TechTarget SearchCloudApplications

E-guide

PRO+ Content

Tom Nolle's Guide to App Lifecycle Management in the Cloud/DevOps Era

In this e-guide:

Tom started his career in the 60's as a programmer, and moved through various programming and programmer/analyst positions to manage software project teams and entire software projects.

His background as a software programmer and architect makes him a hands-on kind of expert, a blend of strategic insight and in-the-trenches practical knowledge.

Ahead, Tom explains what developers need to know about ALM as businesses adopt DevOps and migrate to the cloud.

In this e-guide

ALM and DevOps tooling still a critical part of orchestration

Tom Nolle, President, CIMI Corporation

The cloud, the componentization of software and demands for continuous delivery are all combining to profoundly change applications. Nowhere are these changes more significant than in the area of deployment and operations lifecycles, where what was once called "DevOps" seems to be morphing into something called "orchestration."

Developers need to understand these trends in order to ensure development and operations continue to evolve in synchrony. This requires that developers understand the differences between DevOps and orchestration, rethink their development and operations strategies, and stay aware of important emerging trends.

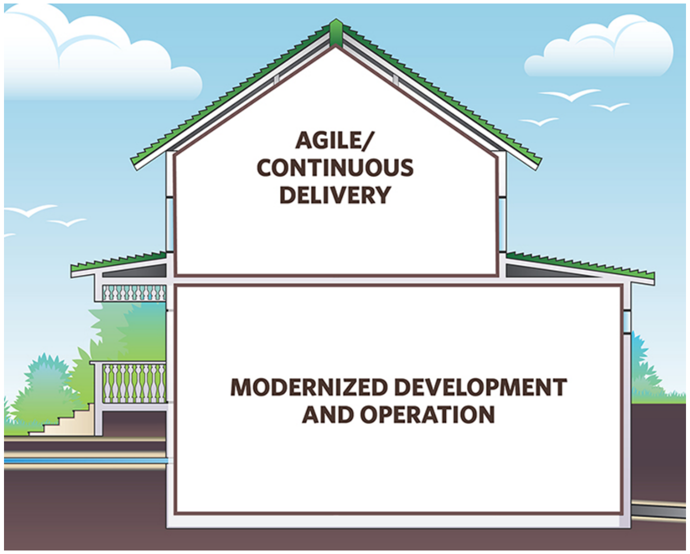
What orchestration really means

The evolution of applications from development to deployment has always been a marriage of multiple practices. There are four distinct sets of tools and practices involved in applications. Version control and development management bind the development processes together and link them back to enterprise architecture models. DevOps tools then link the development processes to deployment and application lifecycle management.

The term "orchestration" has confused many developers and architects because it's not always clear exactly what changes are occurring. Orchestration describes the broad process of applying software automation techniques to, potentially, all of the four practice areas involved in application development and operations. It modifies the tools that are already largely automated, DevOps in particular -- so much that orchestration is sometimes seen purely as an evolution to DevOps. It also adds an automation dimension that cuts across much of the rest of the development-deployment spectrum.

The common driver for orchestration is the need for greater agility and personalization in applications. This has manifested itself in mobile-based, front-end processes; mobile backend as a service and microservice-based applications. Combining these trends with virtual resources and cloud computing creates two dimensions of variables. This is why automation is needed to speed processes and reduce errors.

The new development and deployment structure



It's best to visualize modern development and deployment as a two-layer structure. The top is the Agile or continuous delivery layer of software development that unites development processes and is now embodied in application lifecycle management. The bottom is the modernized form of DevOps, designed for highly componentized ("microservicized," if you will) applications and fully virtualized resources.

The most important point in making this new structure work is to ensure that the ALM tools are fully integrated with DevOps tooling throughout the application lifecycle. There have always been benefits to this kind of integration. But if continuous delivery and innovation through software automation is the goal, there is no other path to success. In the modern application model, continuous delivery efforts and Agile development must be directed both at the services themselves and at the DevOps frameworks that deploy and sustain them. That's what orchestration defines.

Processes become like services

The notion that continuous delivery and DevOps have to integrate doesn't seem insightful to many. After all, DevOps was always meant to define the relationship between development and operations. The problem is that in a world of cloud and Agile development DevOps processes have to be considered almost as coequal to services or components. This kind of relationship is already visible in the toolkits from integrated software vendors like IBM, Microsoft and Oracle as well as in open source form from Red Hat.

With continuous development and DevOps integration, the DevOps models or recipes associated with each ALM phase have to be designed following the ALM processes. Then it can be codified in DevOps language -- declarative or imperative, as appropriate. When development changes are made to an application, component or service, the changes not only have to be tested in terms of application functionality, security and compliance, but also in how they impact the integration between ALM and DevOps.

The tight coupling of development, ALM and DevOps demanded by continuous delivery has changed DevOps already. The two most popular tools, the imperative Chef and declarative Puppet, have both evolved to support modular declarations of resources. This in turn supports virtualization and modular definitions of component and service deployments to facilitate reuse and modularized deployment. Second-generation DevOps tooling like Ansible or SaltStack have supported these capabilities from the start.

Integration considerations

There aren't many API issues or integration challenges associated with linking DevOps to continuous development and ALM, but there are definitely ways to ensure that critical things don't get overlooked.

The magic word is "modularize." DevOps models or scripts should always be built from component- or service-level elements upward to applications. Anywhere that the ALM phases require a mix of models, the models should be linked in a simple way to the production deployment models.

Event control

Another critical consideration is event control in lifecycle management. Without event control, resource processes can't signal conditions to lifecycle management and deployment processes to allow for automatic handling.

Event-handling in a DevOps tool is critical if it's to be used in orchestration. Generally, DevOps tooling that supports the emerging "Infrastructure-as-Code" model (as both Chef and Puppet do) will handle events, but check any other tools for compatibility with event-driven automation.

Keep an eye on the future

It's possible that the concept of orchestration as the automation of everything related to continuous delivery and Agile development will eventually subsume all the current tools into one continuous lifecycle-flow process. In fact, it seems that this is the only outcome that can really ensure that future software development practices can keep pace with business.

Vendors who offer all the right tools have the best chance of quickly assembling an "orchestrated" combination of them, so keep an eye on these players even if you rely on other tools. At the very least, they'll help frame your vision of an orchestrated future.

↘ Next article

How hybrid cloud adoption impacts ALM

Tom Nolle, President, CIMI Corporation

Unless you don't believe in public cloud computing at all, your company will become a hybrid cloud user. That means all of your application testing and deployment will have to accommodate the hybrid cloud, and most companies' application lifecycle management processes just aren't ready. Getting yours ready means mapping the specific differences in ALM driven by hybrid clouds, understanding how your hybrid goals expose your applications to hybrid issues, reviewing your ALM practices and tools to address these areas of change, and projecting the impact of hybrid cloud adoption to future proof your solutions.

Application lifecycle management is driven by a lot of small issues on both the technology and business side, but basic ALM approaches are usually threatened only by something big. Hybrid cloud adoption will change small things, but the big thing it changes is the mapping between applications and resources. Architects and development managers are already addressing personalized productivity enhancement and mobility with Agile application architectures. Hybrid clouds then demand these applications be deployed on a resource pool with vast differences in cost and performance.

Hybrid clouds are often presented as a step in evolution from a pure data center-driven IT model to a pure public cloud model, but most enterprises know they aren't going to totally replace data centers. The real driver in hybrid cloud is the need for ad hoc resource augmentation. You can't easily expand and contract your data center, but you can add cloud capacity when needed and retire it when the need passes. Resource control is challenging in hybrid clouds, but resource dynamism is probably the only current justification for them. The question is what specific driver demands the dynamism?

The most common reason businesses look to a hybrid cloud adoption strategy is that they are building worker-customizable or mobile-driven front-end processes to augment their current transactional applications. In effect, the front-end processes are migrated to the cloud and integrated with legacy workflows at a fixed point. Both sides of the hybrid relationship are largely independent except at this hand-off.

Cloud bursting, or the allocation of public cloud resources to augment data center capacity when workloads increase or in a period of failure, is the second reason for hybridization. With cloud bursting, the pool of resources expands and contracts with work, and, in theory, each application component that is capable of being used in a load-shared way can be replicated then scaled back as needed.

Most businesses agree that ALM in front-end-driven hybridization can be addressed in the same way as Web front-end integration or even multi-platform -- Linux and Windows, for example -- data center ALM is addressed. Detailed ALM tools and processes are first selected for the public cloud and private IT resources, and the two are then harmonized at the point, in time and technology, where they meet.

Cloud bursting is more difficult because the application's components move in terms of resource commitments. Each piece of an application, each step in a test or workflow, might be supported in or out of the cloud. Information must then flow along different paths, with different integration and different quality of service.

Making ALM work in a cloud bursting application starts with having massive test data generation capability. While some businesses rely on simple tools for data generation in ALM, that's not going to work in cloud bursting ALM because you'll be unable to force the scaling in and out of resources. Random data generation for functional testing is still important, but you may want to separate it from volume test data generation to cut ALM testing time and complexity. Verify workflows overall, then verify cloud bursting performance and integrity second.

One of the important things to test in hybrid cloud ALM is the specific details of the DevOps processes used for application component deployment and decommissioning. For cloud bursting, it's critical that you get a new component instance up and running quickly and taken down gracefully, meaning that all the pathways used for connection of the instance are removed without impacting the rest of the application.

While dictating application design and operation may not normally be considered an ALM mission, hybridization dictates the use of DevOps if there's to be any hope of creating a robust application environment and a stable and effective test process. In the cloud, there is a visible trend toward model-based rather than script-based DevOps, and model-based systems will require careful testing of each model and all of the model's operating states. In fact, truly integrated development and operations practices makes hybrid cloud ALM easier and more reliable.

In most cases, the best results in hybrid cloud ALM will be obtained from integrated development-testing tools, or from suites of tools that are designed to work together. Users of development tools from IBM, Microsoft, CA and others can be assured the pieces will all fit. For those who want to assemble development and ALM tools, it may be wise to use a hybrid cloud application as a testbed to ensure you encounter all the conditions hybridization will create.

For those who may have felt comfortable with the notion that their cloud-front-end mission for hybrid cloud adoption would keep them out of the most complex areas of ALM impact, then be prepared. The long-term trend is toward "Agile applications" whose workflows and structures are assembled almost ad hoc according to worker need, and that use both data center and public cloud resources based on worker and data location and pricing and performance policies.

This Agile-application future for ALM raises many questions, even without considering hybrid cloud adoption resources. The traditional notion of an "application" is weakened as worker productivity gains depend increasingly on microservices tuned to events. ALM will have to evolve to meet this challenge and also to accommodate the dualism of resource ownership that hybrid clouds demand -- now and likely for a long time to come.

Next article

Effective app testing and ALM techniques in the cloud era

Tom Nolle, President, CIMI Corporation

Some early cloud adopters have had a happy and successful cloud project experience marred down the line by testing and application lifecycle management issues. A successful project is one that meets the business goals as well as security and compliance requirements.

To be sure that's all accomplished, cloud planners, architects and developers should start with ensuring that test data and processes validate the user experience and visualize the technology groups within their cloud deployment. They should vertically integrate user experience with security and compliance in each group, and horizontally integrate within and between groups using proven workflow techniques that can be effectively tested.

Cloud applications are a double threat in terms of user experience. Most are launched to optimize productivity, and the variables of cloud computing can impact quality of experience significantly. Because the cloud presents many operational scenarios growing out of the wide choice in component placement, significant testing is required to hit them all. This promotes the use of test data generation, which can compromise the whole testing and ALM process.

Effective test data generation for the cloud has to follow the entire path of real user data and provide for response time measurement and reporting, or there's no way to determine whether the top-level user experience is even being tested. This is a particular challenge for mobile applications, which are increasingly the prime target for cloud deployment.

There are two ways to couple test data effectively:

- Introduce data at multiple points just as mobile users would
- Measure mobile data variations independently and combine the results with application performance data to assess response time and QoE

It is very difficult to drive a realistic user-to-application-to-user workflow in the cloud without making the test process into a cast-of-thousands effort. It's smarter to take user front-end access statistics and combine them with the rest of the testing and ALM techniques. That means getting user access statistics from the widest possible user distribution from other applications. Combining this with testing and ALM QoE statistics is easiest if you design your cloud application with a test GUI hook into the front-end process, something that parallels the real user connection and duplicates its internal processing delays to the greatest extent possible.

Testing and ALM middleware tools

Another aspect of cloud applications that complicates testing is the variability of middleware and platform tools across the various elements of the application. Most cloud applications have a front-end GUI process that feeds a deeper transaction processing element. The front end is often based on a Web-oriented technology like Java and the back end might be traditional SOA using a service bus. While it's important to test through this entire flow, it may be helpful to test each segment independently to spot problems and aid in isolation and repair.

If you group your application components by middleware, you may be able to define two or three technology groups within your application. In most cases these technologies won't be distributed randomly through the application but will be exercised on a linear path from worker to database. Application testing during development, and ongoing ALM, can exercise these technology groups first to insure they're functioning according to specifications, then do end-to-end testing aimed primarily at validating overall QoE and functional coupling and integration. To do this, each of the technology groups should be vertically linked with the functional business requirements it's contributing to, and the security and compliance issues that it must address.

The technology-group approach may be particularly helpful in adapting existing ALM techniques to new cloud deployments because the back-end technology groups are usually largely unchanged in cloud applications. Those cloud changes focus on the user front-end processes, and these new processes are also more likely to introduce new security and compliance issues. Testing at this level should validate that the back-end processes are unchanged (or accommodate minimal changes actually made) and then move to validating the behavior of the front end. The fact that a technology group was selected based on harmonized middleware choices facilitates both tool selection and testing.

Horizontal integration

For private cloud and Docker or container users, a technology cluster is also a good place to start portability and scaling plans. The technology cluster is harmonious in terms of OS and platform, so it can be mapped to a single resource pool or become the basis for a Docker Swarm. Horizontal scaling and failure recovery are then facilitated and the standard framework you've built this way will be easier to test.

Horizontal integration is normally the focus of both testing and ALM techniques. For the cloud, it's often easiest to first do horizontal integration testing within a technology group by following the workflow through that group of components. Most issues with security and compliance can be picked up at this level of testing, and it's simpler to conduct the tests on a group of components than across the entire workflow. It's probably not wise to try to do QoE testing by adding process times for all the groups; the relationship between group times and total time is more complicated and end-to-end tests of performance can be done as the final step.

When doing horizontal integration testing, the primary goal should be to ensure that all the variants in component positioning and deployment are tested. This can be difficult with public cloud services because you probably won't have the ability to force deployment changes in the basic cloud services. It's best to contact your public cloud provider(s) to get assistance on how to ensure you can deploy components in all the possible hosting points for testing.

Most security and compliance problems generated by technology or design flaws will creep in because of flaws in cloud integration. Often the processes of locating deployed or moved components open flaws in the security of the directory system being used, and in many cases simple testing is unlikely to uncover these issues. A thorough design review should be conducted prior to the first field tests of the application, and any changes to directory management should be made only after another such review.

The cloud doesn't need new testing tools as much as it needs cloud-centric testing planning. This should start with application planning and not when you're ready to test. Carefully blending knowledge of the cloud with knowledge of the application allows you to organize both application testing and ALM techniques to their optimal state for your business.

Next article

In this e-guide

About SearchCloudApplications.com

SearchCloudApplications.com is dedicated to serving the information needs of organizations which have, or are planning to add, business applications in the cloud. The site spans cloud delivery models and challenges, with articles, tips, tutorials and other resources covering Software as a Service (SaaS), Platform as a Service (PaaS), integration and cloud development.

IT, business professionals, and developers will find unbiased information across cloud topics including technology strategy, Java, vendor selection and database management, architectural considerations, deployment strategies, mobile applications, development and big data strategies, and more.

For further reading, visit us at:
http://searchCloudApplications.techtarget.com

Images; Fotalia