



CASE STUDY

# The Role of CloudBees and Kubernetes in Moving DevOps to the Cloud



## Industry

Insurance

## Geography

Global

## Product

CloudBees CI

*“From the beginning, we felt that CloudBees was committed to our success. We knew we could reach out if we got stuck, and they would point us in the right direction—which saved us a lot of time and frustration.”*

## Gurushyam Mony

Former Director, DevOps and Quality Engineering  
Markel

*Contributed by Gurushyam Mony, Former Director, DevOps and Quality Engineering, Markel*

DevOps has revolutionized software development. It empowers organizations around the world to deliver applications and services at lightning speed by eliminating bottlenecks in the traditional coding pipeline. Using DevOps practices, teams can provide value to their company in a fraction of the time.

All teams and organizations, however, are not created equal. How does this speed and streamlined workflow translate to a multinational operation with 2,000 IT people running dozens of legacy systems?

I'm the director of DevOps and quality engineering at [Markel](#), headquartered in Glen Allen, Virginia. I joined Markel in August 2017 and was given the opportunity to streamline software development, releases, deployments and quality assurance practices across the company. I was excited about the challenge, but it was indeed a big challenge. Markel is a global specialty insurance, reinsurance and ILS (insurance-linked securities) firm with 58 offices in 17 countries. Our specialty products insure items that aren't covered by traditional insurance policies, products such as racehorses, diamond-studded shoes and rock stars' vintage guitars. In addition, our Markel Ventures division owns and operates a family of companies outside of insurance.

To achieve our goal of a more streamlined development and deployment process, I began to work with a team of three full-time associates and contractors from our specialty and commercial divisions. We immediately embarked on a ruthless consolidation of tools and elimination of redundancies by assembling behavioral-driven development and automation frameworks to support software delivery and releases.

## Streamlining DevOps and Quality Control



We first introduced automation tools to four of our teams and then scaled this initiative to six teams across two divisions. As they began writing scripts to automate the software development cycle, these coders asked my team for an orchestrator. We introduced them to the open source server-based version of Jenkins. We also started promoting Jenkins to Markel's nascent DevOps team as a centralized tool for software builds and releases.

Within a year of rolling out Jenkins, we had become victims of our own success. By August 2018, we had one massive Jenkins platform for all our test engineering and build automation. What started as weekly runs of software

releases had grown into 22,000 build pipelines running on a 19-node Jenkins server farm. We also pushed out dozens of integrations into testing tools such as Cucumber, Pester, Specflow, Test Complete and Behave. Over time, DevOps and quality engineering teams had merged into a single entity, and today, my team comprises 12 polyglot full-time engineers supporting DevOps services centrally for Markel IT teams.

In some ways, we were doing great—but we had lost sight of our initial goal. We no longer helped our developers build consistent DevOps practices. Instead, we had become full-time integrators, pursuing tools to add into our Jenkins platform.

We were bogged down in build requests and had no visibility into our Jenkins farm productivity and performance. There was another round with sprawl of practices and tools. The logs were of various schemas, and consistency suffered when trying to mine them for intelligence. The result was a semi-disconnected value stream.


## We Needed to Move to the Cloud

There was no way my team could keep up with the growing number of tools and build requests. We had written a ton of integration code, and our developers were using a plethora of shared libraries. The sheer volume of pipelines was crushing our infrastructure, and we could not scale our physical servers to meet demand. The next logical step was moving our DevOps environment to the cloud.

We started to experiment with Jenkins X, the cloud-based open source version of Jenkins, but our engineering team couldn't handle the workload necessary to consolidate practices across Markel's many IT divisions. We didn't have the time or human resources to manually process every integration request and set up every tool needed by dozens of application teams worldwide.

With Jenkins X, my team had to authorize and authenticate every master server, which took a lot of time. Our applications engineers couldn't spin up a new master, and we weren't in a position to ask them to learn Jenkins X themselves so they could set one up or maintain their pipelines. We wanted to automate this and other processes to accelerate more teams to the cloud.

That's why I started looking at [CloudBees](#) CI. I had a hunch it was going to make our deployment processes easier and get our platform established faster. We also figured it would allow for application teams to operate in more of a self-service mode, rather than leaning on the DevOps team.



*“CloudBees provided all the capabilities we needed without having to build them ourselves. Given the complexity of engineering required for an organization as large as Markel, this was a huge weight lifted from our shoulders.”*

– Gurushyam Mony, Former Director,  
DevOps and Quality Engineering


CloudBees provided all the capabilities we needed without having to build them ourselves. Given the complexity of engineering required for an organization as large as Markel, this was a huge weight lifted from our shoulders.

## Piloting and Training

We rolled out a CloudBees CI pilot in April 2020. Our initial agreement covered 25 users who could spin up master servers and use the platform on Microsoft Azure using the power of Azure Kubernetes Services. This setup provided one immediate upgrade: We could now use Kubernetes to automate the creation of containers instead of constantly engineering new ones in Jenkins. The team had to refactor and re-engineer our approach to automation from the ground up.

The team had already launched centers for enablement and excellence within Markel for various training purposes. We didn't have a formal training process, though, and we'd been using our own compilation of ad hoc learning materials. Our CloudBees CI subscription gave us access to their expansive educational resources, including [CloudBees University](#) online classes and personal training from the company's experts.

When we rolled out the pilot, we began to enroll some of our associates in the CloudBees University courses right away. We also invited CloudBees solutions architects to sit in on our weekly dojo sessions and help us overcome challenges from



the app dev teams. It's one thing to learn peer to peer, but having your solutions provider sit in your training sessions takes education to another level. These dojo sessions were about more than learning the platform. They helped build a relationship with CloudBees, and all of our equal contributions were a major factor in our acceleration journey.

From the beginning, we felt that CloudBees was committed to our success. We knew we could reach out if we got stuck, and they would point us in the right direction—which saved us a lot of time and frustration. Our relationship with CloudBees is probably the strongest among all of our other partners and vendors, largely because these sessions fostered such excitement and honest exchange that we all moved faster.

## Our Air Lift Strategy

To make our move to the cloud, we started by securing our production environment and optimizing our storage and security. Within three months, we launched a beta version of the platform and opened it up to four teams. Our goal was to burden the teams as little as possible, so we asked them to focus on deliverables while we worried about moving their infrastructure to the cloud.

We referred to this as our 'air lift' strategy. We went to teams who were using Jenkins pipelines on-premise, and we moved their operations to the cloud. We made the transition as painless as possible by transferring their existing tools and processes from physical servers to Azure instead of forcing them to use an entirely new environment.


By shifting their activities to CloudBees, we lifted them to the next level, and we started to see some dramatic improvements in terms of build/deployment speed and throughput.

## Immediate Returns and Real-Time Visibility

Within weeks of adopting CloudBees CI, one of our teams began to use the power of CloudBees on Kubernetes to reduce the compilation time of our global reinsurance system's codebase. Every change to this codebase required a 25-minute build on Jenkins but was ready in just four minutes with CloudBees CI. Multiply that time savings by 30 teams and dozens of developers across four divisions, and we save thousands of hours every month.

CloudBees CI has also given us actionable metrics and real-time visibility into our processes. We can now see what's going on "under the hood" and use this information to prevent and resolve issues. When we were using an open source stack, our Jenkins platform would go down every Wednesday and Thursday. We saw the same incident ticket week after week, and we established a pattern of putting out the fire without the ability to fully understand the underlying problem as we were abstracted away from the infrastructure and had to rely on custom monitoring to find root causes.

As it turns out, one of Market's underwriting application teams generates tens of thousands of builds on Wednesdays and Thursdays, which requires massive Jenkins resources. We discerned the issue with CloudBees CI because we can now see how everyone is using the platform in real time, and we can spin up extra compute capacity on the fly to handle the extra load getting into more of predictive workload management for the build orchestration engines.



*"Every change to this codebase required a 25-minute build on Jenkins but was ready in just four minutes with CloudBees CI."*

– Gurushyam Mony, Former Director, DevOps and Quality Engineering

We didn't have this type of visibility or scalability before. Seeing it in action was eye-opening, and we were eager to push our CloudBees CI adoption far beyond the 25-user pilot project.

## Accelerating and Increasing Adoption

To accelerate and increase the adoption of CloudBees CI, we decided to use blueprints and patterns. Instead of chasing application teams and asking them to move to the cloud, we simply mirror their open source environments on Azure. We go into their codebase, abstract their build solution, add on our blueprint recipes, and all of sudden, they are operating on the cloud. We had to do this exercise for Java, Python and Node projects as well. These teams can then run their builds on-premise and on the cloud simultaneously to compare and contrast the two systems. Instead of touting the virtues of CloudBees CI, we let them see the platform in action and judge for themselves. Now, they come to us requesting upgraded features instead of us having to ask creating a 'pull' vs. 'push' model.


We began our transition to CloudBees CI from the bottom up. We went to our application teams, generated excitement through our Center for Enablement forums/meetings, and guided our developers toward those "Aha!" moments that let them experience the advantages of a cloud-based DevOps environment. That organic approach was successful, but as adoption grew, we embraced a top-down strategy as well.

Markel has dozens of legacy platforms that continue to generate substantial revenues for the company. The app dev teams managing these platforms sometimes don't get to spend time re-engineering nor realize the need to upgrade their tools, and sharing success stories does only little to sway them. Our strategy there is to firmly establish expectations: CloudBees CI is the new standard and we expect them to move to Azure next year. We want to focus on supporting one cutting-edge platform instead of splitting our time and attention between multiple systems, some of which are already obsolete given the recent explosion of cloud-native design paradigms.

## Philosophy in Action

Markel's application teams have quickly transitioned from a traditional on-premise Jenkins environment to a cloud-based containerized CloudBees CI platform running on containerized Kubernetes clusters. My DevOps and quality engineering team has created an easy-to-follow upgrade path for our app dev teams and they are now free to focus on improving their value streams, instead of integrating solutions into their pipelines.

We have modernized our DevOps environment and supercharged our CI/CD pipelines in a fraction of the time it would have taken using an open source solution, and we have leapfrogged a few steps along the way. Despite our size, we turned the DevOps philosophy on ourselves—and we're much better for it.



*"We have modernized our DevOps environment and supercharged our CI/CD pipelines in a fraction of the time it would have taken using an open source solution, and we have leapfrogged a few steps along the way."*

– Gurushyam Mony, Former Director, DevOps and Quality Engineering