engineering? Is it related or is it all part of the same thing?

Talia Nassi:
Yeah. So, chaos engineering is based on building your confidence in your system. And that has to do a lot of times with maybe performance testing or load testing. You just want to make sure that whatever happens in production, you're ready for, and testing in production is definitely a part of that. So, when you're doing chaos testing, I would say the first thing you need to do is define some sort of like steady state as something that defines normal behavior, and then hypothesize what's going to happen when you implement some sort of experiment, whether it's through a canary or through an A/B test, and then figure out what you want to happen, what you think is going to happen, and then prove or disprove the hypothesis. So, testing in production is a type of way to do chaos engineering. Netflix, actually, I think started this whole chaos engineering thing back in the day. They have a really great blog post on it. Yeah, but I'm all for breaking things in production. You never know how your company and how your application is going to handle things breaking.

And something that I get asked a lot is, “Oh, but you can't performance test in production. You can't load test in production.” Yes, you can, because the only way to know that your application can withstand that much load or whatever performance, is to test it in production. So, I would say do it at a time with low traffic and be prepared for outages because you want to know that, like on Black Friday your company can support whatever amount of traffic.

Mike Kavis:
So, I've worked a lot in the chaos engineering space, and one of the things we tell our clients is if your system's already in chaos, don't go to chaos engineering. You have to have a certain level of maturity and reliability before you go there. But it feels like in this case, just testing in production in general, taking chaos engineering out, it might be a way to fix a lot of that chaos. So, the question is do you need a certain level of maturity before you attempt testing in production? Or is this a fix for some of the chaos that you already have?

Talia Nassi:
I think you definitely have to have automated tests in place because as soon as something breaks in production, you want those tests to run. So, I would say before you do anything it's just impossible to manually test every user flow, or everything when something goes down, or when you're testing in production. So, I would say the bottom-most thing that you have to have is automated tests in place. And I would say you should also have a good, continuous integration pipeline, so constantly merging with master. What else? No, I think that's it.

Mike Kavis:
Well, cool. Well, really enjoyed the talk today about testing in production. I know this is a topic that scares a lot of people, but I know you've talked and written about this a lot. So, where can we find you on Twitter and where can we find some of these talks to learn more about this concept?

Talia Nassi:
Yeah, so I'm on Twitter. You can find me at @Talia_Nassi. And I speak at conferences and meetups pretty much globally, so –

Mike Kavis:
Virtually now, right?

Talia Nassi:
Yeah, unfortunately it's all virtual now. Coming to a meetup near you in 2023, it looks like.

Mike Kavis:
So, where's the next one we can catch you on? Do you have one coming up?

Talia Nassi:
Yeah, I'm speaking at a meetup in Argentina.. And then I also have DevOps Days Bogota, so if you're in Colombia again. And then I have React Ottawa, so anyone who's in Canada, and then React Meetup Bay Area. So, I'm going to be speaking about testing in production, setting up feature flags with React, and then also controlled mobile rollout with React Native. And you can find all this on my Twitter. I post about events and speaking.

Mike Kavis:
Cool. Well, make sure you follow her out there on Twitter. That's our talk for today on Architecting the Cloud. To learn more about Deloitte or to read today's show notes, head over to www.DeloitteCloudPodcast.com. You'll find more podcasts by me and my colleague Dave Linthicum just researching for Deloitte On Cloud Podcast on iTunes or wherever you get your podcasts. I'm your host Mike Kavis. If you'd like to reach me directly you can reach me at MKavis@Deloitte.com or you can always find me on Twitter @MadGreek65. Thanks for listening and we'll see you next time on Architecting the Cloud.

Operator:
Thank you for listening to Architecting the Cloud, part of the On Cloud Podcast with Mike Kavis. Connect with Mike on Twitter, LinkedIn and visit the Deloitte On Cloud blog at www.deloitte.com/us/deloitte-on-cloud-blog. Be sure to rate and review the show on your favorite podcast app.