VMware Cloud Management

(http://blogs.vmware.com/management)

Insights into managing your virtualized datacenter

Why Model-Driven Application Provisioning is Better than Workflow Deployments

Posted on January 2, 2013 (http://blogs.vmware.com/management/2013/01/why-model-driven-application-provisioning-is-better-than-workflow-deployments.html) by vFabric Team (http://blogs.vmware.com/management/author/klambert)

The goal of the Software-defined Datacenter (SDDC) (http://www.vmware.com/solutions/datacenter/software-defined-datacenter/index.html) is that every datacenter function can be delivered as a service on any cloud at any time. At a basic level, this implies that these services need to have a high level of reuse in order to truly be multi-cloud capable and fit into the SDDC mantra.

Much of the datacenter is being reinvented to fulfill the promise of this new cloud era. One fundamental piece that is ripe for change is software deployment. Scripted or workflow-based deployment methods are out, and model-driven provisioning is in. This article will explain why this shift is necessary and why you should be rethinking your software delivery process. Workflow-driven solutions cause:

Tight coupling of application, deployment, and environment limits app portability and increases the amount of work and maintenance for hybrid clouds

Poor alignment between the development and infrastructure teams

Lack of automation and increased rework in an already error-prone process

Instead of workflow-driven installation and deployment scripts, we should be looking for solutions with both workflow AND model-driven architectures in our deployment automation. Models do make a difference.

What Is the Difference Between Model-Driven and Workflow Driven Deployments?
Today, the most dominant strategy is a workflow driven approach based on scripts. This is where the process of "how the application needs to be deployed" is defined, and a series of steps in the workflow are executed to deploy the entire application. Eventually, this approach is painful because the trend for today's developer is to build automatic scale into their apps and use a strategy of hybrid clouds to gain flexibility, continuity, and/or lower costs. With multiple cloud platforms in mind, a workflow based solution means you need a unique workflow per cloud. This is hard to build, let alone maintain across dozens of applications and environments!

Since the scripts are unique to each combination of workflow, environment, and app, we call them "tightly coupled." As an example, apps often need to be updated after an initial deployment. Typically, this means the workflow scripts need to be updated as well. In fact, the workflow may need to be rewritten several times throughout the lifecycle of the application — any time the application, middleware, or components change.

Additionally, each time you change something about the target environment, your deployment plan and workflow scripts should change and be retested. Why? This is because "tightly coupled" makes your deployment script brittle — if anything changes, the workflow script needs to change with it. With tight coupling and a volume of permutations, work increases exponentially for script developing, testing, and maintenance.

Instead of tight coupling, we should look in the opposite direction toward abstraction (http://en.wikipedia.org/wiki/Abstraction_(computer_science)) . Abstraction has been used repeatedly in computer science to solve this exact issue. (As a side note, look at how abstractions are revolutionizing (http://opennetsummit.org/talks/shenker-tue.pdf) the world of networking.)

When we separate the elements and the actions, we can change individual pieces without the need to touch or change all pieces. Abstraction is essentially what allows us to create a model-driven architecture, where we can describe all of the components logically. In this model-driven architecture, users can:

Change the **environment** (e.g. the underlying cloud, data center, VMs, OS, IP, etc.) without changing any script information about the **application** or **workflow**.

Change the **application** (e.g. the app server port or version) without changing any script information about the **environment** or **workflow**.

Change the **workflow** (e.g. add a load balancer, app-tier, or data-tier node) without changing any script information about the **environment** or **application**.

What happens when you have to change one of the items above for many nodes at once? Usually, it means the workflow script will need to be rewritten, retested, and finally redeployed. With a model-driven architecture, you just change one piece without re-doing multiple, similar scripts. This means less

work, greater app portability, less maintenance, and faster deployment.

Aligning Teams by Making Information Collaborative

Since the workflow driven approach is unique to each app and cloud deployment, it does not offer the opportunity to align teams around the same toolset. In fact, it encourages individual teams to maintain their own catalog of custom-built scripts and prevents the rest of the organization from benefiting from their investments. This is not the case with a model-driven architecture.

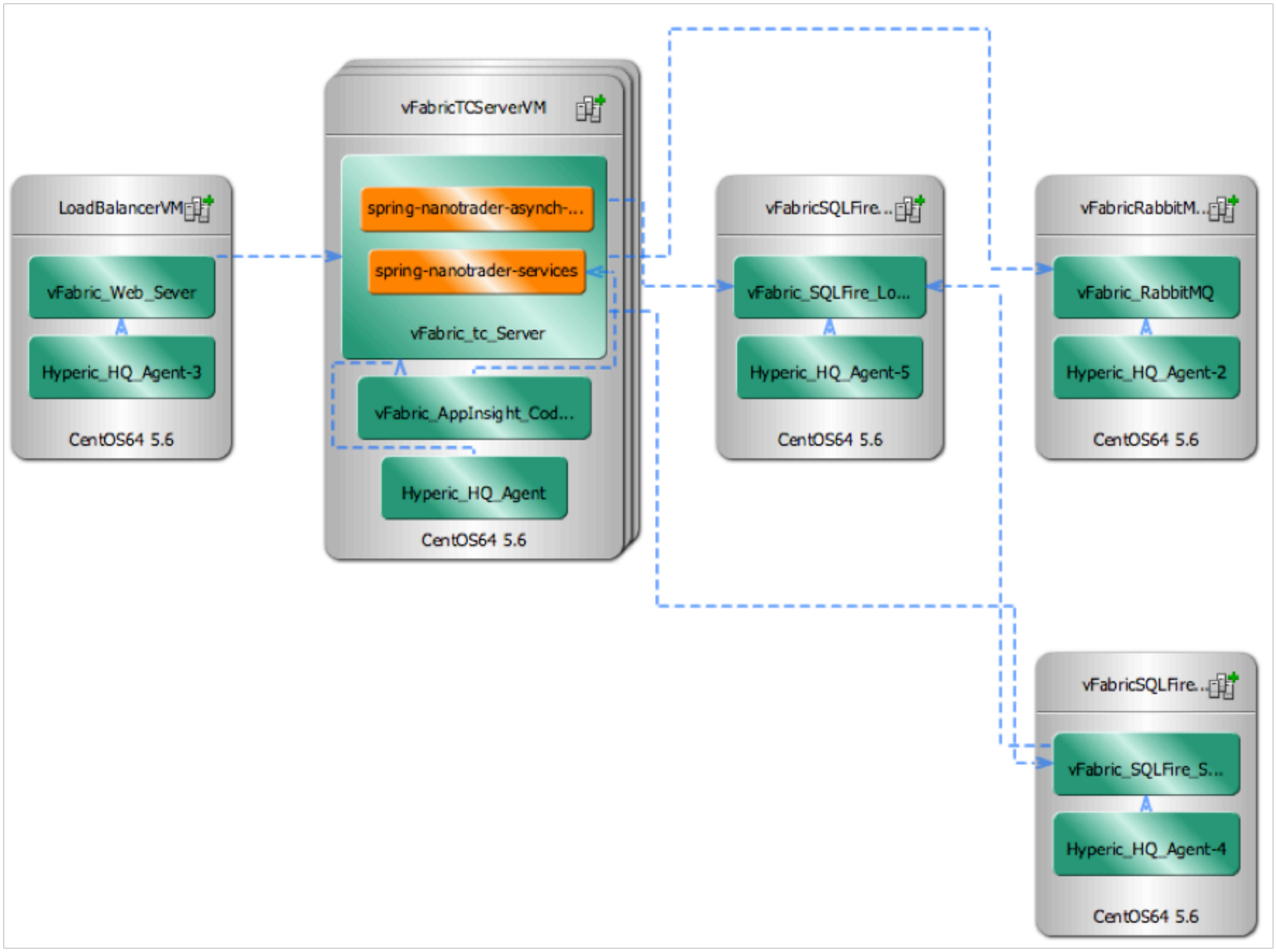
To understand this better, it may be useful to think about it in the terms of building a house. To build a house, general contractors and architects construct a blueprint that provides a visual, portable source of information for various parties to collaborate – those providing the foundation, electricity, HVAC, plumbing, and interior design all have a shared set of specifications to work from. Then, there are separate schematics or views for each group as needed.

When an application's entire topology is built inside a model-driven architecture, it works like a blueprint (in fact, that is exactly what we call it in Application Director (http://www.vmware.com/products/application-platform/vfabric-application-director/overview.html)). Those responsible for environment, application components, and deployment workflow are all able to work together, but they can define and manage key elements separately. There is a clear separation of duties for each area:

Environment (e.g. data center, public cloud, private cloud, VM, OS, storage, network, security)

Application (e.g. web server, app server, database server, WAR, SQL, GEM, etc.)

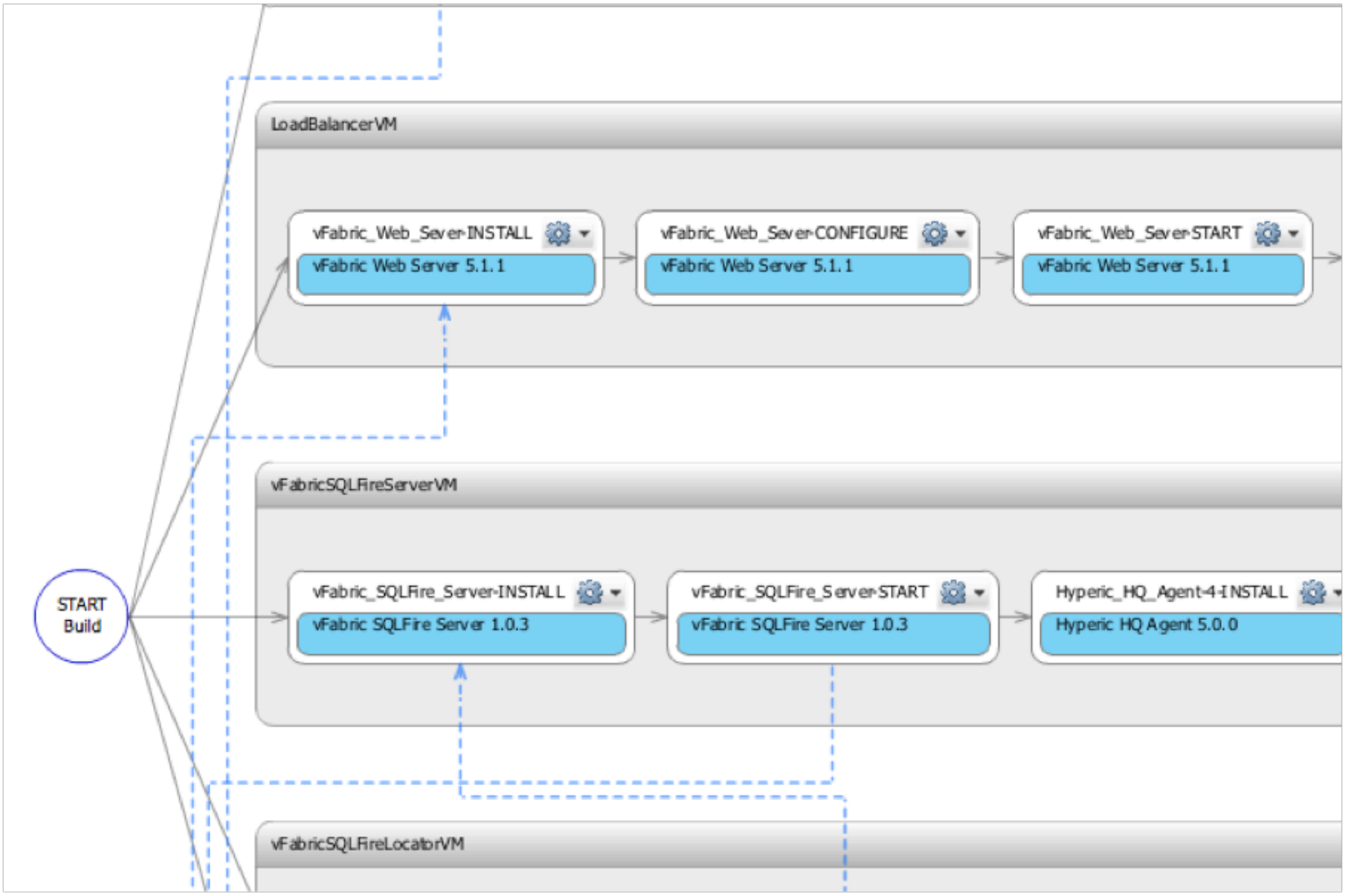
Workflow (e.g. order of install, passing of variables, dependencies, etc.)



(http://blogs.vmware.com/vfabric/files/2012/12/application_blueprint.png)

Figure 1: Application Blueprint in Application Director captures the essence of the application but remains decoupled from the infrastructure.

It is only during deployment time that a DevOps team member (or an automated script) merges the logical application model with a real infrastructure, provides the appropriate attributes for that environment, and runs a workflow to eventually provision the application.



(http://blogs.vmware.com/vfabric/files/2012/12/workflow_chart.png)

Figure 2: Workflow and dependencies between the components are applied in a late-binding process that is specific to the target environment.

Because we have successfully abstracted each part of the installation process (including the environment, the software, and the workflow), we can catalog each of these areas, make them self-serve, and mix and match them more collaboratively. IT can maintain the machines, virtual infrastructure, networking and security, and make it available for app teams to consume through a single tool. App teams are independently empowered to construct their apps without dependencies on IT. How this new dynamic is being played out for organizations is explained more completely in October's post on the new Cloud Operating Model (http://blogs.vmware.com/vfabric/2012/09/how-it-will-come-out-on-top-of-the-cloud.html) .

Faster Release Cycles AND More Stability

Earlier this year VMware's own IT department deployed Application Director internally and reported impressive savings of up to 90% on deployment times and 30% of the costs (http://blogs.vmware.com/vfabric/2012/08/qa-wvmware-it-reducing-provisioning-time-by-90-and-costs-by-30.html) . With greater levels of reuse, it also means there are less opportunities for error. If it's that much faster, easier and stable to build apps in this model, why haven't we done it like this before?

Good question.

Editors Note: As we appreciate not everyone will move to a model-driven deployment strategy tomorrow, Hyperic and other vFabric products will continue to have installers too. However, each of these products are also now available on the VMware Cloud Applications Marketplace (https://solutionexchange.vmware.com/store/category_groups/application-management?) in a ready-to-use application blueprint that you can quickly add into your own Application Director Service Catalog.

About the Author: Raghvender Arni is a staff systems engineer focused on VMware's Cloud Application Platform who has over 10 years of deep and multifaceted experience in networking, systems and enterprise software across multiple roles including engineering, product management and sales. Prior to joining the VMware team, Arni's last three roles have been sales consultant positions at Progress Software, CAST and Danucom. Previously he worked for Lucent Bell Labs where he eventually co-founded ISPSoft, one of the first IP Services Provisioning companies in the industry.

This entry was posted in Application Director (http://blogs.vmware.com/management/application-director) and tagged Application Director (http://blogs.vmware.com/management/tag/application-director) , Cloud Applications Marketplace (http://blogs.vmware.com/management/tag/cloud-applications-marketplace) , cloud operating model (http://blogs.vmware.com/management/tag/cloud-operating-model) , deployment (http://blogs.vmware.com/management/tag/deployment) , hybrid cloud (http://blogs.vmware.com/management/tag/hybrid-cloud) , Hyperic (http://blogs.vmware.com/management/tag/hyperic) , installation (http://blogs.vmware.com/management/tag/installation) , model-driven (http://blogs.vmware.com/management/tag/model-driven) , provisioning (http://blogs.vmware.com/management/tag/provisioning) , workflow (http://blogs.vmware.com/management/tag/workflow) on January 2, 2013 [http://blogs.vmware.com/management/2013/01/why-model-driven-application-provisioning-is-better-than-workflow-deployments.html] by vFabric Team (http://blogs.vmware.com/management/author/klambert) .

3 thoughts on "Why Model-Driven Application Provisioning is Better than Workflow Deployments"

Pingback: The Definition of "Cross Platform Cloud Applications" | VMware Virtualization & Cloud Management - VMware Blogs (https://blogs.vmware.com/management/2013/01/the-definition-of-cross-platform-cloud-applications.html)

Pingback: VMware Virtualization Management Blog: The Definition of "Cross Platform Cloud Applications" | Strategic HR (http://strategic-hr.com/blog/2013/01/07/vmware-virtualization-management-blog-the-definition-of-cross-platform-cloud-applications/)

Pingback: How to Use Spring with Data Aware Monitoring for Cost and SLA Management | VMware Virtualization & Cloud Management - VMware Blogs (https://blogs.vmware.com/management/2013/01/how-to-use-spring-with-data-aware-monitoring-for-cost-and-sla-management.html)

Comments are closed.