Cloud Testing

Cloud-Based Test Dev. 101
By Chris Riley

Make Design a Priority for Cloud Testing
Joe Luthy

What Exactly is Cloud Testing Anyway?
Ole Lesmar

SLAs and Cloud Testing
Michael Hackett
If you are reading this issue, you are probably aware of the impact on the business world of cloud computing. Most people do not have a good grasp on what the cloud is or how people and products can use it. By the way, you are already a cloud user. If your email is stored somewhere “on the internet,” like YahooMail, Gmail, Hotmail - you are using the cloud. If you use an online tax service, TurboTax or others - you use that software-as-a-service in the cloud. The cloud is where they calculate your tax and store your tax data. If you create or share Google docs, store or transfer files using Dropbox, or stream movies on Netflix, you are interacting with the cloud.

From Salesforce to Groupon to LinkedIn, the rapid growth of all things mobile is driving cloud development. Many different forces are converging to push applications, products and services into the cloud.

As is often the case for new technologies, test teams are at the end of the information trail. What cloud architecture issues need to be tested, what services need to be tested, what SLAs need to be tested? The explosion of devices and appliances using the cloud impacts compatibility and UI testing.

We need to better understand what the cloud is, what people mean when they use phrases like a cloud of APIs, personal cloud, hybrid cloud, etc.

On the other hand, remember - cloud is delivery, not development. The differences between cloud development and other development implementations is a very hot topic in the dev. world. Forrester’s 10 Cloud Predictions for 2013 include Developers waking up to the fact that development isn’t all that different in the cloud.

Many testers say testing things in the cloud is no different than web testing, thick server, thin client testing. It’s no different, or the difference is in the specific testing issues like scalability, security, performance. There is no difference in functionality, cross browser, mobile, UI and usability, localization - most aspects of testing remain the same. Cloud is delivery, not development.

In this issue we try to demystify the cloudy cloud. The articles and the glossary, in addition to past articles and publications on cloud computing give you a broad and better understanding of the testing implications around cloud computing.

In my article I outline how to approach SLAs when working in the cloud; SmartBear’s Ole Lesmar looks at some special considerations that should be applied to applications running in the cloud; Ben Rothke reviews the book Testing Cloud Services: How to Test SaaS, PaaS and IaaS; Chris Riley of Cloudshare tackles issues relating to software and hardware licensing for cloud projects and LogiGear’s Joe Luthy emphasizes that the right skills and experience is essential for designing tests for large-scale cloud deployments.

You can view past cloud articles here and our 2014 editorial calendar here.

We’re always open to new submissions!

As holidays approach and we finish 2013, we very much look forward to a bigger, brighter, happier coming new year!
In this Issue

4 IN THE NEWS

5 WHAT EXACTLY IS CLOUD TESTING ANYWAY?

Ole Lesmar, SmartBear Software

Special considerations that should be applied to an application running in the cloud.

7 CLOUD-BASED DEV. 101

Chris Riley, Cloudshare

Throw away clunky hypervisors, and stop thinking about computer hardware and software licenses during your development projects.

9 SLAs AND CLOUD TESTING

Michael Hackett, LogiGear Corporation

A test team’s job is to report test results, not set or guarantee you will meet the SLAs.

10 BOOK REVIEW

Ben Rothke, Wyndham Worldwide Corp.

A review of “Testing Cloud Services: How to Test SaaS, PaaS and IaaS” by Kees Blokland, Jeroen Mengerink and Martin Pol.

12 CLOUD COMPUTING GROWTH

Intriguing statistics on the massive growth of cloud computing and what to expect in the future.

13 MAKE DESIGN A PRIORITY FOR CLOUD TESTING

Joe Luthy, LogiGear Corporation

Having the right skills and experience, even if you have to go outside, is essential for designing tests for large-scale cloud deployments.

15 CLOUD TESTING GLOSSARY

Key terms used when discussing cloud testing.

16 CHÈ - VIETNAM’S SWEET SOUPS

Brian Letwin, LogiGear Corporation

Rich or poor, rain or shine, hot or cold, chè is one of Vietnam’s most unique culinary offerings. Once you get over the fact that there are beans in your dessert, you’ll be hooked for life.
Lloyds Bank Using Service Virtualization to Cut Costs

Lloyds Bank’s use of service virtualization has helped it cut app development costs by reducing testing times and highlighting defects earlier so that they can be fixed less expensively.

The bank, which uses CA Technologies’ LISA service virtualization, told the CA Expo 13 conference in London how its DevOps team used the tool to help deliver major business changes - expanding its internet channel, developing its mobile proposition and optimizing business processes - more cost effectively.

"By delivering virtualized environments, we can do testing much quicker and cheaper than traditionally," said Nick Stemp, principal lead architect at Lloyds Banking Group.

LogiGear Magazine 2014 Editorial Calendar Announced

We look forward to an exciting and full 2014. As we start the new year, we will continue to bring you current articles and videos on the latest trends in software quality. Our editorial calendar is filled with engaging themes addressing software developments, ongoing research and leading individuals in the industry.

As it is difficult to keep pace with the speed of which technology advances, we would like to invite those who have a story in mind and interested in publishing to our readers – be it an interesting perspective or memorable experience on software testing, we are interested in publishing articles or columns from professionals in the field. Our focus for 2014 is getting the next generation of software quality experts published and in the news!

Here are the exciting magazines we have lined up for 2014:

**February** - Test Methods and Strategies  
**May** - Test Tools and Automation  
**July** - Embedded Systems and Things That Think  
**September** - Mobile  
**December** - Offshoring & Outsourcing


Video: What is Cloud Computing?

Traditional business applications and platforms are complicated and expensive. They need a data center, a complex software stack and a team of experts to run them.

This short video explains what Cloud Computing is and why it’s faster, lower cost and doesn’t eat up your valuable IT resources.

You can watch the full video from our friends at Salesforce [here](http://www.logigear.com/magazine/issue/news/editorial-calendar-and-submission-guidelines/).
Over the last weeks, I have found myself in several rather intense discussions about “cloud testing”: what it is, what it isn’t, and what it means for testing and QA professionals. The major source of confusion in these discussions usually revolves around the definition of cloud itself; if you try look up cloud computing on the Internet, you will find it hard to get a formal definition. Wikipedia says it outright: “Cloud computing is a jargon term without a commonly accepted non-ambiguous scientific or technical definition.” Duh!

Since the primary goal of this article is to talk about cloud applications and testing thereof, I’ve tried to distill a number of common traits for “cloud applications” based on the above and similar articles (let me know if you disagree!).

Common for “cloud applications” is that they:

- Run on a virtualized hardware and a software stack that can be moved and replicated between physical machines as needed.
- Share common physical resources with other cloud applications (disk, network, data stores, etc.).
- Are built to be highly scalable in real-time – meaning that they can handle high increases in load by dynamically scaling to more physical resources as needed.
- Are predominately accessed using standard network protocols.
- Use HTML and other web technologies for providing both their front-end and management UIs.
- Provide APIs for integration and management – possibly made available to users or third-party application vendors.
- Consume third-party APIs for providing common services and functionality – things like authentication (OpenID, Facebook), storage (DropBox, Google Drive), messaging (Twitter, Gmail), geo-functionality (GooglMaps) etc.
- Tend to use NoSQL data stores – primarily for managing large amounts of unstructured data.

The “cloud” itself comes down to being the infrastructure that hosts a “cloud application”; it is usually either public (Amazon, Rackspace, etc.), private, or a combination of the two – and can offer many different levels of service (IaaS, PaaS, SaaS, etc.).

So, given these basic characteristics, what should testers be thinking of when tasked with testing a “cloud application” – or more likely, a web application or API that is running “in the cloud”? Are there any specific traits related to a cloud application that mandate extra consideration as opposed to if the application is deployed on an old and dusty server in the corner of your office (running Windows 2000?)?
My immediate answer to this question used to be “No.” A web application or API needs to be tested in the same way no matter how it is deployed. It still has to work and perform as required, and testing is no different for different deployment scenarios. Or so I thought - thanks to my colleagues’ persistence, I have started to open up to a more nuanced answer; perhaps there are a number of aspects that a tester needs to give special consideration to when tasked with testing a “cloud application” – many of which are related to the infrastructural nature of the cloud. For example:

- **Performance** – Applications running in a cloud run on hardware that you might not have any control over, and that they share with other applications. Therefore, ensuring both performance and required scalability is extremely important. Be sure to test performance in a cloud environment similar to the one you will be using in production. If you know that your application shares resources with other applications under your control, run load tests on both at the same time to see if they affect each other. In production, be sure to use monitoring as a means to continuously validate both performance and functionality while your application is in production – ensuring that it scales as required. Using cloud resources to scale under load can be costly, so knowing where that breakpoint is and monitoring to see how close you are to it can also help you budget correctly for your infrastructure needs.

- **Security** – Since cloud applications usually share resources and infrastructure with others, you have to give extra consideration to data privacy and access control issues. Is sensitive data encrypted when stored? Are access control mechanisms in place in all possible situations and at all levels? This is just as valid for applications hosted in a private cloud; data intrusion and “theft” could even happen “by accident” if, for example, a backup for one cloud application happens to access resources or data related to another application.

- **Third-party dependencies** – Cloud applications are likely to consume external APIs and services for providing some of their functionality. You should consider testing and monitoring these as if they were part of your own solution (since that’s what they are from your users’ point of view). You want to make sure they work as you need them to and you want to be the first to know when they don’t.

One common interpretation of “cloud testing” that many vendors seem to adhere to is using the cloud to run or manage the tests themselves. For example, testers can use the cloud to generate massive distributed load tests, simulate a large number of mobile devices, or run functional and performance monitors from all over the world. Although these are all extremely valuable offerings themselves, they are not very specific for testing cloud applications. So, calling it “cloud testing” is kind of a stretch in some situations.

So, did this initial insight help me in understanding what “cloud testing” is? Well, to be honest, not really. Although I do agree that there are things to be aware of when testing an application in the cloud, I still struggle with “cloud testing” being something separate than “regular” performance, integration or security testing.

All of these just need some special consideration and understanding when applied to an application running in the cloud.

This piece was originally published on Network World.

**About Ole**

Ole Lensmar is Chief Architect and co-founder of SmartBear Software in Sweden, formerly the maker of SoapUI, Eviware Software, acquired by SmartBear in 2011.

He also co-founded base8, an XML oriented consulting company in 1996, acquired by the publicly traded Mogul in 1998. Ole worked as CTO, product owner and lead evangelist for Mogul’s software portfolio, including an XML based CMS and a high performance search engine. With Niclas Reimertz, Ole created SoapUI, now the most used open source testing tool in the world with five million downloads and one million active users.

Ole’s astute eye for technology trends and the test community led to his promotion to chief architect at SmartBear. Living his passion for software development, he writes a weekly column for Network World about software quality and blogs in general about coding, quality
The first thing you think about when you hear “The Cloud” may not be development and testing. The Cloudy market is filled with SaaS applications, hosting, and cloud-based file systems. All are very useful, and offer a clear value. However, development and Cloud testing is one of the easiest ways to actualize the benefits the Cloud can provide. By utilizing the Cloud for your development projects you immediately save time, maximize efficiency, alleviate frustration, and increase the overall value of your code.

All development projects start with a development environment. A development environment does not consist only of an IDE. It consists of hardware, operating systems, software licenses, APIs, and your favorite IDE. The challenges with local development environments are:

1.) Costly infrastructure: Though the relative cost of computer hardware has dropped dramatically over the years, the cost of ad-hoc development machines is still considerably high. The reason for this is because development environments are not consumed the same as work machines are. There are additional costs associated with software licenses, high-performance hardware, more robust hardware, or pushing hardware limits with a local hyper-visor. Not to mention the opportunity cost associated with using a local machine as a development machine versus a line of business machine.

2.) Limited amount of resources: Most local development environments are not ideal, or remain useful for a limited time. For this reason developers find themselves with too little memory, too little processing power, and having to upgrade regularly. If the project is server related, most local machines are not capable of running server grade software, or applications without heavy hyper-visors.

3.) Environments are used for more than development: Unfortunately, it’s hard to justify dedicated hardware for development projects. Very often developers not only have their IDEs on a development machine. They also have their email clients, their favorite web browser, etc. This may interfere with the development environment. It can be avoided by having a local hyper-visor. However, this puts an even greater burden on the hardware and performance of the machine.

4.) Environment contamination: Because it’s hard to silo development environments, an environment is often contaminated with past projects, APIs, hot-fixes, or other software that could otherwise interfere with the development process. This contamination is often hard to spot when it causes issues, and a common instigator of extreme frustration.

5.) Transition to demo: Even when having a local development machine is successful, transitioning from development to testing, staging, demo, or production can be tricky. Often it’s not just a matter of copying code, it’s also necessary to make sure all pre-requisites are installed, configurations are correct, and the downstream environment is as identical to development as possible. This is often a manual process.
Having a local development environment in the past had only the real advantage of convenience and performance, but anymore this is not the case. Enter the Cloud:

1.) In the Cloud you get the latest hardware and performance: Cloud providers like CloudShare leverage the latest hardware and software. At CloudShare, we are able to utilize economies of scale to leverage the latest and greatest in high-performance hardware and software. Not only that we have put specific R&D effort into accelerating the performance of our hardware and the delivery to you over the internet. Leaving the user feeling like the machine is right in front of them. Many users remark how using CloudShare virtual machines actually out-perform their local hardware.

2.) In the Cloud deploying pre-installed licenses is easy: How many times has the completion of a software install slowed you down? The Cloud infrastructure allows cloud providers to pre-install critical software. At CloudShare, we identify for you the most common development environment software, and pre-install it for you. We pre-install the operating system, Office, SharePoint, Visual Studio, for example. We also offer Linux, Eclipse, and other common development environments. This means you have to spend zero time or effort thinking about software. Not only that the time it takes to install all that software is significant, with CloudShare we reduce the time dramatically.

3.) In the Cloud it’s easy to replicate and start over: Sometimes things go wrong in development, and often when they do the best thing to do is just start over. This can be extremely frustrating. In the Cloud, it’s easy, and frustration free. With CloudShare its four steps and 15 minutes, not hours, or days. If your development environment has gone sour, simple delete it, and start over with the same VM template, or a snapshot of a previous version. Easy!

4.) In the Cloud you can silo development environments: Imagine how nice it would be to have one development environment per project. Keeping all configurations, and code separate. Allowing for multiple projects to go on at once without concern of interferences. With CloudShare, you can fully silo all your development projects with ease by creating a VM or even a complete environment for each project. This allows you to easily manage the progress of one project and keep projects from impacting each other.

5.) In the Cloud projects are contamination free: Now you can fully separate your local work machine from your development machines without the added overhead of a local hypervisor. In CloudShare, you can separate your development tasks from all your personal and business applications. This prevents the use or existence of these applications from interfering with development. Not only that, one project’s pre-requests will not impair an others.

6.) In the Cloud you can scale on the fly: Use the hardware you need when you need it. You can toggle how many CPUs, how much memory, how much hard-drive space you need for your development environment. With CloudShare, if you need to max out hardware for performance testing, you don’t have to be committed to it like you are with a local machine. Simply throttle your hardware resources up and down as you need them.

The Cloud is now even better than you thought it was. It is not just about production business applications anymore. It’s a more efficient way to develop your applications. Throw away clunky hypervisors, and stop thinking about computer hardware and software license during your development projects.

About Chris

Chris represents the coders and practitioners that use CloudShare as technology evangelist. He fancies himself a technologist & and a recognized industry expert in Content Management, Information Architecture, BigData, Text Analytics, and Cloud Virtualization. He commonly speaks and authors content on these topics.
SLAs and Cloud Testing

A test team’s job is to report test results, not set or guarantee that you will meet the SLAs.

By Michael Hackett, LogiGear

In the rush to cloud services, with everything-as-a-service, you will hear people talking about SLAs. What is this about and what does it have to do with testing?

A Service Level Agreement, or SLA, is a contract a service provider promises for a defined level of service, such as response time, throughput or capacity.

When a customer signs up for service, the provider promises, in contract, certain levels of service. The most important aspect is usually availability.

Availability is the ability to access the system. Everyone wants their service available all the time. This is an impossibility for both good and bad reasons. Good reasons - downtime, patches, new build migration and system upgrades. Bad reasons - system crashes, security problems - denial of service, network/infrastructure problems.

Downtime happens and SLAs are meant to provide a promise from the provider, of how available the system will be.

This is an important part of marketing, sales and contracts for any cloud service provider, from HP and Amazon to consumer products in the cloud like Netflix and Foursquare.

Gartner analyst Lyida Leong blogged that Amazon Web Services, which Gartner named a market-leader in infrastructure-as-a-service cloud computing, has the "dubious status of ‘worst SLA (service level agreement) of any major cloud provider.” She also wrote, “HP’s newly available public cloud service could be even worse.”

What are reasonable SLAs for availability? What is common? The answer differs based on the service. For example, many people use “4 9s” which represents 99.99% uptime:

<table>
<thead>
<tr>
<th>Availability %</th>
<th>Downtime per year</th>
<th>Downtime per month</th>
<th>Downtime per week</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.99% (&quot;four nines&quot;)</td>
<td>52.6 minutes</td>
<td>4.32 minutes</td>
<td>1.01 minutes</td>
</tr>
</tbody>
</table>

Think about this: four nines availability allows 1 minute downtime per week. Wow. How safe do you think it is for a company to guarantee this? In one full year, that means down time of less than an hour.

So what does this mean for testing? Testing SLAs is all about system performance testing; load testing, stress testing. It is measurement of the various attributes of the product; capacity, response time, against agreed upon standards. What you have to remember is a test team’s job is to report test results, not set or guarantee you will meet the SLAs.

About Michael

Michael Hackett co-founded LogiGear in 1994 and leads the company’s LogiGear University training operations division, setting the standard in software testing education programs for many of the world’s leading software development organizations.

Michael is co-author of the popular Testing Applications on the Web (Wiley, 2nd ed. 2003), and has helped many clients produce, test and deploy applications ranging from business productivity to educational multimedia across multiple platforms and multiple language editions.
The book is an incredibly effective and valuable guide that details the risks that arise when deploying cloud solutions. More importantly, it provides details on how to test cloud services, to ensure that the proposed cloud service will work as described.

It is a great start to the topic. The 6 chapters detail a paradigm that cloud architects, managers and designers can use to ensure the success of their proposed cloud deployments.

The first two chapters are a very brief introduction to cloud computing. In chapter 3, the authors detail the role of the test manager. They write that the book is meant to give substance to the broadening role of the test manager within cloud computing. They encourage firms to make sure the test manager is involved in all stages of cloud computing; from selection to implementation. In fact, they write that it is only a matter of time until this service will be available in the cloud, in the form of TaaS – Testing as a Service.

Besides the great content, the book is valuable since it has many checklists and questions to ask. One of the reasons cloud hype is so overly pervasive, is that the customers believe what the marketing people say, without asking enough questions. It would have been an added benefit if these questions and checklists would be made available in softcopy to the reader.

In chapter 4, the book details performance risks. As to performance, an important aspect of selecting the correct cloud provider is scalability of the service. This then requires a cloud specific test to determine if the scaling capacity (also known as elasticity) of the provider will work efficiently and effectively in practice.

An extremely important point the authors make is that when choosing a cloud service, many firms don’t immediately think of having a test environment, because the supplier will themselves test the service. The absence of a test environment is a serious risk.

About 2/3 of the book is in chapter 5 – Test Measures. The chapter mostly details the test measures for SaaS, but also does address IaaS and PaaS testing. The chapter spends a lot of time on the importance of performance testing. An important point detailed in the chapter is...
that of testing elasticity and manual scalability. This is an important topic since testing elasticity is a new aspect of performance testing. The objectives of elasticity tests are to determine if the performance of the service meets the requirements across the load spectrum and if the capacity is able to effective scale. The chapter details various load tests to perform.

In the section on guarantees and SLAs, the authors make numerous excellent points, especially in reference to cloud providers that may guarantee very high availabilities, but often hide behind contract language. They provide a number of good points to consider in regards to continuity guarantees, including determining what is meant exactly by up- and down-time; for example, is regular maintenance considered downtime or not.

Another key topic detailed is testing migration. The authors write that when an organization is going to use a service for an existing business process, a migration process is necessary. This includes the processes of going into the cloud, and backing the service out of the cloud.

With all of the good aspects to this book, a significant deficiency in it is that it lacks any mention of specific software testing tools to use. Many times the authors write that “there are many tools, both open source and commercial, that can” but fail to name a single tool. The reader is left gasping at a straw knowing of the need to perform tests, but clueless as to what the best tools to use are. Given the authors expertise in the topic, that lacking is significant.

The only other lacking in the book is in section 5.3 on testing security, the authors fail to mention any of the valuable resources on the topic from the Cloud Security Alliance. Specifically the Cloud Controls Matrix (CCM) and Consensus Assessments Initiative (CAI) questionnaire.

With that, Testing Cloud Services: How to Test SaaS, PaaS IaaS should be on the required reading list of everyone tasked with cloud computing. This is the first book to deal with the critical aspect of testing as it related to cloud computing. The ease of moving to the cloud obscures the hard reality of making a cloud solution work. This book details the hard, cold realities of turning the potential of cloud computing, in the reality of a working solution.

Had the designers of the Obamacare website taken into consideration the key elements of this book, it is certain that the debacle that ensued would have been minimize and the administration would not have had to send out a cry for help. The Obamacare website will turn into the poster child of how to not to create a cloud solution. Had they read Testing Cloud Services: How to Test SaaS, PaaS IaaS, things would have been vastly different.

About Ben

Ben Rothke, CISSP, CISM, CISA is an information security manager with Wyndham Worldwide Corp., the world’s largest hospitality firm. Rothke has over 15 years of industry experience in information systems security and privacy. His areas of expertise are in risk management and mitigation, security and privacy regulatory issues, design & implementation of systems security, encryption, cryptography and security policy development, with a specialization in the financial services and aviation sectors.

He is the author of “Computer Security - 20 Things Every Employee Should Know” (McGraw-Hill). He is also a frequent speaker at industry conferences, such as RSA and MISTI, and holds numerous industry certifications. - See more at: http://www.rsaconference.com/speakers/ben-rothke#sthash.vszoeEyT.dpuf
CLOUD COMPUTING GROWTH

Total Size of Cloud Computing Industry to be $150 Billion in 2014 as compared to $46 Billion in 2008.

The current spending for global public IT services in 2012 was more than $40 billion, but with a CAGR (compound annual growth rate) of 26.4 percent for the period of 2012–2016, it is expected to reach the $100 billion milestone in 2016.

By 2014, 60% of server workload will be virtualized.

Of all the Cloud Computing revenues, nearly 50% will come from the US alone.

The overall Cloud Computing growth rate is 5 times more than the overall IT growth rate globally.

As of 2013, nearly 40% of the CRM systems sold globally are Cloud based.

Source: Awesome Cloud Services
Moving existing applications to a cloud environment adds new dimensions to testing. One of the primary reasons for moving to the cloud is scalability. Capacity to handle traffic and data transfer can be easily ramped up by access to additional computing resources. How this happens, and the configuration of those resources, can vary. Designing tests that can simulate these real-world scenarios and that are maintainable takes a significant amount of up-front planning.

When Cisco’s Telepresence division migrated the initial PC based applications to the cloud one of the primary goals was to automate as many of the test routines as possible, and to create the automation in a way that would allow it to scale as the application itself grew and changed. Rather than try to build the automation around a particular automation software, the focus was on test design first and tool second. In the end this proved to be the right approach.

Cloud Testing Considerations

Cloud deployments require testing the application, platform and infrastructure. Depending on whether it’s a private or public cloud there may be access issues that need to be addressed, and these will need to be accommodated in the test design. Below are several of the primary considerations the test teams for this project had to address.

Application changes and on premise interfaces
- Would moving the application to the cloud require changes to the application to fit the environment available on the cloud platform?
- How should the test cases be designed and test automation implemented to provide the required test coverage for these changes?

Data migration
- Would new tests be needed for the storage services available on the cloud platform?
- Data moved from managed servers to cloud storage may have to be encrypted for security and compliance needs. How would the additional security requirements be tested?

Performance
- Was a plan needed to load and stress test the application on the cloud platform to ensure that system response is as per SLAs?

Scalability
- Would the test scenarios be robust enough to simulate on-demand needs of the application?

**Feature**

Make Design a Priority for Cloud Testing

Having the right skills and experience, even if you have to go outside, is essential for designing tests for large-scale cloud deployments.

By Joe Luthy, LogiGear
Priorities for Testing

Testing the applications being migrated to the cloud required a careful assessment from the perspective of what changes were required for cloud deployment. This assessment helped the test team understand application areas that needed specific focus during testing on the cloud platform. In addition, knowledge of the cloud platform capabilities for scalability, availability and disaster recovery had to be disseminated to the teams to assure successful test planning.

The testing plan was based on a model that included testing across the application, platform and infrastructure.

The Process

The project consisted of separate teams for portal testing and testing the supported devices. The Portal team was responsible for testing web applications that included a user portal and an administration portal that controlled all account information and updates, subscriptions information and modifications. Proper setup of the test environment required the team to become familiar with the target cloud platform.

Both LogiGear and Cisco teams collaborated to develop the ideal test plan for the existing web applications to be deployed in the cloud. Test engineers designed and developed python-based automated test cases utilizing the Action-Based-Testing (ABT) method. The big advantage of using ABT for this project was the ability to create well-defined test cases that were able to run independently of each other. This approach localized the impact of functional changes under test allowing the automation to be easily maintained.

Maintainability of automated tests is essential to any automation project, but critical for testing of cloud applications where the application owner may not be involved in changes to the infrastructure the application runs on. It is precisely these types of projects where module-based test design proves its value.

This project required additional resources for computations and data storage to provide a true representation of the production data and expected load. Using virtual machines (VMs) proved to be the most effective way to accomplish this, plus VMs made it possible to scale quickly. After the initial testing, the VMs were saved as images in cloud storage, and all the cloud resources were released to keep the overall testing costs low. After the initial tests the VMs were restored as needed to recreate the test environment for subsequent releases.

Key Learning

Test design planning for the cloud migration project was ultimately the primary component in achieving success for this project. Automation was the only way to handle the number of tests, and the amount of data to be tested, and in turn, the module-based approach made it possible to maintain tests easily even in a very fluid development environment.

When moving to the cloud there are two important test considerations. The first is the need to test all aspects of the cloud environment, starting with the cloud client through the application (SaaS), platform (PaaS) and Infrastructure (IaaS) layers. The second consideration is the capability of the team to plan and design the tests. Having the right skills and experience, even if you have to go outside, is essential for designing tests for large-scale cloud deployments.

About Joe

Joe Luthy is the marketing director for LogiGear. Joe has over 10 years experience in product marketing for enterprise IT applications and mobile devices.

He has a strong background in TQM and implementing organizational wide quality processes.
Glossary: Cloud Testing

Cloud Computing: Cloud computing is a concept used to describe a variety of computing concepts that involve a large number of computers connected through a real-time communication network such as the Internet. In science, cloud computing is a synonym for distributed computing over a network, and means the ability to run a program or application on many connected computers at the same time. The phrase also more commonly refers to network-based services, which appear to be provided by real server hardware, and are in fact served up by virtual hardware, simulated by software running on one or more real machines. Such virtual servers do not physically exist and can therefore be moved around and scaled up (or down) on the fly without affecting the end user - arguably, rather like a cloud.

API: A language and message format used by an application program to communicate with the operating system or some other control program such as a database management system (DBMS) or communications protocol. APIs are implemented by writing function calls in the program, which provide the linkage to the required subroutine for execution. Thus, an API implies that a driver or program module is available in the computer to perform the operation or that software must be linked into the existing program to perform the tasks.

Cloud API: Application programming interfaces (APIs) used to build applications in the cloud computing market. Cloud APIs allow software to request data and computations from one or more services through a direct or indirect interface. Cloud APIs most commonly expose their features via REST and/or SOAP. Vendor specific and cross-platform interfaces are available for specific functions.

Platform as a service (PaaS): Along with software as a service (SaaS) and infrastructure as a service (IaaS), it is a service model of cloud computing. In this model, the consumer creates the software using tools and/or libraries from the provider. The consumer also controls software deployment and configuration settings. The provider provides the networks, servers, storage, and other services that are required to host the consumer’s application.

Infrastructure as a service (IaaS): In the most basic cloud-service model, providers of IaaS offer computers – physical or (more often) virtual machines – and other resources.

IaaS clouds often offer additional resources such as a virtual-machine disk image library, raw (block) and file-based storage, firewalls, load balancers, IP addresses, virtual local area networks (VLANs), and software bundles. IaaS-cloud providers supply these resources on-demand from their large pools installed in data centers. For wide-area connectivity, customers can use either the Internet or carrier clouds (dedicated virtual private networks).

Testing as a Service (TaaS): Testing as a service is executing your testing activities in the cloud. Particularly using infrastructure tools such as test case management, bug tracking, virtualized environments and test automation tools hosted and maintained by a service provider.

Thin Client: A computer or a computer program which depends heavily on some other computer (its server) to fulfill its computational roles. This is different from the traditional fat client, which is a computer designed to take on these roles by itself. The specific roles assumed by the server may vary, from providing data persistence (for example, for diskless nodes) to actual information processing on the client’s behalf.

Hybrid Cloud: Hybrid cloud is a composition of two or more clouds (private, community or public) that remain unique entities but are bound together, offering the benefits of multiple deployment models.

Sandbox: A testing environment that isolates untested code changes and outright experimentation from the production environment or repository.

Many cloud service providers have a sandbox to test and experiment with your cloud service integration before moving your product using those services to production.

Sources: Wikipedia, Techtarget.com, PC Mag
Vietnam View

Chè - Vietnam’s Sweet Soups

Rich or poor, rain or shine, hot or cold, chè is one of Vietnam’s most unique culinary offerings.

By Brian Letwin, LogiGear Corporation

Chè carts can be found at busy intersections, outside of schools and alongside parks.

While the myriad of flavors and colors are impressive unto themselves, perhaps more so is the dedication chè sellers apply to their trade. Many wake up before sunrise to prepare 6 – 10 offerings before hitting the hot, mid-day pavement. Since the soups go bad within a day, most chè vendors sell until they run out.

Due to its low price, usually around $.25, chè is particularly popular with students, who need a little something between meals, and office workers who eat them as a dessert after lunch.

Beyond the culinary importance of the soup, it serves as a focal point for social interaction as well. It’s common for groups of friends to go have chè and talk about life, gossip about co-workers, and relax.

For some westerners, the soup’s ingredients may seem a bit strange and, admittedly, it is an acquired taste. But once you get over the fact that there are beans in your desert, it’s hard to get enough.

Is it a desert? A breakfast? A snack? Chè, a variety of sweet soup, is one of Vietnam’s most versatile foods, consumed by the old and young, the rich and the poor.

There are literally dozens of versions of these pudding-like soups with ingredient's ranging from mango to mung beans. While flavors and colors differ for each type, most are prepared with many varieties of beans and rice and are garnished with coconut milk.

No matter the ingredients or the temperature when served, each name starts with chè and is followed by qualifying adjectives referring to the soup’s main ingredients.

It’s common to be served chè at restaurants or at home after a meal, but the main distribution platform for these sweet treats are pushcarts.

For some westerners, the soup’s ingredients may seem a bit strange and, admittedly, it is an acquired taste. But once you get over the fact that there are beans in your desert, it’s hard to get enough.
Speed and Collaboration
The winning combination for large-scale test automation

TestArchitect™
Go to market with confidence

TestArchitect is the modern module-based keyword-driven software test automation tool that enables teams to create, maintain and execute large-scale test automation with groundbreaking speed.

Accelerate test creation and maintenance. TestArchitect’s module-based, keyword-driven authoring platform and library of preprogrammed actions enables team members to create and update tests without programming scripts, making your entire team more productive.

Scale resources and collaborate with ease. TestArchitect makes it easy to scale up project teams as testing demands require. By combining test design and automation into one tightly-integrated process, team members with diverse skill sets can collaborate easily.

Find out more at www.testarchitect.com