WHITE PAPER

Automating Your Way to Simplified Application Management

By Nick Cavalancia

TABLE OF CONTENTS

puppet labs®

In today's data center environment, the toolsets you have may not be sufficient as you move forward.

Maintaining applications used to be a lot easier and less time consuming – a simple matter of installing and updating every few years. But things are different today. Applications now follow a development paradigm where software iterations happen far more quickly; now management is pressuring IT to address business concerns and market opportunities that can be helped by faster application deployment.

But you're already concerned with more than just implementing or updating applications; your eye is on the bigger picture of maintaining your IT environment's operational readiness as well. You want to ensure the data center itself is well updated and ready to deploy and maintain those applications.

Because it involves supporting and maintaining so many tier two and three applications, this modernized approach is as concerned with time-to-value as it is with time-to-deployment. And this has some of you looking at new ways to provide these same applications at a lower cost.

The need to lower the expense of maintaining a home for them with the option of additional capacity has many looking to the cloud. But, you've already invested in toolsets, staffing, and education, so you need to take advantage of those, whether on-premise or in the cloud, to see a return.

Others of you are thinking more along the lines of new DevOps initiatives, potentially changing how you approach application development, deployment and operations altogether, taking a more agile approach.

In either case, there's a bit of politics somewhere in the discussion, as well as conversation around which technologies will help you solve your application deployment and maintenance problems. As if there wasn't enough pressure already…

In today's data center environment, the toolsets you have may not be sufficient as you move forward. And the processes you have in place for the configuration of machines, operating systems, and applications are probably using older technologies and methods. For example, you may be relying on scripts written by someone who is no longer part of your IT organization.

So, you've got to understand the resources at your disposal and will need to determine how you will maintain them going forward. And if you're using golden images, a valid methodology, to maintain your state, but the number of golden images starts to proliferate, you'll find yourself with an image maintenance problem as well.

Regardless of the mix of approaches and legacy tools you're using, you may just be adding complexity to the work you do today, while not being adequately prepared to iterate software more quickly so you can get applications to market faster.

So, how should you go about addressing these concerns and new initiatives, and the opportunities you have to streamline application management?

By looking to adopt and expand the use of automation, IT spends less time fighting fires.

Taking an Automated Approach

To keep up with a data center environment that is, in essence, in a constant state of flux, it's going to take an IT paradigm shift from legacy one-off toolsets to a focus on repeatable automated processes. By looking to adopt and expand the use of automation, IT spends less time fighting fires, and focuses more on the goal of rapidly bringing internally developed software to market or business-critical applications into production.

This also means your IT team can easily address ongoing management needs in the same fashion, using a catalog of automated tasks to keep your virtual and cloud infrastructures, your network and storage devices, your machines and the software that's running on them all in a consistent state.

With automation, you define the state you want your systems to be in. You determine what applications, what packages, what services, and what settings you want your data center to have across different lines of business. You enforce the deployment of those applications and maintain their state in your environment.

By increasing your ability to perform rapid and repeatable changes in your data center and with your applications, and then forcing consistency into the state of that environment, your application time-to-market is decreased and IT productivity remains high, all while the organization remains updated, operational, and secure.

So what are the necessary parts of an automation strategy to simplify application management?

Automation is more than just having a few scripts lying in wait to execute a task. For automation to truly have a positive impact in your IT organization, it's going to have to have a few capabilities that will require either some additional work on IT's part, or a third-party solution.

Automation is more than just having a few scripts lying in wait to execute a task.

Context

Automation isn't a script you run across the board for every location, every server, and every application. Your environment is far too complex for that. For automation to be both efficient and effective, it needs to be paired with some level of context each and every time it's used. Without context, you have no ability to target specific applications, machines, and even the conditions where it's appropriate for the automation to complete the task at hand.

So, what kind of context is necessary for successful automation?

Simple applications running on a single server pretty much establish their own context. But when you're looking to deploy or keep a set of multi-tiered applications (e.g.: a three-tiered application with a front-end application server, a load balancing middleware server, and a backend database server) consistently updated, you need to consider the specific needs of each application tier, as well as the dependencies between them.

Some applications – especially those internally developed – can easily have technical dependencies: compatibility constraints require services on another server be already running at startup, or the validation of credentials on another system before continuing. It's these kind of dependencies you'll need to ensure your automation methodology takes into consideration as an integral part of the automation itself.

Also think about the entire set of servers that make up the context of the multi-tiered application. That, too, may need to be deployed, configured, and/or updated as a single entity. This would further require that you have the ability to define which components are dependent upon each other to ensure, for example, the database server is spun up before the front-end application is.

To achieve this level of contextual automation, it's obvious you'll need more than scripting alone. Without a third-party solution, you'll have to have a tremendous amount of documentation that keeps track of every OS and application configuration, technical and operational dependency, or specific requirement therein.

To achieve this level of contextual automation, it's obvious you'll need more than scripting alone.

Even with an ability to establish which automation should impact what servers and applications, in an enterprise where operational efficiency is critical, you can't simply roll changes out and impact production without simulating the automation first. This is why a proper automation methodology includes *modeling*.

Modeling

Automation is little more than a set of tasks to be completed. So, when you plan on implementing applications and/or servers via automation, it's important to realize each of the environments in which you may use automation may be very different. Think about it: your data centers, development and production environments, as well as your private and public clouds are *clearly* not identical environments.

So, how do you ensure your automation will work consistently, no matter the target environment?

In order to have confidence the changes you're pushing out into production will not only be consistent even in heterogeneous environments, but also will be successful, you'll want to use modeling to define every aspect of the environment you want automation to create. Think infrastructure, virtualization, network storage, middleware, applications, and configurations. *Everything*.

Whatever you are going to want deployed, created, installed, updated, *Once you have models in place, you don't want to keep it all in the hands of only a few.*

or configured as part of the actual automation should be part of the model. The goal is to model once, simulate the automation, and then sit back, knowing you can deploy applications (and their supporting environments) anywhere.

Unfortunately, this is one aspect of automation where you're not going to be able to adequately find a "do it yourself" solution. In theory, you *could* cover this with quality assurance-like testing in various environments, but that would take up so much time and resources, it really defeats the purpose of automation.

Once you have models in place, and context established to provide the right automation based on the needs of the organization, you don't want to keep it all in the hands of only a few. What you need is the ability to appropriately allow IT and non-IT users alike to kick off automation requests themselves.

Self-Service

Automation is as much about scale as it is speed and consistency. The whole idea of automating a process is so you can do run the process repeatedly and at will *without needing to do an equal amount of work each time*. That line of thinking shouldn't be limited to the task itself, but also to the *launching* of the task.

Your IT organization has too many other responsibilities beyond installing and implementing applications to spend its time taking and launching automation requests from users, line of business owners, and cross-functional teams.

For automation to scale, the initiating of automation needs to be shifted from the hands of IT and placed in the hands of the users themselves. It's the appropriate move for an IT organization that's transitioning from a team that reactively provides services (including automation) ad-hoc to their users, to an organization proactively providing *IT as a service* (where automation is one of the services offered).

It's a slight, but very important differentiation. Users today are looking to move, in some cases, faster than you can accommodate. By placing automation of application management into a user's hands, complete with a set of appropriate controls, you can focus on building the output of the automation as a service, and leave the utilization of that service to the user base.

Ideally, users need a self-service catalog – a task automation app store of sorts – to centrally store the application-related tasks (and supporting infrastructure-related tasks necessary, as is appropriate) available to them. Now, this obviously wouldn't be an "everyone can launch everything" kind of service catalog. The right execution of this aspect of automation would leverage permissions, entitlements, and workflows to define which automated tasks are visible, can be launched, and require approvals before running.

Ideally, users need a self-service catalog to centrally store the application-related tasks.

This catalog isn't something you can easily build yourself, but it's necessary to mention it as part of automating application management, as it's the next logical step once you have a mature level of automation in place.

Putting it all together

In enterprises where applications – from the simple copy of Office to the complex rollout of a virtual testing environment – are an everyday necessity, automating application management – from deployment, to configuration, to updating – takes on a new level of intensity. IT will either fail, succumbing to the weight of the sheer amount of work needed to fulfill these daily requests, or find a way to automate not just the *doing* of the work, but also the requesting, approving, and initiating of the work.

In some ways, by automating not just application management but also the creation and configuration of the necessary underlying components, servers, virtualization, and storage needed to support an enterprise application, you are going farther than just offering "applications as a service". In some cases, you're offering "infrastructure as a service".

The challenge will be deciding whether to do it yourself, or seek out a third-party solution. A homegrown option will require a multitude of scripting languages that can, in total, deploy and configure various operating systems, spin up clusters, create new virtual machines, deploy applications, patches and updates, and can probably handle additional tasks you haven't thought of yet. A third-party solution will need to address these tasks, as well as provide the context, modeling and self-service needed.

By establishing an automation methodology that contextually models, advertises, and deploys application management tasks, you will achieve the faster time-to-value and quicker time-to-deployment you're looking for, and will be able to provide the same applications to your organization at a lower cost. ■

With nearly 20 years of enterprise IT experience, Nick Cavalancia is an accomplished consultant, speaker, trainer, writer, and columnist and has achieved certifications including MCSE, MCT, MCNE and MCNI. He has authored, co-authored and contributed to over a dozen books on Windows, Active Directory, Exchange and other Microsoft technologies. He has spoken at conferences such as the Microsoft Exchange Conference, TechEd, Exchange Connections, and on countless webinars and at tradeshows around the world