HENTSŪ
SYSTEMATIC AGILITY

Cloud Adoption Decision Making

REHOST, REFACTOR OR REBUILD

Uros Pavlovic (Technical Writer): To recap on what we've been chatting about in our previous interview (Re-factoring Legacy Applications for the Cloud) what are the best steps for firms (large hedge fund or financial services) to prepare for that?

Des Holmes (co-CTO): I think it's important for firms to evaluate the business value of refactoring. That involves things like understanding the effort to refactor, and not just cost but man hours that go into that and to understand how the application could be broken out into smaller components and easily separated into external services. As part of that is defining comparable metrics – so if you've got things like costs, and performance, comparing your legacy state to your new state is a good way of understanding what the business value is. And there's probably two more things two consider: to understand the target services and architecture and as we mentioned the last time to iteratively re-factor (avoiding big bangs) – if you can break the components out and utilize external services, it's certainly worth considering.

> If you can break out your legacy applications into smaller components and get those to use the native services in an iterative process, in terms of an effort reduction and incident management and stress level, you get less maintenance overheads, because you have the benefits of the public cloud providers looking after the cloud infrastructure.
>
> *- Des Holmes, Hentsū CTO*

UP: Yes, I totally get that. Breaking up the components is a step in the right direction then. So, moving on. When we talk about migrating legacy applications it, typically, involves a ton of associated data, security, and networking configurations, the powerful underlying infrastructure and serverless tech. This time around, I'd like to dig in a bit more into the specifics of serverless services.

DH: Sure. Serverless computing allows companies to purchase cloud services 'pay-as-you-go' model, which is different to the capex private cloud model. With serverless there's no server/VM to manage. This is handled and offset with the public cloud providers. There are plenty of serverless services, which provide scale and excellent cost reduction (compared to the legacy VM-based approaches). A few that are worth mentioning are **AWS Lambda** or **Azure Functions**. They scale up and down to zero as you need them. These are great for short-running tasks, anything around automation, small ETL jobs and they are even used now for the backend of API endpoints. So, they are excellent.

AWS and **Azure Batch** are used by clients for scheduled tasks, (again: pay as you go), ETL processes. These allow you to focus on the code, rather than the patching or the servers. One more thing that comes to mind is **Azure App Service**, which is excellent for hosting the frontend and backend development of web applications, and for handling authentication between services. That is great because it means there's one less problem to worry about.

UP: Yeah, it sounds like there is a flow to it, so these tools and services are all interconnected in their roles. They sort of help smoothen the process, if you will.

DH: Totally, yes. Plus, you get the scalability, the cost reduction and a wide range of additional benefits of serverless technology.

UP: Continuing with our main theme, the serverless approach to migration. Tying into what you said previously, it's possible to replace the components of the application with serverless versions of them, one at a time, so does this mean that simultaneously users utilize a kind of hybrid version of the application? Also, let's dig into the advantages of this methodology.

DH: I think that's a good point. We mentioned it earlier. It's just getting the benefits sooner, I think. So, if you can break out your legacy applications into smaller components and get those to use the native services in an iterative process, the kind of benefits you're gonna get is obviously, cost. Rather than needing long-running VMs, you've got ephemeral services, disposable services as and when you need them. So, if you're not paying, they get turned off. Also, in terms of an effort reduction and incident management and stress level, you get less maintenance overheads, because you have the benefits of the public cloud providers looking after the cloud infrastructure. And you can scale. As well as going down to zero, which is great.

Again, you cannot do that within the private cloud, because it's a capex model and this is an opex model, where you talk about the top end of the scale, which is mind blowing in certain cases. However, at the lower end of the scale you just turn them off altogether. One more important thing to mention is that if you pick the services off, and refactor your application in smaller parts, you also reduce the learning curve. You need to get used to them as an organization. It's not going to happen overnight. The iterative process just creates less headaches and it's going to be easier to onboard for sure.

UP: What you're saying is that it is a bit of a challenge at first. But once you're set up, you're good to go you're fired up.

DH: Yes. Well, also, there is a bit of fatigue when it comes to learning the services; you've got to get them up to speed, you've got to bed them in. You need to get used to them as an organization. It's not going to happen overnight. The iterative process just creates less headaches and it's going to be easier to onboard for sure.

UP: Sounds pretty good. So, what I'm interested in now specifically is how does Docker tie into all this and what are the main benefits? So, we're talking about code right now.

DH: As you know, here at Hentsu we prefer to have everything `Powered by Code`: using Infrastructure as code tools like Terraform and Ansible. A natural extension to this, and utilizing serverless technologies, are solutions like Docker. If companies haven't used Docker before, they ought to be aware that it is a real game-changer. It provides immutable infrastructure with the ability to reliably execute your code in a repeatable way across any environment. Whether that's a local development environment, in a cloud service, in a private cloud environment, on a mobile device - it's write it once, run it anywhere.

> If companies haven't used Docker before, they ought to be aware that it is a real game-changer. It provides immutable infrastructure with the ability to reliably execute your code in a repeatable way across any environment.
>
> *- Des Holmes, Hentsū CTO*

As I said, it's immutable, so you can push that through various environments and they all behave in the same way. It reduces a lot of the overhead, which you have with legacy applications and the legacy release processes. It's an essential tool and removes all the legacy issues around dependency management and inconsistencies between environments. We just don't hear developers say "Well it worked in Dev" when using docker. It also allows for rapid bootstrapping of environments locally by using the compose functionality to stand up entire application stacks with a single command. So, it's really powerful and honestly, hand-on-heart, we've not really gone back and would not consider going any other way, other than the Dockerized approach for any application development. Once you've used it, there really is no turning back.

UP: Yeah. If I'm not mistaken, it is a component that dramatically improved our own workflow and added more to the automation, right?

DH: Yeah, definitely. It's a huge benefit.

UP: Cool, that's a pretty big deal. Next up, all of this is basically an effort to replace the monolithic API approach correct with more powerful and flexible cloud environment, correct?

DH: Yes, there are really good services provided by the likes of AWS, specifically around like API gateways and management. The good thing is rather than having a monolithic API, which written in one tech and one language and deployed often in a complicated way to provide scale, Amazon API Gateway and Azure API Management allow you to mix and match legacy APIs with new APIs and present them all behind a single consistent API layer.

One of the great benefits of this tool is that it allows you to iteratively refactor your applications. Using these API services allow you to gradually replace elements of your legacy monolithic API into micro services that are powered using serverless technology. This brings allows you to then use cloud-native services like Azure Functions or AWS Lambda to power parts of your API. So, they are very powerful and low scale and even at high scale very cost-effective too.

UP: Great, so it sounds like there are a lot of benefits there actually.

DH: Yeah, huge. It's a real game-changer.

UP: So, let's wrap up with looking at our internal process a bit more, because we've had a lot of experience with this. How has the public cloud helped Hentsu with deployment? Let's take a moment to talk about DevOps - CI/CD pipelines, Bitbucket and Bamboo and the helpful tools we're all used to.

DH: Yeah, there are lots of services which work well together to create CI/CD pipelines. Internally we use a combination of the Atlassian stack, Azure DevOps and GitHub to manage application development, network and service state management and a lot of the automation. It depends on the use case, but it is good not to be stuck with one tool and to mix and match services that suite the organization. So, getting your CI/CD pipeline setup to automate and ensure quality should be a key focus and will prove invaluable in the long run.

> “
>
> It is good not to be stuck with one tool and to mix and match services that suite the organization. So, getting your CI/CD pipeline setup to automate and ensure quality should be a key focus and will prove invaluable in the long run.
>
> *- Des Holmes, Hentsū CTO*