MSPOG – Accepting the Reality of Multiple Single Panes of Glass

You probably dread the phrase as much as I do. We hear it all the time on a sales call or a product demo: “this is the single pane of glass for you and your team”. The problem is that I’ve been working in the industry a long time and have been using a lot of single panes of glass…at the same time. Many of my presentations have been centered around the idea that we must embrace the right tool for the right task, and not try to force everything through one proverbial funnel because the reality is that we cannot do everything with any single product.

For this reason, it’s time to embrace MSPOG: Multiple Single Panes of Glass

Many Tools, Many Tasks, One Approach

Using a unified approach to something is far more important than the requirement to using a single product to do it. I’m not saying that you should just willy nilly glue together dozens of products and accept it. What I am saying is that we have to dig into the core requirements of any task that we performa and think about things in a very Theory of Constraints (ToC) way. Before we even dive into some use-cases, think about what we are taught as architects: use the requirements to define the conceptual, logical, and then physical solution. All the while, understanding and making our decisions based on risks and constraints.

If you have a process that requires two or three different processes within it, you may be able to use a single tool for those processes. What if one of the processes is best solved with a different tool? This becomes the question of the requirements. Is it a risk if we embrace a second tool? More importantly, is it a risk or a constraint to use a single tool? This is the big question we should be asking ourselves continuously.

Imagine a virtual machine lifecycle process. We need to spawn the VM from a template, give it a network address, deploy an application into it, and then make sure it is continuously managed by a patch management and configuration management system. I know that you’re already evaluating how we should do this at the physical level by saying “use Ansible!” or “use Puppet!” or “use vRealize Automation!”. Stop and think about what the process is from end-to-end.

Our constraints on this is that we are using a VMware vSphere 6.5 hypervisor, a Windows 2016 guest, and using NGINX and a Ruby on Rails application within the guest.

1. Deploy a VM from template – You can do this with any number of tools. Choose one and think about how we move forward from here
2. Define IP address – We can use vRO, vRA, Puppet, Chef, or any of a number of tools. You can also even do some rudimentary PowerCLI or other automation once the machine is up and running
3. Deploy your application – App deployment can be done with something like Chef, Puppet, or Ansible, as well as the native vRO and vRA with some care and feeding
4. Patch management – Now we get more narrow. Most likely, you are going to want to use SCCM for this one, so this is definitely bringing another pane of glass in
5. Configuration management – Provided you use SCCM because of the Windows environment, you can use that as well for configuration management…but what about the nested applications and configurations such as websites and other deeper node-specific stuff. Argh!!!

Even if you came out of the bottom of those 5 steps with just two tools, I would be thinking you may need to reevaluate because you have have overshot on the capabilities of those two tools. It is easy to see that if we start narrowing to a single pane of glass approach, that we are now jamming square blocks into round holes just to satisfy our supposed need to use a single product.

What we do need to do look for the platforms within that subset of options that has the widest and deepest set of capabilities to ensure we aren't stacking up too many products to achieve our overall goals.

The solution: Heads up Display for your Single Pane of Glass

Automate the background and display in the foreground. We need to think more about having the proverbial single pane of glass be a visible layer on top of the real-time activity that is happening. Make your toolkit a fully-featured solution together with focus on how you can do as much as possible within each product. Also, reevaluate regularly. I can't even count how many times i've been caught out by using something a specific way, only to find out that in a later version that the functionality was extended and I was using a less-desirable method, or even a deprecated method.

There is a reason that we have a mainframe at the centre of many large infrastructure shops. You wouldn't tell them to shed their mainframe just to deploy all their data on NoSQL, right? That would be lunacy. Let's embrace our Multiple Single Panes of Glass and learn to create better summary screens to annotate the activity. This way we also train ourselves to automate under the covers and trust the underlayers.

I, for one, welcome our Multiple Single Panes of Glass.

Image source: https://hudwayglass.com