**Interlude III. Second Attempt at Cloud Native**

*Jenny’s second wake-up call and WealthGrid’s attempts to reapproach their cloud migration from a more creative direction.*

Jenny’s first wake-up call was that WealthGrid needed to take action now on making the move to the cloud. Her second is that, unfortunately, her effort to lead a cloud native transformation using a small internal team (working entirely on their own, and only part-time) simply isn’t working. It’s not getting them ready fast enough to handle the arrival of a disruptive new cloud-powered competitor.

The trouble, she realizes now, comes from treating their cloud migration as just another tech changeover. They ran this project the same way they always handled, say, moving to a new database. This was an approach, it now becomes apparent, that was destined to fail. Going cloud native, she understands at last, requires so much more than simply lifting and shifting WealthGrid’s existing systems to run on the cloud.

It’s fine, though. They may have lost some time, but Jenny has learned a few things from this first go-round. Her team has gained experience as well. She is confident that a second effort—one that has more resources, more people helping out, even a dedicated budget allotment—will go right and go quickly this time. Based on the chatter she has heard from upper management, she is further confident that WealthGrid’s executives will support it.

Jenny gets to work preparing a very significant plan, complete with technology goals, documentation, cost projections, and all the many, many predictive details that WealthGrid execs like to see. The document takes her weeks to pull together. She must involve enterprise architects to create complex diagrams and meet with other program managers and department heads to get them on board too. Finally, however, Jenny creates a comprehensive vision for building a new platform and adopting new technologies—in other words, building cloud native like it’s meant to be.

She takes this plan, complete with deadlines and deliverables, to Steve, WealthGrid’s CEO. Steve is happy to see Jenny taking the initiative on a migration, because he also understands that to survive in this market, the company needs to evolve. In fact, the board has even had some preliminary discussions about whether it’s time for the company to move to the cloud, so this is good timing. Jenny seems knowledgeable, and certainly well prepared. She is given permission to launch her project, with a budget and a mandate for all of WealthGrid to support the initiative.

**Embracing Innovation**

Having learned that cloud native can’t simply be installed like some new software suite, or even a whole new operating system, Jenny has come up with a much more ambitious plan.

The original migration effort had a major block to progress: the team kept getting pulled away from building the new cloud platform in order to work on the current system. In response, Jenny has devised a “divide and conquer” strategy.

The plan is to divide their efforts. There is a legacy platform team, staffed with just a few people, to keep things going while everyone else works on getting the new system put into place. The legacy team will be delivering slowly, if at all, when it comes to any new features or functionality, because this is just a placeholder until the brand-new cloud native platform gets built. Any new upgrades to current functionality and all new features will be part of that. After all, it makes no sense to keep building things for the previous platform when it’s going away very soon. The general idea is that, once up and running on the new platform, teams will be able to produce so much and so fast that they will very quickly pay back any delay in deliverables.

Meanwhile the newly created cloud native team is of course responsible for building this new platform. It’s an all-hands-on-deck effort: WealthGrid is going all in on cloud native. There is a generous budget. A large crew of engineers—comprised, in fact, of most of the company’s developers and IT people, except for the skeleton crew keeping the current system going—has been mobilized to work on the initiative full time. This time, they have no constraints.

Jenny’s promise to WealthGrid’s board is that all of this will take six months.

**Experimentation Time**

The first three months go by quickly, with a lot of exploration. In typical Waterfall predictive style, the teams start by doing a lot of theoretical research and comparing features. They are trying to think ahead about all possible system architecture outcomes while simultaneously trying to guess all future situations that could arise. Their efforts are thoroughly documented, as is standard WealthGrid procedure.

No one really anticipated the sheer number of possible implementation combinations available, though! There are literally hundreds of projects and products available—more than 1,200 as of this writing, both open source and commercial vendor offerings—with more added all the time. The ecosystem is evolving so rapidly that the Cloud Native Computing Foundation offers a cloud native “landscape” that is updated daily to keep track of it all.

The result: 10 different teams have gone out to do research on the various choices available for public and/or private cloud implementations, and they are all doing it in different ways. No one is worried, though, because the idea right now is just to explore the best options for WealthGrid’s new platform.

It seems like things are going pretty well, actually—by now some of the teams even have a small application actually running in production on the cloud. Each team’s app experiment, however, is dramatically different: different clouds, different tools, different technologies—and there are seven of them. It doesn’t help that, because WealthGrid’s technical teams are organized by functionality, each team has built something that works really well for their highly specialized area of responsibility…if maybe not so well for the other teams. They are all working hard on the thing they think is right, even if it looks nothing like what the next group is doing. Of course they are each really proud of their version and believe it to be the right one for WealthGrid to adopt (remember the IKEA effect bias, where people value something much more highly if they assemble it themselves).

Once they realize the level of extreme divergence between efforts—and opinions—on the best platform, Jenny and the other project managers (there is one at the head of each team) try their best to work out a compromise.

Unfortunately, there is no way to standardize the various versions, even a few of them, onto one unified platform. They simply don’t share enough commonalities to be compatible. By the time all of this has been figured out, argued about, and argued about some more, two more months have gone by. They are nearing the project deadline, and the company’s executives are expecting to see some results.

A new argument arises: Why not just let each of the specialist teams use their own cloud implementation, then? They’ve always been able to choose their own tools before, at least within reason. We don’t want to waste all of this effort, after all (remember the sunk cost bias?). Ultimately, however, the Operations team makes it clear that there is no way they could ever possibly run seven different platforms, much less support seven different teams in production. That would result in complete chaos!

All of this takes yet more time to analyze, discuss, and debate. Eventually, Jenny and some other managers go back to the board and the CEO to explain why six months have passed but the promised cloud native platform has not been built. WealthGrid’s execs are supportive: this is a comprehensive initiative affecting the entire company. We can see that six months might not be enough time to get it all done. Take another six months and finish it up, we can wait a little while longer for you guys to do this right—but not *much* longer.

**Back to the Drawing Board**

So, back to the drawing board. This time, the decision is made to try a unified approach. The teams work with one of WealthGrid’s systems architects to make a plan and make sure that everyone is on the same page before anything else gets built. Microservices and Kubernetes are definitely going to be part of the implementation—this is one area where everyone is in firm agreement.

So the architect sits down to design a consistent plan for all the engineers on all the teams to implement Kubernetes running microservices. It’s a big diagram, with a very (very) detailed description of how it will be installed—writing the document alone takes four months before the tool itself can ever be installed. The senior managers are happy; they see a very good document, which to them represents very good progress.

Great, right? Well…Now the next step is for everyone to attempt putting into place a plan for Kubernetes that was written by someone who has never actually used Kubernetes. He doesn’t even understand that he doesn’t understand—though this does not stop him from creating and presenting a complex diagram with the very strong conviction that this is how it should work (this is a classic illustration of the cognitive bias known as the Dunning-Kruger effect, or the tendency for unskilled individuals to overestimate their own ability).

Except that it doesn’t work. Everyone is astonished: here are microservices, there is Kubernetes, those are the core cloud native pieces. So what is wrong?

Yes, you have an impressive document, a tool installed, and maybe even a nice demo for upper management. But what exactly are you showing them? Kubernetes is included, but the one application the cloud native team (which at this point, remember, consists of most of WealthGrid’s engineers) was finally able to get running on it is simply terrible: full of holes, no security, no automation. It’s in a container but still falls apart every two weeks.

Everything is configured wrong. It turns out that even these so-called “full solution” platforms still require a great deal of complex initial configuration, so even this small platform they have built is very difficult to maintain. There is no preparation for production, no monitoring, and no attempt to create the continuous integration and continuous development processes and practices necessary to effectively develop a cloud native app in the first place. The developers don’t know how to use it. The ops team is frankly afraid of it. And everyone is complaining because instead of fewer tasks, there are now so very many more (because the new system is much more complex, but zero automation has been put in place to handle that complexity).

At this point maybe 2% of the implementation is complete. The cloud native team can claim that they do have something running on K8s, but realistically nothing is actually delivered. They tried building a microservice architecture, yet there is not a single microservice in production even though six months have, once again, gone by.

Managerial trust in the engineering team is at an all-time low. The engineers feel this is extremely unfair because they have been working very hard to build the thing they were told to build. They’ve always succeeded before in delivering technical projects, and they are truly surprised that they can’t seem to finish this one.

The problem is, after a six-month extension on the project, it is now one year since the full-company cloud native initiative launched—and two years since Jenny first began trying to shift WealthGrid to the cloud in the first place. The full-company cloud native team is nowhere close to being finished. Maybe 30% of the platform is delivered, and it’s not even close to production ready.

Meanwhile, the skeleton team left in charge of WealthGrid’s original platform has just been maintaining the status quo. They have been holding off on delivering features—features that customers are demanding—because they were waiting for the new system to come online. They wouldn’t have been able to deliver a meaningful amount of new work, anyway, because so much of the team was moved over to building the new system.

**Third Time’s the Charm?**

WealthGrid can’t wait any longer: they *have* to deliver these delayed features to the market as soon as possible or they risk losing customers. Customers expect new functionality and constantly increasing quality of experience; if they don’t get them, they will simply move over to one of WealthGrid’s competitors.

At this point the executive team and the sales team have so much pent-up pressure coming from the market that they finally lose patience. They summon Jenny to a meeting and deliver the following message:

“You know what? We’ve had enough. You promised certain deliverables in six months and this hasn’t happened. We gave you six more months and you still couldn’t deliver. You can’t even tell us what will happen in the *next* six months.

“We know this is complex stuff and we still trust you to deliver this platform, but we really, really need to get these backlogged features out to the customer. We don’t care where they run, but we *must* have these five features out in the next three months. What are you going to do about it?”

Jenny’s third and final wake-up call has arrived.

The message from WealthGrid’s leadership is crystal clear: show us value in 90 days. If she can’t deliver, Jenny could very well lose her job. Her engineers are going to be extremely unhappy as well, since they would have to stop working with this cool new tech and go back to churning out features on the boring old legacy system.

But what to do? There is no practical way to get these five do-or-die features out on the new cloud native platform, because the new platform is not production ready and no one really knows when it will be ready. So they have to do something else. The problem is, *what*? First they tried doing the new system off to the side while continuing to roll out the current system features. That didn’t work. Next they tried going all in, sidelining the current system to focus on building a brand new one as fast as possible. That didn’t work either.

**What else is there?!**

*This is the point where cloud native consultants often get called in to rescue a stalled or failing transformation. But even summoning external experts won’t always help much if a client organization doesn’t take the necessary global approach. That is, consultants who primarily offer Kubernetes knowledge will indeed get a platform up and running pretty quickly*—*but the company will still struggle with the other essential pieces of the cloud native puzzle: microservices architecture, team structure, DevOps, hierarchy, culture, and so on. So, WealthGrid needs to be careful not to call a consultant that will only address the tech aspect of its situation. However, also calling a management consultancy to help with the organizational and process side of things would probably not help the company much, either, as that scenario will almost certainly involve a massive transformation plan that will cost millions and take years.*

*We don’t want to leave you hanging too much (and also this is a book about patterns). So suffice it to say there is a path to success, and we plan to tell you all about it. Coming up next in*[*Chapter 6*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch06.html#tools_for_understanding_and_using)*we will present some tools that will help Jenny solve the problems keeping her team from delivering a functional cloud native platform in a timely fashion. The chapters after that introduce the patterns themselves. Then, in*[*Chapter 11*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch11.html#applying_the_patterns)*and*[*Chapter 12*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch12.html#applying_the_patterns_a_cloud_native)*, we will demonstrate a successful pattern design for actually delivering the system in 90 days*—*while also delivering the five do-or-die features WealthGrid needs now.*