QualiSystems

Building a Cloud-Based "Lab as a Service" (LaaS)

QualiSystems

Introduction

Test labs represent a large ongoing investment and an opportunity for significant savings. Many technology organizations are seeking to consolidate multiple test labs into more central, shared test data centers that can be used by many remote user groups. The key to the success of such initiatives is automation software and accompanying best practices in test data center design and utilization.

Test and lab automation software can provide testing organizations with the technology platform to create successful, scalable and sustainable Lab as a Service (LaaS) clouds.

LaaS clouds not only deliver impressive CAPEX and OPEX savings, but by leveraging the full breadth of automation frameworks, they also pave the way for automation of the whole testing process. This leads to additional business benefits such as speedier time to market, agility in responding to market and customer changes, and increased competitiveness.

Sustainable Automation is Key to LaaS Consolidation

When seeking to consolidate multiple, dispersed test labs into a centralized test data center, a key challenge is effectively enabling remote users. If the highly manual processes typically at play in test labs remain in place after consolidation, users will find that they are road blocked from being productive. This is because remote users will need to rely on local personnel to perform manual tasks such as finding and connecting test equipment into topologies.

Since these processes can take up to a week to perform, and since there will naturally be higher demand on fewer local resources to perform these tasks, remote users will suffer from delayed access to resources. Unless a software-driven approach is implemented to manage the search, reservation and connection of consolidated lab resources into usable test topologies, lost productivity may doom the consolidation effort by causing remote user groups to re-implement their own testing environments.

The state of the art for software that enables LaaS consolidation has advanced dramatically. Modern LaaS automation software should deliver a broad range of capabilities that enable sustainability of the LaaS initiative, including:

- Centralized, live infrastructure and resource inventory that is customizable to make it easily searchable
- Inventory-aware test topology design
- Shared, calendar-based resource and topology reservation
- Connectivity mapping and automated connectivity control
- Easy to create automated provisioning tasks
- Non-programmer friendly automation workflow creation based on a library of highly reusable test objects that can be created from a wide variety of sources and leveraged to create:
  - Auto-discovery, auto base-lining and other automated maintenance routines
  - Full test automation workflows

Another important concept for sustainable automation is that the platform for managing the test lab should avoid the pitfall of using monolithic and fragile script-based approaches to automation, which

cannot scale due to their high maintenance costs. An object-oriented platform that captures and manages all inventory resources, test topologies, provisioning actions and testing tasks in a library of highly re-usable, easy to update object building blocks is the only architecture that ensures automation scalability and long-term and cost-effective sustainability of LaaS cloud administration.

Best Practices for LaaS Consolidation

To achieve a successful LaaS roll-out, best in breed technology is critical but must be accompanied by best practice methodologies:

Highly Automated Physical Layer Connectivity:

Software-based automation benefits from a structured, documented, and easy to operate physical connectivity environment. The state of the art practices in LaaS consolidation include deployment of Layer 1 switching to virtually eliminate manual cable patching. Of course, Layer 1 switching should be combined with sound, TIA standards-compliant data center layout and structured cabling so that the entire physical environment can flex to changing requirements over time.

Resourcing the Automation Infrastructure Service

The most successful lab automation deployments involve dedicating personnel resources with data architecture and programming skills to build and maintain the object library of inventory resources, test topologies, provisioning and shared testing objects and workflows. The broader user community can then leverage this library to build and reserve topologies, easily perform provisioning, and progress into test automation as the library is built out. Dedicating resources to maintaining the object library as an infrastructure service is strongly recommended because if the utility and ease of use of the object library is not maintained at high levels, users will abandon the automation system, wasting the investment.

A Phased Approach:

Successful LaaS automation is typically built in phases, where each phase aims for a visible productivity gain and return on investment in a reasonably short time in order to build user engagement and momentum and create realistic expectations.

Generally speaking, achieving “hands-off” visibility and reservation of lab resources using Layer 1 switching and automation software is the first major goal. This level of automation allows remote users to be on an equal productivity footing with local users, and promotes deep buy-in of all user groups with the consolidation initiatives. Any LaaS consolidation project should ensure that the centralized data center is designed with this in mind.

The second automation phase is to free testers from the time-consuming tyranny of low-level device provisioning tasks. This involves turning manual provisioning processes into easy to invoke, menu-driven tasks from the automation GUI. The best practice in this stage is to ensure the sustainability of the system, by avoiding reliance on fixed scripts. While it may be relatively easy to create a first set of provisioning scripts for some usage scenarios, the time-consuming nature of script maintenance will too often cause the provisioning capabilities of the system to become out of date. The negative experience

of using scripts that don't work will end up alienating users and deepen their reliance on manual processes. Automation of provisioning tasks typically start with the basic provisioning steps needed to get DUTs to a particular state, such as uploading OS images or applying patches. More advanced provisioning tasks involve common configuration steps to ready the logical layer of a test topology, such as configuring VLANs, routing adjacencies, or tunnels on physical or virtual switches. These automated provisioning objects help test engineers more easily accomplish the routine tasks that often dominate their workdays, and allows them to focus more on higher order thinking to achieve maximal test coverage.

A third phase of LaaS automation that is short of full test automation is to create automated maintenance routines. Examples include auto-discovery, which helps keep the inventory up to date, and auto-base lining returns devices back to their default provisioning states on a timed basis. These types of routines require development of a comprehensive set of device control/interface automation objects for all necessary devices in the test infrastructure, so that they can be leveraged across multiple maintenance automation processes.

The Path to Testing Velocity—Full Test Automation

A modern LaaS automation software platform will go beyond hands-off test topology design and automated provisioning. It will also provide the way to implement full test process automation. This more advanced form of LaaS automation means the creation of a library of reusable testing task objects. Best practices here utilize software tools to enable the construction of actual test automation workflows from those objects by non-programmers. This is the only way for test automation to achieve high percentage penetration of the overall testing process. However, organizations that invest the time and programming personnel to develop and maintain the automation task library will find that they can dramatically increase the speed and thoroughness of testing. This results in faster time to market for new products and services and higher market competitiveness.

Test Shell—The Industry's Choice for LaaS Automation

A best of breed commercial solution deployed by industry-leading organizations worldwide, TestShell offers the fastest path to a successful and sustainable LaaS, allowing managers and engineers to:

• Manage data center inventory including physical DUT and testing equipment, L1 switches, and virtual resources such virtual machines and virtual switches in a live, searchable database of resource objects tagged with searchable attributes, eliminating manual searching for equipment in racks and allowing engineers to interface with the data center infrastructure efficiently via software. TestShell's inventory and resource management allows for object hierarchies which can represent relatively simple nested resources such as chassis, blades and ports or complex, pre-integrated resources stacks such as converged infrastructure and "data center in a box" solutions.

• Create test topologies via a software GUI that allows drag and drop of resource objects onto a canvass, visually ascertain availability, design and sanity check connectivity, and save the entire topology as a higher level object in the resource library, so that it can be reused later or by other engineers.
• Schedule resources and entire test topologies through a common calendaring system. Resource conflicts can be easily resolved since it is easy to find out who is using resources at any time.

• Manage connectivity remotely by generating patching or cabling requests to lab administrators, or if Layer 1 switches are in use, to automatically connect test topologies, enabling true 'hands-off' test topology management.

• Make device provisioning error-free by building a library of easily updated automation objects for common provisioning tasks that can be launched from a right click menu in the graphical test topology view. Device provisioning can include uploading OS images, or common configurations such as creating GRE tunnels or routing adjacencies between virtual switches.

• Create auto-discovery and auto-base lining processes that leverage TestShell's extensive array of control interfaces, GUI automation and scripting capabilities to streamline the management of inventory and device states.

• Roll out full test automation by the creation of an automation object library. This includes the ability to integrate legacy automation scripts as testing objects as well as creation of new test automation objects through screen, GUI and other capture processes.

• Generate real-time reports and dashboards on data center device utilization, topology reservations vs. activations, and even comprehensive test results.

Conclusion

TestShell automation accompanied by best practices methodologies offer organizations an opportunity to successfully and sustainably deploy LaaS clouds, and to go beyond resource sharing to automated provisioning and ultimately full test process automation. TestShell's object-oriented architecture provides the state of the art platform that the industry's leaders rely on to move beyond traditional, script-based approaches to automation and modernize their testing processes. The net result is significant CAPEX and OPEX savings, increased business velocity and agility, and higher levels of market competitiveness. To learn more from the global leader in test and lab automation solutions, please visit http://www.qualisystems.com .