HENTSŪ
SYSTEMATIC AGILITY

Refactoring Legacy Applications for the Cloud

Uros Pavlovic (Technical Writer): To kick off, can you explain a bit more about the basics of the refactoring legacy applications for the public cloud and how vital this process for modern-day business.

Des Holmes (co-CTO): Yeah, sure, I think that when we're talking about migration and refactoring applications (and that's what a lot of our clients come to us for), the three main types of migration that we look at is a sort of lift and shift – taking an existing application and moving it straight to public cloud. The second one is minor refactoring; so, when our clients have a modern application, we adjust that to use cloud native services. Finally, on the other end of scale, we've got large rewrites or total rewrites of incredibly old applications that are not suitable or have an end of life wherever they are currently hosted.

So, those are the three basic types of migration. In terms of the basics of refactoring, you need to take a few things into considerations, one of which is just evaluating the value to the business of refactoring the application. Just because you want to move or migrate an application from on-prem or private into public cloud, does not necessarily mean that you might need to refactor it.

You need to consider how long are you going to be using the application, when is its end of life, and understand the amount of effort that might go into refactoring the application. So, if it is something that is going to be around for a long time and it does add a lot of value to your business, then it is worth considering. However, if it is a legacy application that you're looking to phase out in the next few months, or year or so, then refactoring that to be compatible with cloud native services, probably isn't a worthwhile investment of time and effort.

> Another crucial thing to note, if you can iteratively refactor an application, you should definitely do that as opposed to doing a sort of 'big bang' – just so you can understand how the application behaves in cloud and you can start getting the benefits of that refactoring, rather than waiting several months before you release it.
>
> *- Des Holmes, Hentsū CTO*

UP: From your words, I guess you could say, that this is something that's here to stay?

DH: Yeah, absolutely. And like I say the scale of it could be small applications that get moved or it could be big enterprise applications, that are core to the business, that need to be migrated and refactored as well. So, we talked about it briefly before, but refactoring is a natural part of any development and life cycle – it is just how and when you should do it.

Refactoring is a natural part of any development and lifecycle – it's just how and when you should do it.

- Des Holmes, Hentsū CTO

UP: Right, and basically businesses and companies knowing the nature of their business and where they're at and having a general view on what they should do specifically.

DH: Absolutely.

UP: Let's dig into it a bit more by talking about these enterprise-level facets of refactoring and the chief benefits.

DH: The key thing, especially on an enterprise level, is to leverage the development investment from the public cloud providers. You know they're spending billions to make your life easier and cheaper. I'd consider that as one of the big benefits of the process. The other thing to consider is reducing your overheads and focusing on your core business as well. So, if you've got a legacy app or elements of your business on an enterprise scale that is causing you headaches, and you're constantly having to look after it and monitor it, then why not off-set that and use cloud native services instead and actually focus on your core business? Then again, if DevOps is not a core part or core enough part of your business, then just look to off-set that and leverage cloud services – so, you don't want to be running endless patching cycles, or worrying about the infrastructure itself. Make that somebody else's problem, and in this case it's the public cloud providers.

UP: Yes, and I think that neatly ties into the topic, or a question, whether there are benefits of the public vs. the private cloud environment.

DH: Definitely. I think that the three main things are the scale, and opex vs, capex, but also considering the amount of access to the tools and services that you've got. We help clients scale up and down using public cloud, so it's not always about "I need to infinitely scale," which is obviously what public cloud offers, but also if they're looking to reduce their capex they can move to public cloud and dial the services down or, ideally turn them off, if they're not being used. That just isn't possible with your private cloud model where you're paying for the resources and when they're on, that's it, they're on the bill.

UP: Yes, and you're sort of stuck with that.

DH: Yeah, you're totally stuck with that, which is annoying because if I was going out shopping for my food and I didn't want something one week, I just wouldn't put it in my shopping cart. Whereas if I had to pay for the same food every week regardless of whether I was going to use it or not, I'd be really annoyed with that.

UP: It makes complete sense, of course. You are free from that long-term obligation that you're locked into.

DH: Yeah, absolutely. And it's great, the scale and the number of tools and the services that the public cloud providers are developing. But on the other side there's quite a lot of fatigue as well, because it takes a lot of time and effort to keep up-to-date with all the new tools and services, just to make sure you're using the right ones.

UP: Okay, let's jump on ahead a bit, can you talk a bit more about serverless application refactoring?

DH: Yeah, sure. When we refer to 'serverless' we're just talking about not going with the traditional route of having a virtual machine or server where something is hosted and deployed to. In Hentsū we are constantly using the strength of the public cloud to carry this out internally, while relying on powerful tooling such as Azure App Service, AWS Lambda, AWS/Azure Batch, ECS and so on. All these things are available and able to deploy your applications too and turn them on and off as well.

So, the good thing about serverless is: "I'm not maintaining a server, I don't have the patching cycles and headache with that. But also, I don't necessarily need long-running instances." So, with things like Lambda and Azure Batch, you only pay for them as long as you need them; rather than having to pay for a server which could be 300-500 dollars a month, you're going down to sense in terms of running the processes themselves.

UP: Right, and this has been a hot topic for us as of late; this kind of serverless ambiance and grid computing, ephemeral computing is really taking center stage.

DH: Totally. It is big. With these services, you cannot just throw anything at them; you need to have highly potent processes as well as these nicely ephemeral services that you can just turn on and off. And, in some cases to inject failure into on purpose just to make sure things are behaving as expected. But yeah, it's the way forward, I think. It's no longer a question why you need a virtual machine turning on now.

UP: That totally puts things into perspective, yeah. And I think another really big thing I personally noticed with all the big-tech companies, and those who are going for cloud adoption, is automation, apart from scaling and auto-scaling. I think in its current state, the cloud is a cool environment to have. But without broadening the topic too much, we can leave that for another discussion. Anything else you'd like to add while we're talking about legacy applications?

DH: Yes. One point, just to mention quickly. How tech and the application landscape is changing. I think the good news now is, if people start developing applications they won't have the same problems in future when it comes to refactoring, because they're going to be using things like Docker and they're going to be breaking out APIs from the UI. So, they won't be having the same problems they're encountering now.

I think the tech's gone to a mature state, and there are established patterns, which means you won't potentially have the same expensive refactoring or large-scale refactoring that's required at the minute. There's a lot of legacy applications written on old tech that unfortunately were unsuitable, so it's just something to consider. There is some solid tech out there to consider helping future-proof yourself more than was possible before.