LaaS: ENABLING ON-DEMAND LAB SERVICES

Andrew Bartley
Lab Systems Engineer
EMC Corporation
andrew.bartley@emc.com

Jochen De Smet
Principal Software Engineer
EMC Corporation
jochen.desmet@emc.com

Jonathan Hayden
Software Engineer II
EMC Corporation
jonathan.hayden@emc.com

Table of Contents

Abstract

As globalized markets become more competitive, businesses can no longer afford to wait several days for basic lab tasks to be completed manually. Business units are looking for quicker response times and on-demand lab services. Development cycles are getting shorter and shorter—what used to take three years must now be completed in three months. Software developers are in short supply in the US—the lack of personnel will need to be made up for with efficiency. There just isn’t time to roll a terminal out into the lab and manually reinstall hardware. The time has come to rethink data center operations and begin transitioning toward offering Labs as a Service (LaaS).

Although the virtualization of data centers has allowed for more dynamic and rapid allocation of resources, a large portion of data center activities still require physical hardware to meet certain requirements. This physical hardware still requires time-intensive, manual procedures to make any major changes or repairs to the system. These hands-on procedures not only add to lab costs based on salary of operations employees, but more importantly they can lead to delays in lab response time. If lab response time is in the critical path, each hour of hands-on time is an hour later that the product will be released.

This Knowledge Sharing article will introduce EMC Proven Professionals to one innovative method of delivering LaaS. One EMC lab implementation case-study explores how LaaS principles were introduced into a lab’s architecture resulting in much quicker lab service times. LaaS also enables the complete automation of certain basic tasks (such as reimaging or power cycling a system), allowing for on-demand self-service lab operations. All of these benefits can be realized by meeting the following requirements according to what is outlined in this article:

- Lab request ticketing service
- Role-based access control (RBAC) of lab data
- Globally available, virtualized lab equipment
- Automated system reimaging

By adding intelligence to existing data center best practices, LaaS provides customers with the self-service tools necessary to minimize lab service times to minutes. Reducing lab service time will add value to the data center, allowing customers to stay productive rather than wait idly while a hands-on lab procedure is performed.

This Knowledge Sharing article will be particularly useful to lab or data center managers, along with solution architects that work in this space.

Introduction

All software development efforts will require development (and test) labs of some kind in order to be completed. These development labs are frequently reconfigured due to changing product requirements. Depending on the exact feature or test case at the time, radically different lab configurations will be required on a weekly or even daily basis. These different configurations will be felt in terms of lab requests; as Figure 1 illustrates, the number of lab requests can vary more than 100% week-to-week (as seen from 10/5 to 10/12), depending on the business requirements of the time. This data was collected from a Massachusetts development lab environment. In this environment, each lab request for a new server or for repairing existing equipment is processed as a “service request”. This dynamic and sometimes unpredictable workflow presents development labs with a unique challenge that requires an agile and adaptive solution framework in order to meet these rapidly changing business requirements.

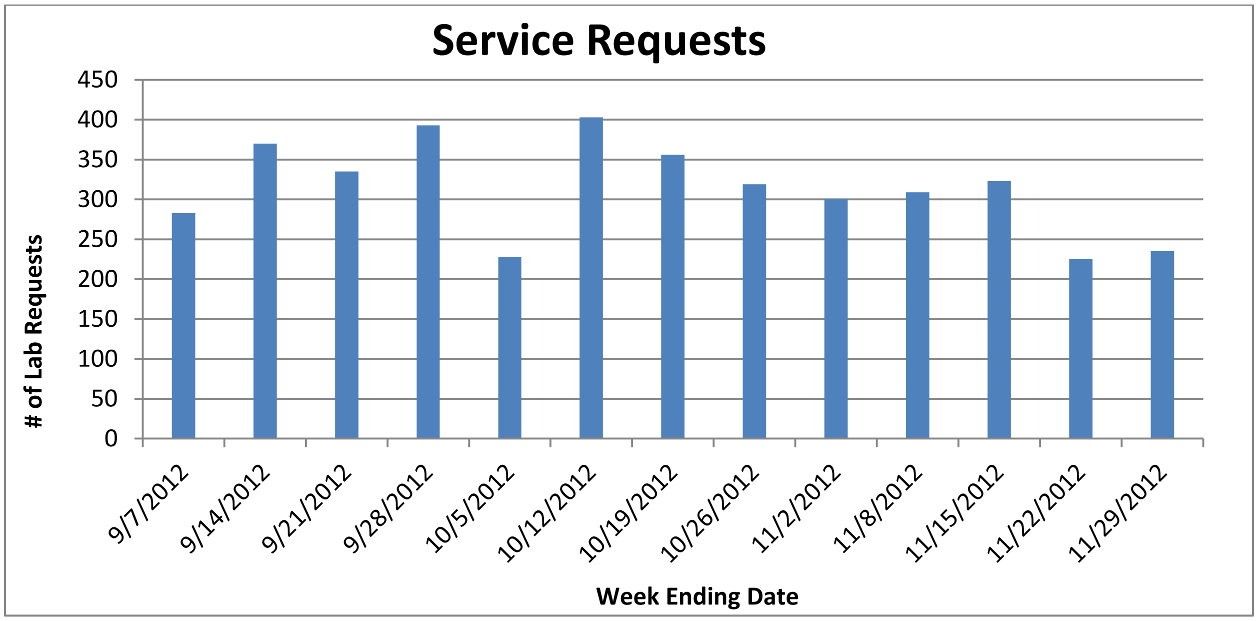


Figure 1: Sample Quarterly Lab Request Graph, sorted by week

Additionally, software development is shifting toward globalized teams that can accommodate a 24x7 Global Software Development architecture with the promise of shorter development cycles (James Herbsleb, 2001) and reduced development costs (Sami ul Haq, 2011). This continuous development environment has major implications on lab management teams. Without decreasing the scope or timeline of a project, a common response to large spikes in workloads is the use of overtime or increasing available resources. Continuously increasing budget requests is never easy, making this an unsustainable solution to increasing lab workloads. Decreases in productivity and increases in health risks—caused by long-term overtime—have been well documented (Sabine Sonnemag, 1994) (A E Dembe, 2005) (H. Randolph Thomas, 2012), making overtime another unsustainable solution for higher lab workloads. A new solution is required for the challenge of increasing lab workloads.

Solution Overview

Automation of common lab tasks is the key to keeping up with lab request demand. Power cycling a host should not require a 15-minute trip to the lab and back; this can be done right from the user's desk. Reimaging an array does not always need to take days waiting on lab staff; most instances can be initiated with a few clicks and can be completed in well under one hour. As traditional IT continues its transition towards IT-as-a-Service (ITaaS), development labs and data centers must also follow this path. By embracing virtualization technologies and adding intelligence, development labs can be delivered as on-demand services available to customers 24x7. Some of the key benefits from offering LaaS include:

- Self-service host/array reimaging
- On-demand power cycles for hosts/arrays
- Virtual Machine (VM) deployment in less than 30 minutes
- Self-service lab equipment reporting

To realize these benefits for additional lab agility, there are certain systems that must be configured properly. Figure 2 shows a high-level visualization of this case study's LaaS environment. While there is some flexibility for configuration customization, the systems listed in Figure 2 must be running and available within the lab environment to achieve the LaaS benefits described in this article; each component will be discussed in greater detail along with examples from our real-world case study.

This case study is based on live development and test labs at EMC, though many implementation-specific details have been omitted for business security.

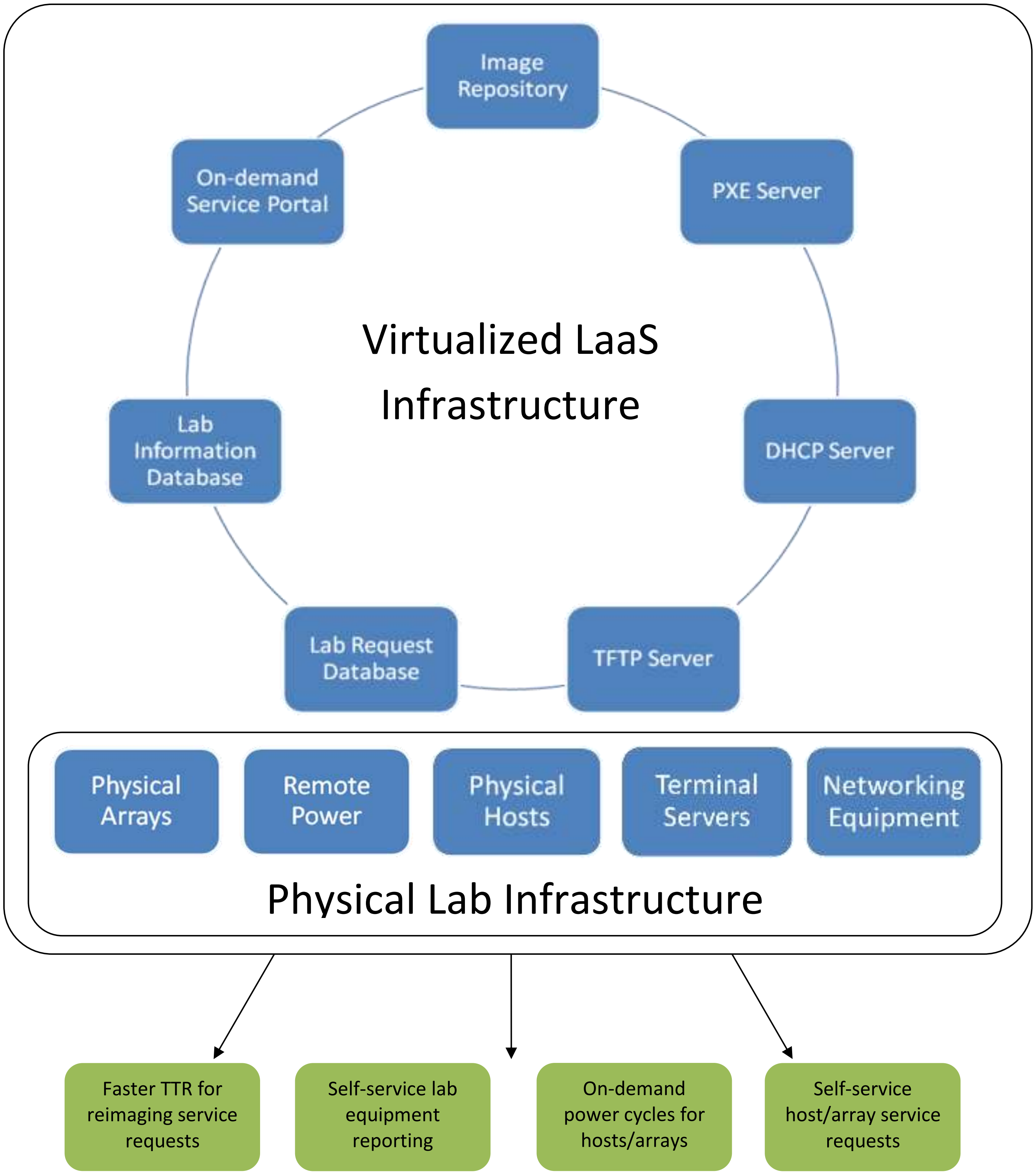


Image Repository
PXE Server
DHCP Server
TFTP Server
Lab Request Database
Lab Information Database
On-demand Service Portal
Virtualized LaaS Infrastructure
Physical Arrays
Remote Power
Physical Hosts
Terminal Servers
Networking Equipment
Physical Lab Infrastructure
Faster TTR for reimaging service requests
Self-service lab equipment reporting
On-demand power cycles for hosts/arrays
Self-service host/array service requests

Figure 2: LaaS Architecture Overview

As noted in Figure 2, two sub-environments have been configured that complement each other for LaaS to run properly. The first environment, referred to as “Physical Lab Infrastructure”; is a typical lab. In this category will be hardware used directly by the customers of the labs and its accompanying infrastructure (cabinets, networking equipment, etc.).

The second environment is the Virtualized LaaS Infrastructure; this is the “production” area of the labs. Production VMs that will be used by customers would go in here, along with any hardware that runs lab services such as DNS, DHCP, PXE, etc. Our case study has virtualized all production services, but if another lab chooses to use physical servers for production services instead of virtual machines, those physical servers would still fall under this category.

Physical Lab Infrastructure

Physical lab infrastructure is comprised of elements of a basic lab or data center: hosts, arrays, networking equipment, etc. The exact specifications of each type of infrastructure will vary significantly depending on the specific lab workloads at the time, but lab equipment will need serial concentrators, enterprise-level networking equipment, and remote power units for full LaaS functionality. Table 1 lists required infrastructure and their uses.

Infrastructure Required	**Functionality of infrastructure**
Storage Arrays	For customer use
Servers / Hosts	For customer use
Serial Concentrators	Remote console access to equipment
Remote Power Units	Remote self-service power cycles
Enterprise-level networking equipment	Network traffic segregation Remote configuration changes

Table 1: Physical Lab Infrastructure Overview

All storage arrays and hosts in the lab should be connected to a serial concentrator, remote power unit, and to networking equipment. These pieces of infrastructure will not only allow for some routine lab tasks to be performed remotely (power cycling, console access, etc.), they will also assist in the process of automating certain tasks. This case study will go over the details of setting up these automated tasks later in this article.

Virtualized LaaS Infrastructure

Virtualized LaaS Infrastructure is what many labs would consider “production” services for the labs. All core services and equipment such as network cores, DNS servers, DHCP servers, email servers, and so forth, would be in this category of equipment. This is also where most of the intelligence is added in order to maximize the value of the labs.

These services have been virtualized using VMware, though other virtualization solutions can be used with varying efficacy. If the environment is more conducive to configuring some or all of these services using physical hardware, that should not affect the functionality of this LaaS case study. However, some of the high-availability and business benefit figures may vary from what is presented within this article.

The choice to virtualize is based on the business benefits of reduced hardware costs, reduced operating expenses, shorter server procurement times, and improved business continuity (VMware, 2011). By providing customers with virtual machines (rather than physical machines) whenever possible, server procurement times are reduced (VMware, 2011). Physical deployments require ordering, installing, and configuring new hardware which can take up to weeks depending on hardware vendors. Meanwhile, assuming proper infrastructure forecasting, virtual machine deployment can be completed within 30 minutes in conjunction with the PXE environment utilized in this case study. Virtual machine requests/deployments can be integrated with a ticketing system, assuming there is flexibility to customize the ticketing system. Since a comprehensive list of benefits can be easily found from virtualization solution vendors, only a few of these benefits are outlined in this case study.

The following subsections describe each system in the Virtualized LaaS Infrastructure in detail. Table 2 summarizes Virtualized LaaS Infrastructure systems and their functions.

Virtualized LaaS Infrastructure System	**Functions**
DHCP Server	Allows simplified equipment and VM deployment Enables PXE boot functionality
Image Repository	Stores array/host images for PXE boot New images automatically added (policy dependent)
PXE Server	Allows hardware to boot from network Automatically updates options from image repository
TFTP Server	Serves the images over the network for PXE boot
Lab Request Ticketing System	Monitors lab service requests and resolution stats Allows on-demand equipment info lookup
Lab Information Web Application	Allows self-service equipment reservation from RBAC-allocated equipment pools Allows on-demand, automated equipment reimaging Monitors equipment configuration and usage stats
Service Portal	Allows users to easily browse all lab services

Table 2: Virtualized LaaS Infrastructure Overview

DHCP Server

It is important to be able to integrate new machines into an existing lab environment as quickly and with as little manual intervention as possible. Dynamic Host Configuration Protocol (DHCP) technology is very useful for this. DHCP allows a client to automatically obtain configuration information from a central server. Typically, this includes network information such as the client IP address, name servers, time servers, and so on. A DHCP server enables PXE booting in a lab, allowing for reimaging clients via the network—no physical installation media required. PXE will be discussed later.

DHCP enables management of IP addresses from a central location. This eliminates the need for one to manually configure the network on their machine, reducing the chance of duplicate IP addresses due to human error when configuring static IPs. By linking DHCP and Domain Name System (DNS) servers, every machine can always be addressed by its hostname. This link prevents human error in DNS entries by automating DNS record updates (even if an IP address has changed) and eliminates hands-on configuration time. NTP can also be sent to any machine requesting an IP address, if configured. This process is depicted in Figure 3.

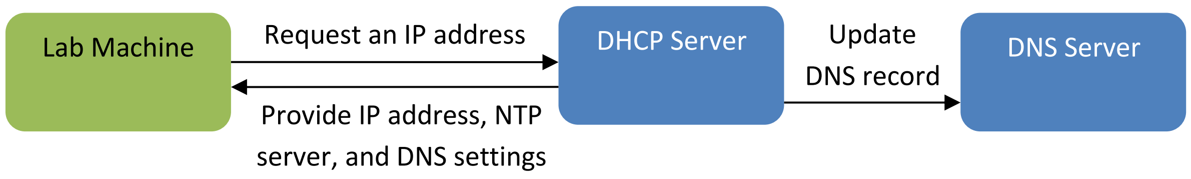


Figure 3: DHCP and DNS Server Configuration

Two of the most common configurations for DHCP leases in a lab environment are either setting up long DHCP lease times or permanently mapping the IP address for a given machine on the server side. The latter essentially provides a static IP address on the client side while still keeping the management of it on the server side; though this provides greater IP address consistency, it requires additional time for these manual entries.

PXE and TFTP Servers

Pre-eXecution Environment (PXE) allows a machine to obtain its boot image from the network, bypassing problems that may be on the local disk's operating system. PXE-compatible network installation images are available for a variety of operating systems and distributions, or custom images can be created if required. In cases where new equipment comes into the lab without any software, this setup can greatly speed up the configuration time of new hardware. If a software issue on an existing machine is preventing it from booting, in most cases the equipment will still be capable of a PXE boot (except for certain hardware failures); this allows the hardware to get back to a usable state.

The case study in this article uses PXE on an as-needed basis for fresh installation of new equipment and for reimaging equipment to a known, stable image; this required a "PXE map" which stores configuration entries on the PXE server for each client that is going to PXE boot. Clients PXE booting for the first time will go through a short script that adds an entry onto the PXE server; both lab staff and end users may update this entry to change which image the client will obtain on the next PXE boot.

After the first PXE boot, a client will request a DHCP address for the image transfer. Once an address is obtained, the client will request an image from a PXE server on the same network. The PXE server will then redirect each client to its specified boot image based on the entry on the PXE map for that client.

On some machines, the Intelligent Platform Management Interface (IPMI) can be used to enable PXE over the network. Other machines require entry into BIOS to change the boot order, which means a terminal server connection is required to avoid the need for physical access. The Lab Information Web Application in this case study has written scripts that allow end users to simply choose the image that they would like on their client. Once the image is selected, the Web Application then automates the PXE process. Depending on the length of installation, the client will be ready for use again in 20-60 minutes with the requested image—no hands-on time required. Figure 4 provides a visualization of the self-service reimaging process.

A fair amount of configuration is required on the server side of the PXE process. At minimum, a Trivial File Transfer Protocol (TFTP) server and a DHCP server are required. The DHCP server must be configured to send clients the hostname and IP of the TFTP server, as well as the filename to load for each client. Boot images are required for each type of software available for installation; this will likely be done on a separate storage device. A set of configuration files, or the “PXE map”, is required to determine which image needs to be loaded on a given machine. Depending on the TFTP server and/or bootloaders used, PXE can be set up to select an image based on the MAC address, IP address or subnet, GUID, or hostname. In this case study, a simple web interface enables people to configure which image their clients will pull from a PXE boot.

In development environments, it is possible to automatically add new software revisions to the PXE server’s available images. This requires a dedicated image repository on hardware separate from the PXE server. Development teams and lab teams must work together closely to ensure that this automation stays up-to-date.

It may make sense to have several servers that can supply the needed images. This is particularly likely for labs with a large number of machines or for lab environments that are geographically separated. The case study in this article allows users or lab staff to pick which server to use to ensure the fastest installation time.

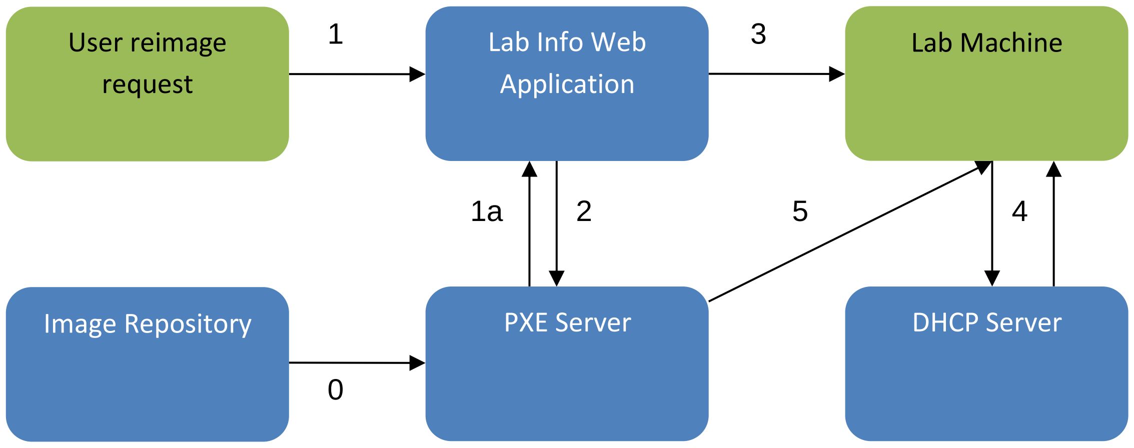


Figure 4: Self-service Reimaging Diagram

0. Images are periodically forwarded to the PXE server; this allows self-service reimaging
1. A user requests a new image for their machine via the Lab Information Web Application
    a. The images available for use are displayed to the user through the Web Application
2. The image selected is updated to that machine's PXE record
3. The Web Application sets the machine to PXE boot and power cycles the machine
4. Upon reboot, the machine requests an IP address via DHCP so that it may PXE boot
5. The machine boots the requested image from the PXE server

In addition to the web page intended for human use, a PXE server can offer REST API that other applications can use to configure a client's PXE entries. One application of this REST API is the "one-button-PXE" functionality (briefly mentioned earlier), available in the Lab Information Web Application.

APIs also provide a way for client machines to provide feedback to the server regarding installation progress or issue. This information can then be displayed where an organization feels it may be relevant. In the case study, this information is available from the same web page used to configure a client's PXE settings. The API also allows machines to upload the installation logs to another server. Storing the logs ensures that users can see their equipment's progress during installations. This centralization of installation logs can be extremely useful during development; debugging software can be accelerated with a central location of all installation logs.

Lab Request Ticketing System

A productive lab environment requires effective, documented communication between lab customers and lab operators. This case study will explore one Lab Request Ticketing System solution to this end.

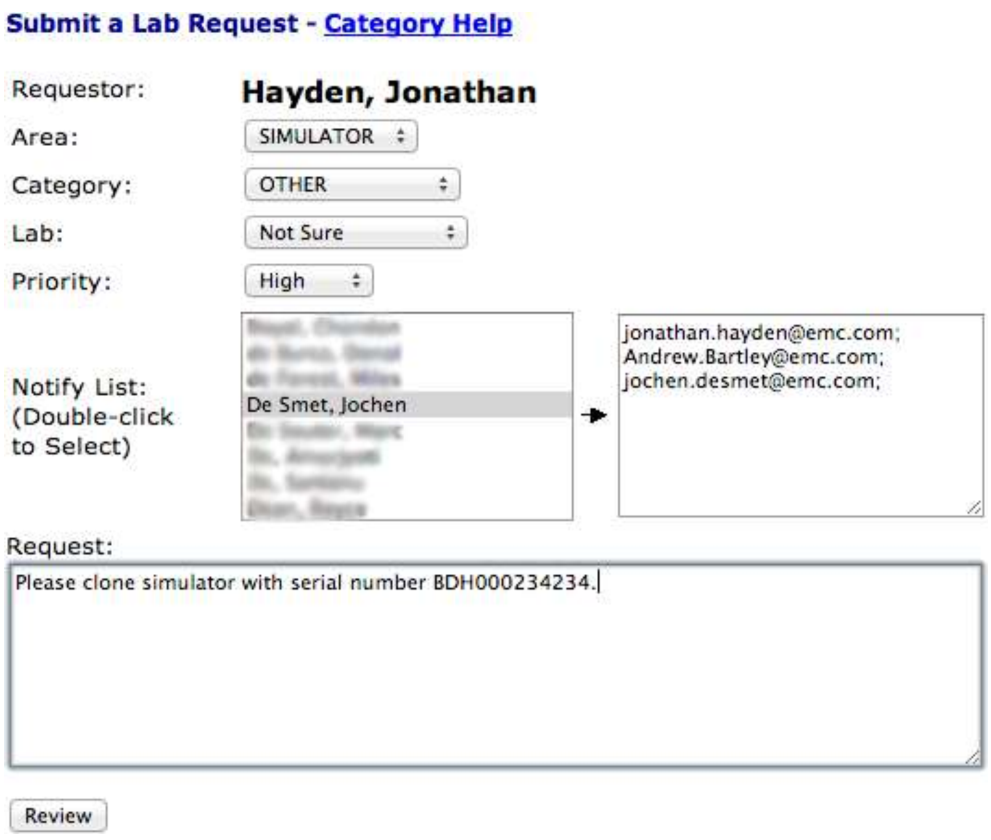
The ticketing system implemented in this case study has the following minimum requirements.

- Role-based access
- Collect requests from customers
- Route requests to the appropriate lab operators
- Communicate progress of requests to customers
- Produce metrics on requests
- Integration with external tools

The Lab Request Ticketing System is a web application that enables users to create requests and lab operators to take action based on those requests. Authentication to the system is integrated with EMC corporate LDAP (Lightweight Directory Access Protocol). Users of the application may be either a customer or a lab operator.

The customer is presented with the ability to file a request. The request can be supplemented with additional information for routing purposes, such as category of the request, or a specific location where the work must be performed. A confirmation email is sent to the customer when the request has been filed, and an additional email is generated for each action on the request, including assignment to a qualified lab operator. The request is viewable and updatable by both the customer and the lab operator throughout the lifecycle of the request. Figure 5 shows the user interface for creating a request (user data has been removed for business security reasons).

The lab operator is presented with a list of requests currently assigned to him or her, as well as a list of requests that have not yet been routed. Process dictates request routing, often with information provided by the customer being used to pre-sort requests to one or more managing lab operators.



Submit a Lab Request - Category Help

Requestor:	**Hayden, Jonathan**
Area:	SIMULATOR
Category:	OTHER
Lab:	Not Sure
Priority:	High
Notify List: (Double-click to Select)	[redacted] De Smet, Jochen [redacted] → jonathan.hayden@emc.com; Andrew.Bartley@emc.com; jochen.desmet@emc.com;

Request:

Please clone simulator with serial number BDH000234234.

Figure 5: Lab Request Ticketing System - Enter Request Interface

Proper management of lab operations requires deep understanding of how volume of requests and resolution times relate to business objectives. To this end, data collected on the requests are leveraged to produce metrics and reporting. Adjusting temporary staffing for projected swells in workload is an effective approach to enable lab operators to perform tasks within the accepted SLA (Service Level Agreement). High volume of similar requests may be evaluated as the target of an automation effort to further drive efficiencies.

Armed with the metrics and reporting provided in the Lab Request Ticketing System, several common, automatable tasks have been identified which are enabled through integration with other systems in the LaaS ecosystem. One such task is the deployment of a virtual machine from a standard catalog of images. By selecting the appropriate parameters when creating a ticket, the customer is able to initiate an automated VM deployment. This automation is the result of custom software that interacts with both the Lab Request Ticketing System and the Virtual Infrastructure environment to deploy VMs based on specifications set forth in the request. After the virtual asset has been created, connection detail and documentation is sent through the ticketing system as an update to the request.

The Lab Request Ticketing system is implemented as a web application with a SQL database for persistent data storage. The database and application servers are virtualized. An email integration system is also implemented which allows email-based notification on the progress of a request. Types of data stored include detailed user information, the content of a request, as well as metadata required for a lab operator to take action on the request. Additionally, history on all requests is stored in the database. History-generating events trigger an email notification to the requestor, the currently assigned lab operator, and any email address explicitly added to the notify list. Figure 6 shows a high-level diagram of the systems architecture.

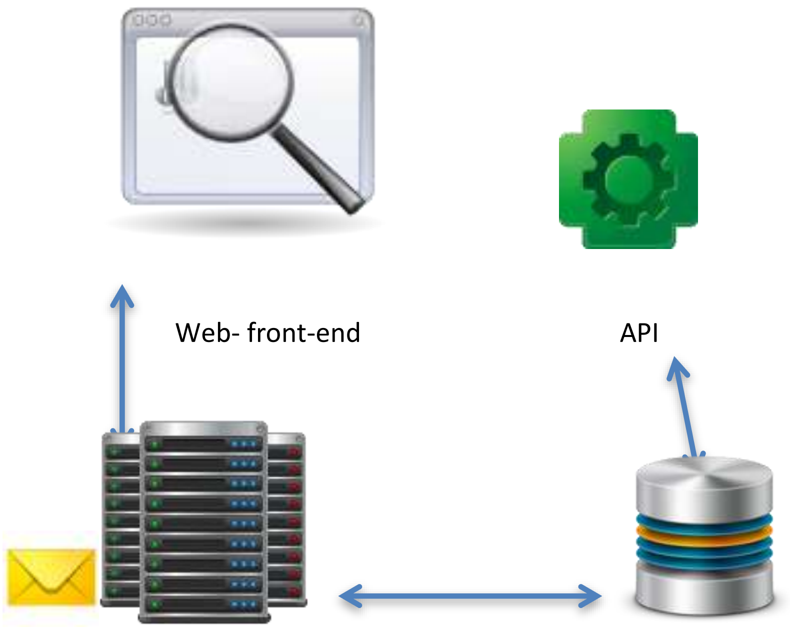


Figure 6: Lab Request Ticketing System Architecture

The Lab Request Ticketing system is a crucial building block in the LaaS environment, as it provides structured communication between lab operators and customers, produces metrics necessary to make business decisions, and provides a point of integration to the remainder of the environment.

Lab Information Web Application

Organized access to lab assets is a key factor in the success of any LaaS effort. This article will introduce the Lab Information Web Application implemented by our team.

The web application provides the following feature set to the LaaS environment.

- Role-based access to lab assets
- Self-service actions
- Automated data collection from assets
- Reporting

Authentication for the Lab Information Web Application is integrated with EMC LDAP, and authorization is maintained in the application. The application uses a concept referred to as "pool" to group collections of users and machines. A machine may be in many pools to facilitate equipment sharing, and a user may be in many pools (see Figure 7). A user's default view is comprised of a list of machines in their pools with some high-level information about the systems. Additional details for each machine are available at the click of a button. So-called pool owners may choose to divide equipment and user access to best align to various development styles or projects.

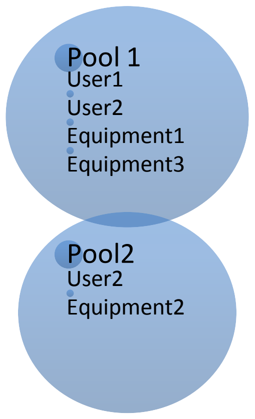


Figure 7: Pools Diagram

In addition to viewing details about a machine, the Lab Information Web Application allows the user to create a reservation against a system. A reservation provides the user with uninterrupted access to a system and enables the user to perform several self-service actions against the system. The self-service actions available to the reservation holder include filing a request to the Lab Request Ticketing System, performing an automated reimage, and creating annotations on the system. A request generated through the Lab Information Web Application automatically includes detailed system information as well as routing information in the request, reducing overall request resolution time. The automated reimage functionality is another example of integration between the building blocks of the LaaS environment. The automated reimage requires connection information stored in the Lab Information Web Application, records stored in the PXE Server, and access to the image list maintained in the Image Repository.

Much of the detailed information displayed for each system in the Lab Information Web Application is collected directly from the systems. A full inventory of hardware, firmware, and software is collected and displayed for the user to see. This information allows the user to make an educated decision about which system is appropriate for their development or test work. Data collection is achieved through a number of means, depending on the system being collected from. One method is an agent that runs on the system. The application server may send instructions to the agent to collect data and return with the results. Another method used is a “crawl”, wherein the data collection server connects to each system without an agent and runs collection scripts. Collected data is stored in the database.

The Lab Information Web Application is implemented as a client-heavy web application that communicates to the application stack through a RESTful API. Persistent data is stored in a SQL database. Figure 8 depicts the Lab Information Web Application System Architecture.

The data collected by the Lab Information Web Application drives actions in the LaaS environment. Out-of-date hardware revisions can be targeted for mass replacement. Firmware and software revisions can be targeted for mass update or to drive notifications to the community about violations of Minimum Acceptable Level (MAL) for various testing or development applications. Data about current software configurations and runtime metrics can be used to flag systems that are likely sitting idle for closer inspection. High-level views of the pool structure can be leveraged to visualize system allocation by functional team, by program feature, or other pivot. The data makes it possible to plan for large-scale actions and to keep the labs proactive rather than reactive.

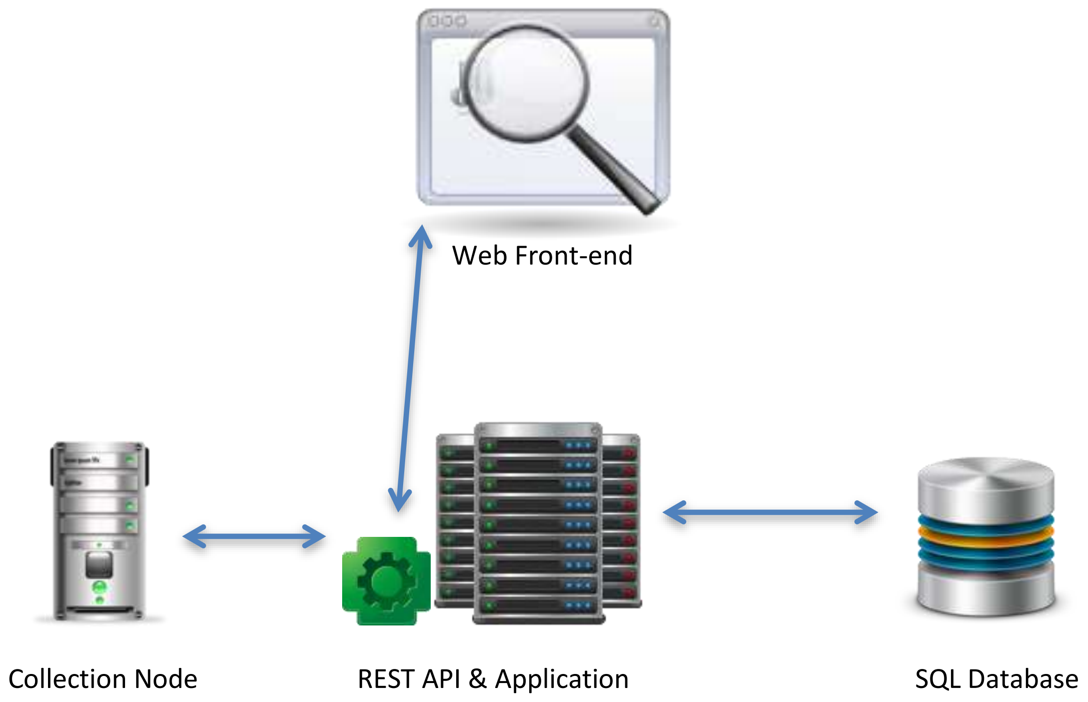


Figure 8: Lab Information Web Application System Architecture

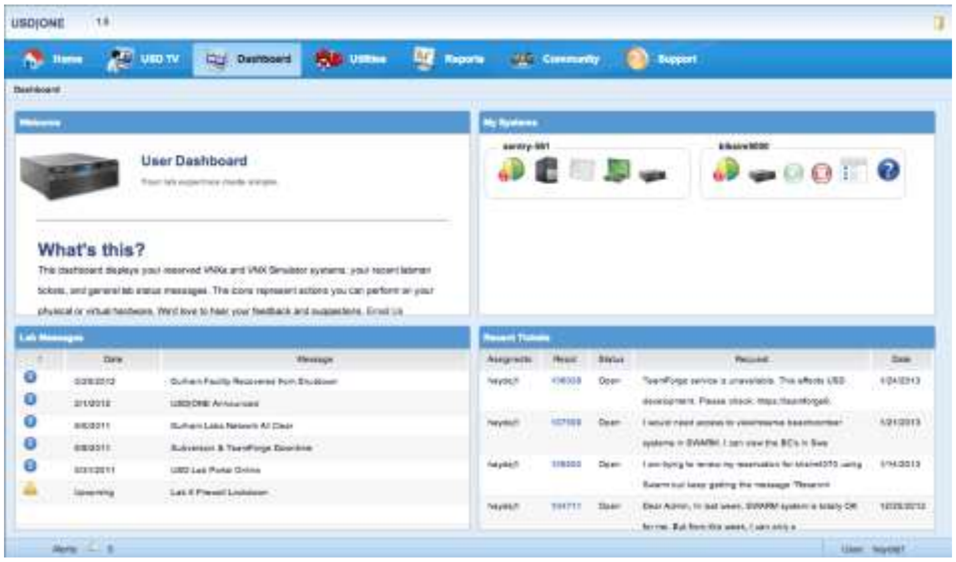
The Lab Information Web Application gives LaaS customers efficient self-service options and provides the business with critical data.

Service Portal

The Service Portal is a web application designed to bring together numerous tools and web applications into a single point of access. Among the services accessible through this interface are the Lab Request Ticketing System, the Lab Information Web Application, and the PXE server web interface. Integration between the Service Portal and the other services mentioned in this case study is not limited to simple links. Data from the various sources are combined into views and reports such as a developer dashboard, which shows systems reserved by the user, as well as status on requests.

From the Service Portal, a user has complete access to their development and test environment in the labs. Simple navigation and quick access to the automation around the user’s equipment allows the user to easily initiate self-service actions outlined above and communicate effectively with the lab operators.

The Service Portal is implemented as a client-heavy web application with a SQL database backend. Integration with other tools and automation scripts is achieved mainly through accessing data and performing actions through the APIs exposed through other LaaS systems, including the Lab Information Web Application, the PXE Web Page, and the Lab Request Ticketing System. Visualized styling of the Service Portal is modeled after the control interface for the main product lines of the business.



User Dashboard

[illegible]

What's this?

This dashboard displays your reserved VNXe and VNX Simulator systems, your recent labman tickets, and general lab status messages. The icons represent actions you can perform on your physical or virtual hardware. We'd love to hear your feedback and suggestions. Email Us

	Date	Message
	3/28/2012	Durham Facility Recovered from Shutdown
	2/1/2012	USD|ONE Announced
	8/8/2011	Durham Labs Network All Clear
	8/8/2011	Subversion & TeamForge Downtime
	3/31/2011	USD Lab Portal Online
	Upcoming	Lab X Firewall Lockdown

Assignedto	Result	Status	Request	Date
[illegible]	[illegible]	Open	TeamForge service is unavailable. This affects USD development. Please check: https://teamforge/	1/24/2013
[illegible]	[illegible]	Open	I would need access to [illegible] systems in SWARM. I can view the BCs in Swa	1/21/2013
[illegible]	[illegible]	Open	I am trying to renew my reservation for [illegible] using Swarm but keep getting the message "Reservati	1/14/2013
[illegible]	[illegible]	Open	Dear Admin, till last week, SWARM system is totally OK for me. But from this week, I can only a	12/28/2012

Alerts 0 User: [illegible]

Figure 9: Service Portal

The Service Portal gives LaaS users an effective one-stop-shop for accessing all of the LaaS features discussed in this case study.

Business Benefits

While the intangible benefits of these best practices may make the lab environment friendlier, only by identifying business benefits will a proposal for a lab architecture change ever be approved. Using the architecture described in the sections above, this case study has deployed a LaaS solution at a Massachusetts lab site; some of the business benefits are presented in the following section.

While there are numerous benefits of implementing LaaS, for space reasons this article will only quantitatively demonstrate the most prominent business benefits: faster time-to-resolution for reimaging requests, and time saved through self-service, automated service requests.

The self-service lab equipment reporting is capable of reporting on IP management, connection information, configuration information, even usage statistics; this tool can serve as the single source for lab-related information within a lab organization, enabling different functional groups to make strategic decisions using “one version of the truth”. The benefits of on-demand power cycling are more tangible: customers can reboot their equipment on-demand, rather than filing a service request and waiting for lab staff to complete the request. This can rapidly accelerate projects within the labs, particularly if the labs do not have overnight or weekend staffing.

The first quantitative benefit to be discussed is the faster time-to-resolution (TTR) for reimaging services requests. In a development environment, a common service request is to put a new OS image onto lab hardware. Certain components of the Virtualized LaaS Infrastructure (specifically, the PXE server, TFTP server, Image repository, and DHCP server) facilitate this type of request, resulting in faster service request TTR.

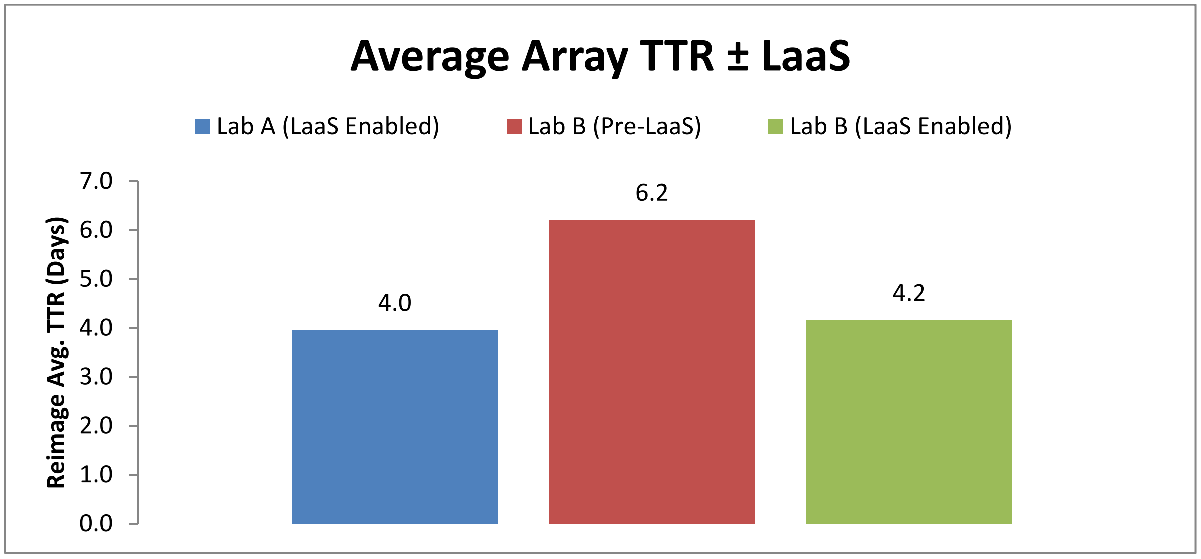


Figure 10: LaaS Reimage TTR Comparison

Though the specifics of this case study have been removed for business security, Figure 10 shows that labs with LaaS enabled (specifically the PXE server portion) have a 32% reduction in average time-to-resolution. Lab A data points have always had LaaS, while Lab B has both pre- and post- LaaS data points to observe; the LaaS-enabled time-to-resolution averages are comparable in both labs. Reduced hands-on-time will benefit any lab organization, and that benefit will scale along with the organization’s growth.

Our LaaS environment takes full advantage of virtualization's facilitation of automation (ISACA, 2010). Using the Lab Information Web Application, the case study in this article has created the opportunity for users to access lab services on-demand. Scripts have been written for the Lab Information Web Application that enable users to initiate certain reimaging functions which will then automate the reimaging of their system. To date, this automated reimaging has been executed over 1,000 times. That is: over 1,000 requests that have been completed without any hands-on time, allowing lab staff to focus on other requests. More importantly, this enables the developers and testers of the organization to return to their work more quickly after their machines experience severe software problems.

Summary

This article presents best practices for deploying LaaS within a text/development lab or data center, and provides several examples with data from a real-world deployment in a text/development lab environment. It provides an overview of the LaaS architecture, followed by a more detailed explanation of the individual systems. Finally, the article identifies business benefits of implementing LaaS in hopes that other organizations will begin their transition to this new model of lab service delivery.

Through the architecture presented within this article, development/test labs and data centers can better meet their service goals as budgets shrink and performance expectations rise. Virtualizing key elements of lab services is the first critical step toward offering LaaS to customers. The standardization and agility of virtualization will enable the organization to begin automation of certain tasks, enhancing the team's ability to respond and provide customers with self-service opportunities for even faster lab responses.

Works Cited

A E Dembe, J. B. (2005). The impact of overtime and long work hours on. *Occup Environ Med*, 588–597.

H. Randolph Thomas, K. A. (2012). Scheduled Overtime and Labor Productivity: Quantitative Analysis. *American Society of Civil Engineers*, 0733-9364.

ISACA. (2010 йил October). *Virtualization: Benefits and Challenges.* Retrieved 2013 йил 10-January from ISACA: http://www.isaca.org/Knowledge-Center/Research/Documents/Virtulization-WP-27Oct2010-Research.pdf?id=237fd5a5-009d-443c-8396-725ed8b3d1a1

James Herbsleb, a. A. (2001). Challenges of Global Software Development. *Proceedings of the Seventh International Software Metrics Symposium*, 1-3.

Sabine Sonnemag, F. C. (1994). Stressor-burnout relationship in software. *Journal of Occupational and Organizational Psychology*, 327-341.

Sami ul Haq, M. R. (2011). Issues in Global Software Development: A Critical Review. *J. Software Engineering & Applications*, 590-595.

VMware (2011). *Business and Financial Benefits of Virtualization.* Palo Alto: VMware.