Computing Department, Letterkenny Institute of Technology, Port Road, Letterkenny, Co. Donegal, Ireland.

A comparison between Cloud Solutions with focus on DevOps tooling

Author: Marcelo Rodrigues Costa
Supervised by: Ruth Lennon

A thesis presented for the award of Master of Science in Computing in Enterprise Application Development

Submitted to Quality and Qualifications Ireland (QQI)   May 2015

Declaration

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of Master of Science in Computing in Enterprise Application Development, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Signature of candidate: ____________________ Date: ___________________

Acknowledgements

I dedicate this work to my beloved wife Ana Karolina who was kind, patient and supported me during my absence and many sleepless nights. To my son Heitor, my daughter Anabela, my family in Brazil: my brother Bruno, his wife Beatriz, my niece Cecilia, my sister Larissa, her fiance Hiury and, specially my parents: Paulo and Ozelia, for believing and investing so much in me, I am truly glad to know that, despite the distance, they are proud of me and doing well.

A Special thanks to Marcelo Beckmann for assisting me with my academic research. Thanks to Pramerica Systems Ireland for letting me be part of an amazing team of Systems Engineers for two wonderful years in Donegal, also thanks for bringing me and my family to this country where I found so many interesting opportunities. Thanks to IBM, my current employer, which gives me a dynamic work environment where I learn something new every single day.

I would also like to express my deepest appreciation to my advisor Ruth Lennon, who guided me through this difficult, yet, enlightening path to produce this research and open doors to ambitious opportunities, thanks to my other lecturers: Mark Leeney, William Farrelly, Gerard McCloskey and Thomas D Devine. Thanks to Dr. Averil Meehan for her brief, yet brilliant, talk about memory management in the Java Virtual Machine. Many thanks to Thomas Dowling for the amazing opportunity to study in this institution, LYIT is a great learning environment and it expanded my academic knowledge in ways I never thought would be possible.

Abstract

The DevOps philosophy keeps gaining popularity due to its practices that allow companies to quickly deliver software changes to their customers. Along with this advent, Cloud Solutions continue to expand their customer base for Infrastructure-As-A-Service (IaaS) and Platform-As-A-Service (PaaS) offerings. The amalgamation of these two concepts result in an opportunity to significantly reduce the risks of managing the integration and delivery processes involved in the Software Development Life Cycle, by relying on agile practices and the abstraction of complex infrastructure elements.

This study illustrates how a Cloud-based DevOps environment can bring benefits to Enterprise level development teams through the comparison of two Cloud solutions: Google Cloud Platform and IBM Bluemix, by assembling a Deployment Pipeline on each one and exploring additional features that support DevOps tooling.

Keywords: Cloud, DevOps, Continuous Integration, Continuous Delivery, Google, Google Cloud, IBM, IBM Bluemix, IaaS, PaaS, Agile, Cloud Computing.

Contents

1. Introduction

This research focuses on the evaluation of two different services from the point of view of productivity, such services allow the user to interact with the infrastructure of the companies that offer them so they can create their own solutions to run in the "cloud", which means, the infrastructure is abstracted to the end-user and they can focus on the development of their application and generating value to their customers. However, even by abstracting most of the non-functional elements, that, as Mark Cade, et al. (2010, p.20) describe, are comprised by "performance, scalability, reliability, availability, extensibility, maintainability, manageability, and security", developing the software itself with the functional requirements also presents its challenges and it is pointless to have a mature reliable infrastructure if the application itself is susceptible to failures and the process to deliver code changes is not efficient.

Based on this context, this research analyzes two main points: Cloud Solutions and DevOps tools, with focus on two specific options: Google Cloud Platform and IBM Bluemix, so, to proceed with the research, it is necessary to understand what is DevOps and review some concepts involving Cloud-based services.

1.1. Brief Overview of DevOps

Strongly aligned with Lean and Agile concepts, DevOps is an idea that involves cultural changes to break "silos", which means, eliminate red tape barriers between teams in the IT Department and allow more collaboration that results in more effective and efficient processes, this collaboration supports practices to integrate and deliver software through automation. Enterprise level applications might be comprised of dozens of components and, even with the abstraction of many complex services that mature commercial middleware offers, such as, Transaction Management, Messaging or Secure communication through Cryptography, a single failure on the application side might affect the entire orchestration of a critical use case, assuming a reliable architecture, this failure can only happen due to some code that is committed to the Source Control Management (SCM) system that is not properly tested, or some component that is erroneously deployed to a "Production environment" through manual tasks and it is missing some dependencies, which could be another artifact or some configuration that was not applied correctly, the solution for such scenarios is a combination of two practices known as Continuous Integration (CI) and Continuous Delivery (CD).

1.2. Cloud Solutions

Cloud computing is leveraged in the IT Industry due to its power to handle high demand requests and the abstraction of the low-level architectural components, such as Servers, Network configuration and Load Balancing tools, with different subscription models, users can benefit from Cloud Solutions to achieve more reliable and cost-effective services. To mention some of the popular Cloud Solutions found in the market: DropBox provides storage in the Cloud, Google Application Engine allows the users to deploy Web applications written in Java or Python to a high-available and scalable application server, Amazon EC2 provides a self-service Virtual Machine provisioning platform, Microsoft Azure also allows the users to deploy Web Application to an Application Server instance that is hosted in the Cloud; these services and vendors might be slightly different but they are all trying to help their users by relieving the burden of dealing with the actual infrastructure and the complexity associated to it.

These Cloud services are categorized as Infrastructure-As-A-Service (IaaS), Platform-As-A-Service (PaaS) and Software-As-A-Service (SaaS). Starting from the most low-level option: IaaS provides Virtual Machines, storage and a network that is fully configured to connect and manage each one of these servers, in other words, a set of Infrastructure components that are available to the end-user without exposing all the details about the actual architecture, including the automated provisioning underneath the administration control panel. A PaaS system raises the level of abstraction by providing a mechanism for the user to deploy any Web Application and rely on the hosting Platform to respond to the clients demand, it reacts automatically to guarantee non-functional requirements like scalability, availability and security. The last cloud service category, SaaS, is basically a fully-developed Web Application that offers some value for the end user whereas all its internal configuration and architecture details are completely abstracted, the user usually pays some subscription fee and consumes the services provided by the application without any access to backend services like Application Servers, Storage and Databases.

The adoption of Cloud Solutions should, depending on the proper evaluation of the options mentioned above, eliminate most of the requirements of an ideal “Operations” criteria, but not all of them. The application that is deployed to the Cloud still dictates some of the guidelines of its own maintainability. The developers should not abstract the tasks that will be performed against the application once it is already deployed. The Development and the Operations team need to collaborate so the application can be designed with these considerations in mind, unfortunately, it is not always the case, as Thomas A. Limoncelli (2014, p.31) mentions, “This is in contrast to strategies where operations is an after-thought and operations engineers are forced into a position of ‘running what other people build’. That’s the outdated way”.

By leveraging Cloud Solutions, most of the Operations objectives of the application can be achieved with the PaaS solutions available in the market, such solutions accommodate capabilities like Monitoring, Graceful degradation, Redundancy and Software upgrades.

In this research, two Cloud Solutions were analyzed from the DevOps perspective, an evaluation framework was elaborated to guide the comparison and evidence which one presents the most ideal set of non-siloed tools and services for Continuous Integration (CI) and Continuous Delivery (CD). The author verified the interfaces provided to interact with the solutions’ components, ran an experiment involving the creation of a Deployment Pipeline by leveraging both PaaS and IaaS elements of each Cloud solution, deployed a sample application and changed its code to simulate a Development environment. Additional features like monitoring and bug tracking were also verified to check if the solutions facilitate tasks that fall in the scope of an Operations team.

This research aims to answer the following question: “Which Cloud Solution offers the best tooling for the support of DevOps?”.

2. Literature Survey

Cloud services can be presented through different components: Software, Platform and Infrastructure, all these services can be leveraged to facilitate the process of developing software, to name a few examples: a secure bug tracking system can keep all the users stories in the Cloud in a cost-effective way, a scalable Web Application that receives thousands of requests every day or even a reliable Cloud Storage that can be used by another application to host a large set of documents, for such services, different Key Performance Indicators (KPIs) can be defined to create evaluation frameworks that help users decide which vendor provides the best Cloud Solution that is more aligned with their needs.

In a scenario where the Software evolves, it is necessary to make sure that the integration of the components of the end product and its delivery to the customers happen in a smooth way. There are other relevant points beyond the abstraction of non-functional requirements that the Cloud offers, the tools offered by each Cloud Solution can also be evaluated to verify how a development environment can be assembled and how the changes can be integrated and delivered effectively and efficiently, such tools must be aligned with DevOps practices.

Cloud-Based solutions that can align their offerings with DevOps practices have not been widely explored and, based on the advantages of such combination, this is a valuable information for the IT industry. Such Cloud Solutions need to be properly evaluated from the DevOps optic to gather a better understanding of which features would be a better fit to hire a service that supports a highly-productive Software Development Life Cycle (SDLC).

2.1. The relevance of DevOps and Cloud Solutions

DevOps introduces a new paradigm towards efficient Software Delivery Life-Cycle (SDLC), according to Thomas A. Limoncelli (2014, p.172): “DevOps is an emerging field in operations. The practice of DevOps typically appears in web application and cloud environments, but its influence is spreading to all parts of all industries”, whereas cloud solutions continue to evolve into a reliable cost-effective set of services that can allow companies of different dimensions to reduce on-premises assets, JIA Xiaojing (2010) says that "No doubt, cloud computing is the most popular topic of the IT industry in 2009. Google, Amazon, Yahoo and other Internet service providers, as well as IBM, Microsoft and other IT companies have proposed their own cloud computing strategy”, therefore, the amalgamation of both ideas is surely relevant to the IT Industry as it lets the team focus more on the end product that they want to present to their customers, they can achieve the right abstraction of non-functional requirements with the assistance of Cloud Services and high-quality stable builds and deployments produced by DevOps practices and tools.

To reinforce how wide the scope of DevOps is, Soon K. Bang, et al. (2013) define the idea as "a combination of development and operations” and it is stated that “”diverse stakeholders are involved in DevOps, including business analysts, software developers, software testers, and quality assurance personnel for development, and system administrators, database administrators, network administrators, web masters, and security officers for operations”, this emphasizes how valuable cloud-based DevOps services can be for an organization, with the right set of tools, they can be leveraged throughout different departments.

Cloud Solutions’ offerings grow substantially and opportunities arise to leverage them to improve the Quality of Service (QoS) of existing applications, to illustrate the benefits of the Cloud adoption, Daniel Cukier (2013, p.1) describes how he successfully moved a Web Application to a Cloud infrastructure and connected other cloud-based services to improve the non-functional aspects like storage, email, load balancing and enhancing the ease of Operations responsibilities: "Today, we are running using more than 20 virtual servers. We use memcached servers to store user browser session data, Amazon S3 to store products photos and static assets.”, he even mentions some adoption of DevOps practices: “”We have a test and staging environment with continuous integration and deployment software”, however, without further details about the configuration of this development and staging environments.

In his article, he presents the rationale of each decision involving the Cloud Services they have chosen to solve a specific problem. They developed a “HTML as a Service” system that allows their users to interact, in an automated fashion, with the Brazilian Post Office website and retrieve a posting code through Selenium operations that run in background, although that seems to be more involved with a functional requirement than with the development of the application itself, automation is a very important practice within the DevOps paradigm.

There are a few “patterns” he illustrates that do not really touch the main characteristics of DevOps (Continuous Integration and Continuous Delivery), reinforcing the need for more studies involving DevOps services in the cloud. Instead of describing patterns to leverage DevOps tooling to improve the Software Development process, he presents the chosen Cloud services based on specific needs involving the application use cases: files storage, messaging queue, PaaS and email. It demonstrates that the application’s use-cases were improved, not only by the Cloud services themselves, but the smooth integration and automated deployment that is supported by such services. The DevOps elements could have been more emphasized considering the title of the article. One other pattern described in his article, “Load Balancing Application Server with memcached user sessions”, presented a good practice for a distributed architecture, which is to store user sessions’ data in a way that makes the application redeployment more resilient. This idea connects really well with Continuous Delivery since it prevents any disruption of service and allows more automation on the deployment side.

The last patterns discussed in Daniel’s article focus on both Logging and Monitoring, which is also one of the pillars of DevOps, Juan F. Pérez, et al. (2105, p.1) describe how important is the visualization of the application’s data from the DevOps perspective: “Recent years have seen the rise of the DevOps approach for software development, which aims at closing the gap between development and operations, providing timely feedback to the application developer to speed-up the development cycle.”. The collaboration results in greater knowledge sharing involving the application and the infrastructure requirements. Several examples of Real User Monitoring solutions are presented, for example, Zabbix, Statsd and Pingdom, all these recommendations are valuable pieces of information when it comes to assemble a DevOps-focused development environment.

The author agrees with the statement from Daniel Cukier’s conclusion (2013, p.10) where it is mentioned that he “did not cover very important topics like continuous delivery, deployment, configuration management, software quality process and tests. These are some issues that could complement this catalog to form a Pattern Language with DevOps practices to scale web-based business using the cloud.", however, from the scalability point of view the selected services serve as a good reference for the integration of existing on-premises applications with Cloud solutions.

2.2. Security considerations for DevOps practices in the Cloud

Security is also a fundamental concern that is mostly handled within the Operations scope, however, the idea of introducing DevOps to the enterprise software development process revolves around the elimination of the "silos" between teams, security engineers and developers must work together to share knowledge and solve technical debts on both sides to achieve a cooperation that results in changes that will not affect the stability of the system and keep the compliances involving security.

To illustrate some of the security concerns and elaborate how they are handled within enterprise-level vendors, the author analyzed the issues presented by cloud services that were built on top of Open Source systems. According to Rao (2003), in her white paper for the SANS Institute, the usage of open source software for enterprise requirements is associated with risks because it presents "Absence of meticulous evaluation", "Spurious open source" and "Lack of sponsorship". Once the open source system is used as a foundation for the elaboration of a larger proprietary solution, it expected that the companies behind such development should mitigate the risks by setting the appropriate security policies within the elements of the infrastructure and verify the source code for any Common Vulnerabilities and Exposures (CVE), reducing, this way, the risks introduced to enterprise level applications.

The creation of a cloud-based Deployment Pipeline can be aligned with security standards, such as, the Payment Card Industry (PCI), through the introduction of measures like: strict authorization and authentication management solutions to handle multi-tenant component, firewall configuration of the elements hosted in the cloud and even encryption mechanisms for the customer data. Some Cloud Solutions also offer "Private Cloud" environments that would still present the benefits of the abstraction of infrastructure elements and allow the provisioning of the DevOps tools. The usage of fictitious data in the development pipeline is also an option for mitigate the friction of introducing any changes to the application, this is one alternative that can be leveraged but different solutions can always be implemented as knowledge sharing and experimentation for continuous improvement are some of the pillars of the DevOps paradigm, although the cultural change and collaboration between teams must be discussed to drive this initiative in an enterprise-level environment.

2.3. Continuous Integration in the Cloud

Continuous Integration or CI is the practice of having a centralized place for testing and building Software artifacts, eliminating any inconsistencies associated with builds that are produced in a developer’s machine, many PaaS solutions offer a “Push-to-Deploy” service that automatically deploys the application package to an Application Server once the code is committed to the source-code repository (such service is used in cloud platforms like Google Application Engine, IBM Bluemix and Heroku), this exercise of Continuous Integration is very aligned with the adoption of Cloud Services as it relies on an abstraction of the infrastructure associated with the source-code repository, the build machine and the Application Server to where the application is deployed, some of these solutions even offer a Development Pipeline, which not only trigger the deployment “on demand” but also allow the user to set up a scheduled build to increase the quality of the software through constant testing and verification of its use-cases.

The biggest advantages of implementing CI in the Cloud are related to the availability of the build servers and, specially, the scalability of them, Gopularam, et al. (2012, p.4) illustrate an interesting experiment in their article where they demonstrate how to run a high amount of test cases in a shorter time-frame by introducing more Virtual Machines to share the workload of the Selenium tests, they state the following: “The time taken gradually decreases with addition of a greater number of testbeds. The time taken for running 324 Firefox test cases is 54 minutes with 10 testbeds and where as it is 15 minutes with 30 testbeds.”, this reveals the advantage of the combination of Cloud Computing and DevOps practices.

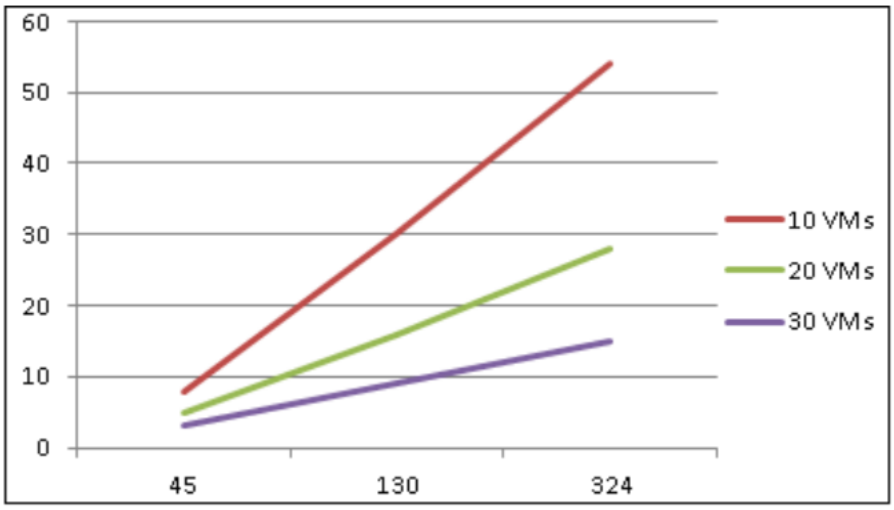


Figure 2.1. *Test cases executed with multiple VMs.*

Another concept that was absorbed during this research concerns the abstraction of the additional components required for proper Continuous Integration tests, assuming that a given functionality relies on a virtual machine, a database or a Lightweight Directory Access Protocol (LDAP) server, it is interesting to design the Cloud-Based solution so it will base the provision of such components on the Topology and Orchestration Specification for Cloud Applications (TOSCA), which basically turns them into “service topologies”. According to Johannes Wettinger, et al. (2012, p.4): “The result is a self-contained, portable, and executable service model that can be used to deploy and manage service instances in the Cloud ”, the author also checks if the cloud platforms evaluated in this research follow such concept.

With such specification, it is reinforced the abstraction of the infrastructure and focus on the interaction between each entity within the Cloud-Based Solution, always considering services as opposed to nodes. Following the same paradigm, it is possible to leverage the elasticity of the build server and seize the opportunity to “fan out” the workload of the Unit and UI testing to different “slave” machines that can build and test the desired Software.

2.4. Continuous Delivery in the Cloud

Continuous Delivery or CD can be translated as Automated Deployment that comprises all the dependencies and settings required for a given application to produce value to its end-users, it follows the paradigm of “Infrastructure as Code” so it facilitates the consistency between different environments since the application is deployed against a topology whose nodes are provisioned programmatically.

Another article from Johannes Wettinger, et al. (2013, p.1) focus on both DevOps and Cloud services, describing the motivation and elaboration of a framework to perform automated deployment of the application utilizing DevOps tools: Chef and Puppet, to install the required middlewares of the application, which implies a preference for the usage of IaaS over PaaS because they are "geared towards deploying whole application stacks in Virtual Machine images (VMs), following the IaaS model that has been dominant in the Cloud service market”, this leads to a discussion involving the tread-off of abstracting the infrastructure configuration, PaaS solutions are indicated if the application does not require a high level of customization on the Operational System (OS) or the Application Server side, IaaS allows the user to perform more low-level changes specific to the application, however, if the PaaS Solution allows the user to apply such configurations without compromising the factor of low complexity and auto-scalability, it should be the preferable option.

Other articles present interesting opinions around the Cloud Services evaluated in this research, on his paper, Tudor Alexandru (2014, p.19) talks about some peculiarities of PaaS and specifically mentions some details about Google Application Engine, unlike Daniel Cukier, he advocates the adoption of IaaS over PaaS by exemplifying that "the application code must conform to specific APIs. Google App Engine, one of the most successful products in this setting, supports only applications written in Java and Python and in the Java code, threads are not allowed", this raises some awareness involving the limitations that are introduced by the PaaS Application Servers.

The advantage of IaaS over PaaS, considering the level of flexibility, is very clear, however, it should be a requirement for enterprise level applications with specific points of complexity. In this research, IaaS is leveraged purely to enable the Continuous Integration environment, assembling Jenkins and Selenium Hub Virtual Machines, and only when this is necessary, in IBM BlueMix, per instance, Jenkins is made available through Software-As-A-Service (SaaS), so, it works for the conventional Continuous Integration scenario to assemble a Development pipeline, on top of that, BlueMix was built on top of Cloud Foundry, which is an open-source technology that works with multiple IaaS back-ends, it is very flexible and facilitates the configuration of different technology stacks (NodeJS, Java, Python, Ruby), it is also extensible and allows the user to customize the configuration of the Application Server hosted in the cloud.

2.5. Comparison of Cloud Solutions from the DevOps Perspective

The research within this specific perspective did not result in many articles, among the few articles that were gathered, the one that better connects the previously-mentioned topics (DevOps and Cloud Solutions), was written by Daniel Cukier but, as it was stated in his own conclusion, there were important topics associated with Continuous Integration and Continuous Delivery that were not addressed in his research, he presented many Cloud Services that were connected to his application along with alternatives, resulting in an interesting case study that also presents multiple candidates for a successful Cloud integration. However, while focusing on the comparison of different Cloud Solutions, Saurabh Kumar Garg, et al. (2013, p.2) state that: “it is not sufficient to just discover multiple Cloud services but it is also important to evaluate which is the most suitable Cloud service. In this context, the Cloud Service Measurement Index Consortium (CSMIC) has identified metrics that are combined in the form of the Service Measurement Index (SMI), offering comparative evaluation of Cloud services”, the article documents the creation of a framework named as Service Measurement Index for the Cloud (SMICloud), which evaluates Cloud Solutions based on metrics and criteria that focus on Service Level Agreement (SLA), the measurement index is a pragmatic approach to analyze and compare services, once it is provided as a framework, it helps the user to properly evaluate services so the right decision can be made.

The comparison of Cloud Solutions is also explored in a white paper published by Chris Haddad (2011, p.5), where he even mentions the “DevOps Tooling” as one of the criteria for the evaluation of the best vendor for Cloud Services, he elaborates some of the characteristics of DevOps tools in the following way: “PaaS offerings often support DevOps practices, which include self-service, automated provisioning, continuous integration, and continuous delivery”, these practices act like valuable criteria points to indicate which Cloud Solution can provide services that facilitates user operations.

The approach selected by Saurabh Kumar Garg, et al. is very interesting as it is based on an actual consortium that received support from CA Technologies, Accenture and universities like University of Melbourne and City University London; the Cloud Services Measurement Initiative Consortium (CSMIC) separates their evaluation points in the following topics: Accountability, Agility, Assurance, Financial, Security and Privacy, Performance and Usability. In addition to that, the criteria selected by Haddad covers the DevOps tooling and the basic Cloud characteristics, the characteristics are documented by the National Institute of Standards and Technology, also known as NIST (2012), they are: On-demand self-service, Broad network access, Resource pooling, Rapid elasticity and Measured service; this list presents a good set of capabilities for Cloud Solutions. The objective of this research is to use such evaluation frameworks as a reference and adjust its criteria to achieve a method that can verify which solution offers the tools that best align with the DevOps paradigm.

3. Methodology

The main goals of this research are the successful configuration of a Deployment Pipeline on both Cloud platforms and the analysis of the capabilities for comparison purposes. This experiment answers the following questions related to the Cloud Solutions under analysis:

- Which one contains the best set of offerings to build and test a Web Application?
- Which one provides the best set of tools to help the user to quickly deliver a change to a Production environment without affecting the quality of the application?
- Which one offers the best set of monitoring tools to support decisions involving additional changes and operations?
- Which one has the best set of components to assemble a Deployment Pipeline, including testing tools and IaaS elements, such as virtual machines?
- Which one presents a cost-effective model to provision a Deployment Pipeline and Virtual Machines?

The purpose of this research is to analyze Cloud Solutions and point out which one would provide appropriate features to achieve better integration with DevOps practices. For this analysis, two of these solutions were chosen: Google Cloud Platform, that allows you to “build and run applications on Google's infrastructure” (Google, 2014), and IBM Bluemix, whose main objective is to “simplify the delivery of an application by providing services that are ready for immediate use and hosting capabilities to enable internal scale development” (IBM, 2014).

IBM Bluemix should present an advantage over Google Cloud Platform due to its awareness of the modern needs of Software Development and its strong offerings involving DevOps services.

Through this research, a couple of evaluation frameworks and scorecards were mapped to be used as reference to the comparison between these two Cloud Solutions, the author deploys a sample application through the pipeline and verifies several points of the Software Development Life-Cycle, considering the DevOps tooling of both Google Cloud Platform and IBM Bluemix.

The results of this research should present a tangible example of how to explore the features of such PaaS solutions and point the one that stands out among the evaluated options.

The selection of these two specific solutions was based on key similarities, such as, the strong brands associated with both solutions (Google and IBM), the

the DevOps alignment presented by both companies and the focus on their Java Enterprise Edition PaaS offerings: Google Application Engine started with Python but, a year later, Google met the demand from Java developers and released a version of GAE for Java, whereas IBM Bluemix was released with a Websphere Liberty Profile option among its supported technologies.

There are other strong candidates that could have been included in this research: Microsoft Azure, Salesforce, Amazon Beanstalk, Heroku (Which was acquired by Salesforce in 2010) and Red Hat's OpenShift, all these Cloud Solutions are associated with industry leaders and were adopted by a substantial number of users, they can be found the latest Gartner's Magic Quadrant reports (Figures 3.1 and 3.2) for IaaS (Gartner, 2014) and PaaS (Cloud Computing, 2014). Some of them do not show up in both quadrants as they are focused exclusively on the PaaS Model, which is the case of the Salesforce solution. The other ones fall in a different category of Cloud Solutions that allow the users to leverage both IaaS and PaaS offerings.

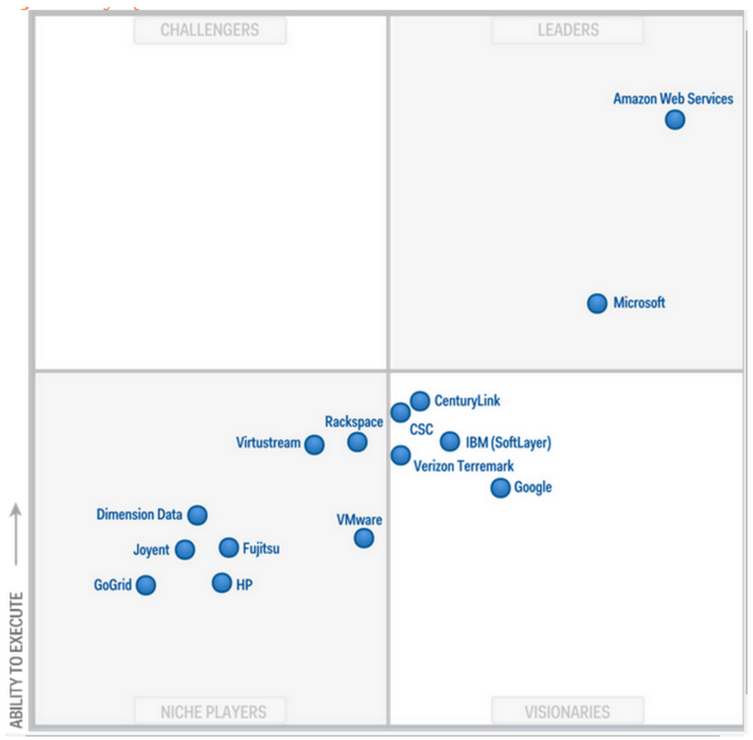


Figure 3.1. *Gartner's Magic Quadrant for IaaS 2014, (Gartner, 2014).*

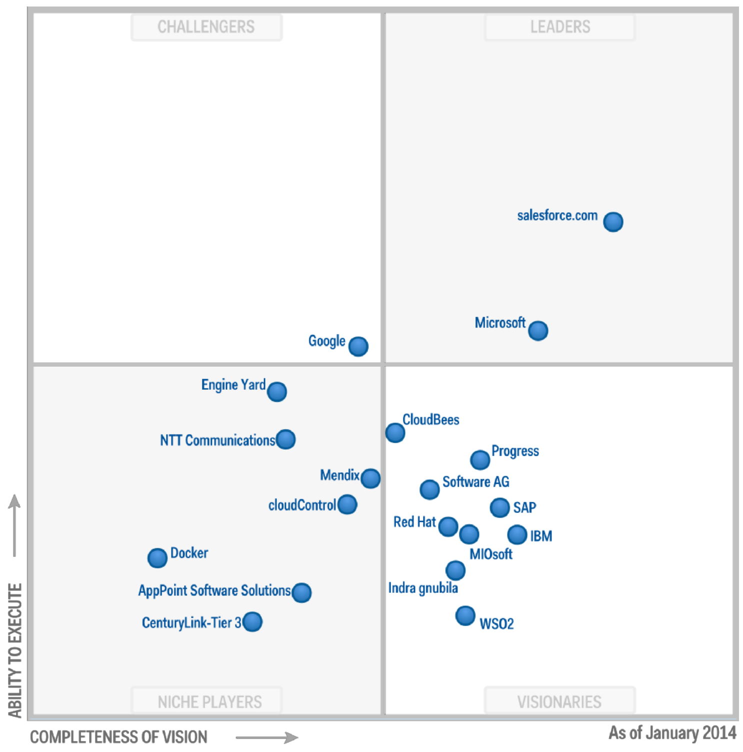


Figure 3.2. *Gartner's Magic Quadrant for PaaS 2014 (Gartner, 2014).*

Google Cloud Platform was selected due to its popularity, which is mostly concentrated on Google Application Engine. This PaaS Solution has been in the market since 2008 and it attracted users from multidimensional companies, it is among the fastest and cheapest options according to online cloud comparison articles. Peter Wayner (Computerworld, 2014) conducts a benchmarking exercise with Virtual Machines provided by three different Cloud Solutions: Amazon EC2, Google Compute Engine and Microsoft Azure, this exercise was performed with the DaCapo benchmarks, elaborated by Stephen M Blackburn, et al. (2006, p.1), which is "a set of general purpose, realistic, freely available Java applications" elaborated by the Open-Source community, in the ComputerWorld article, this benchmark approach is used to run instances of Java Virtual Machines in the cloud-provisioned machines, among the description of the results it was stated that "a Google machine had the fastest time in 13 of the 14 tests. A Windows Azure machine had the fastest time in only one of the benchmarks. Amazon was never the fastest.", another section describes the results based on the pricing criteria in the following way: "A Google machine was the cheapest option in eight of the 14 tests. A Windows Azure instance was cheapest in five tests. An Amazon machine was the cheapest in only one of the tests.", although Google Cloud Platform have not been pointed as the top-leader in Gartner's Magic Quadrant for IaaS or PaaS, such analyses reinforce how it still presents itself as a strong candidate in the Cloud Computing world.

In the DevOps perspective, Google has also adopted and established Continuous Integration and Continuous Delivery practices within the Company, their Engineering Director (Melody Meckfessel 2014) stated the following: “we do, internally, each day: 800 thousand builds, 2 Petabytes of build outputs, 100 Million Test Cases and 30 thousand Changelists; the rate of code generation, it's not about the number of changes, I'm trying to indicate how quickly the iteration cycle could be and how fast, for me, my mission is how fast I can make it for an engineer to take that idea and get it out reliably to the user community”. The same paradigm is applied to their IaaS and PaaS offerings, Google Cloud Platform offers a Release Pipeline that automates builds, test and deployment operations. They also offer a monitoring feature to accelerate the troubleshooting and the act of resuming the correct state of the application and the services supply for end-users.

Google Cloud Platform also offers Cloud services associated with Big Data, in his article, Stefano Bellasio (Cloud Academy, 2014) says that “today Google has some of the most advanced labs in the world about data mining, machine learning and, of course, Artificial Intelligence”. Based on the potential of Big Data services, Google Cloud tends to reach higher ranks in the overall evaluation of Cloud Solutions and gain a wider market adoption. The collection of large amounts of data is particularly interesting if leveraged to improve the interaction with the infrastructure by learning more about anomalies such as network outages or slowness. This can be achieved by parsing log entries, the improvement of services through knowledge sharing and monitoring is aligned with DevOps practices.

IBM Bluemix is one of the latest Cloud Solutions introduced in the market (it was released in February 2014), it was strongly advertised as “a set of DevOps services”, Jason Verge (Datacenterknowledge, 2014) presents Bluemix as a solution “based on IBM’s Open Cloud Architecture and Cloud Foundry, with SoftLayer acting as the underpinning of the platform. Bluemix provides DevOps in the cloud. After selecting a variety of open source technologies and back-ends, Bluemix provisions the entire environment in the cloud”, this association with the DevOps practices makes the Cloud Solution the perfect candidate for this research.

In Bluemix PaaS, Java applications are deployed to the Websphere Liberty Profile runtime. Liberty is a lightweight Java Application Server that is easily configured and allows Java applications to be deployed seamlessly. According to Alex Mulholland’s article (DeveloperWorks, 2014), Liberty Profile is the ideal Application Server to host Cloud applications, she mentioned that “The Liberty runtime is available through the Liberty build-pack, which can automatically bind your application to many of the Bluemix services, so they are quick and easy to use.”. This easy decoupling of components facilitates the usage of such Application Server. Bluemix is built on top of Cloud Foundry, the open-source PaaS Solution, it provides an abstraction of runtime engines and it can accommodate all technologies (Java, PHP, Python, Ruby, Node JS), it also work with custom “buildpacks” that allow the developers to package the entire application server and push it to the Cloud, increasing the flexibility of the Cloud offerings.

Microsoft’s Azure was pointed as one of the top choices in Gartner’s Magic Quadrant for both IaaS and PaaS in 2014 (as seen in Figures 3.1 and 3.2). It is a robust Cloud Solution that supports multiple technologies and even offers a Continuous Integration pipeline in its Online Visual Studio management console. According to Keith Mayer (Blogs.technet.com, 2014), Microsoft’s Senior Technical Evangelist, “Together with Microsoft Azure, Visual Studio Online promotes a consistent, cloud-enabled approach to delivering reliable enterprise solutions throughout all development and operations phases”. However, beyond its wide adoption from .NET developers and Microsoft enthusiasts, Windows Azure is still enhancing its Java support. According to one of Microsoft’s News from Central and Eastern Europe (Openness CEE, 2014): “Azul Systems, the award-winning leader in Java runtime scalability and Microsoft have partnered last year on a Windows distribution build of the community-driven open source Java™ implementation, known as OpenJDK™, for Windows Server on the Azure platform”, based on that the author decided to evaluate Google and IBM solutions only.

Amazon Web Services (including its PaaS offering, Elastic Beanstalk, also known as “EB”) is also among the leaders of IaaS Solutions. However, AWS are often associated with high investments and management overhead (decreases the abstraction in its PaaS model and requires configuration of underlying components like the Application Server and the Network). This complex management aspects are not very aligned with the DevOps paradigm, so it is not ideal for the objectives of this research.

Google Cloud Platform and IBM Bluemix present a cloud services model that leverages both PaaS and IaaS offerings: Bluemix has a special focus on its PaaS Model with a wide variety of services in its catalogue, whereas Google offers additional IaaS provisioning options, such as Google Storage. Both companies provide a solid and reliable infrastructure and, as per the practices mentioned previously, are known DevOps evangelists, such similarities reinforce the decision towards these two vendors.

In this research, the evaluation of Cloud Solutions’ DevOps Tooling is conducted through 3 stages:

1) Deployment Pipeline

In order to empirically explore the offerings of each Cloud Solution from the DevOps perspective, the author assembled a Deployment Pipeline (combining Continuous Integration and Continuous Delivery concepts) in both Google Cloud Platform and IBM Bluemix.

The application's code is subject to a Development Pipeline that performs Unit Testing and produces continual builds to increase quality assurance (Continuous Integration). The application is then deployed to its respective PaaS (Google Application Engine, also known as GAE, or IBM Bluemix's Websphere Application Server Liberty Profile) automatically. As part of the Continuous Delivery evaluation, the source-code stream and versioning were configured to properly identify the builds that, based on the automated testing against the sample application's interface, became eligible for Production deployment, this eligibility is set by Tagging builds.

Each Cloud Solution was also evaluated based on the flexibility of their deployment mechanisms, security aspects and the presence of tools dedicated to assist with Release Management and Defect Tracking.

2) Analysis of results

As discussed in the Literature Survey, there are two evaluation tools whose format is aligned with the analysis proposed in this research. The first is the framework created by Saurabh Kumar Garg, et al. (2013), based on the Cloud Service Measurement Index Consortium (CSMIC), which was named as “Service Measurement Index Cloud framework” (SMICloud). The second is the “PaaS Scorecard” created by Chris Haddad (2011). Based on the criteria presented by these tools, the author defined a set of capabilities to evaluate the components of each Cloud Solution from the DevOps perspective, this set of capabilities represent a new evaluation framework that is discussed in the next section of this chapter.

3) Reporting

The report is composed of scorecards to illustrate the gathered metrics followed by a description of results, strong facilitation points and challenges related to specific capabilities, the last section is related to the overall conclusions.

In order to measure the qualitative capabilities, the author sets a specific score for each of the DevOps capabilities on each Cloud Solution, similar to the “PaaS

Scorecard”, where Chris Haddad (2011) describe the process stating that “PaaS offerings score high (10) when they are well integrated with preferred software development life-cycle tooling across all application life-cycle phases. PaaS offerings score low (1) when they deliver a disconnected and siloed design, development, deployment, and management experience”. The author decided to use this range for the scores (1-10), each capability will be classified according to its level of compliance with the DevOps practices. Beyond the qualitative points, the elapsed time to build and deploy the app should also be analyzed, including the “Pricing” based on the projection of costs that is provided by each Cloud Solution.

The data was obtained through empirical exercises with both Cloud Solutions, the results are mostly based on a sampling approach involving a Java EE Application, a substantial set of libraries and a relational database interaction. The author performed multiple tests by delivering code changes through the pipeline to check the performance of each operation that is carried out in the Cloud environment.

3.1. The evaluation framework - DOMICloud

As mentioned previously, the approach for this research is to utilize two already-existing evaluation tools as reference to create a new one that shall focus on the DevOps components, these evaluation tools are: the “Service Measurement Index Cloud framework” (SMICloud) and the “PaaS Scorecard”.

These tools were chosen because they contain a well-organized list of Key Performance Indicators (KPIs) and were elaborated specifically for the evaluation of Cloud Solutions, this is aligned with the concepts that are explored in this research: DevOps and Cloud Solutions. By using these measurement indexes and criteria categories as a reference, the author defined a new set of criteria points based on the experiments conducted in both Cloud Platforms. It was necessary to create a new framework because these two tools do not cover the full scope of the elements being analyzed in this research, SMICloud analyzes Cloud Solutions from the services point of view, without a more specialized analysis of the DevOps aspects of the actual platform, whereas, the “PaaS Scorecard”, although it presents some focus on DevOps, it is limited to the aspects of PaaS solutions and disregards the IaaS elements, it does not focus on the evaluation of Cloud solutions from the point of view of both IaaS and PaaS, the white paper also does not present any details about how the evaluation data were obtained from the cloud platforms.

The SMICloud is composed of Accountability, Agility, Assurance, Financial, Security and Privacy, Performance and Usability; based on these points the author mapped a set of qualitative and quantitative items, this set of capabilities should be referred to, from now on, as **DevOps Measurement Index for the Cloud (DOMICloud)** framework. Each item of the DOMICloud reflects one of more items of the CSMIC, creating a layer of specialization that is more DevOps-oriented, they are 6 items: **Friendly, Simple, Automated, Operable, Fast and Cost-Effective**. Each item is listed below to illustrate how they are correlated with the CSMIC categories, the list also includes a description of the capability and the rationale behind the data gathering methods.

DOMICloud's qualitative capabilities are comprised of the following items:

- **Friendly**
  - Description: Bootstrapping**.** The ease of subscribing to the Cloud service and creating a simple application.
  - How can it be determined: The author describes his initial experience with both Cloud Solutions and analyzes the bootstrapping process based on the official documentation of each vendor.
- **Simple**
  - Multiple offerings for similar objectives with different features, low level of complexity involving the specific requirements to deploy on the PaaS.
  - How can it be determined: By checking the effort to refactor the sample application code according to the requirements of the PaaS where it will be deployed, checking if the provisioning of services does not present too many obstacles, identifying siloed components.
- **Automated**
  - The set-up process of a Deployment Pipeline, create and bind services with the application, create Virtual Machines automatically.
  - How can it be determined: By evaluating the creation of the Deployment Pipeline and its dependencies based on the official documentation and presents evidence to support which option has more out-of-the-box Self-Service mechanisms.
- **Operable**
  - Focused on Operations' tasks, options within the Administration Dashboard (Auditing and logs).
  - How can it be determined: By explores the features related to the visualization of data produced by the Deployment Pipeline and the application itself, considering scenarios of a simple release, log analysis, PaaS monitoring and patch deployment.

- **Cost-Effective**
  - Good storage and services quotas even on basic packages, better pricing.

- How can it be determined: By checking, at the end of the exercises on both Cloud Platforms, the billing details presented on their management console, analyzing and comparing the data to attest which one offers the best investment based on DevOps tools.

Whereas one quantitative capabilities is defined as:

- **Speed**
    - Description:
        - Elapsed time to perform Deployment Pipeline operations:
            - commit
            - build
            - test
            - deploy
            - GUI Testing

        - Elapsed time to:
            - Create PaaS Infrastructure for new applications
            - Provision Virtual Machine

    - How can it be determined: By triggering the deployment through the pipeline and taking note of the time spent on each transaction (commit, build, test, deploy, GUI Testing); this can be mapped in the Deployment Pipeline's logs (Maven build logs). By verifying how fast new applications and Virtual Machines are provisioned verifying, when possible, the logs of the respective command line utilities provided by each vendor.

These are the capabilities defined by the DOMICloud evaluation framework, it is a framework to guide the evaluation of the DevOps offerings and interpret results of empirical exercises with Cloud Solutions.

3.1.1. The DOMICloud Scorecard

The score utilized for the interaction with the tools of each platform, as described previously, is based on the "PaaS Scorecard", this method was elaborated to classify the operations. For the specific comparison proposed in this research, the evaluation plan is composed of the following interactions:

1. Subscribe to the service
2. Learn how to operate the management dashboard

3. Set up Command-Line Interface (CLI) utilities locally and prepare “off-line” development environment through the configuration of a J2EE Application Server and a local database instance
4. Adjust the settings of the sample web application to match the specific requirements of the PaaS J2EE container
5. Bind and configure a Cloud Database service
6. Create the Deployment Pipeline, making sure that the sample web application is working as expected
7. Create Virtual Machines to set up Graphic User Interface (GUI) automated tests and link it with the sequence of the steps within the pipeline
8. Evaluate additional features offered by each platforms, such as: Monitoring tools, logging and Bug Tracking solutions.

The author analyzed each these steps and took screenshots to gather evidence of the results of each operation, the points granted to each operation are classified from (1), for non-satisfactory criteria, to (10), for results aligned with the DOMICloud capabilities, each classification is described in the table below.

Points	User Experience	Technical Components
1-2	**Complex.** This score is associated with operations that present a high level of “friction”. They are not performed efficiently.	**Not automated and involves excessive configuration** The platform does not provide any tool to facilitate the configuration or provisioning of a given component of the application development lifecycle, the features associated with its enhancement and maintainability are also included in the classification, for example, absence of Bug Tracking system among the tools offered in the Cloud platform.
3-5	**Sensitive.** Operations that introduce some complexity, either through non-intuitive or unnecessary steps.	**It is either partially automated or does not offer orchestration with other components** The platform provides automated methods to install dependencies of a given component but not the component itself, there is still need for manual interaction to achieve the objective. Per instance: a component for GUI Tests that cannot be linked to the good builds tagging.
6-7	**Average.** No critical impact to any	**Automated but presents strictly limited configuration.** The component can be automatically

	interaction with the interface or the configuration of its services.	provisioned and posses an interface to communicate with other components, however, due to the nature of its automatic configuration, it affects the objective of the overall automation, per instance, a Jenkins VM that is automatically provisioned but only runs a set of predefined jobs.
8-10	**Ideal** Very intuitive and helps the user to achieve all objectives efficiently and effectively.	**Fully automated with ideal abstraction of configuration.** The component is activated and all the basic configuration is applied seamlessly, but the user is still allowed to customize it and leverage its features to improve the application development lifecycle. For example: a Deployment Pipeline that is created along with the application and is still open to extensibility.

Table 3.1. *DOMICloud Scores and their respective characteristics.*

In order to perform all of these operations, the author allocated two weeks to set up and verify the results on each Cloud Solution. That included assembling the Deployment Pipeline and the configuration of the sample Web Application. All steps are performed with constant consulting of the Google Cloud Platform and IBM Bluemix official documentations.

Regarding the limitations of the methodology: the research conducted here was performed with a limited financial investment, so the scalability aspects of the Pipeline could not be explored as the computation of the activities incur in high expenses.

3.2. The Deployment Pipeline

The pipeline is composed of steps that produce the application's artifacts and deliver them in a way that they can be consistently consumed by the end-users, this consistency is achieved through testing and automation, tests should flag any issues introduced by any change, stop the flow of the software delivery and notify the team responsible for such change, the automation makes the deployment less error-prone and accelerates the overall set of tasks, in real-world scenarios, multiple environments are created to verify all these steps and even perform some environment-specific operations to increase the quality of the final version that is delivered to the end-users, to name some of these environments.:

1. Development
2. Quality Assurance
3. Performance Testing
4. Staging
5. Production

The common practice is explained through the diagram below (Figure 3.3). The initial environment (Environment 1) is the main development environment, its pipeline might be executed every night, on demand or scheduled to run every N hours. Normally, this environment's pipeline is not associated with any manual task and it must redeploy very often to make sure that each change can be safely accommodated into the next part of the release, the steps are normally comprised of:

1. **Build/Test:** pulls the code from the GIT Source Control Management (SCM) system, run static analysis, unit test and produce packages that should be deployed.
2. **Deploy:** interact with the application server in order to install the application so it will be available within that given environment.
3. **Graphic User Interface (GUI) Test:** automates the verification of the application's interface, along with its functionality, it is usually performed by tools that automate operations in a browser, simulating users' actions.
4. **Tag Good Build:** Once the changes are successfully processed by the other steps of the pipeline, the practice is to either run a specific SCM command should create a logical grouping of the code files (build tagging), or just "tag" the actual artifacts produced by the build using some other tool that manages such artifact.
5. **Manual Tests:** Human-based testing is required to ensure that the builds are not being validated erroneously by some problem with the automation. This is usually performed in other environments like Quality Assurance or User Acceptance Tests. Multiple environments might exist to accommodate this objective (ENVIRONMENT 1*N).
6. **Load Tests:** To ensure that the application will also operate consistently in different performance scenarios, a set of tests introduce some work load to the application so it can be monitored and produce reports that will help the teams involved to make decisions regarding required changes, this should indicate adjustments that need to be applied to the application or even re-engineering of the architecture, there are different kinds of tests: Load tests, Stress tests and Capacity tests. This kind of task can also be performed in multiple environments, with different components and different set of test cases (ENVIRONMENT 1*Z).

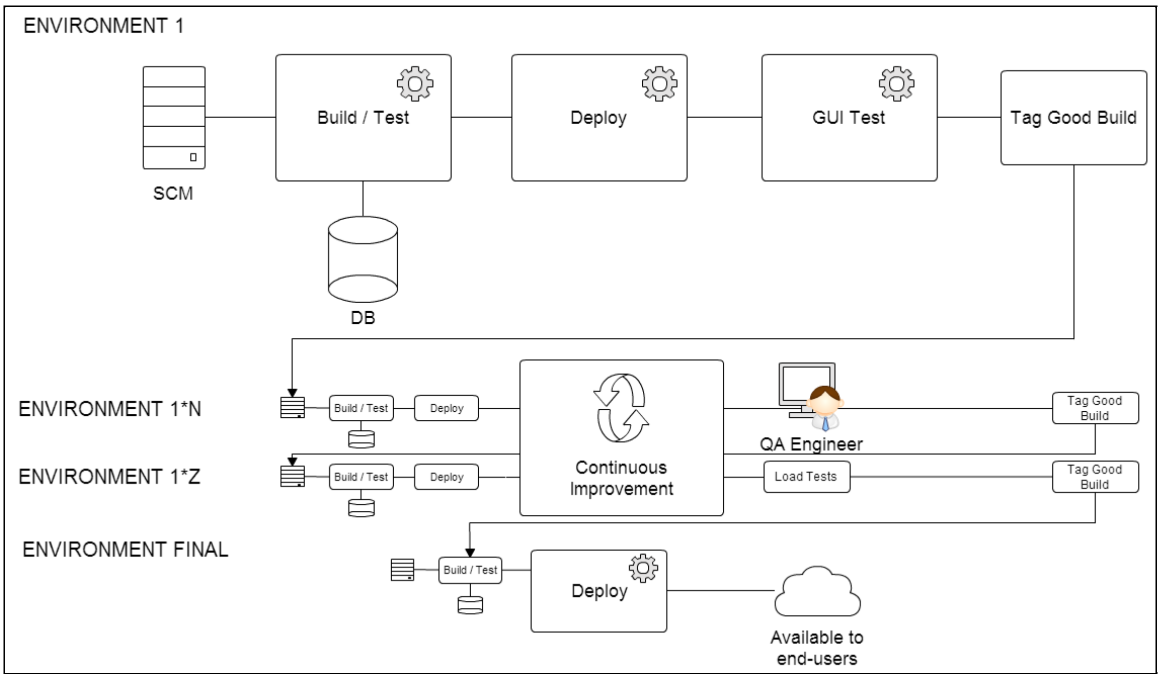


Figure 3.3. *Basic diagram illustrating pipelines in multiple environments.*

There are alternatives that involve just a subset of these steps: some development teams work with a simple pipeline that just produces the artifacts, perform a manual deployment against a Test environment and then wait until the QA team finishes their tests, verify the QA report. The application is then deployed to the Production environment by documenting steps and delegating the deployment to the "Operations" team, based on the requirements of the release, several other teams also need to follow a set of steps to assist with the deployment of the application (Network configuration, Database updates, Application Server adjustments). This practice presents itself as an inefficient approach for the following reasons:

- Presence of manual operations that can be automated (Tests, configuration, deployment)
- Silos between teams
- Lack of environments to evaluate different kinds of tests and perform adjustments

The Deployment Pipeline is in the center of the evaluation conducted in this research, the author created a pipeline in both platforms to compare and contrast the steps of the setup process and the tools associated with it considering the capabilities described in the DOMICloud framework. For the purposes of this research, the pipeline should contain the basic steps expected in a Development environment (ENVIRONMENT 1): Build, Test, Deploy, GUI Test and Build Tagging. The diagrams below illustrate the components of the pipelines created in both platforms.

The Google Cloud Platform offers a “Release Pipeline” that automatically provisions a subset of the required steps (build, test and deploy), however, due to the lack of a GUI Testing service and to a compatibility problem between the artifacts produced by the build and the Java support in the GAE (refer to Chapter 4), the author decided to leverage the IaaS provisioning features of the Google Compute Engine (GCE). To do this two Virtual Machines were created: one of them with a Jenkins instance and the other one running a Selenium Hub to facilitate the GUI Test, the tool was configured to build, test, deploy the application to GAE and trigger the GUI Test through Selenium.

Even though the Continuous Integration server could not be fully configured seamlessly through automation, the official Google Cloud documentation (2015) provides full guidance to “set up an App Engine application so that is deployed automatically whenever you push your code to your connected Git repository. It uses Jenkins to configure and manage automatic build and deployment.”, so the Google Cloud pipeline (Figure 3.4) was configured based on these steps, including the provisioning of a Google SQL database that is used by both the Unit Tests and the application itself, this configuration is further elaborated in the “Results” chapter.

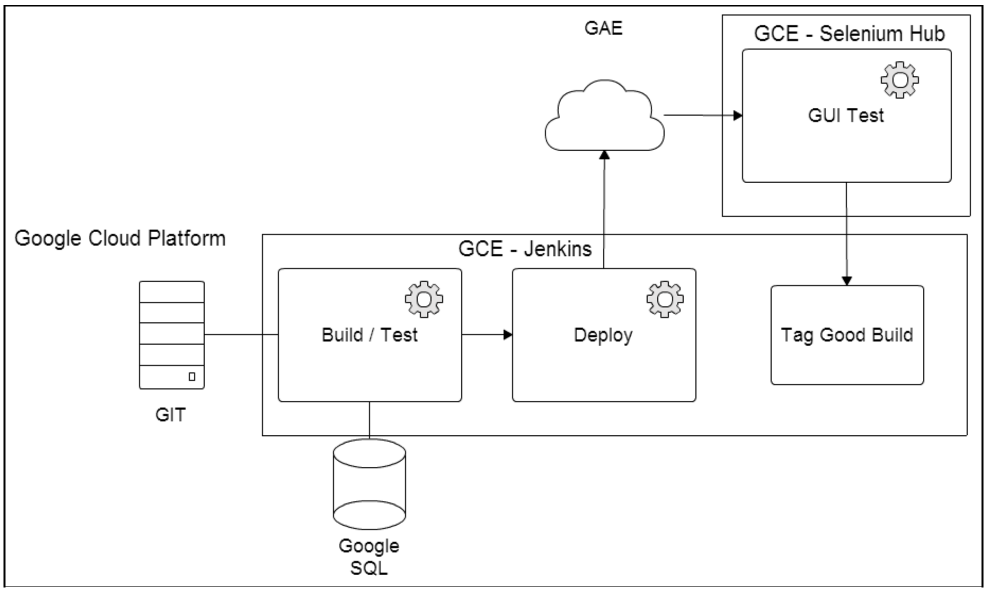


Figure 3.4. *Deployment Pipeline in Google Cloud Platform.*

In IBM Bluemix, once the user creates a GIT repository, it will automatically provision a basic Deployment Pipeline containing the build, test and deploy steps, the author identified one issue related to a specific component of the sample application: the presence of the Spring framework libraries. That was solved by configuring a custom version of the Liberty Profile application server

(Refer to Appendix D), a new step had to be added to the pipeline to perform the GUI Test, this was performed through the creation of a Virtual Machine within Bluemix's IaaS and the configuration of a Selenium Hub to allow the execution of Selenium test cases triggered by the Pipeline.

A database service was selected to support the sample application's functionalities and the Unit Tests within the pipeline. At the time of this research, ClearDB MySQL was the only valid MySQL Database option that would be similar to the Google SQL database. The Bluemix Pipeline (Figure 3.5) presents a considerable level of abstraction of the underlying technology and complexity of the infrastructure, but it still allows the user to perform some configuration on each stage.

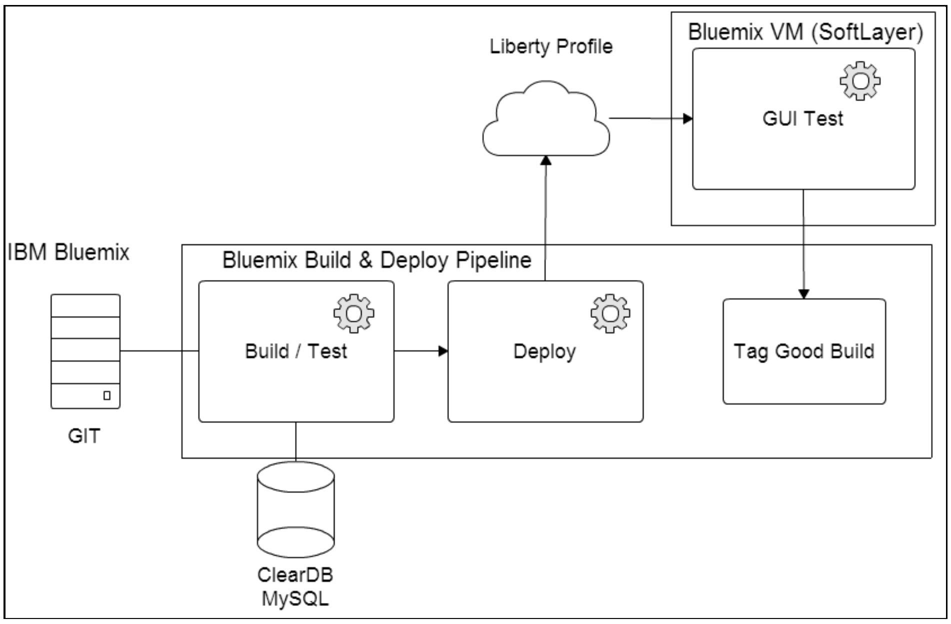


Figure 3.5. *Deployment Pipeline in IBM Bluemix.*

Once the Deployment Pipelines are fully configured, the author tested them by creating code changes to modify the interface and backend functionality of the sample application. Each step was observed and analyzed through the criteria points established by the DOMICloud evaluation framework, scoring high (10) or low (1) according to how it responded to the exercise.

4. Results

This chapter presents the results of the experiments with both Google Cloud Platform and IBM Bluemix (the steps taken to assemble the deployment pipelines are located in appendices C and D respectively), including the analysis from the point of view of the DOMICloud qualitative points and the reports followed by scorecards, as elaborated previously.

4.1. Interacting with Google Cloud Platform

The service subscription (Table 4.1) presented some friction since it presented a form containing numerous fields and based on the strict dependency associated with the credit card, the integration with a service like Google Wallet could turn that into a more seamless operation.

Interaction	**Capabilities**	**Score**	**Weighting**
Service Subscription	Friendly	7	7
	Simple	5	10
	Automated	-	0
	Operable	-	0
	Fast	6	5
	Cost-Effective	-	0
Summary Score:		5.86	

Table 4.1. *DOMICloud score for Service Subscription with Google Cloud.*

Google Cloud Platform offers a variety of services in an organized and intuitive interface, the Dashboard (Table 4.2) presents an adequate amount of configuration options and it is supported by clear instructions found on each page, it would have been ideal if the Deployment Pipeline could be monitored and configured within the same area, the DevOps Services are placed in a separate web context.

Interaction	Capabilities	Score	Weighting
Management Dashboard	Friendly	8	10
	Simple	8	8
	Automated	-	0
	Operable	5	8
	Fast	7	7
	Cost-Effective	-	0
Summary Score:		7.06	

Table 4.2. *DOMICloud score for* Management Dashboard *with Google Cloud.*

The Command Line Interface utilities (Table 4.3) are easy to use and, for some operations, they are preferred over the web interface as it facilitate operations such as the network configuration to allow communication through specific ports and the Virtual Machine provisioning, the embedded Google Application Engine instance that is integrated with the Google Cloud SDK is very useful since it allows the user to verify any compatibility issues prior to the deployment to the cloud.

Interaction	Capabilities	Score	Weighting
Client-side Utilities	Friendly	7	4
	Simple	8	6
	Automated	8	10
	Operable	9	8
	Fast	8	7
	Cost-Effective	-	0
Summary Score:		8.11	

Table 4.3. *DOMICloud score for* Client-side Utilities *with Google Cloud.*

The GAE Application presented some issues, the most critical ones were the compatibility issues with the JSPs compiled with the JDK version 8 and the limitations associated with the Java libraries that could not be deployed to the

Application Server container (See Appendix C), however, the embedded GAE instance that is available in Google SDK facilitates the testing on the local development environment, also the deployment to the cloud GAE has proven itself to be very fast, once the first deployment is performed, the subsequent changes are quickly integrated into the application.

Interaction	**Capabilities**	**Score**	**Weighting**
PaaS J2EE Container	Friendly	5	4
	Simple	8	6
	Automated	6	10
	Operable	5	9
	Fast	9	7
	Cost-Effective	8	5
Summary Score:		6.73	

Table 4.4. *DOMICloud score for* PaaS J2EE Container *with Google Cloud.*

The services binding was verified, mainly, by provisioning a MySQL Database service in the Google Cloud platform and linking it with the GAE application, the initial configuration of the Google SQL instance involved the creation of the database, the user management operations and the network access control adjustments, however, this special Google SQL database acts as a custom layer on top a regular MySQL 5.5. Database, this requires a special JDBC driver that is only created once the application is deployed to the cloud, this special requirement demands database connection properties changes that are particular to Google SQL, resulting this way in a “vendor lock-in” scenario.

Interaction	Capabilities	Score	Weighting
Services configuration and binding	Friendly	7	5
	Simple	4	6
	Automated	5	9
	Operable	7	9
	Fast	8	7
	Cost-Effective	8	8
Summary Score:		6.52	

Table 4.5. *DOMICloud score for* Services configuration and binding *with Google Cloud.*

The creation of the Deployment Pipeline demanded numerous manual steps and the "Release Pipeline" offering presented problematic results due to the lack of support on GAE for the JDK version 8 (refer to Appendix C). The Jenkins server was assembled easily by leveraging the existing Bitnami image but all the remaining steps and plugins had to be configured manually, this lack of comprehensive automation impacted on the final score.

Interaction	Capabilities	Score	Weighting
Deployment Pipeline	Friendly	7	7
	Simple	5	9
	Automated	5	10
	Operable	7	8
	Fast	7	7
	Cost-Effective	6	8
Summary Score:		6.06	

Table 4.6. *DOMICloud score for* Deployment Pipeline *with Google Cloud.*

The author could not identify any Bitnami image available with a pre-assembled Selenium Hub, although there is a web page that collects feedback from users regarding the plan to release such image (Bitnami, 2015). It is not available as yet so a Virtual Machine had to be configured from its initial state. The provisioning of VMs is facilitated by the Command Line Interface tool (gcloud), however, the configuration of the service itself took numerous manual steps. Further, the Windows Server environment provided by the VM had various security restrictions. It was necessary to add several domains to the list of trust sites in Internet Explorer in order to be able to download the JDK 7 and Mozilla Firefox.

Interaction	**Capabilities**	**Score**	**Weighting**
Virtual Machines	Friendly	7	7
	Simple	5	8
	Automated	5	9
	Operable	7	8
	Fast	9	7
	Cost-Effective	8	7
Summary Score:		6.71	

Table 4.7. *DOMICloud score for* Virtual Machines *with Google Cloud.*

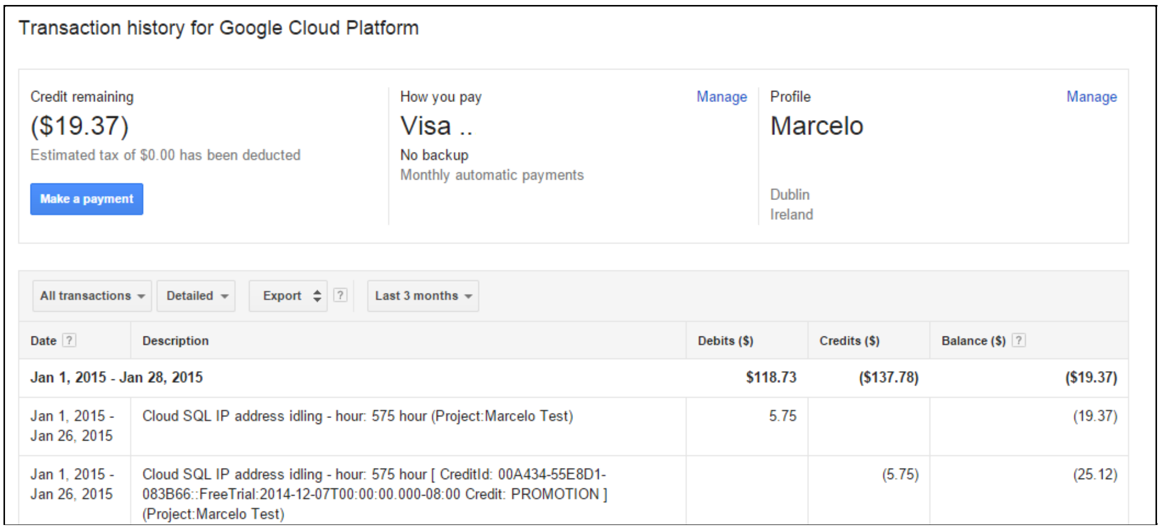
The monitoring for Google Compute Engine VMs and the Google Application Engine instances is particularly interesting as, although it presents a simple interface, it can be changed to present different types of data, such as CPU, Memory and Network traffic. The “Overview” page presents a good summary of all the activity in both GAE and GCE. Beyond that, the “Logs” page presents a comprehensive log aggregation solution where the user can filter the log entries and search for any word, this is really helpful for troubleshooting.

Google Cloud Platform does not offer any bug tracking system for the user. One could be configured within a new GCE Virtual Machine but would also involve a number of manual steps, the author did not identify any similar feature among the offerings of Google Cloud Platform.

Interaction	**Capabilities**	**Score**	**Weighting**
Additional features (Monitoring, Logging and Bug Tracking)	Friendly	6	7
	Simple	6	9
	Automated	7	8
	Operable	7	8
	Fast	7	5
	Cost-Effective	7	6
Summary Score:		6.62	

Table 4.8. *DOMICloud score for* Additional Features *with Google Cloud.*

To support the scoring for the "Cost-Effective" criteria in all the interactions analyzed previously, the billing report below was verified while approaching the end of the trial period (Figure 4.1). The cost of running such experiments on Google Cloud Platform for thirty days consumed approximately one third of the value that was granted to the user within the trial period of sixty days ($300). The services were not used constantly throughout this period and the initial tests demanded numerous redeployments and adjustments to the application's code, with the proper knowledge of the overall configuration the value of this investment could be substantially decreased.


Transaction history for Google Cloud Platform

Credit remaining
($19.37)
Estimated tax of $0.00 has been deducted

How you pay
Visa ..
No backup
Monthly automatic payments

Profile
Marcelo
Dublin
Ireland

Date	Description	Debits ($)	Credits ($)	Balance ($)
Jan 1, 2015 - Jan 28, 2015		$118.73	($137.78)	($19.37)
Jan 1, 2015 - Jan 26, 2015	Cloud SQL IP address idling - hour: 575 hour (Project:Marcelo Test)	5.75		(19.37)
Jan 1, 2015 - Jan 26, 2015	Cloud SQL IP address idling - hour: 575 hour [CreditId: 00A434-55E8D1-083B66::FreeTrial:2014-12-07T00:00:00.000-08:00 Credit: PROMOTION] (Project:Marcelo Test)		(5.75)	(25.12)

Figure 4.1. Billing report from *Google Cloud Platform.*

4.2. Google Cloud as a DevOps services provider

The following table presents the summary DOMICloud Scorecard that was assigned to the key interactions with the Google Cloud Platform solution. The justification of each score is based on the analysis presented in the previous sections.

Interaction	Score
Service subscription	5.86
Management Dashboard	7.06
Client-side Utilities (Application server, Command-Line Interface tool and Plugins)	8.11
PaaS J2EE Container	6.73
Services configuration and binding	6.52
Deployment Pipeline (Build, Tests and Deployment)	6.06
Virtual Machines	6.71
Additional features (Monitoring, Logging and Bug Tracking)	6.62
TOTAL	53.67

Table 4.9. *DOMICloud score for Google Cloud Platform.*

The research work conducted with Google Cloud Platform indicates that the platform has reliable offerings. There was no disruption of services during any of the exercises, reinforcing once more that the solution has been available in the market long enough to have reached a good level of maturity. It presents friendly interfaces (both Web-based and CLI-based). It does not exhibit a critical level of complexity for the configuration of its services. Although it could improve its offerings regarding the automation of a Deployment Pipeline. It offers good monitoring and logging capabilities but it does facilitate any cloud-based bug tracking system. The speed is very satisfactory in every aspect: Application server, VM provisioning, build, tests and deployment. Regarding the cost, the investment involved in the experiments is certainly reasonable considering the infrastructure and computing power allocated for all the tasks.

4.3. Interacting with IBM Bluemix

To sign up for the trial period of thirty days in Bluemix, a credit card is not required. With the IBM ID in place the free trial limits the applicant to the "free tier" options available for the services in the catalogue.

Interaction	**Capabilities**	**Score**	**Weighting**
Service subscription	Friendly	8	7
	Simple	9	10
	Automated	-	0
	Operable	-	0
	Fast	7	5
	Cost-Effective	-	0
Summary Score:		8.22	

Table 4.10. *DOMICloud score for* Service subscription *with IBM Bluemix.*

The dashboard presented a good separation of Applications, Services, Containers and Virtual Machines, it also helps the user understand what is the overall state of all the applications through the "App Health" monitor and keep track of the number of services allocated to the account, one negative point is that the "Horizon Dashboard" is part of a completely different console and the configuration of the IaaS components is not very intuitive, although it did not present any critical impact in the configuration of the virtual machine.

Interaction	Capabilities	Score	Weighting
Management Dashboard	Friendly	9	10
	Simple	8	8
	Automated	-	0
	Operable	5	8
	Fast	7	7
	Cost-Effective	-	0
Summary Score:		7.36	

Table 4.11. *DOMICloud score for* Management Dashboard *with IBM Bluemix.*

The Cloud Foundry Command Line Interface tool allows the user to perform management operations involving Bluemix applications, such as, create applications, create and bind services, and check logs. During the research, the tool was used in one specific operation, to "push" the application's deployment artifact to the cloud and install the custom instance of the Liberty Profile Application Server through the community buildpack, however, while trying to interact with the IaaS components, the author could not find any CLI tool to manage virtual machines and its network access, the installation of the local instance of the Application Server was not facilitated through the CLI utility either.

Interaction	Capabilities	Score	Weighting
Client-side Utilities	Friendly	5	4
	Simple	6	6
	Automated	5	10
	Operable	4	8
	Fast	7	7
	Cost-Effective	-	0
Summary Score:		5.34	

Table 4.12. *DOMICloud score for* Management Dashboard *with IBM Bluemix.*

Applications that are deployed to Websphere Liberty Profile do not require any special configuration (no “vendor lock-in”), the only change that had to be performed was related to a rendering issue with the SiteMesh framework (which was replaced later by Apache Tiles), however, there are no limitations related to specific application resources, all the Java classes within the sample application were deployed to the cloud. The ability to use Community Buildpacks is an interesting feature that allows a high level of customization and extensibility, however, the fact that the default instance is consistently failing based on the presence of Spring JAR packages is a negative point.

Interaction	**Capabilities**	**Score**	**Weighting**
PaaS J2EE Container	Friendly	7	4
	Simple	8	6
	Automated	9	10
	Operable	9	9
	Fast	7	7
	Cost-Effective	8	5
Summary Score:		8.19	

Table 4.13. *DOMICloud score for* PaaS J2EE Container *with IBM Bluemix.*

The process to bind services to the application is very straight-forward, although it requires a restart of the Liberty Profile instance for the re-staging operation. The specific database service that was targeted for this research was in Experimental Mode so the ClearDB MySQL database (hosted outside Bluemix’s infrastructure) was used as an alternative, the author could not identify an online interface to perform any configuration against the database instance. The limited number of connections (4 active connections) in the “free tier” model caused an impact during the Unit Tests so some of the tests had to be disabled. Bluemix offers a Cloud Foundry mechanism (CloudFoundry Documentation, 2015) to facilitate the connection with the database and other services. Once the services are bound to the application, a JSON object is appended to the value of a variable called “VCAP_SERVICES”. It can be retrieved and parsed by the application code so it can interact with the service. Although it is an interesting feature to leverage in runtime, it impacts the configuration phase due to the lack of a friendly interface.

Interaction	Capabilities	Score	Weighting
Services configuration and binding	Friendly	7	5
	Simple	8	6
	Automated	8	9
	Operable	6	9
	Fast	8	7
	Cost-Effective	8	8
Summary Score:		7.47	

Table 4.14. *DOMICloud score for* Services configuration and binding *with IBM Bluemix.*

The creation of the Deployment Pipeline presented itself as a smooth process, the basic stages (Build and Deploy) were already pre-assembled and ready to use with Ant, they just had to be adjusted and connected to additional stages to achieve the Deployment Pipeline that was idealized. There were connectivity issues that were preventing any communication with the database server and the Selenium Hub, this impacted the configuration and tests that were being processed in the pipeline, the issue was corrected in the latest stages of the research and that allowed further testing within the pipeline. One issue that remained unresolved is the conditional trigger based on the status of the GUI Test, the author could not identify any mechanism that could invoke different stages based on the result of the tests, however, the overall configuration of the pipeline was performed very quickly and intuitively, the interface is also a good asset to visualize all the stages in action and check any problems related to the release.

Interaction	**Capabilities**	**Score**	**Weighting**
Deployment Pipeline	Friendly	9	7
	Simple	9	9
	Automated	9	10
	Operable	7	8
	Fast	7	7
	Cost-Effective	8	8
Summary Score:		8.22	

Table 4.15. *DOMICloud score for* Deployment Pipeline *with IBM Bluemix.*

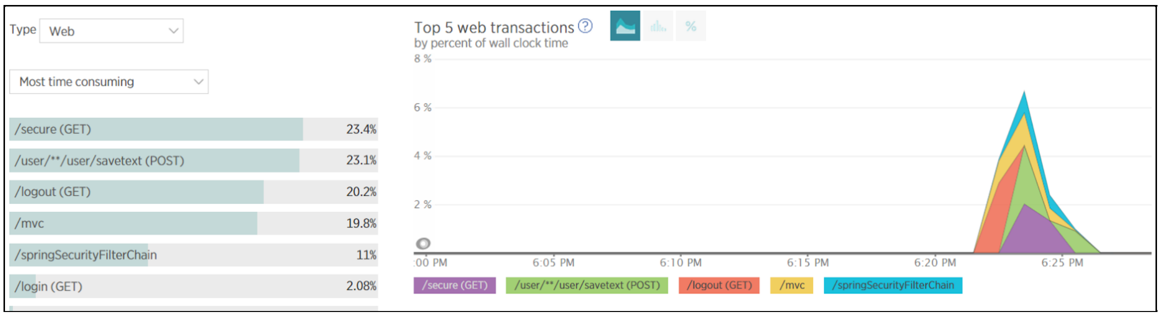
Both containers and Virtual Machines are still in BETA mode, so the utilities to interact with them are limited at the moment. The creation of SSH keys could have been done seamlessly via command line utilities. Some of the operations available in the Horizon Dashboard could be performed on the actual “cover page” of the VM itself. The “free tier” model, once more, caused a negative impact since it does not offer VMs with a desktop to facilitate the configuration of the Selenium Hub. Several steps had to be performed on the Linux VM to configure the GUI Test service. The offering needs improvement to relieve the friction for the users, however, the potential of such technologies is relevant for the adoption of the platform. The provisioning of cloud nodes in SoftLayer and the components can be easily achieved through solutions like Docker, which leverages Linux Containers (LXC) to deploy software more efficiently against specific environments. Regarding pricing: Virtual Machines are still in Beta mode so they can be used free of charge until they move to the General Availability (GA) stage.

Interaction	**Capabilities**	**Score**	**Weighting**
Virtual Machines	Friendly	6	7
	Simple	5	8
	Automated	7	9
	Operable	9	8
	Fast	8	7
	Cost-Effective	8	7
Summary Score:		7.15	

Table 4.16. *DOMICloud score for* Virtual Machines *with IBM Bluemix.*

The application monitoring that is presented in Bluemix's dashboard is limited to a summary of the memory usage and the overall application health. To gather more detailed information about the application activities, the "NewRelic" service (Figure 4.2) can be bound to it, the "free tier" model of this service offers some monitoring options, such as:

- JVM Performance analyzer
- Web transactions
- Database call response time and throughput
- Errors



/secure (GET)	23.4%
/user/**/user/savetext (POST)	23.1%
/logout (GET)	20.2%
/mvc	19.8%
/springSecurityFilterChain	11%
/login (GET)	2.08%

Figure 4.2. Monitoring of the sample application in NewRelic.

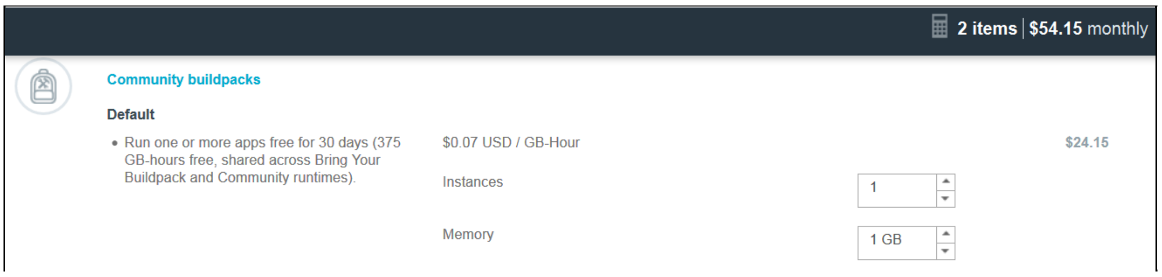
The Liberty Profile logs are available in the "Files and Logs" section of the application overview page. It presents an extensive list of folders and files and, based on the investigation of the issues faced during the experiments conducted in this research, the majority of them are not relevant for a more directed troubleshooting. The Application Server's log is under "logs/messages.log" so perhaps the interface could give more emphasis on this one and move the other files to another section of the interface, in case some other information is necessary to investigate different issues.

IBM Bluemix offers a “Plan and Track” solution as part of its DevOps Services. This allows the creation of “tickets” that can be distributed to a number of developers and organize the tasks and defects for them. It associates these items with “story” work items. As part of the Agile methodology, it would be interesting to automatically create defects for any build or GUI test failures that occur in the Deployment Pipeline but the author could not find any configuration to link these features. It is still a valuable feature as it concentrates more aspects of the development lifecycle within the same platform.

Interaction	**Capabilities**	**Score**	**Weighting**
Additional features (Monitoring, Logging and Bug Tracking)	Friendly	8	7
	Simple	7	9
	Automated	9	8
	Operable	8	8
	Fast	8	5
	Cost-Effective	8	6
Summary Score:		7.97	

Table 4.17. *DOMICloud score for* Additional Features*.*

IBM Bluemix has a wide range of options in its services catalogue, but for the objective of this research, only 3 services were utilized: Community buildpacks, ClearDB MySQL and NewRelic monitoring, the last two are free so the focus of the billing is on the Liberty Profile instance that is running on the cloud. In IBM Bluemix, the user is charged based on “GB-Hour” (Bluemix, 2015), which means, the cost of the services is based on how many gigabytes of RAM are allocated to the application, the number of application instances and hours of activity (Total GB/App x Number of App Instances x Total Hours running). The estimate provided by Bluemix states that, in a month, the cost should be $ 54.15 (Figure 4.3), however, according to the website, the first 375 GB-hours are free every month, based on that, it is considered a reasonable investment.


2 items | $54.15 monthly

Community buildpacks

Default

- Run one or more apps free for 30 days (375 GB-hours free, shared across Bring Your Buildpack and Community runtimes).

$0.07 USD / GB-Hour

Instances: 1

Memory: 1 GB

$24.15

Figure 4.3. Billing report from *IBM Bluemix.*

4.4. IBM Bluemix as a DevOps services provider

The following table presents the DOMICloud Scorecard that was assigned to the key interactions with IBM Bluemix:

Interaction	**Score**
Service subscription	8.22
Management Dashboard	7.36
Client-side Utilities (Application server, Command-Line Interface tool and Plugins)	5.34
PaaS J2EE Container	8.19
Services configuration and binding	7.47
Deployment Pipeline (Build, Tests and Deployment)	8.22
Virtual Machines	7.15
Additional features (Monitoring, Logging and Bug Tracking)	7.97
TOTAL	59.92

Table 4.2. *DOMICloud score for IBM Bluemix.*

The experiments performed with IBM Bluemix presented positive results even though it is one of the latest Cloud Solutions (with both PaaS and IaaS offerings) to be released into the market. It is a competitive option and it leverages third-party technologies, connecting them through its own service catalogue and making the DevOps Services features available to the users. It is a friendly solution that facilitates the creation of applications and binding of services. It does not present considerable friction to provision and configure most of its services and components (Simple), it facilitates the integration of services that improve the monitoring and logging considerably. The community buildpack deployment introduces some delay to the actual deployment operation. Despite this the Liberty Profile J2EE container is very fast and its local instance exposes all the configuration for any adjustments that might be necessary. The billing model is fair and the initial quota that is offered is helpful to assist new users not to allocate too many resources while they are still adapting.

5. Conclusions

5.1. Context

The IT Industry presents a growing adoption of DevOps practices, however, the initiatives normally target on-premises infrastructures. There is an opportunity to explore how the same practices can be implemented by leveraging Cloud Solutions. Due to the popularity of Cloud Services and the increasing struggle to improve the software development process, it was identified a need to map which Cloud offerings in the market could mitigate this problem by, not only allowing the user to abstract the non-functional requirements like scalability and availability, but to improve the actual software delivery process by quickly provisioning a development environment with cloud-based DevOps tools.

In this research, two Cloud Solutions were analyzed from the DevOps perspective, an evaluation framework was elaborated to guide the comparison and evidence which one presents the most ideal set of non-siloed tools and services for Continuous Integration (CI) and Continuous Delivery (CD). The author verified the interfaces provided to interact with the solutions' components, ran an experiment involving the creation of a Deployment Pipeline by leveraging both PaaS and IaaS elements of each Cloud solution, deployed a sample application and changed its code to simulate a Development environment. Additional features like monitoring and bug tracking were also verified to check if the solutions facilitate tasks that fall in the scope of an Operations team.

The research was conducted by subscribing to both services and exploring the features to assemble a Deployment Pipeline: the author installed and configured several tools, adjusted the code of the sample application, provisioned databases and virtual machines and verified the flow of the pipeline to deliver the application to the PaaS J2EE Containers. For all the operations involved, the evaluation was guided by a framework, the DevOps Measurement Index for the Cloud (DOMICloud). It defines capabilities for each cloud solution: checking if a given interface is user-friendly, does not present complicated configuration steps and offers automatic provisioning. It can be monitored and it does not require a high investment. The author also documented two DOMICloud scorecards to illustrate the performance of each cloud solution by mapping specific operations, such as, creating a Deployment Pipeline or provisioning a virtual machine, points were given according to the compliance with the capabilities of the DOMICloud framework.

5.2. The Result

At the end of this research, it was discovered that IBM Bluemix was the choice that presented the best alignment with the DevOps paradigm, proper combination of PaaS and IaaS elements, strategic partnership with third-party services to improve the interaction with the application and the creation of the Build and Deploy Pipeline. Bluemix presented itself as a competitive option. Both vendors (Google and IBM) offer valuable services but Bluemix introduces a more adequate level of extensibility and customization. Refer to Appendix E for a combined scorecard.

An enterprise-level application can benefit from such combination of tools as it will substantially mitigate the friction of assembling a Continuous Integration and Continuous Delivery environment even if it is leveraged to support only a portion of its components. By reducing the management overhead involved in build and test servers, including the maintenance of CI systems or GUI Test solutions the development team can rely on the “Service” model to abstract the underlying maintenance around these elements. The model also enables the automation of the provisioning of a Deployment Pipeline, allowing the verification of separate components of the enterprise application through the pipeline before they are all connected and ready to be promoted to the next step of Quality Control.

Bluemix was pointed out as a more recommended option for the enterprise due to the following points:

- No vendor lock-in
- The presence of DevOps Services out-of-the-box (Git, Build, Deploy Pipeline). Including an adequate level of IaaS and PaaS elements to achieve a custom Deployment Pipeline
- Built on top of Cloud Foundry (Open source cloud computing PaaS)
- Offers Agile Planning and Bug Tracking mechanisms

As enterprise-level applications normally present multiple components and a certain level of complexity, Bluemix stands out as it has the flexibility offered by Cloud Foundry’s management options and the capability to bind multiple third-party services to its applications, this facilitates the orchestration with other tools that can even be introduced to its service catalogue in the future.

The Deployment Pipeline from Google Cloud Platform was created successfully but it was not possible to leverage all the automatic features introduced by the platform. The “Release Pipeline” feature can still be used to assemble a simplified version of a proper Deployment Pipeline comprised of the build, test and deploy steps. Another option would be to have a hybrid Deployment Pipeline that would execute part of the steps locally and some other steps on the cloud. It could trigger the deployment to the cloud through a GIT commit operation and delegate the basic steps (Build, test and deploy), then it could rely on some Google SDK CLI command to retrieve GAE logs, confirm the version was deployed and trigger the “GUI Test” and “Tag Build” operations.

5.3. Challenging old paradigms

It was expected that, Google, as a more mature platform, would have more DevOps focused offerings in its catalogue, some technologies provided are aligned with the concepts of DevOps, such as: the automatic provisioning of Application Servers, Virtual Machines and Databases, however, the interaction between these tools, from the point of view of a Deployment Pipeline, still needs to be improved. Cloud Solutions can leverage more opportunities to facilitate the development process by helping the user to connect tests, deployment and code stream management, which involves the separation of source-code that resides in the Development workflow from code that is promoted to a QA or Stabilization environment, including Production.

This research indicates the need for more sophisticated cloud offerings involving CI and CD features that can improve the quality of the software that is being developed and delivered to the PaaS servers, by improving their service catalogue and increasing the consistency of the deliverables of a project through the DevOps tooling. The abstraction of the infrastructure and the automated provision of tools for testing and management of software artifacts, including monitoring components, are valuable features that allow a software development team to focus on delivering more value to their customers instead of allocating efforts to manage and connect components to achieve a proper orchestration of DevOps tools, the study of such features is what makes this research significant. The elaboration of DOMICloud is a starting point to analyze, measure and correlate DevOps features offered by Cloud Solutions and assist users to make decisions regarding which service should be hired.

5.4. Limitations

This research was affected by some limitations involving financial investment and time. One exercise that was not conducted due to financial limitations refers to an experiment involving multiple Development Pipelines, similar to the test conducted by Bhanu Prakash, et al. (2012) with multiple Virtual Machines running Selenium test cases. They showed that by increasing the number of Virtual Machines, a higher number of test cases could be performed in a shorter time-frame. A similar test could be done with multiple Development Pipelines reproducing a scenario where a given change could block the pipeline due to an error that would be deliberately committed into the source-code repository and other Development Pipelines would work in parallel without the offending code-change to validate other commits that are delivered through a simulation by different developers. That would present an interesting vision of how multiple Development Pipelines could allow developers to continue leveraging an instance of the Development environment while another developer that delivered some code that “broke” the environment would allocate some time to work on the issue and fix the bug. The author also did not explore the elasticity of the components in the Deployment Pipeline to present metrics regarding availability and scalability, such test with a high workload incurs in more expenses and it is not properly conducted with a trial package with basic quota services.

One other exercise that could not be conducted is related to a simulation of the development cycle of a complex application that integrates multiple components. The main application would be comprised of different web contexts, a separate API that would be included in a JAR package (perhaps even some orchestration with a different runtime languages like Python or Ruby), the unit tests could involve more dependencies: connecting to a Messaging Queue (MQ) solution and a Lightweight Directory Access Protocol (LDAP) server. This exercise would improve the evaluation of the DevOps tooling by experimenting with an application that would simulate more enterprise-level components and, it could also point out more compatibility issues, any impacts to the configuration of the Deployment Pipeline could be a negative point for the Cloud Solution under evaluation, that would confirm which Cloud Solution provides better compatibility between its services.

One last material that the author could not integrate to the work performed in this analysis was a qualitative research to gather the impressions towards the DevOps tooling of both platforms, due to time constraints, it was not possible to organize a research population and elaborate a questionnaire. That would reinforce the results of the comparison, specially for the DOMICloud capabilities “Friendly” and “Simple”.

5.5. Final considerations

This research is a starting point that should foment the study of different aspects of Cloud Computing, while PaaS and IaaS are commonly leveraged for non-functional requirements that support the state of the already-assembled application, there are elements of the development process itself that also require their own infrastructure and collaboration with other tools and services. This introduces an opportunity to the vendors to offer new services, allowing users to abstract the non-functional requirements of the assets involved in their Development process, improving the quality of the functional requirements implemented in the application, an example of that would be the elaboration of a Deployment-Pipeline-as-a-Service (DPaaS) solution.

6. Bibliography

1. Mark Cade and Humphrey Sheil, 2010. Sun Certified Enterprise Architect for Java™ EE Study Guide, Second Edition. Prentice Hall.
2. Paul M. Duvall, with Steve Matyas and Andrew Glover, 2007. Continuous integration: improving software quality and reducing risk. Addison-Wesley Professional.
3. Jez Humble, David Farley, 2011. Continuous delivery: reliable software releases through build, test, and deployment automation. Addison-Wesley Professional.
4. Kent Beck, 2003. Test-driven Development: By Example. Addison-Wesley Professional. Preface.
5. Google, 2015. What Is Google App Engine? [online] Available at: <https://cloud.google.com/appengine/docs/whatisgoogleappengine> [Accessed 19 November 2014].
6. IBM, 2015. Bluemix Overview. [online] Available at: <https://www.ng.bluemix.net/docs/#overview/overview.html#overview> [Accessed 19 November 2014].
7. Limoncelli, T., Chalup S. and Hogan C., 2014) The Practice of Cloud System Administration. United States: Addison-Wesley Professional.
8. Soon K. Bang, Sam Chung, Young Choh, Marc Dupuis, 2013. A Grounded Theory Analysis of Modern Web Applications - Knowledge, Skills, and Abilities for DevOps. RIIT '13 Proceedings of the 2nd annual conference on Research in information technology.
9. JIA Xiaojing. (2010). Google Cloud Computing Platform Technology Architecture and the Impact of Its Cost. 2010 Second WRI World Congress on Software Engineering.
10. Daniel Cukier. (2013). DevOps Patterns to Scale Web Applications using Cloud Services. Proceedings of the 2013 companion publication for conference on Systems, programming and applications: software for humanity.
11. Johannes Wettinger, Vasilios Andrikopoulos, Steve Strauch, Frank Leymann. (2013). Enabling Dynamic Deployment of Cloud Applications Using a Modular and Extensible PaaS Environment. Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on.
12. Saurabh Kumar Garg, Steve Versteeg and Rajkumar Buyya. (2013). A framework for ranking of cloud computing services. Future Generation Computer Systems 29 (2013) 1012–1023.
13. Chris Haddad, 2011. Selecting a Cloud Platform : A Platform as a Service Scorecard. [online] Available at:

<http://wso2.com/whitepapers/selecting-a-cloud-platform> [Accessed 22 March 2015].

14. Gianluigi Zavattaro, 2010. Automatic Deployment of Applications in the Cloud. Alma Mater Studiorum - Universita di Bologna. 10.
15. NIST, 2012. . [pdf] National Institute of Standards and Technology. Available at: <http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf> [Accessed 27 April 2015].
16. Google, 2012. Jenkins, meet Google App Engine. [online] Available at: <http://googlecloudplatform.blogspot.ie/2012/10/jenkins-meet-google-app-engine.html> [Accessed 6 December 2014].
17. Computerworld, 2014. Amazon vs. Google vs. Windows Azure: Cloud computing speed showdown . [online] Available at: <http://www.computerworld.com.au/article/539633/amazon_vs_google_vs_windows_azure_cloud_computing_speed_showdown/> [Accessed 4 April 2015].
18. Blackburn, S. M., Garner, R., Hoffman, C., Khan, A. M., McKinley, K. S., Bentzur, R., Diwan, A., Feinberg, D., Frampton, D., Guyer, S. Z., Hirzel, M., Hosking, A., Jump, M., Lee, H., Moss, J. E. B., Phansalkar, A., Stefanovic, D., VanDrunen, T., von Dincklage, D., and Wiedermann, B., 2006. The DaCapo Benchmarks: Java Benchmarking Development and Analysis. OOPSLA '06: Proceedings of the 21st annual ACM SIGPLAN conference on Object-Oriented Programing, Systems, Languages, and Applications.
19. Cloud Academy, 2014. 5 reasons why Google Compute Engine will be the next cloud choice. [online] Available at: <http://cloudacademy.com/blog/5-reasons-why-google-compute-engine-will-be-the-next-cloud-choice/> [Accessed 4 April 2015].
20. Gartner, 2014. Magic Quadrant for Cloud Infrastructure as a Service. [online] Available at: <http://www.gartner.com/technology/reprints.do?id=1-1UM9419&ct=140529&st=sb> [Accessed 4 April 2015].
21. Melody Meckfessel, 2014. How DevOps and the Cloud Changed Google Engineering. [video online] Available at: <http://www.infoq.com/presentations/google-devops-cloud> [Accessed 4 April 2015].
22. Cloudcomputing.info, 2014. Windows Azure now Leader in Gartner Magic Quadrant for Enterprise Application Platform as a Service. [online] Available at: <http://cloudcomputing.info/en/news/2014/01/windows-azure-now-leader-in-gartner-magic-quadrant-for-enterprise-application-platform-as-a-service.html> [Accessed 4 April 2015].

23. Blogs.technet.com, 2014. Accelerating DevOps with the Cloud using Microsoft Azure and Friends: The Series. [online] Available at: <http://blogs.technet.com/b/keithmayer/archive/2014/05/01/accelerating-devops-with-the-cloud-using-microsoft-azure-and-friends-part-1.aspx> [Accessed 4 April 2015].
24. Opennes CEE, 2014. Java on Azure – Contribution to OpenJDK, Java SDK for Azure Management and Resources. [online] Available at: <http://www.opennessatcee.com/post/101155515004/java-on-azure-contribution-to-openjdk-java-sdk> [Accessed 4 April 2015].
25. Data Center Knowledge, 2014. With Blue Mix, IBM Services Meet the SoftLayer Cloud. [online] Available at: <http://www.datacenterknowledge.com/archives/2014/03/27/blue-mix-ibm-services-meet-softlayer-cloud/> [Accessed 5 April 2015].
26. Techworld, 2011. IBM offers free Jazz Hub cloud development tool. [online] Available at:<http://www.techworld.com/news/apps/ibm-offers-free-jazz-hub-cloud-development-tool-3284711/> [Accessed 10 April 2015].
27. DeveloperWorks, 2014. Why Liberty profile is the best Java runtime for the cloud. [online] Available at: <https://developer.ibm.com/wasdev/docs/liberty-profile-best-java-runtime-cloud/> [Accessed 10 April 2015].
28. Google, 2015. The JRE Class White List. [online] Available at: <https://cloud.google.com/appengine/docs/java/jrewhitelist> [Accessed 14 April 2015].
29. Googleappengine, 2015. Issue 9537: Java 8 support. [online] Available at: <https://code.google.com/p/googleappengine/issues/detail?id=9537> [Accessed 14 April 2015].
30. Bitnami, 2015. What is Bitnami? [online] Available at: <https://bitnami.com/learn_more> [Accessed 14 April 2015].
31. Google, 2015. Push-to-Deploy. [online] Available at: <https://cloud.google.com/tools/repo/push-to-deploy> [Accessed 14 April 2015].
32. Google, 2015. Logging into a Windows virtual machine instance. [online] Available at: <https://cloud.google.com/compute/docs/operating-systems/windows#rdp-manual> [Accessed 18 April 2015].
33. Bitnami, 2015. Selenium Cloud Hosting, Selenium Hosting - Installers and VM. [online] Available at: <https://bitnami.com/stack/selenium>. [Accessed 18 April 2015].
34. IBM Developer Works dW Answers, 2015. Spring 2.4.2 App at Bluemix. [online] Available at: <https://developer.ibm.com/answers/questions/167743/spring-242-app-at-bluemix.html> [Accessed 18 April 2015].

35. WasDev, 2014. Download just the Liberty profile runtime. [online] Available at: <https://developer.ibm.com/wasdev/downloads/liberty-profile-using-non-eclipse-environments/> [Accessed 18 April 2015].
36. Cloud Foundry Documentation, 2015. Cloud Foundry Environment Variables. [online] Available at: <http://docs.cloudfoundry.org/devguide/deploy-apps/environment-variable.html> [Accessed 18 April 2015].
37. IBM, 2015. Pricing Calculator - IBM Bluemix. [online] Available at:. <https://console.ng.bluemix.net/?ace_base=true#/pricing> [Accessed 18 April 2015].
38. Sreenivasa Rao Vadalasetty, 2003. Security Concerns in Using Open Source Software for Enterprise Requirements. [online] Available at: <http://www.sans.org/reading-room/whitepapers/awareness/security-concerns-open-source-software-enterprise-requirements-1305> [Accessed 24 May 2015].

7. Appendices

Appendix A - Continuous Integration

CI aims to improve quality assurance by eliminating inconsistencies in the build and testing phase, the most common one being, the scenario where a developer would state that the latest change "is working on his machine". That introduces a risk as, at that point, it is not confirmed how that new change set will behave once it is deployed to a real environment.

CI is achieved by assembling a flow where the Development code stream is pulled by a centralized service that tests new code and build the deployable software artifacts, Paul M. Duvall, et al. (2007, p.40) describe the general idea of CI by stating that the steps in a CI scenario will typically go something like this:

1. First, a developer commits code to the version control repository. Meanwhile, the CI server on the integration build machine is polling this repository for changes (one approach is to perform the check every few minutes).
2. Soon after a commit occurs, the CI server detects that changes have occurred in the version control repository, so the CI server retrieves the latest copy of the code from the repository and then executes a build script, which integrates the software.
3. The CI server generates feedback by e-mailing build results to specified project members.
4. The CI server continues to poll for changes in the version control repository.

Figure 1 illustrates the elements of the CI system.

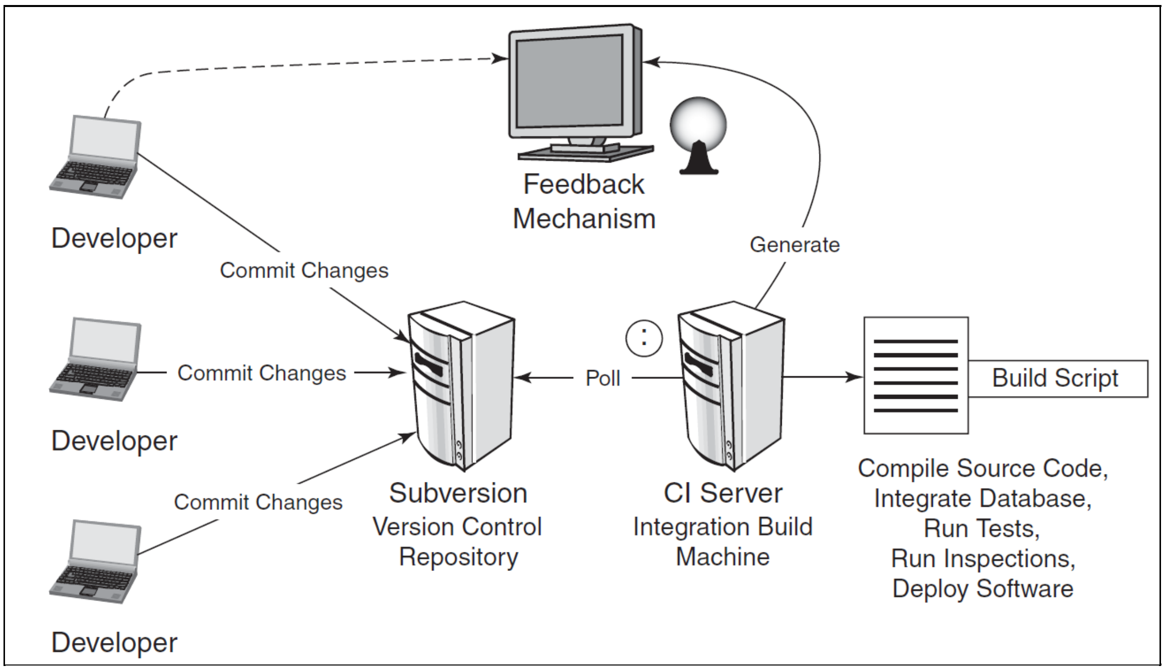


Figure A.1. *The components of a CI system.*

CI aims to automate and constantly repeat operations that are considered risky (error-prone), set up a centralized service to handle all changes and, mostly important, Testing. These tests can be comprised of Static Analysis (Checking for Code conventions issues, compilation problems or anticipating runtime exceptions by verifying variables that are not being handled properly) and Unit Tests, which is a practice aligned with Test-Driven-Development (TDD).

The TDD practice combined with the continual centralized build approach of CI improves the consistency of the tests and produces more robust software, it prevents the accumulation of bugs and, as each build cycle handles a small set of change sets, it is easier to troubleshoot and unblock the development pipeline, achieving, this way, a scenario where the team would have more time to invest being proactive instead of reactive.

Appendix B - Continuous Delivery

CD focuses on facilitating the deployment of the application by turning a manual, time-consuming, error-prone process into a stable and automated operation. This helps developers to efficiently turn ideas and changes into real value for their customers. It involves the development life-cycle of the application, focusing on improving the process of the deployment of the artifacts that are generated by the build phase into a “Production” environment. The deployable artifacts should be initially deployed to a "Development Environment". After they are deployed and the application is available, there is another round of automated tests that should simulate the operations performed by the end-users. This practice is very important to make the Software Development Life Cycle faster and more consistent. It achieves this by eliminating manual tasks and mitigating the deployment risks, the application is “Production-ready” if the resulting deployment matches the criteria of a functional application and it is able to deliver the services required by the end-users.

The most important concept within CD is the Deployment Pipeline (Figure B.1), Jez Humble, et al. (2011, p.38) state that “Every change that is made to an application’s configuration, source code, environment, or data, triggers the creation of a new instance of the pipeline. One of the first steps in the pipeline is to create binaries and installers. The rest of the pipeline runs a series of tests on the binaries to prove that they can be released. Each test that the release candidate passes gives us more confidence that this particular combination of binary code, configuration information, environment, and data will work. If the release candidate passes all the tests, it can be released”. This flow is illustrated in Figure B.1.

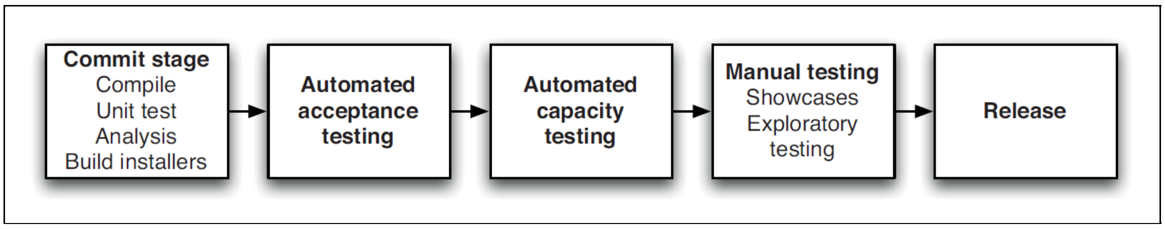


Figure B.1. *The Deployment Pipeline.*

The advantages introduced by CD comprise:

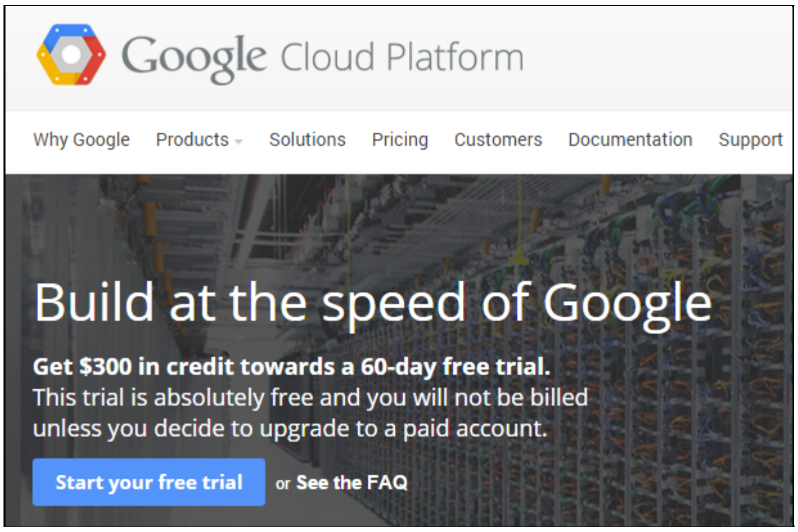
- More reliability in the deployment process through automation
- Elimination of risks by properly separating development, staging and production environments
- Effective understanding of the elements of the same through Configuration Management and control of such elements with version-control mechanisms

It also focuses on auditing through recorded operations and logging so the results of each deployment can be verified and even enhanced through continuous improvement.

Appendix C - Building a Deployment Pipeline with Google Cloud Platform

The following section describe how to create a Deployment Pipeline by leveraging IaaS and PaaS options from Google Cloud Platform.

The sign-up procedure was simple, when this research was being elaborated, Google was offering a 60-day free trial with a $300 credit (Figure C.1) that allows one to explore the services and evaluate some of its features. The access to the Developer Console can be enabled after linking a Google Account and providing Credit Cards details, this information is mandatory even for the free trial access.



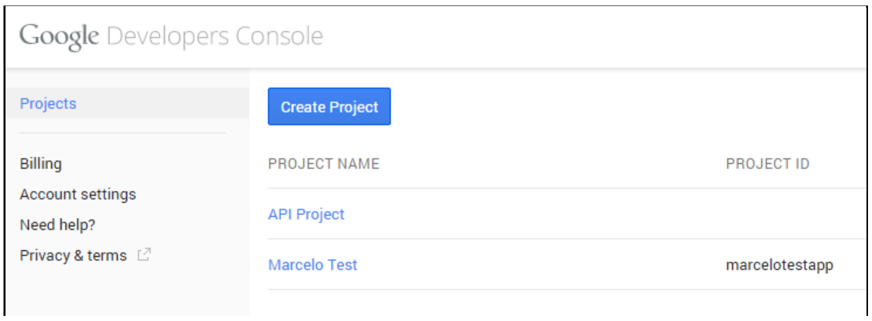
Build at the speed of Google

Get $300 in credit towards a 60-day free trial.
This trial is absolutely free and you will not be billed unless you decide to upgrade to a paid account.

Figure C.1. *Google Cloud Platform home page.*

C.1 The Developers Console

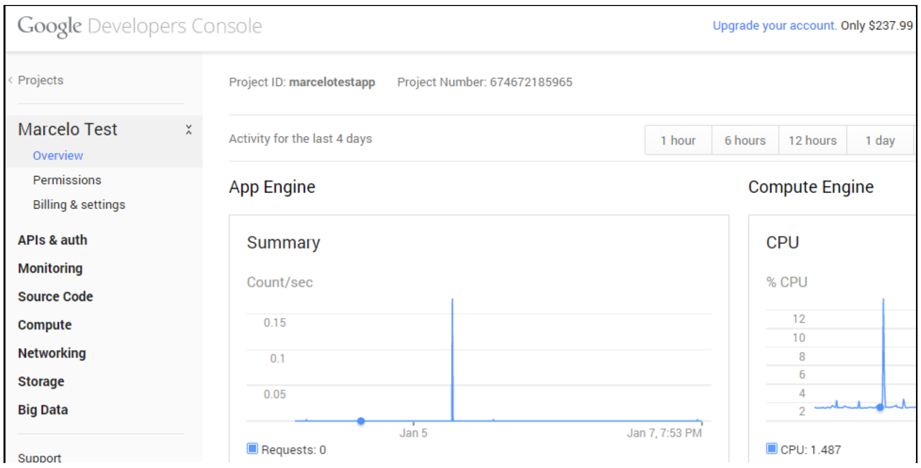
Once the sign up is completed, the user is presented to the Developers Console, this section is comprised of an initial menu that is used to manage projects and change general account configurations, including billing. A project is created through the "Create Project" button, it only requires a name and a Project ID (Figure C.2).



PROJECT NAME	PROJECT ID
API Project	
Marcelo Test	marcelotestapp

Figure C.2. *Creating a Google Cloud project.*

By accessing a specific project, the console expanded the menu showing all the services available to the project, it also presents a monitoring dashboard which includes data related to App Engine, Compute Engine, Errors, Disk IO and APIs (Figure C.3).



Project ID: marcelotestapp Project Number: 674672185965

🟦 Requests: 0

🟦 CPU: 1.487

Figure C.3. *Project's menu and monitoring dashboard.*

The console, at first glance, was overwhelming due to the amount of options but, for the purposes of this research, 4 main services were analyzed:

1. Monitoring:
   - Logs: For troubleshooting purposes. To check GAE logs.
2. Compute:
   - App Engine: Application Server instances to run the application that was deployed.
     Compute Engine: Virtual Machines to support additional tasks related to the pipeline.
3. Storage:
   - Cloud SQL: MySQL Instance that was used with the test application.
4. Source Code:

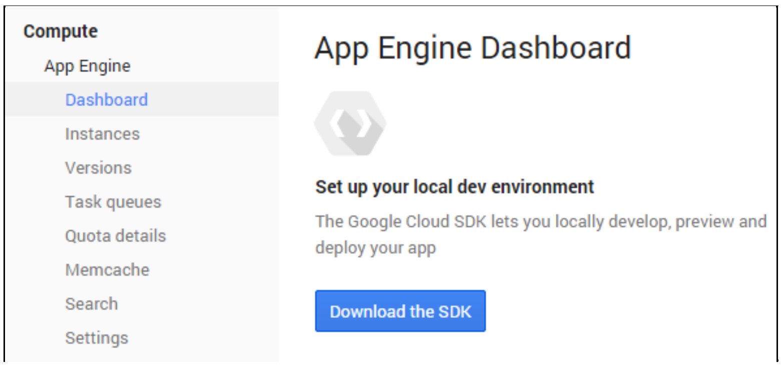
- Browse: To read and edit the source-code that was committed through Git, used for small configuration changes.
  Releases: This feature was explored in this research but later abandoned due to a misconfiguration that causes compatibility problems with GAE.

More details about each one of these services are shared in the sections documented below.

C.2 Google Application Engine (GAE)

In order to proceed with the evaluation of the DevOps tools, it was necessary to prepare the integration and delivery mechanisms to illustrate the Software Development Life Cycle within the Google Cloud Platform environment, a vital part of this process is the actual component that manages the application once it is deployed, that is the Google Application Engine (GAE). The GAE represents the Platform-as-a-Service portion of the services provided by Google Cloud, it is seen as an abstraction of an application server that supports different technologies like Java, Python or PHP, beyond hosting the application, it also covers important non-functional requirements with features like Automatic Scaling and Load Balancing, including the integration with other Google Cloud Services and APIs. As per Google’s official documentation (2015), “Google App Engine makes it easy to build and deploy an application that runs reliably even under heavy load and with large amounts of data”.

The console presents several "Quick Start" options within each category, in order to gather some understanding about the deployment steps and the expected template of a GAE Application, the author followed the instructions to "Set up your local dev environment" (Figure C.4), which is found within the "Compute – App Engine – Dashboard" section:

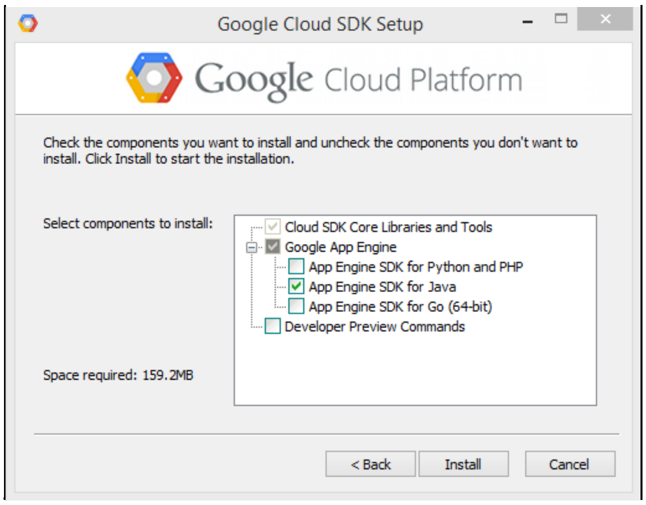


Set up your local dev environment

The Google Cloud SDK lets you locally develop, preview and deploy your app

Figure C.4. *Setting up local Development environment.*

The instructions are comprised of 3 steps: "Install Cloud SDK", "Get Started with Google Cloud Platform Service" and "Submit Questions and give us feedback". The first step (Install Cloud SDK) is very straight forward, in this research, the "GoogleCloudSDKInstaller.exe"file was downloaded and installed on a Windows workstation after selecting the "App Engine SDK for Java" option in the installation wizard (Figure C.5). This step installs the client-side libraries for all the Google Cloud API calls that are used to interact with the services programmatically, it must also install some sort of security mechanism, like Secure Shell (SSH) Keys, that allows this particular workstation to invoke such operations through the command-line.



Check the components you want to install and uncheck the components you don't want to install. Click Install to start the installation.

Select components to install:

- [✓] Cloud SDK Core Libraries and Tools
- [✓] Google App Engine
  - [ ] App Engine SDK for Python and PHP
  - [✓] App Engine SDK for Java
  - [ ] App Engine SDK for Go (64-bit)
- [ ] Developer Preview Commands

Space required: 159.2MB

Figure C.5. *Google Cloud SDK Setup wizard.*

The second step (Get Started with Google Cloud Platform Service) involves many suggestions involving all services, in this research the "Try Google App Engine Now" link was chosen. In this next page, the user follows a sequence of quick instructions to prepare a small sample application, run it locally and deploy it to the App Engine. This sample application is called "appengine-try-java.zip" and it is available for download on the same page, the structure of this application is very similar to any regular Maven web application ("maven-archetype-webapp"), except for the presence of an additional Deployment Descriptor (DD) found in the "src/main/webapp/WEB-INF" directory: "appengine-web.xml".

This additional file, appengine-web.xml, is used to "specify the app's registered application ID and the version identifier of the latest code", it also defines system properties, environment variables, configure Secure URLs (SSL) and manages other pieces of web-related configurations.

Based on this examination of the sample app, the initial structure of the test application was created using Maven's webapp archetype generation command.

```
mvn archetype:generate -DgroupId=com.themarcelor.calmlywriting -DartifactId=CalmlyWriting
-DachetypeArtifact=maven-archetype-webapp -DinteractiveMode=false
```

Code listing C.1: Maven command to build the basic structure of the app.

The project was then populated with the source-code of the sample Web Application, the pom.xml file was edited with the main components used by the Java application, the Project Object Model file is used by Maven to manage the project dependencies and the other steps of its building and deploying lifecycle.

The project was imported to Eclipse to facilitate the adjustments to the source-code and the "pom.xml" file, some dependencies had to be adjusted based on the restrictions associated with GAE (Google, 2015), which only allows certain Java Library classes to be loaded into its JVM. For every dependency that is removed from the "pom.xml", it is recommended to re-create the project's classpath within Eclipse to avoid any compilation issues on the Integrated Development Environment (IDE) interface, this is done through the following Maven command: "mvn eclipse:eclipse".

In order to test the application locally, the following command is used: "mvn appengine:devserver -DskipTests", with this approach, the Unit Tests are skipped to conduct an initial evaluation of the local GAE development server. The database connection details are not properly configured so the sample application is not fully functional at this point.

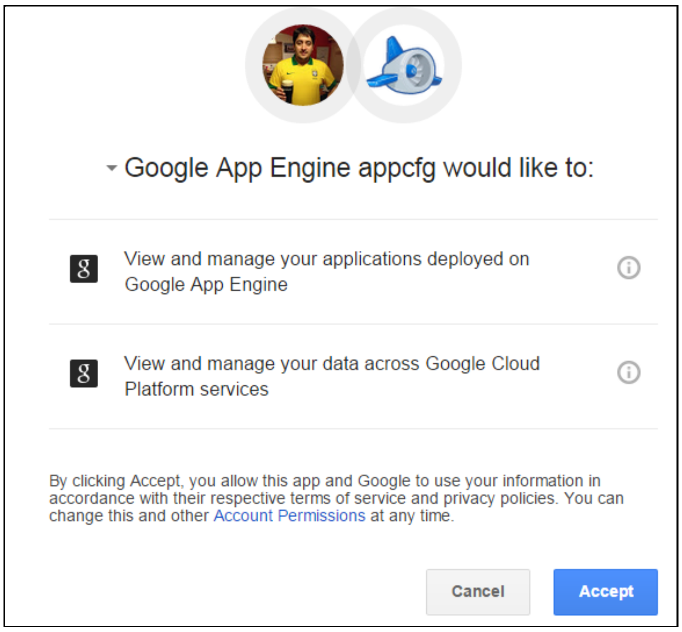
```
[INFO]
[INFO] Google App Engine Java SDK - Running Development Server
[INFO]
[INFO] Retrieving Google App Engine Java SDK from Maven
Downloading: https://oss.sonatype.org/content/repositories/comgoogleappengine-10
04/com/google/appengine/appengine-java-sdk/1.9.17/appengine-java-sdk-1.9.17.zip
Downloading: https://repo.maven.apache.org/maven2/com/google/appengine/appengine
-java-sdk/1.9.17/appengine-java-sdk-1.9.17.zip
15728/166765 KB
```

Code listing C.2: Downloading GAE Development Server through Maven.

The application's code also needs to be changed so it won't require any Web Context, this was done to follow the expected URL mapping of the GAE. After running this command, the application is accessed locally through the following URL: "http://localhost:8080".

The Maven build process triggers the Unit Test classes by default so, since the sample application is using "JUnit" to test the common operations against the database (create, read, update, delete, list items), the "-DskipTests" system property was used to skip this step, allowing this way, the initial deployment against the local instance of GAE that is installed along with the Google Cloud SDK. This system property that skips the Unit Tests is no longer necessary once the MySQL Database is configured and the application tables are created, this approach was used just as a sanity check to verify if the application can be deployed against the local GAE instance.

To publish the application to Google Cloud and automatically provision an instance of GAE, the following command is used: "mvn clean appengine:update -DskipTests". This command deploys the application's main deployable artifact, which, in this case is a ".war" package that is produced by the Maven build. This approach uploads the packaged application to the GAE without involving any source-code repository. The “appengine:update” command builds the artifact and exceptionally, the first time it is executed, it prompts the user for an OAuth token. This token is an authorization code that allows this client to upload the application package to the cloud. This is a security check triggered by the command line operation, it opens an instance of the default browser, prompts the user to login with a Google account (Figure C.6) and asks the user to authorize the local GAE Configuration to manage applications in the Google Cloud platform through Open Authentication (OAuth). once the user accepts it, a code is presented on the browser interface and the same must be copied and pasted into the command line interface, this operation creates a file under the user’s home directory (.appcfg_oauth2_tokens_java) to allow all subsequent requests to update the application in Google Cloud.


Google App Engine appcfg would like to:

View and manage your applications deployed on Google App Engine

View and manage your data across Google Cloud Platform services

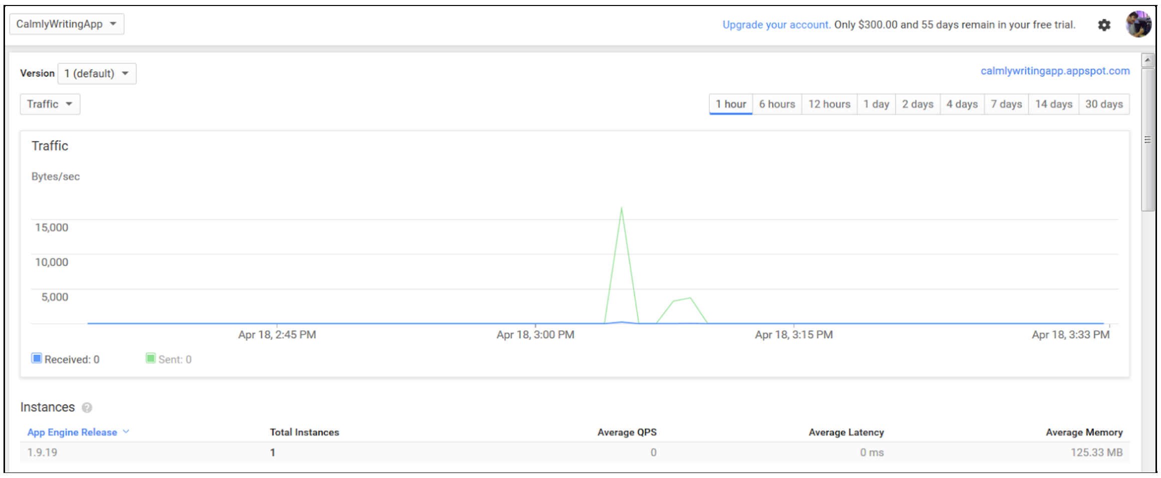
By clicking Accept, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

Figure C.6. Authorizing local GAE Configuration to manage cloud applications.

Once the OAuth token is provided in the command-line interface, the Maven build process is completed and the GAE dashboard in the Developers Console becomes available (Figure C.7), it presents a dropdown with several monitoring options:

- Requests by type
- Latency
- Loading latency
- Error details
- Traffic
- Utilization
- Instances
- Memory Usage
- Memcache Operations
- Memcache Compute Units
- Memcache Traffic
- Memcache Total Cache Size
- Memcache Hit Ratio

At the bottom, it shows a summary of Instances, Billing Status, Current Load, Server Errors and Client Errors, on the top-right corner it shows the URL of the recently-deployed application.


CalmlyWritingApp
Upgrade your account. Only $300.00 and 55 days remain in your free trial.
Version 1 (default)
calmlywritingapp.appspot.com
Traffic
1 hour 6 hours 12 hours 1 day 2 days 4 days 7 days 14 days 30 days
Traffic
Bytes/sec
15,000
10,000
5,000
Apr 18, 2:45 PM
Apr 18, 3:00 PM
Apr 18, 3:15 PM
Apr 18, 3:33 PM
Received: 0
Sent: 0
Instances

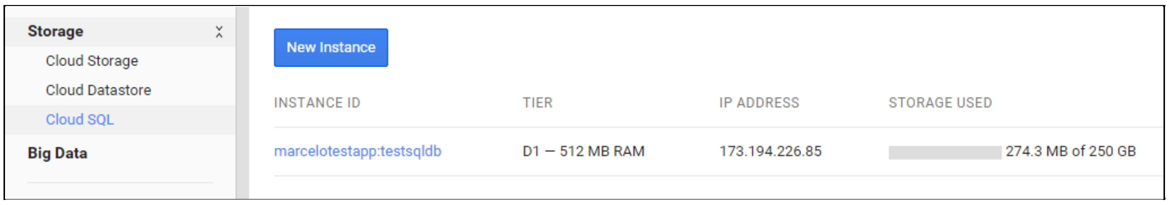
App Engine Release	Total Instances	Average QPS	Average Latency	Average Memory
1.9.19	1	0	0 ms	125.33 MB

Figure C.7. Google Application Engine dashboard.

That is result of the newly created GAE instance that is automatically provisioned once the “appengine:update” command is executed. From this point onwards, the Google Cloud platform automatically provisions additional instances of the GAE server depending on the workload within the application.

C.3 Storage – Cloud SQL

The sample application is only functional on the Google Cloud environment if it can connect to a database, so an instance of the Cloud SQL component was created to accommodate this need. To create a Cloud SQL instance, the user needs to navigate to the Developers Console, select the option "Cloud SQL" under the "Storage" section of the menu and click on the "New Instance" button (figure C.8).

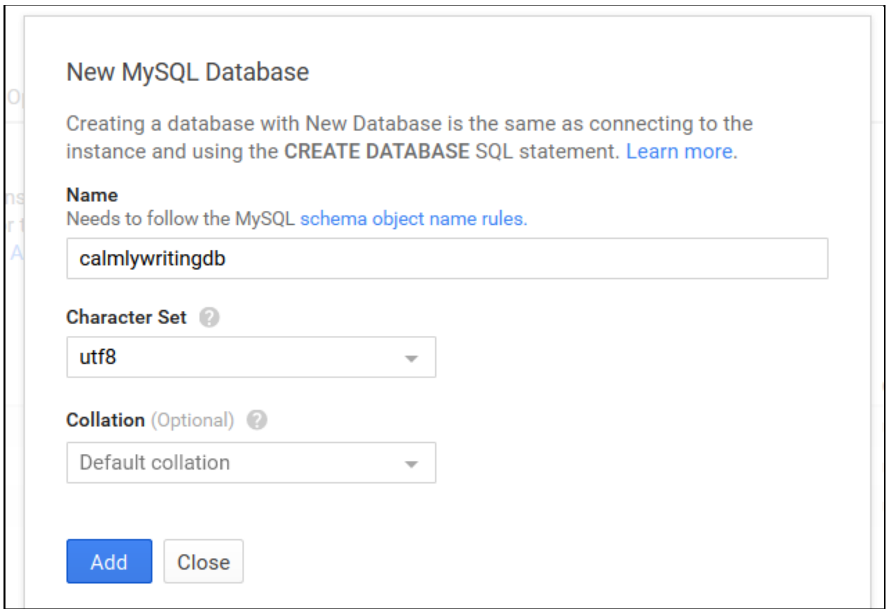


INSTANCE ID	TIER	IP ADDRESS	STORAGE USED
marcelotestapp:testsqldb	D1 — 512 MB RAM	173.194.226.85	274.3 MB of 250 GB

Figure C.8. Creating *Cloud SQL instance.*

In the “Create Cloud SQL Instance” page, it asks the user to specify an “Instance ID”, region (Make sure the same region that hosts the GAE instance is selected), tier (the default is “D1 - 512MB” of RAM) and options involving the backup schedules and the “Activation policy”, the latter specifies how the database system is activated, the default option is “On Demand”, which means, the database is activated as soon as it receives any incoming requests and it shutdown automatically after a few minutes of inactivity.

This provisions an instance of MySQL 5.5 by default in Google Cloud's infrastructure, the next steps are the creation of the "root" user associated with the "%" client host (the percentage symbol allows connections from any client) and the creation of the actual database object, which was named as "calmlywritingdb" (Figure C.9).


New MySQL Database

Creating a database with New Database is the same as connecting to the instance and using the **CREATE DATABASE** SQL statement. Learn more.

Name
Needs to follow the MySQL schema object name rules.
calmlywritingdb

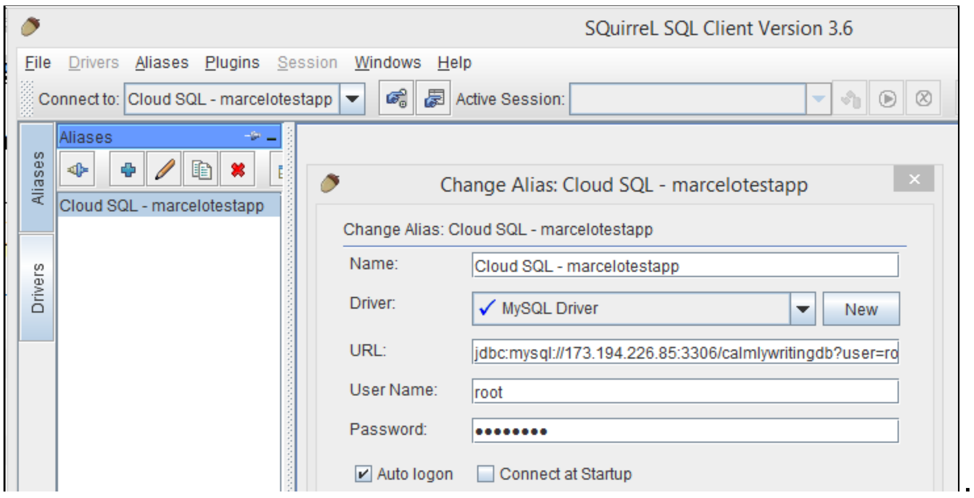
Character Set
utf8

Collation (Optional)
Default collation

Figure C.9. Creating *database instance for the sample application.*

Once the database is created it is necessary to authorize the existing Google Application Engine application to access the database, this is done by clicking on the database instance, navigating to the "Access Control" tab and adding the Application ID in the "Authorized App Engine Applications" section.

Google Cloud's database is a special MySQL Database instance that requires a special driver when applications are deployed to GAE. For the purposes of this research, the Cloud SQL database was configured remotely by using a SQL Client called "Squirrel SQL Client" (Figure C.10), a connection alias containing a MySQL Driver was created and the SQL scripts were executed to create the proper tables required by the sample application.


SQuirreL SQL Client Version 3.6
File Drivers Aliases Plugins Session Windows Help
Connect to: Cloud SQL - marcelotestapp
Active Session:
Aliases
Drivers
Cloud SQL - marcelotestapp
New

Change Alias: Cloud SQL - marcelotestapp

Name:	Cloud SQL - marcelotestapp
Driver:	✓ MySQL Driver
URL:	jdbc:mysql://173.194.226.85:3306/calmlywritingdb?user=ro
User Name:	root
Password:	••••••••

[✓] Auto logon [ ] Connect at Startup

Figure C.10. *Using SQuirrel SQL Client to configure the application's database.*

The Spring MVC configuration file (mvc-servlet.xml) defines how the Java Database Connectivity (JDBC) Connection Pool is generated, it basically loads a properties file that is placed in the application's "resource" folder and, based on its data, it creates the proper connection to the database instance.

```
<!-- database -->
<bean
   id="dataSource"
   class="org.apache.commons.dbcp.BasicDataSource"
   destroy-method="close" >
   <property name="driverClassName" value="${jdbc.driverClassName}" />
   <property name="url" value="${jdbc.url}" />
   <property name="username" value="${jdbc.username}" />
   <property name="password" value="${jdbc.password}" />
</bean>
```

Code listing C.3: Spring JDBC Connection Pool configuration.

Each environment has its own properties file containing the connection details for its own MySQL Server instance, the "Development" environment has local connection information and credentials to the local database instance whereas, the "Production" (GAE) environment, is different based on the special requirements demanded by Google's Cloud SQL database instance, within the Google Developers Console, it is possible to specify which applications have access to a given Cloud SQL instance so no password is necessary when connecting to the Cloud SQL database from the GAE environment. GAE should inject the required Google Driver into the application once it is deployed into GAE, another requirement that is described by the documentation is that the "Connector J" (Special connector for MySQL) needs to be defined in the application's "appengine-web.xml", this connector is defined by adding

"<use-google-connector-j>true</use-google-connector-j>" to Google Application Engine's deployment descriptor.

```
###################          MySQL          Configuration          –          Development.properties
######################
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/calmlywritingdb
jdbc.username=root
jdbc.password=root
jdbc.dialect=org.hibernate.dialect.MySQLDialect
```

Code listing C.4: JDBC properties for local deployment.

```
###################          MySQL          Configuration          –          AppEngine.properties
########################
jdbc.driverClassName=com.mysql.jdbc.GoogleDriver
jdbc.url=jdbc:google:mysql://marcelotestapp:testsqldb/calmlywritingdb?user=root
jdbc.username=root
jdbc.password=
jdbc.dialect=org.hibernate.dialect.MySQLDialect
```

Code listing 4.5: JDBC properties for App Engine deployment.

In this research, the steps were elaborated to try to automate everything as much as possible, therefore, in order to avoid any changes to the source-code to specify the properties file, the Spring MVC configuration was adjusted to load the name of the properties file from an environment variable called "ENV_SYSTEM", the default value of this configuration property is "AppEngine" so, if the environment variable is not defined, the application is deployed loading the data from "AppEngine.properties". In order to test the application locally, it is necessary to run the command "export ENV_SYSTEM='Development'" to define the variable before starting the unit tests and deploy the application to the local GAE instance with "mvn appengine:devserver".

```
<context:property-placeholder location="classpath:${ENV_SYSTEM:AppEngine}.properties" />
```

Code listing C.6: Spring MVC configuration to load environment-specific JDBC connection data (Default option: AppEngine).

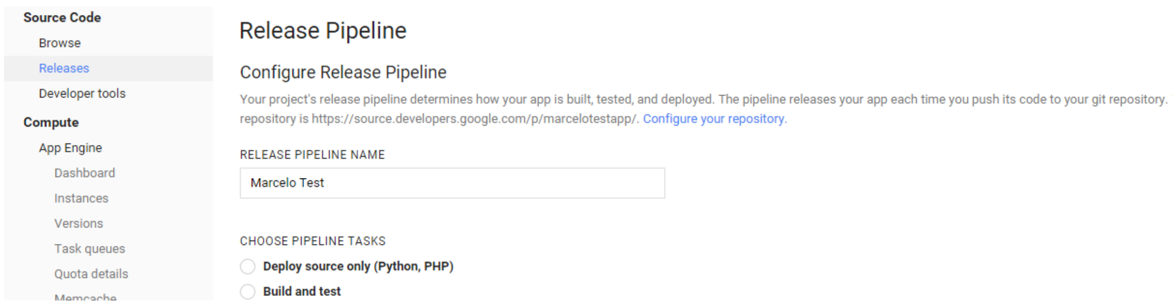
This configuration is revisited when the Continuous Integration environment is assembled, a third properties file is created to store database connections details specific to the Google Cloud SQL database.

C.4 Source Code – Release management

Google Cloud also offers a Source Code Management (SCM) system called "Git", which manages the application's source-code in the Cloud, this repository is used for the "Push-and-Deploy" approaches that are offered by the Google Cloud Platform: "Automatic Releases pipeline" and "The GCE Jenkins VM".

The “” service automatically creates a Deployment Pipeline and connects it to the source-code so it can build, test and deploy the package to the GAE instance. The proposal of the feature is to provide this basic Continuous Integration environment with minimal configuration effort.

In order to configure the releases’ pipeline in the Google Cloud Platform, the user needs to navigate to the Google Developers Console and select the option "Releases" under the "Source Code" section. A form (Figure C.11) is presented so the user can specify some options involving the pipeline, for this research, "Build and Test" was selected along with the "Java: Use Maven to build, test and deploy" option (Figure C.12).



Release Pipeline

Configure Release Pipeline

Your project's release pipeline determines how your app is built, tested, and deployed. The pipeline releases your app each time you push its code to your git repository. repository is https://source.developers.google.com/p/marcelotestapp/. Configure your repository.

RELEASE PIPELINE NAME

Marcelo Test

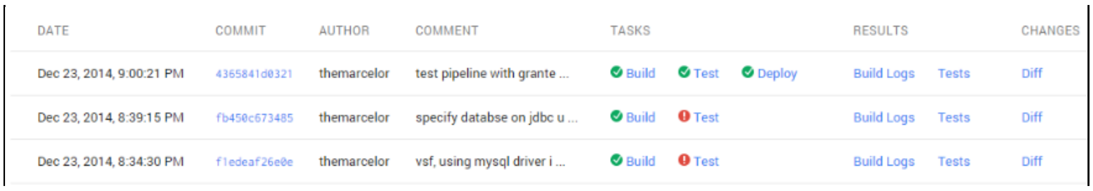
CHOOSE PIPELINE TASKS

○ Deploy source only (Python, PHP)

○ Build and test

Figure C.11. *Configuration of the automatic Release Pipeline.*

This option is presented two major issues, the first one is that the access to Jenkins is limited and none of the build steps can be configured, the second issue is related to a Java compilation problem that was identified during the research, the automatic configuration creates, by default, a Jenkins VM within the Google Compute Engine (GCE) Infrastructure-as-a-Service (IaaS) environment, this VM uses a Java Development Kit (JDK) version 8 to compile the application code and the GAE does not support Java Server Pages (JSPs) compiled with this particular version, only the ones compiled with JDK 7 or earlier versions, a workaround to change the JDK version within the service was not identified, the issue was already identified by a number of users and it is being tracked in a bug report (Googleappengine, 2015).



DATE	COMMIT	AUTHOR	COMMENT	TASKS			RESULTS		CHANGES
Dec 23, 2014, 9:00:21 PM	4365841d0321	themarcelor	test pipeline with grante ...	✓ Build	✓ Test	✓ Deploy	Build Logs	Tests	Diff
Dec 23, 2014, 8:39:15 PM	fb450c673485	themarcelor	specify databse on jdbc u ...	✓ Build	! Test		Build Logs	Tests	Diff
Dec 23, 2014, 8:34:30 PM	f1edeaf26e0e	themarcelor	vsf, using mysql driver i ...	✓ Build	! Test		Build Logs	Tests	Diff

Figure C.12. Google Cloud's *Release Pipeline.*

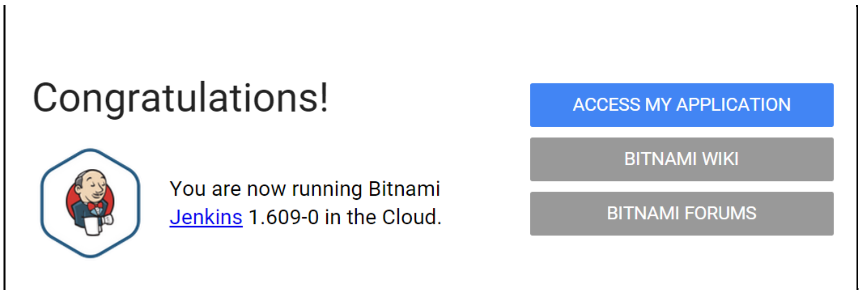
Currently, there is no solution for this problem as the VM is automatically configured and the access to the Jenkins management console is limited. This issue delayed the evaluation of the Continuous Integration and Continuous Delivery components as an alternative had to be considered.

C.5 Creation of GCE Jenkins VM

This approach involves the creation of a Jenkins VM through the Google Cloud Command Line Interface (CLI) component, followed by its configuration and, in addition to the first approach, as the access to the Jenkins management console is now available, it also includes the creation of a second GCE VM that hosts a Selenium Hub, allowing a pipeline that involves build, Unit Testing, deploy to GAE and Graphic User Interface (GUI) tests through the Selenium server. This approach also comprises a conditional build tagging mechanism that is added to the end of the pipeline, builds that are successfully deployed and provide expected results on user interface tests to indicate which build needs to be tagged properly to be eligible for promotion to the next environments.

The creation of the Jenkins VM is done through the CLI tool called "gcloud" and its mechanism to automatically load Bitnami images, according to their online page (Bitnami, 2015), "Bitnami is a library of popular server applications and development environments that can be installed with one click, either in your laptop, in a virtual machine or hosted in the cloud. We take care of compiling and configuring the applications and all of their dependencies (third-party libraries, language runtimes, databases) so they work out-of-the-box". The "gcloud" tool easily controls the services provided by the Google Cloud Platform, the command to provision the Jenkins VM (Figure C.13) contains details like user credentials and Virtual Machine's metadata.

```
C:\tech\google\cloud_apps\marcelotestapp>gcloud compute instances create bitnami
-jenkins --project "marcelotestapp" --image-project bitnami-launchpad --image bi
tnami-jenkins-1-587-0-linux-debian-7-x86-64-image --zone us-central1-a --machine
-type n1-standard-1 --metadata "bitnami-base-password=*********" "bitnami-def
ault-user=user" "bitnami-key=jenkins" "bitnami-name=Jenkins" "bitnami-version=1-
587-0" "bitnami-url=//bitnami.com/stack/jenkins" "bitnami-description=Jenkins."
"startup-script-url=https://dl.google.com/dl/jenkins/p2dsetup/setup-script.sh" -
-scopes "https://www.googleapis.com/auth/userinfo.email" "https://www.googleapis
.com/auth/devstorage.full_control" "https://www.googleapis.com/auth/projecthosti
ng" "https://www.googleapis.com/auth/appengine.admin" --tags "bitnami-launchpad"
```
Code listing C.7: Creating Jenkins Bitnami VM through the "gcloud" command.

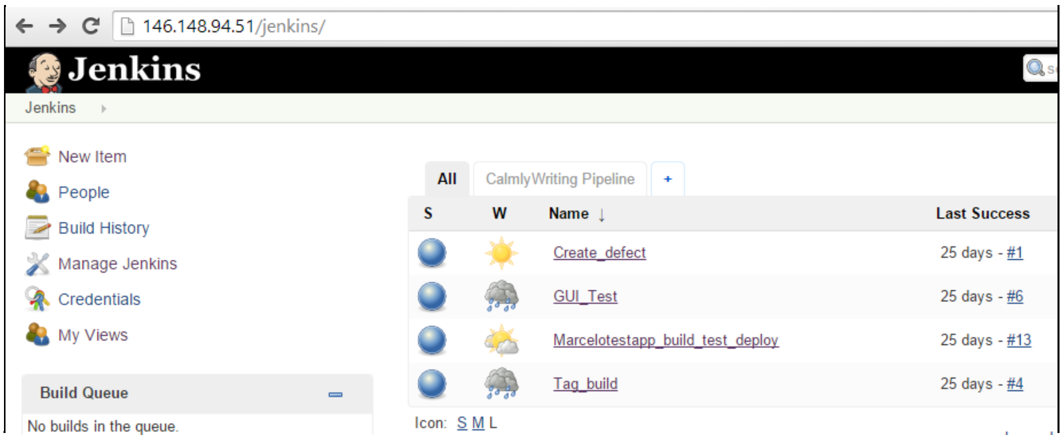

Congratulations!

You are now running Bitnami Jenkins 1.609-0 in the Cloud.

Figure C.13. *Bitnami Jenkins provisioned through the Google Cloud Platform.*

The configuration of the Jenkins jobs is done through the Jenkins management console, for the purposes of this research, four jobs were created to assemble the Continuous Integration and a basic Continuous Delivery pipeline (Figure C.14):

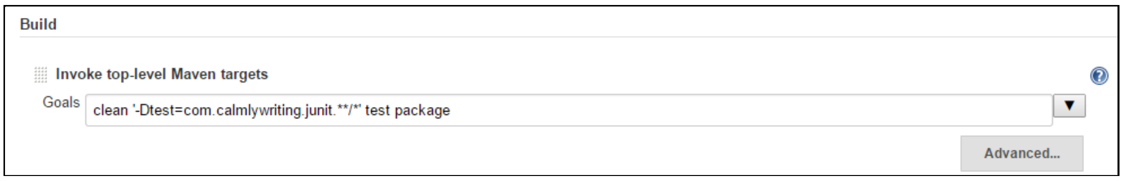
1. **Build_test_deploy**
2. **GUI_Test**
3. **Tag_build**
4. **Create_defect**



S	W	Name ↓	Last Success
●	☀	Create_defect	25 days - #1
●	🌧	GUI_Test	25 days - #6
●	⛅	Marcelotestapp_build_test_deploy	25 days - #13
●	🌧	Tag_build	25 days - #4

Figure C.14 *Jenkins interface showing Deployment Pipeline jobs.*

1) Build_test_deploy: This step connects to the Git repository, retrieves the code, run the required Unit Tests and builds the web application package (.war) so that it can be deployed to GAE. It has a downstream Jenkins project called "GUI_Test" that is triggered only if this initial steps are completed successfully. The Maven command used in this step specifies only Unit Tests (Figure C.15) and it skips the test cases declared under the "selenium" package (clean -Dtest=com.calmlywriting.junit.**/* test package). A second step is added to perform the deployment operation, the following "Shell Script" command was used: "gcloud --project=calmlywritingapp preview app deploy target/CalmlyWriting" (without the .war extension).



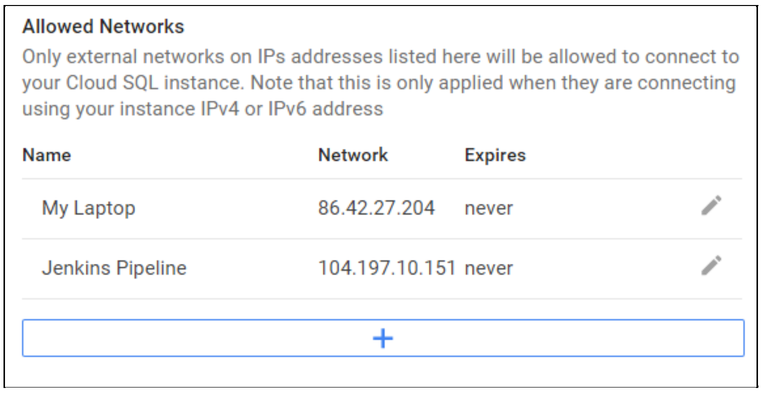
Build

Invoke top-level Maven targets

Goals: clean '-Dtest=com.calmlywriting.junit.**/*' test package

Figure C.15. *Jenkins build configured with Maven goals.*

In order to properly configure the access to the Cloud SQL database and allow the Unit Tests to run, a third properties file was created (Pipeline.properties) to be consumed exclusively by Unit Tests executed in the Jenkins build step. This is done through the configuration of the "ENV_SYSTEM" environment variable under the Slave machine that is used by this Jenkins Bitnami VM (Under the configuration of the "cloud-dev-java" slave, there is a "config" option that leads to a form that allows the user to add environment variables). Another dependency is related to "Access Control" within the Google SQL instance, the user needs to authorize the communication between the Jenkins Pipeline and the Database server, this is done by navigating to the Google SQL access configuration page and adding the IP Address of the Jenkins Bitnami VM (Figure C.16).



Allowed Networks

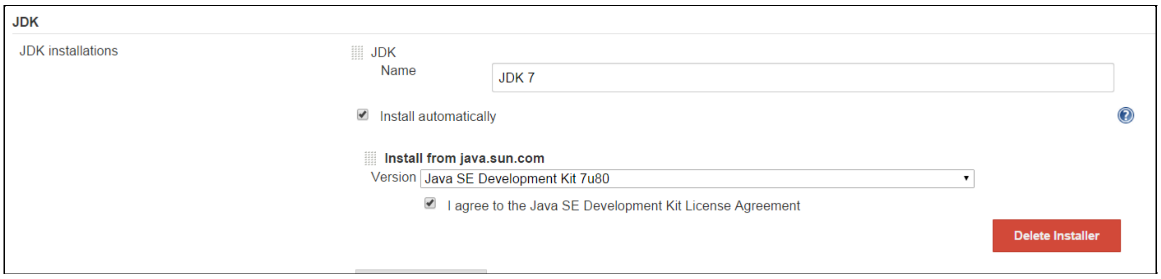
Only external networks on IPs addresses listed here will be allowed to connect to your Cloud SQL instance. Note that this is only applied when they are connecting using your instance IPv4 or IPv6 address

Name	Network	Expires
My Laptop	86.42.27.204	never
Jenkins Pipeline	104.197.10.151	never

Figure C.16. *Google SQL Access Control.*

With this additional configuration applied, the “Build_test_deploy” project is almost ready to be executed, however, the initial trigger of the pipeline needs to be automated, this is achieved by configuring the project once more and select the “Poll SCM” option under the “Build Triggers” section, the official documentation suggests the “H/5 * * * *” for the configuration for the scheduled task, this performs a check on the GIT repository for any new commits every 5 minutes.

The Jenkins Bitnami image is provisioned with the JDK 8 by default and Google Application Engine is not compatible with JSPs compiled with this version, to solve this problem the author had to adjust Jenkins configuration by navigating to the “Manage Jenkins > Configure System” page and adding the JDK 7 (Figure C.17), the Oracle credentials must be provided and once the configuration is saved, the job configuration page presents a drop-down so the user can select the new JDK, this is the workaround to prevent compilation failures while trying to deploy the application package to GAE.


JDK

JDK installations

JDK
Name: JDK 7

☑ Install automatically

Install from java.sun.com
Version: Java SE Development Kit 7u80

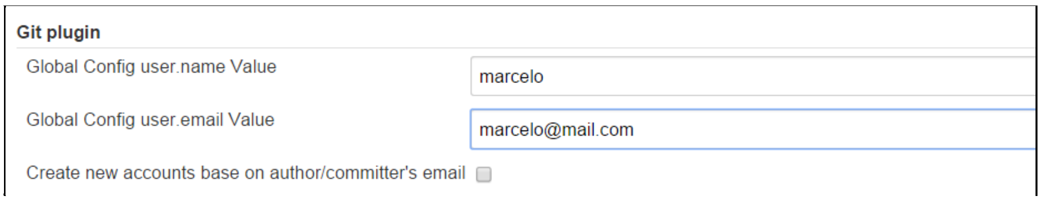
☑ I agree to the Java SE Development Kit License Agreement

Figure C.17. *JDK 7 must be installed to solve JSP compilation issues in GAE.*

2) GUI_Test: This step executes the Selenium Classes that validate the interactions with the sample application's interface (using the following Maven command: -Dtest=com.calmlywriting.selenium.** test), it is triggered by a successful execution of the previous project (Build_test_deploy). The Selenium classes need to connect to a Selenium Hub that delegates the test processing to a Remote WebDriver, based on this need, a second GCE Virtual Machine was created to host both the Selenium Hub and the Web Driver to run a Firefox browser for each "GUI_Test" build. The creation of this VM is discussed in the next section.

3) Tag_build: The last two Jenkins projects are also conditional, if the "GUI_Test" build fails, it should create a defect, if it is executed successfully, it should tag this build within the GIT repository so the files associated with this package of the web application are marked as eligible to be promoted to the next code stream and environment.

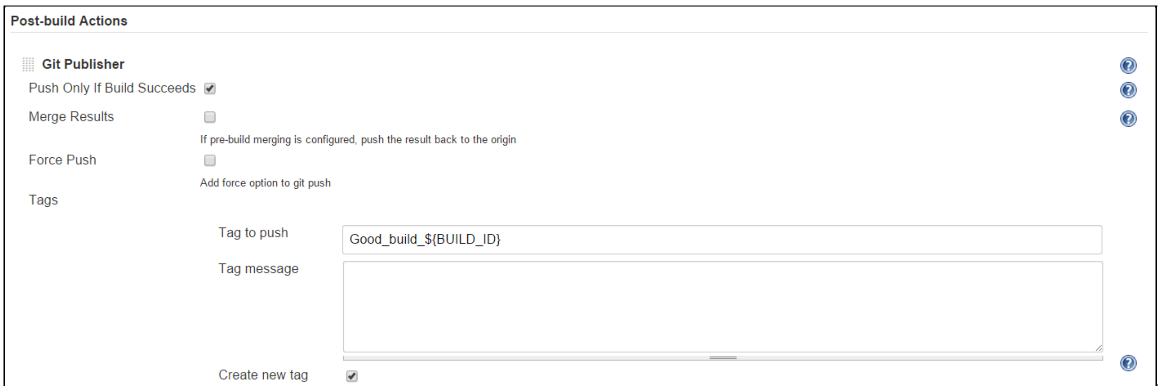
Before the build can be tagged, Jenkins needs to configure the global user for the GIT system, this is done in “Manage Jenkins > Configure System”, within the “GIT Plugin” section, the author provided a username and an email address to satisfy this dependency (Figure C.18).


Git plugin

Global Config user.name Value	marcelo
Global Config user.email Value	marcelo@mail.com
Create new accounts base on author/committer's email	☐

Figure C.18. *Global user name configured in Jenkins’ GIT Plugin.*

The tagging is configured through a post-build action configured within the Jenkins project (Figure C.19), the Git Publisher option allows the user to create tags within the GIT repository to create a logical snapshot of the changeset that was committed, since multiple builds are processed throughout the development cycle, the tags were named as “Good_build_${BUILD_ID}”, the “BUILD_ID” variable is appended to the number of the “Tag_build” iteration to the tag name, therefore, each tag is unique.


Post-build Actions

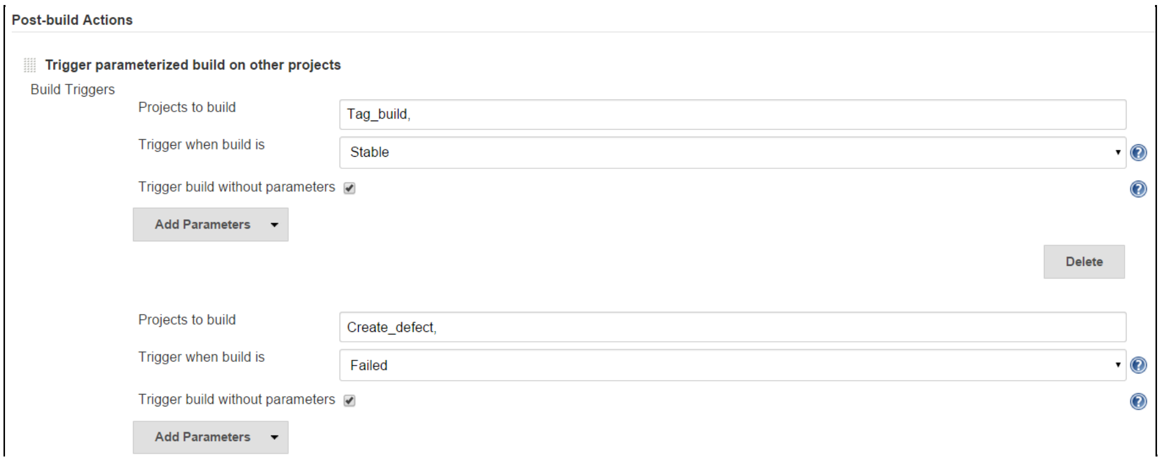
Git Publisher

Push Only If Build Succeeds	✓
Merge Results	☐ If pre-build merging is configured, push the result back to the origin
Force Push	☐ Add force option to git push
Tags	
Tag to push	Good_build_${BUILD_ID}
Tag message	
Create new tag	✓

Figure C.19. *Create tags in the GIT repository.*

4) Create_defect: This step should, ideally, create an entry in a bug tracking system if the GUI Test fails. It can trigger an Application Programming Interface (API) call to a Bugzilla or JIRA and assign to the author of the class associated with the feature that was being tested. Since there is no orchestration available with a bug tracking service, the author just placed a Shell Command to print a message, just for testing purposes.

To automatically trigger the last two projects (“Tag_build” and “Create_defect”) conditionally, the author used the “Parameterized Build” plugin (Figure C.20) and adjusted the “GUI_Test” project to invoke each build based on the status of its job, builds are tagged when the tests are executed successfully, whereas defects are reported if something fails.


Post-build Actions

Trigger parameterized build on other projects

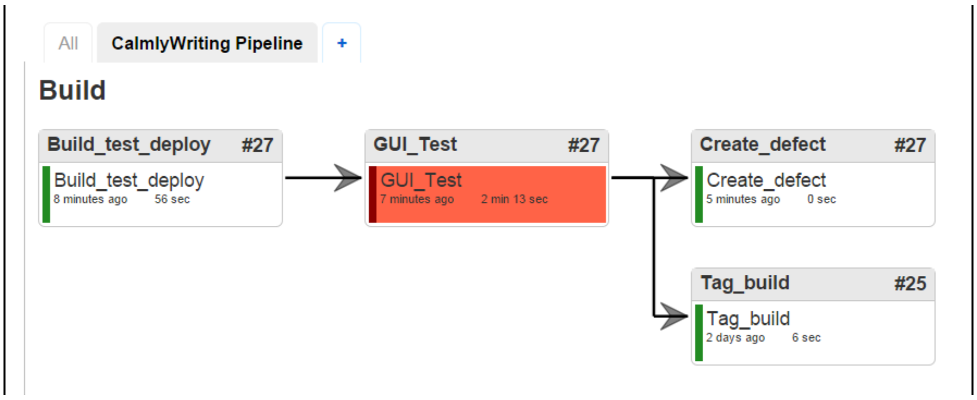
Build Triggers

Projects to build	Tag_build,
Trigger when build is	Stable
Trigger build without parameters	✓

Projects to build	Create_defect,
Trigger when build is	Failed
Trigger build without parameters	✓

Figure C.20. *Configuration of conditional builds.*

To help the user visualize the flow of the Deployment Pipeline, a plugin called "Delivery Pipeline Plugin" was installed, the plugin installation does not require a restart and it is done through the "Manage Jenkins" page, the author specified all four projects in the plugin configuration and created a new Jenkins view (Figure C.21).



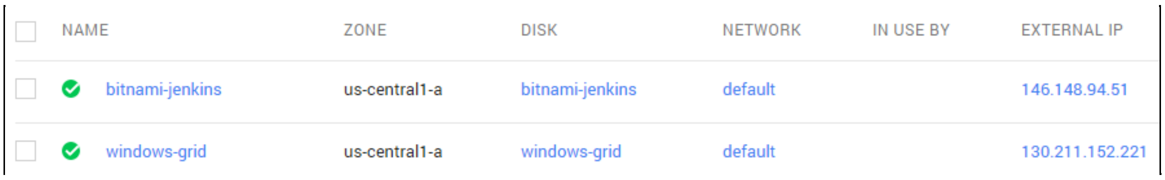
All
CalmlyWriting Pipeline
+
Build
Build_test_deploy #27
Build_test_deploy 8 minutes ago 56 sec
GUI_Test #27
GUI_Test 7 minutes ago 2 min 13 sec
Create_defect #27
Create_defect 5 minutes ago 0 sec
Tag_build #25
Tag_build 2 days ago 6 sec

Figure C.21. *The CalmlyWriting Pipeline hosted in Google Cloud.*

The pipeline was tested by applying a small change to the text on the index page, the author modified the contents of the page and committed the code through GIT, then the Deployment Pipeline reacts to the change (as per the SCM Polling that was configured in Jenkins) and process the testing flow to validate the change, a build is tagged at the end of the process as a result of a good build.

C6 Creation of GCE Selenium Hub VM

The Selenium VM was also created with the "gcloud" CLI tool, however, there is no image with a Selenium Hub pre-installed. The author could not identify a more efficient way to provision a GUI Test service automatically. This VM is necessary to allow the invocation of the Firefox browser from the Selenium Web Driver that runs the sample application's JUnit test cases. A Windows Server 2008 Virtual Machine was created (Figure C.22) to facilitate the configuration of the components. The instructions were taken from the "Logging into a Windows virtual machine instance" page (Google, 2014).



NAME	ZONE	DISK	NETWORK	IN USE BY	EXTERNAL IP
✓ bitnami-jenkins	us-central1-a	bitnami-jenkins	default		146.148.94.51
✓ windows-grid	us-central1-a	windows-grid	default		130.211.152.221

Figure C.22. *The Jenkins server and the Selenium Hub GCE VMs.*

The VM was created with the command below, it specifies details like the name of the image, the region and it defines the VM user's password based on the contents of a separate text file.

```
C:\tech\google\cloud_apps\marcelotestapp>gcloud compute instances create windows-selgrid
--image windows-server-2008-r2-dc-v20141120 --image-project windows-cloud --zone
us-central1-a --metadata gce-initial-windows-user=marcelorjava --metadata-from-file
gce-initial-windows-password=password.txt
```

Code listing C.8: Windows Server VM creation through the "gcloud" command.

Once the VM was created, the access through the Microsoft Terminal Services Client (MSTSC) became available (The credentials are also defined as metadata parameters of the "gcloud" command) but, before the Selenium test cases can access, it is necessary to create a Firewall Rule in Google Cloud's network to allow the connection from the local machine to perform tests through local builds.

```
C:\tech\google\cloud_apps\marcelotestapp>gcloud compute firewall-rules create selenium --allow
tcp:4444 --source-ranges 86.42.30.226
Created [https://www.googleapis.com/compute/v1/projects/marcelotestapp/global/fi
rewalls/selenium].
NAME     NETWORK SRC_RANGES   RULES    SRC_TAGS TARGET_TAGS
selenium default 86.42.30.226 tcp:4444
```

Code listing C.9: Creation of a Firewall Rule to allow Selenium test cases to connect from local machine.

When the VM access was obtained, the following steps were executed in order to prepare the Selenium Hub:

a. Install JDK 7: This is a dependency to run the Java components associated with the Selenium Hub and the Web Driver.
b. Install Firefox: So it can be invoked by Selenium to test the sample application's interface.
c. Download "selenium-server-standalone-*.jar" package from "www.seleniumhq.org", place it in a specific folder and prepare two batch files to start the required components: “startHub.bat” and “startNode.bat”.

```
java -jar selenium-server-standalone-2.44.0.jar -role hub -timeout=20
-browserTimeout=60
```

Code listing C.10: *Contents of startHub.bat.*

```
java -jar selenium-server-standalone-2.44.0.jar -role node  -hub
http://localhost:4444/grid/register
```

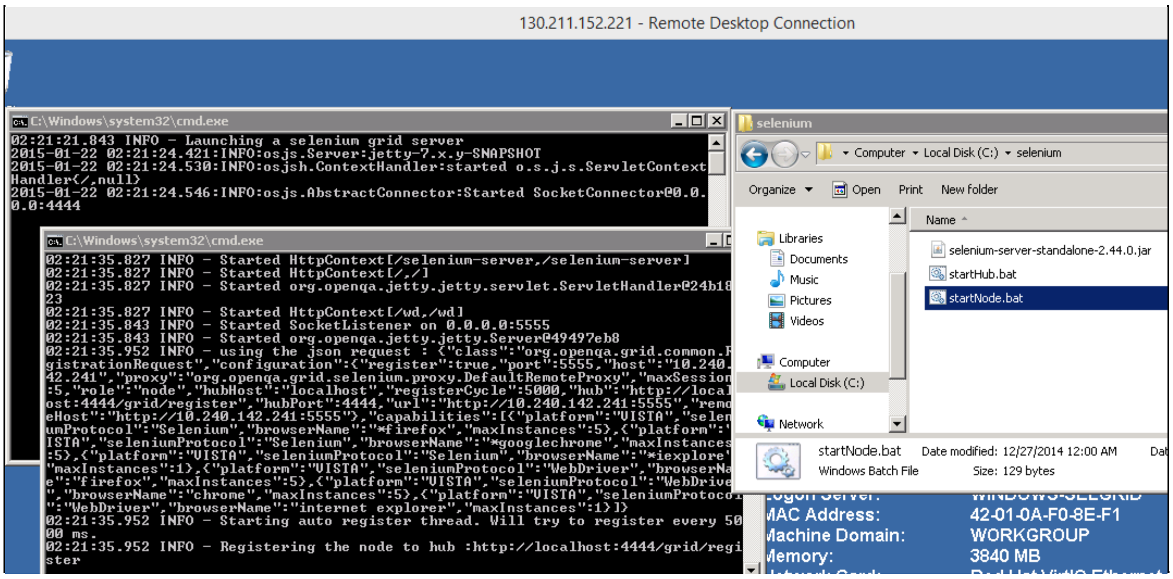
Code listing C.11: *Contents of startNode.bat.*

Once both components are started, there is one final step that is required in order to allow the access between the Jenkins VM and the Selenium VM, once again, the "gcloud" CLI tool is used to create a new Firewall Rule that allows that communication between the Jenkins "GUI_Test" build and the Selenium Hub on port 4444.

```
C:\tech\google\cloud_apps\marcelotestapp\default>gcloud compute firewall-rules c
reate seljenkins --allow tcp:4444 --source-ranges 146.148.94.51
Created [https://www.googleapis.com/compute/v1/projects/marcelotestapp/global/fi
rewalls/seljenkins].
NAME       NETWORK SRC_RANGES    RULES    SRC_TAGS TARGET_TAGS
seljenkins default 146.148.94.51 tcp:4444
```

Code listing C.12: Creating Firewall Rule to allow access between the Jenkins and Selenium VMs.

At the end of this process, the Selenium VM is ready to start accepting requests and execute Firefox interactions against the sample application (Figure C.23).


130.211.152.221 - Remote Desktop Connection
C:\Windows\system32\cmd.exe
selenium
Computer ▸ Local Disk (C:) ▸ selenium
Organize
Open
Print
New folder
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
Network
Name

```
02:21:21.843 INFO - Launching a selenium grid server
2015-01-22 02:21:24.421:INFO:osjs.Server:jetty-7.x.y-SNAPSHOT
2015-01-22 02:21:24.530:INFO:osjsh.ContextHandler:started o.s.j.s.ServletContext
Handler{/,null}
2015-01-22 02:21:24.546:INFO:osjs.AbstractConnector:Started SocketConnector@0.0.
0.0:4444
```

```
02:21:35.827 INFO - Started HttpContext[/selenium-server,/selenium-server]
02:21:35.827 INFO - Started HttpContext[/,/]
02:21:35.827 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@24b18
23
02:21:35.827 INFO - Started HttpContext[/wd,/wd]
02:21:35.843 INFO - Started SocketListener on 0.0.0.0:5555
02:21:35.843 INFO - Started org.openqa.jetty.jetty.Server@49497eb8
02:21:35.952 INFO - using the json request : {"class":"org.openqa.grid.common.R
gistrationRequest","configuration":{"register":true,"port":5555,"host":"10.240.
42.241","proxy":"org.openqa.grid.selenium.proxy.DefaultRemoteProxy","maxSession
:5,"role":"node","hubHost":"localhost","registerCycle":5000,"hub":"http://local
ost:4444/grid/register","hubPort":4444,"url":"http://10.240.142.241:5555","remo
eHost":"http://10.240.142.241:5555"},"capabilities":[{"platform":"VISTA","selen
umProtocol":"Selenium","browserName":"*firefox","maxInstances":5},{"platform":"
ISTA","seleniumProtocol":"Selenium","browserName":"*googlechrome","maxInstances
:5},{"platform":"VISTA","seleniumProtocol":"Selenium","browserName":"*iexplore"
"maxInstances":1},{"platform":"VISTA","seleniumProtocol":"WebDriver","browserNa
e":"firefox","maxInstances":5},{"platform":"VISTA","seleniumProtocol":"WebDrive
","browserName":"chrome","maxInstances":5},{"platform":"VISTA","seleniumProtoco
":"WebDriver","browserName":"internet explorer","maxInstances":1}]}
02:21:35.952 INFO - Starting auto register thread. Will try to register every 50
00 ms.
02:21:35.952 INFO - Registering the node to hub :http://localhost:4444/grid/regi
ster
```

Name
selenium-server-standalone-2.44.0.jar
startHub.bat
startNode.bat

startNode.bat   Date modified: 12/27/2014 12:00 AM
Windows Batch File   Size: 129 bytes

MAC Address:	42-01-0A-F0-8E-F1
Machine Domain:	WORKGROUP
Memory:	3840 MB

Figure C.23. *GCE Virtual Machine with Selenium Hub.*

The Java code is prepared so it can point to the correct Remote WebDriver, the Selenium VM's IP Address is specified within the code so the GUI Testing can happen during the build process.

```
public class SimpleInterfaceTest {
  private RemoteWebDriver driver;

  @Before
  public void firefoxSetUp() throws MalformedURLException {
    DesiredCapabilities capability = DesiredCapabilities.firefox();
    driver = new RemoteWebDriver(new URL("http://130.211.152.221:4444/wd/hub"), capability);
    driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
    ...
```

Code listing C.13: Java code containing the definition of the RemoteWebDriver, which points to the GCE Selenium VM.

Appendix D - Building a Deployment Pipeline with IBM Bluemix

In IBM Bluemix, as opposed to Google Cloud Platform, in order to sign up for the 30-day trial period, the user does not require a credit card for the initial subscription, only an email address that is used for the creation of an IBM ID that allows the user access to the dashboard, the user just needs to follow the instructions on Bluemix website (Figure D.1).

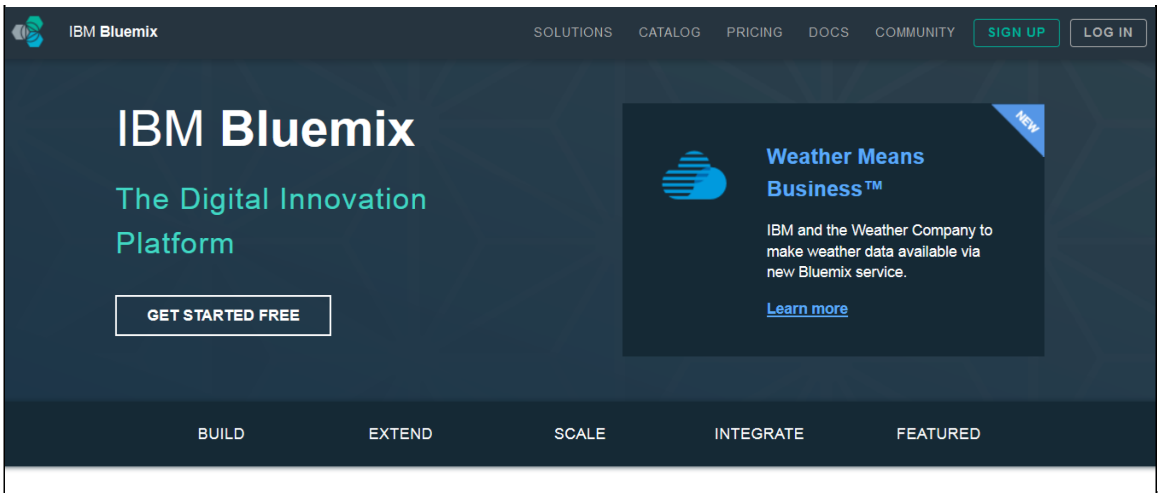


Figure D.1. *IBM Bluemix home page.*

D.1 The Dashboard

The dashboard menu is composed of four options: CF Apps (Cloud Foundry applications), Services, Containers and Virtual Machines (Figure D.2), the applications' section presents the technologies available in the Bluemix PaaS and allows the user to automatically provision one of the available platforms and deploy an application, the Services page shows the full Service Catalogue (discussed in upcoming sections) and the last two options are related to Bluemix IaaS offerings: IBM Containers (also known as Docker Containers), which is in Beta version and Virtual Machines that are already available for some users.

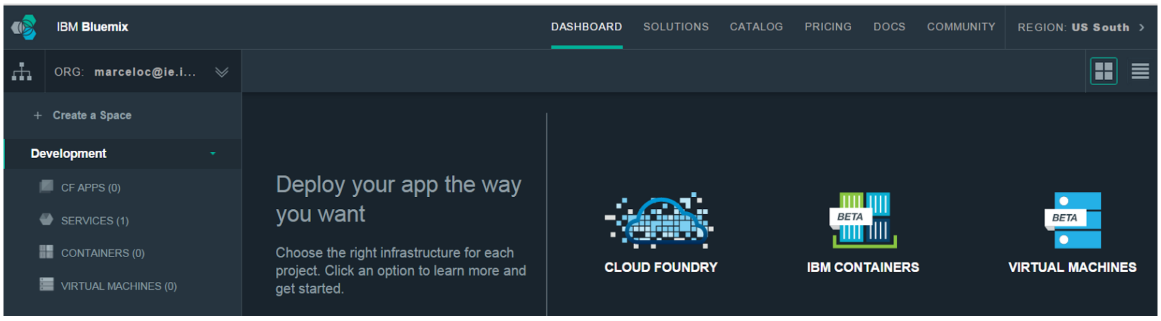


Figure D.2. *The Bluemix Dashboard.*

In the applications pages, there is a report showing the Memory Usage, the Application Health and the binded Services (Figure D.3), this facilitates the monitoring of the overall resources provisioned Bluemix. In order to create an application, the user clicks on the "Create an App" button and then it follows a wizard to set up the type of the application that is created.

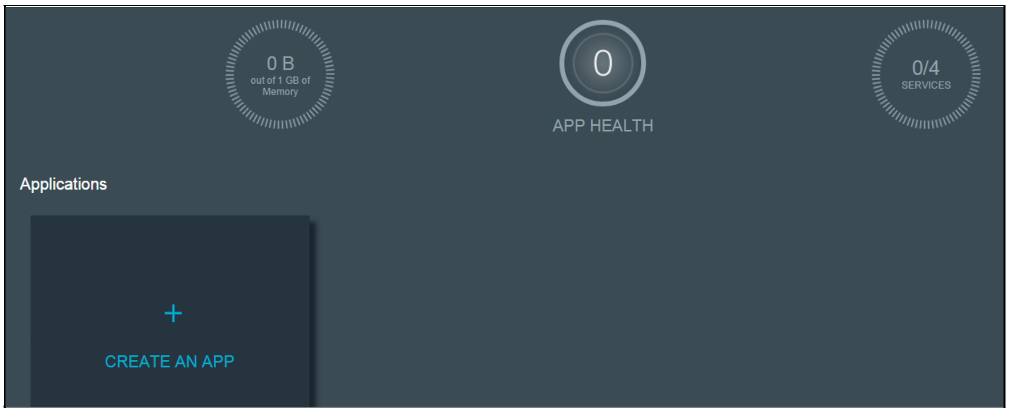


Figure D.3. *Applications page.*

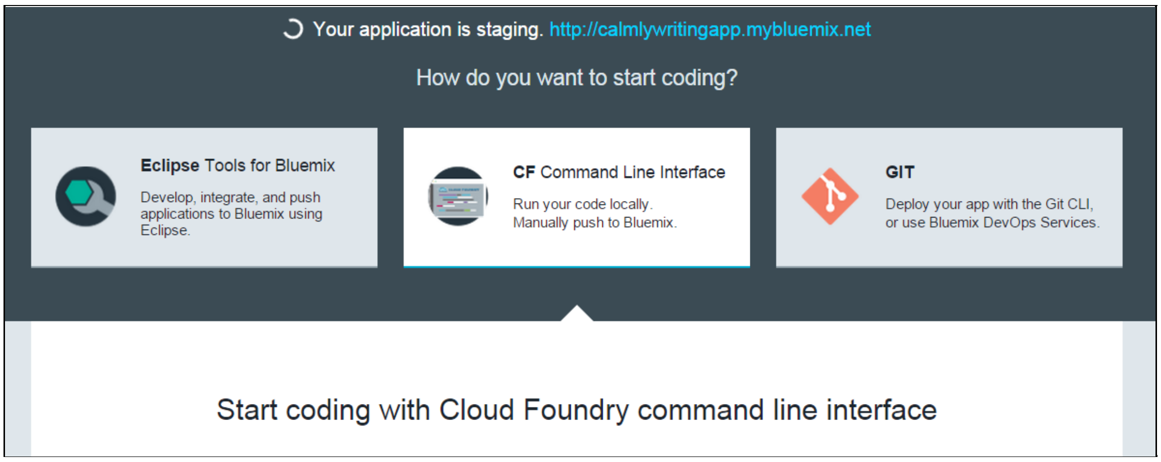
The user must select between a "Web" or a "Mobile" app and then the technology that is used, Bluemix offers options like: Liberty for Java, SDK for Node.js, Go, PHP, Python and Ruby. For the purposes of this research, a Java Web Application was created. The PaaS technology associated with this choice is the Websphere Liberty Profile.

D.2 Websphere Liberty Profile

Websphere Liberty Profile is a lightweight application server that supports a subset of the technologies specified on the Java Enterprise Edition (JEE) 6 stack, it is fast and easily configurable, its main configuration file (server.xml) is very short and simple, this application server is the default option for Java applications that are deployed to Bluemix.

Once the user creates an application (clicking on the “Create an App” button), the application name must be defined, in this exercise, the application was named as “CalmlyWritingApp”, after that, the system starts to provision the underlying infrastructure immediately and presents three different methods to start working with the new web application (Figure D.4), all these methods offer an option to download a “Starter Code”, which is a simple application containing a “Hello World” page in a standard structure of a Java Enterprise Edition application, the three options are:

- **Eclipse tools for Bluemix**: Involves a plugin in the Integrated Development Environment (IDE) that allows the developer to upload a packaged Websphere Liberty Profile server and the application package to Bluemix.
- **Cloud Foundry**: The user installs a Command Line Interface (CLI) tool and executes commands to log into the Cloud Platform and “push” the application to the PaaS server.
- **GIT**: It leverages the Source-Code Management service to create an online version-control repository which can also be used as a web-based IDE, this option also offers a live-sync service where the developer can make changes to the code locally and the changes are immediately reflected in the online repository.


Your application is staging. http://calmlywritingapp.mybluemix.net

How do you want to start coding?

Eclipse Tools for Bluemix
Develop, integrate, and push applications to Bluemix using Eclipse.

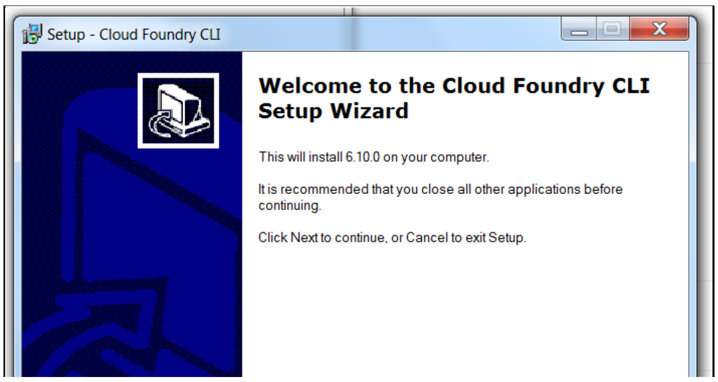
CF Command Line Interface
Run your code locally. Manually push to Bluemix.

GIT
Deploy your app with the Git CLI, or use Bluemix DevOps Services.

Start coding with Cloud Foundry command line interface

Figure D.4. *Application provisioning and initial coding options.*

In order to prepare the local configuration to upload the sample application to Bluemix’s Liberty Profile, the Cloud Foundry CLI was installed (Figure D.5), this command line tool manages applications and services in Bluemix (Similar to Google SDK). The “Starter Code” package was also downloaded.



Welcome to the Cloud Foundry CLI Setup Wizard

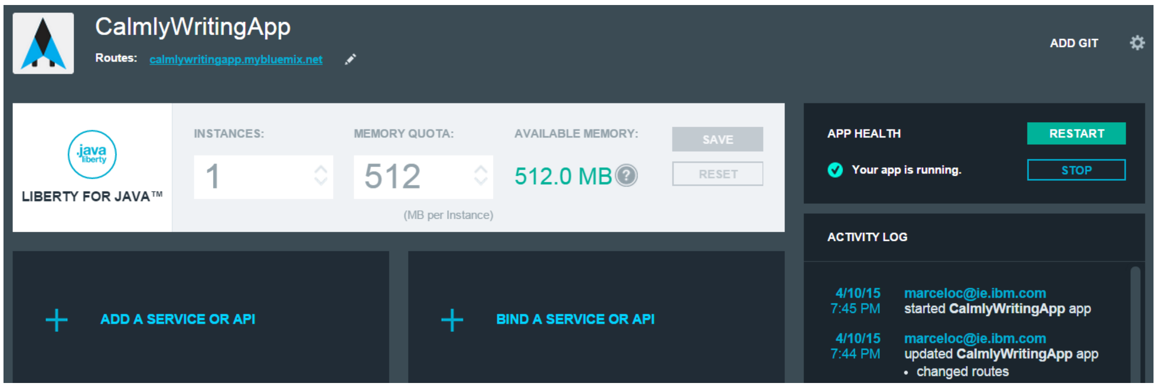
This will install 6.10.0 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Figure D.5. *Installing Cloud Foundry CLI.*

To proceed with the creation of the application leveraging the same SCM offering from Google Cloud, the "GIT" option was selected and, after reading the instructions, the author was directed to the application's overview page (Figure D.6), this page is comprised of a set of status information related to the application, a small icon that represents a Websphere Liberty Profile application, the URL of the Bluemix-hosted application, the number of instances, memory quota, available memory, an "APP HEALTH" report with options to restart the application, buttons to bind Services and "APIs" and an "Activity Log" section with the latest changes applied to the application.



INSTANCES: 1
MEMORY QUOTA: 512 (MB per Instance)
AVAILABLE MEMORY: 512.0 MB

APP HEALTH
Your app is running.

ACTIVITY LOG

4/10/15 7:45 PM marceloc@ie.ibm.com started **CalmlyWritingApp** app

4/10/15 7:44 PM marceloc@ie.ibm.com updated **CalmlyWritingApp** app
- changed routes

Figure D.6. *Application overview page.*

In order to test Websphere Liberty Profile locally to verify if the application is fully compatible with it, the author followed the instructions from the "Download just the Liberty profile runtime" page, which is part of the Liberty Profile documentation (WasDev, 2014), the following steps were applied to install a local instance of the application server:

1. Download the Liberty profile runtime.

2. Run the following command then follow the prompts to install the Liberty profile runtime:

java -jar wlp-runtime-8.5.5.5.jar

3. From the wlp/bin directory in your Liberty profile runtime installation, run the following command to create a new server:

   server create server-name

   The new server is created in wlp/usr/servers/server-name.

4. Run the following command to start the server:

   server start server-name

5. To deploy an application, copy the .war file to the server's dropins directory:

   usr/servers/server-name/dropins

   There is no need to restart the server.

Based on these steps the author was able to deploy the sample application to the local instance of Liberty Profile, however, it failed to present the expected interface(it did not render the background image and other UI elements), some investigation was performed but the root cause was not identified, in order to progress with the experiment it was necessary to replace OpenSymphony's SiteMesh, which was the content presentation framework being used in the sample application, with Apache Tiles, which is an alternative that works with the "Composite View" pattern, as opposed to the "Decorator" pattern used in SiteMesh, after replacing the frameworks and refactoring the code the application interface was successfully rendered in Liberty Profile.

The deployment artifact was produced by the command "mvn clean install -DskipTests" (it was necessary to skip the tests since the database had not been configured at this point), the ".war" package was then copied to the "dropins" directory of the Liberty Profile instance installed locally, no automation was applied for this step.

To deploy the same artifact to the Liberty Profile instance that is running on the cloud, the user needs a "manifest.yml" file, which is Cloud Foundry's deployment descriptor to "push" the Web Application package to the cloud and the Cloud Foundry CLI that must be installed on the local machine. To obtain a "manifest.yml" file, the user can analyze the contents of the "Starter Code" package downloaded previously to edit the parameters and adapt the values to the sample application's configuration.

```
applications:
- disk_quota: 1024M
  host: calmlywritingapp
  name: CalmlyWritingApp
  path: CalmlyWritingApp.war
  domain: mybluemix.net
  instances: 1
  memory: 512M
```

Code listing D.1: Changes applied to manifest.yml to deploy the sample application to Bluemix.

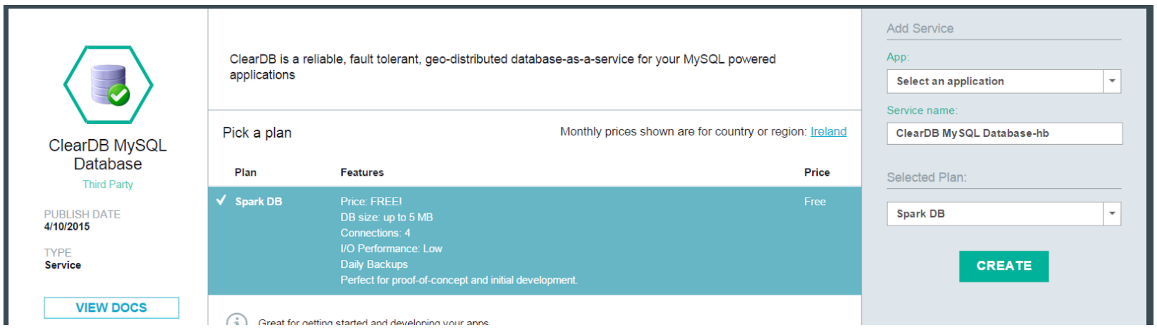
Once the "manifest.yml" file is modified, the following command is used to "push" the application to the Liberty Profile instance in Bluemix: "cf push -p target/CalmlyWritingApp.war", the "-p" parameter refers to the path where the deployment artifact is located, there is an alternative that allows the user to "push" without the "manifest.yml" file, which is the command "cf push CalmlyWritingApp -p CalmlyWritingApp.war", by specifying the name of the application, it eliminates the need to prepare a "manifest.yml" file, however, this deployment descriptor is needed if the configuration of the Liberty Profile Application Server needs to be customized, this was elaborated in the upcoming sections.

D.3 Binding and configuring the Database service

The Bluemix catalogue offers a variety of services, with a free-tier option on each one of them, the services' categories are found on the left-side menu of the "Catalogue" page, below the main categories, the "Support" section gathers other categories like IBM, Third Party, Community, Experimental and Beta. Here are the categories:

- Watson
- Mobile
- DevOps
- Web and Application
- Integration
- Data Management
- Big Data
- Security
- Business Analytics
- Internet of Things

To keep a point of similarity with the Google Cloud solution, that also offers its own custom MySQL database, the author considered two options from the “Data Management” section of Bluemix’s services catalogue: the Experimental MySQL and the ClearDB MySQL Database (Figure D.7), to avoid any tests with “Experimental” services, the latter was selected and binded to the sample application. ClearDB MySQL is a third party service that hosts a fault-tolerant MySQL database instance.



ClearDB MySQL Database
Third Party

PUBLISH DATE
4/10/2015

TYPE
Service

VIEW DOCS

ClearDB is a reliable, fault tolerant, geo-distributed database-as-a-service for your MySQL powered applications

Pick a plan — Monthly prices shown are for country or region: Ireland

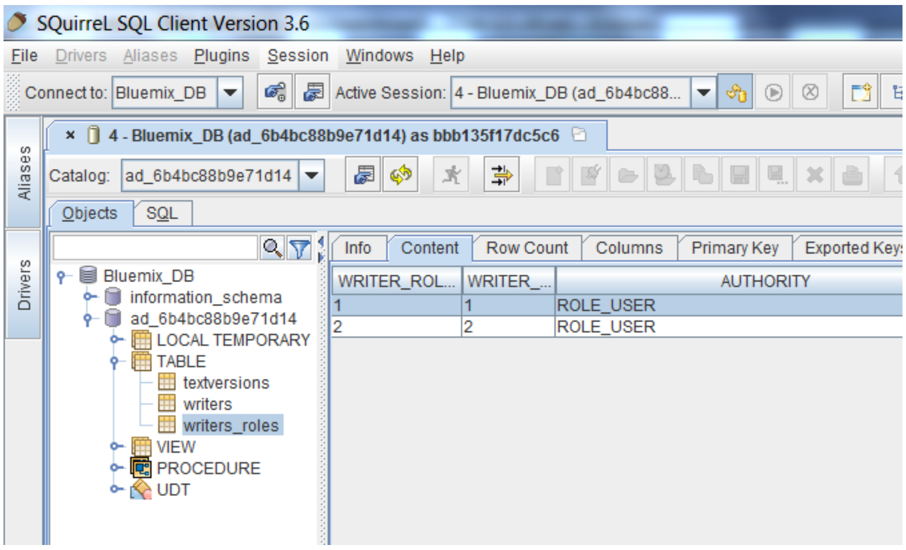
Plan	Features	Price
✓ Spark DB	Price: FREE! DB size: up to 5 MB Connections: 4 I/O Performance: Low Daily Backups Perfect for proof-of-concept and initial development.	Free

Great for getting started and developing your apps

Add Service
App: Select an application
Service name: ClearDB MySQL Database-hb
Selected Plan: Spark DB
CREATE

Figure D.7. Creating the ClearDB MySQL service.

In order to prepare the database to interact with the sample application (CalmlyWriting), a client was used to run the SQL code that creates the tables and insert the required data to create the users, roles and some initial content. Following the same approach applied in the evaluation of Google Cloud Platform, the author used SQuirrel SQL (Figure D.8) to create the connection to the MySQL instance.


SQuirreL SQL Client Version 3.6
File Drivers Aliases Plugins Session Windows Help
Aliases
Drivers
Objects
SQL
Info Content Row Count Columns Primary Key Exported Keys

Connect to: Bluemix_DB
Active Session: 4 - Bluemix_DB (ad_6b4bc88...
4 - Bluemix_DB (ad_6b4bc88b9e71d14) as bbb135f17dc5c6
Catalog: ad_6b4bc88b9e71d14

- Bluemix_DB
  - information_schema
  - ad_6b4bc88b9e71d14
    - LOCAL TEMPORARY
    - TABLE
      - textversions
      - writers
      - writers_roles
    - VIEW
    - PROCEDURE
    - UDT

WRITER_ROL...	WRITER_...	AUTHORITY
1	1	ROLE_USER
2	2	ROLE_USER

Figure D.8. Connecting to the ClearDB MySQL database .

The application’s database connection properties (LibertyProfile.properties) were edited to test the artifact against the PaaS platform, the configuration data includes the name of the database, the hostname, port number, username and password, this information is exposed in the application dashboard (the Application Overview page), each service has a “Show Credentials” link that presents a sliding section containing all the configuration of the service in Javascript Object Notation (JSON) format. Once the service is binded to the application, the interface presents a dialog stating that the it needs to “re-stage” so it can apply the changes, the application is then restarted and all the services become active.

```
################### MySQL Configuration - DEV (Bluemix) ##########################
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://us-cdbr-iron-east-02.cleardb.net:3306/ad_6b4bc88b9e71d14
jdbc.username=bbb135f17dc5c6
jdbc.password=******
jdbc.dialect=org.hibernate.dialect.MySQLDialect
```

Code listing D.2: Database properties file to be loaded by the Spring configuration file.

With the database fully configured with all the application tables, the author moved forward with the adjustments around the application code, more specifically, the Spring configuration file (mvc-servlet.xml), which was modified to consume the “LibertyProfile.properties” file in the absence of the “ENV_SYSTEM” environment variable, so all local builds require the declaration of the variable specifying the Development environment, resulting in a connection to a local instance of MySQL (export ENV_SYSTEM='Development').

```
<context:property-placeholder location="classpath:${ENV_SYSTEM:LibertyProfile}.properties" />
```

Code listing D.3: Spring MVC configuration to load environment-specific JDBC connection data (Default option: LibertyProfile).

With these adjustments in place, the Development environment is ready to accommodate code changes, Unit Tests involving operations on the local MySQL database instance and the deployment to the both instances of Liberty Profile (local and cloud).

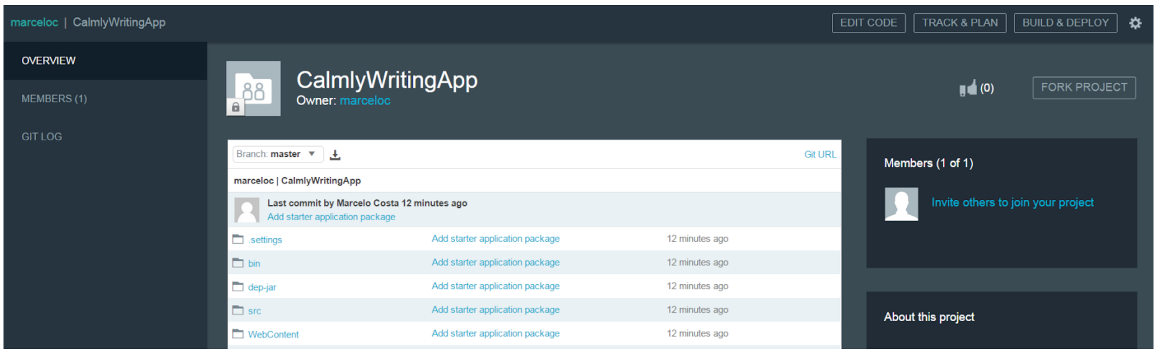
D.4 DevOps Services

This section presents the steps taken to configure the Source Control Management (SCM) system and the Deployment Pipeline within Bluemix, it also presents an overview of the main features associated with the DevOps services:

The options to interact with the code, GIT online components, the Agile planning tool and the Build and Deployment interface.

In the Application Overview page, the “ADD GIT” button presented on the top-right corner initializes the source-code repository for the application, by clicking on it, the user is presented with a dialog that requests the creation of a new account, this time, a “Jazz ID” account, this account is linked with the IBM ID so the user can work with the Source Code Management system, the dialog presents additional instructions stating that the application is deployed every time any code is committed and pushed into the repository (the Push-and-Deploy approach), it also contains a checkbox to enable the addition of the “Starter Code” application to the version-control system, this is offered to assist new users that want to evaluate the code changes in a test application and check the changes right away.

After a few minutes, the GIT repository is ready and this action enables a new “Edit Code” button, also on the top-right corner of the application overview page, this button redirects the user to the “Jazz Hub” page (Figure D.9).



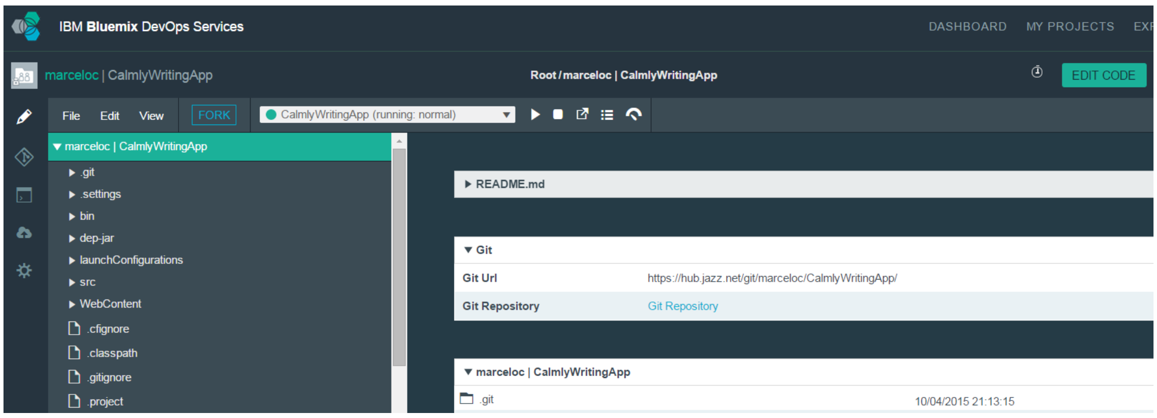
marceloc | CalmlyWritingApp

Last commit by Marcelo Costa 12 minutes ago
Add starter application package

.settings	Add starter application package	12 minutes ago
bin	Add starter application package	12 minutes ago
dep-jar	Add starter application package	12 minutes ago
src	Add starter application package	12 minutes ago
WebContent	Add starter application package	12 minutes ago

Figure D.9. The *Jazz Hub page.*

In an online article (Techworld, 2011), Ellen Messmer defines the Jazz Hub as a cloud-based software development service that “is based on IBM's software development tool Rational Team Concert. Particularly aimed at use by college students, Jazz Hub is an idea from IBM to encourage a new generation of software developers to work more collaboratively and in the cloud.”, this area is where the developer can visualize the application’s files, manage projects and invite other users to join the development. The menu at the top of the page presents three buttons: “Edit Code”, “Track and Plan” and “Build and Deploy”, the first one provides an interface for code editing, along with options to show GIT branches and deployment of the application based on the files currently placed in the workspace (Figure D.10).


IBM Bluemix DevOps Services
DASHBOARD
MY PROJECTS
marceloc | CalmlyWritingApp
Root / marceloc | CalmlyWritingApp
EDIT CODE
File
Edit
View
FORK
CalmlyWritingApp (running: normal)
marceloc | CalmlyWritingApp
.git
.settings
bin
dep-jar
launchConfigurations
src
WebContent
.cfignore
.classpath
.gitignore
.project
README.md
Git
marceloc | CalmlyWritingApp
.git
10/04/2015 21:13:15

Git Url	https://hub.jazz.net/git/marceloc/CalmlyWritingApp/
Git Repository	Git Repository

Figure D.10. *The code editing page.*

The second button, "Track and Plan", presents instructions to enable this tool in the Project Settings page, this is a tool for agile project planning that manages "sprints" and "scrums" by allowing the user to create "Stories", "Tasks" and "Defects", these are known as "Work Items" or "Tickets" and they are used to organize the work that is performed on the application (Figure D.11).

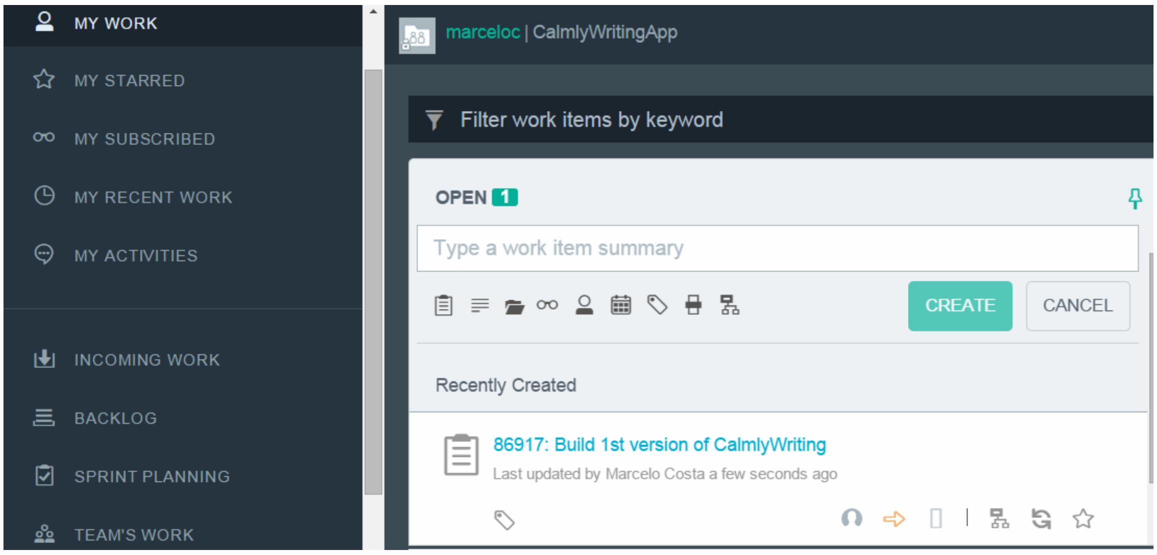
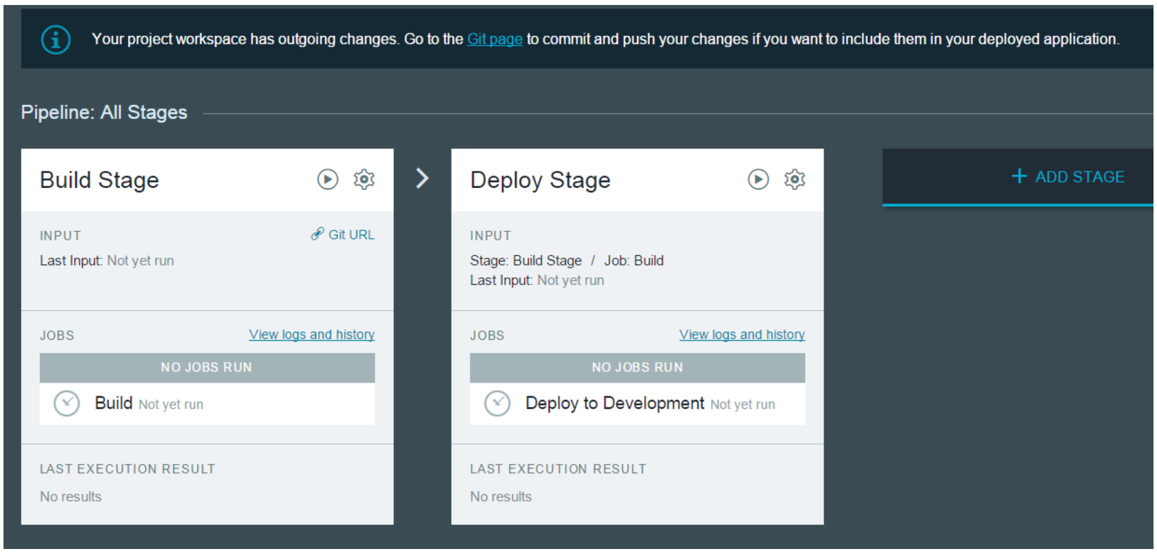


Figure D.11. *The Agile Planning page.*

The "Build and Deploy" button takes the user to the "Build and Deploy Pipeline" page, this area is comprised of "Stages", steps that are defined in the deployment pipeline in order to prepare and deliver the application to the Cloud, this interface illustrates the flow of the build and deployment phases so the developer can track the operations involved in the code delivery (Figure D.12), it also presents the results and exposes any failures that might occur during each of these phases to facilitate the troubleshooting.


Your project workspace has outgoing changes. Go to the Git page to commit and push your changes if you want to include them in your deployed application.

Pipeline: All Stages

Build Stage
INPUT
Git URL
Last Input: Not yet run
JOBS
View logs and history
NO JOBS RUN
Build Not yet run
LAST EXECUTION RESULT
No results

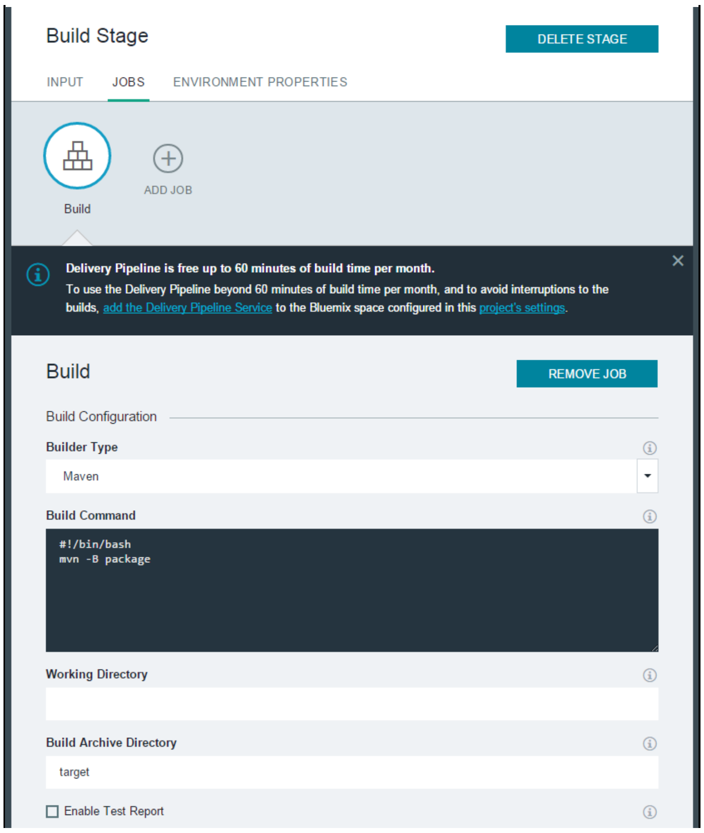
Deploy Stage
INPUT
Stage: Build Stage / Job: Build
Last Input: Not yet run
JOBS
View logs and history
NO JOBS RUN
Deploy to Development Not yet run
LAST EXECUTION RESULT
No results

Figure D.12. The Build and Deploy Pipeline *page.*

Each stage is configurable and can be adjusted for the particular needs of each application (Figure D.13), this is done by clicking on the “configuration” icon at the top-right corner of each “Stage” element, for example: the “Build Stage” presents options to define the “Input” data, which, in this case, is the source-code from the SCM Repository (GIT) and sets the “trigger” for such action (the entry point of the pipeline), the “Jobs” tab allows the user to configure or add operations related to the “build” itself, it provides the following options:

- **Simple:** Package the original files for deployment with no changes.
- **Ant:** Build using Apache Ant.
- **Grunt:** Build using Grunt.
- **Gradle:** Build using Gradle.
- **Maven:** Build using Apache Maven.
- **npm:** Install dependencies with node package manager.
- **Shell Script:** Run a UNIX shell script.
- **IBM Container Service on Bluemix:** Build a Docker Image from a Dockerfile using the IBM Container Service Build Service on Bluemix.

The last tab, “Environment Properties”, allows the user to create custom variables that can be used by the build process or by the next “Stage” of the pipeline.



Delivery Pipeline is free up to 60 minutes of build time per month.
To use the Delivery Pipeline beyond 60 minutes of build time per month, and to avoid interruptions to the builds, add the Delivery Pipeline Service to the Bluemix space configured in this project's settings.

Builder Type: Maven

Build Command:

```
#!/bin/bash
mvn -B package
```

Working Directory:

Build Archive Directory: target

☐ Enable Test Report

Figure D.13. Customizing one of the stages.

Once the first set of changes is committed and pushed by the user, it automatically triggers the deployment pipeline and each "Stage" presents some visual feedback with the progress of the operation (Figure D.14).

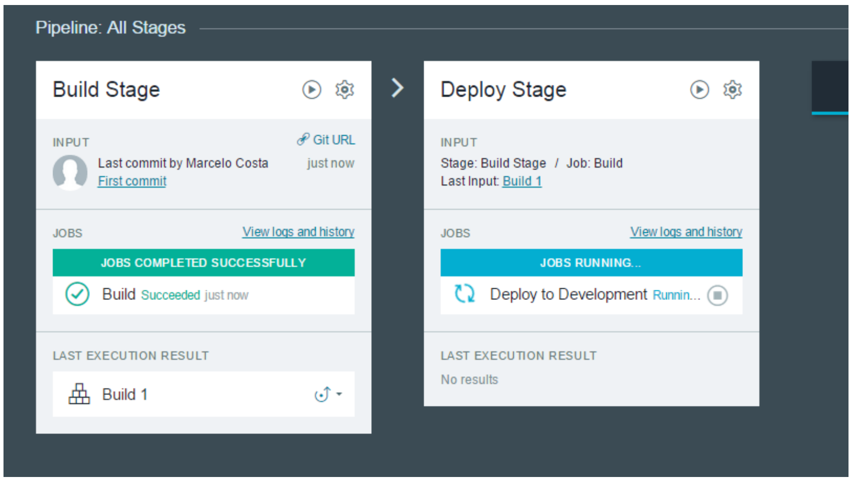
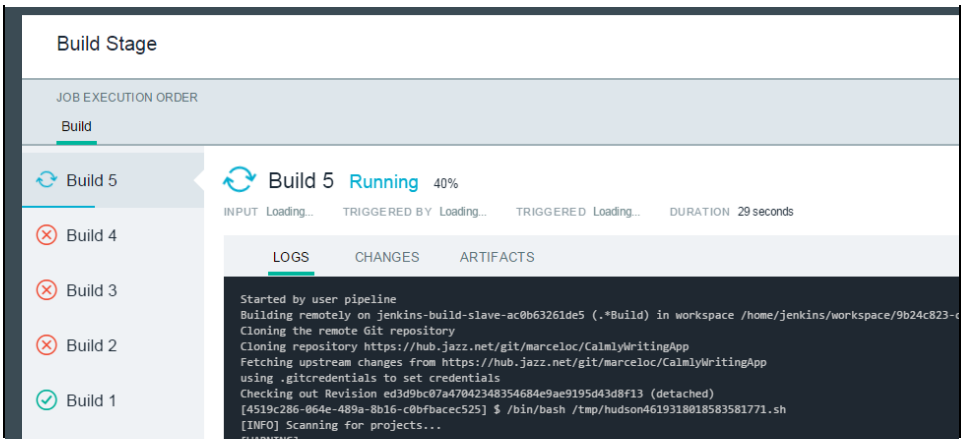


Figure D.14. Pipeline in action.

This simple Deployment Pipeline is automatically created to illustrate how the "Starter Code" application is built and deployed, in case the build presents any failures, the user can click on the job and check the error message in the build log, similar to the Jenkins interface, the log entries show up in real time as the build progresses (Figure D.15).



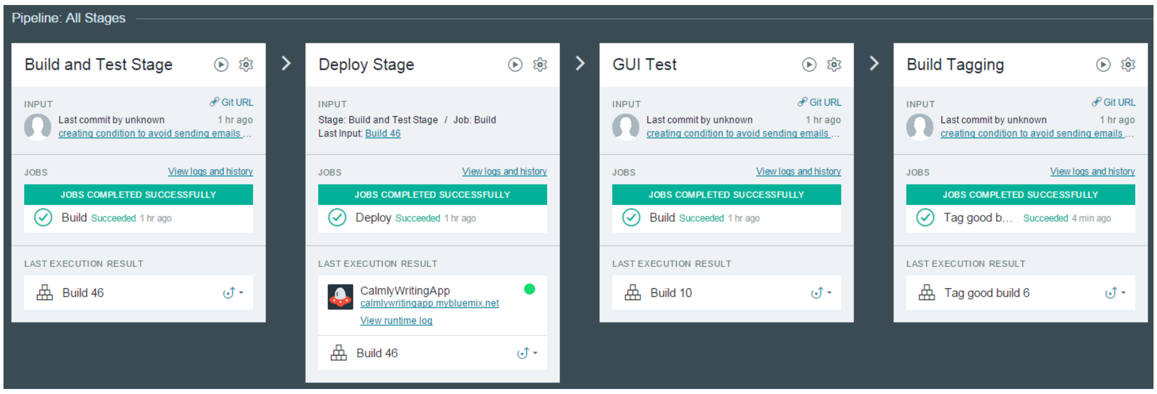
```
Started by user pipeline
Building remotely on jenkins-build-slave-ac0b63261de5 (.*Build) in workspace /home/jenkins/workspace/9b24c823-c
Cloning the remote Git repository
Cloning repository https://hub.jazz.net/git/marceloc/CalmlyWritingApp
Fetching upstream changes from https://hub.jazz.net/git/marceloc/CalmlyWritingApp
using .gitcredentials to set credentials
Checking out Revision ed3d9bc07a47042348354684e9ae9195d43d8f13 (detached)
[4519c286-064e-489a-8b16-c0bfbacec525] $ /bin/bash /tmp/hudson4619318018583581771.sh
[INFO] Scanning for projects...
```

Figure D.15. Checking build results.

This section presented some of the features that aggregate more value to the Software Development Lifecycle: Agile planning tool and the Jazz Hub. Including a basic introduction of the Build and Deployment Pipeline, the next section illustrates the steps to configure the "Build and Deploy" pipeline with the sample application prepared for this research.

D.5 Build and Deploy Pipeline

To configure the "Build and Deploy" Pipeline with the sample application (CalmlyWriting), four stages were defined: Build, Deploy, GUI Test and Build Tagging, these stages were created based on the basic set of steps of a Development environment's Deployment Pipeline (Figure D.16). The author could not identify any conditional configuration that could trigger a fifth stage to create a "Bug Report" in case of any failures.


Pipeline: All Stages

Build and Test Stage
INPUT Git URL
Last commit by unknown 1 hr ago
creating condition to avoid sending emails ...
JOBS View logs and history
JOBS COMPLETED SUCCESSFULLY
Build Succeeded 1 hr ago
LAST EXECUTION RESULT
Build 46

Deploy Stage
INPUT
Stage: Build and Test Stage / Job: Build
Last Input: Build 46
JOBS View logs and history
JOBS COMPLETED SUCCESSFULLY
Deploy Succeeded 1 hr ago
LAST EXECUTION RESULT
CalmlyWritingApp
calmlywritingapp.mybluemix.net
View runtime log
Build 46

GUI Test
INPUT Git URL
Last commit by unknown 1 hr ago
creating condition to avoid sending emails ...
JOBS View logs and history
JOBS COMPLETED SUCCESSFULLY
Build Succeeded 1 hr ago
LAST EXECUTION RESULT
Build 10

Build Tagging
INPUT Git URL
Last commit by unknown 1 hr ago
creating condition to avoid sending emails ...
JOBS View logs and history
JOBS COMPLETED SUCCESSFULLY
Tag good b... Succeeded 4 min ago
LAST EXECUTION RESULT
Tag good build 6

Figure D.16. Pipeline in action.

The first stage (Build) was configured with the “CalmlyWritingApp” code repository as the process input, the only job assigned to it is a Maven command that compiles the code and executes the Unit Tests against a specific package (com.calmlywriting.junit.**).

```
mvn clean test package -Dtest=com.calmlywriting.junit.**/*
```

Code listing D.4: Maven command to run Unit Tests and build the application package.

The author has identified an issue with the deployment where the Liberty Profile instance is unable to initialize the application successfully, the error message is: “NoClassDefFoundError:org.springframework.context.ApplicationContextInitializer”, after some research, the author discovered more details about the default Liberty Profile instance that is hosted in Bluemix, according to Jarek Gawor in his answer from the developer’s forum (dW Answers, 2014), this issue happens while deploying a Spring application to Bluemix because “the Spring auto configuration is triggered when the buildpack finds a spring-core*.jar file in your application”, so he suggests the customization the Cloud Foundry buildpack as a workaround, in order to prepare this buildpack, the author forked the “ibm-websphere-liberty-buildpack” from the Cloud Foundry Github repository and edited the “configs/components.yml” to remove the line containing “- LibertyBuildpack::Framework::SpringAutoReconfiguration”, once this is done, the Deploy stage can be executed with the Cloud Foundry CLI, the command must specify the parameter “-b” to specify the custom buildpack.

```
cf push "${CF_APP}" -b https://github.com/themarcelor/ibm-websphere-liberty-buildpack.git -p
CalmlyWritingApp.war
```

Code listing D.5: Command to deploy the application to Bluemix’s Liberty Profile.

```
containers:
  - "LibertyBuildpack::Container::JavaMain"
  - "LibertyBuildpack::Container::Liberty"
jres:
  - "LibertyBuildpack::Jre::IBMJdk"
  - "LibertyBuildpack::Jre::OpenJdk"
frameworks:
  - "LibertyBuildpack::Framework::Env"
  - "LibertyBuildpack::Framework::JavaOpts"
  - "LibertyBuildpack::Framework::NewRelicAgent"
  - "LibertyBuildpack::Framework::JRebelAgent"
```

Code listing D.6: Custom components.yml file edited for custom buildpack.

One final requirement related to “buildpacks” is related to IBM JVM and IBM Liberty licenses, the license codes can be retrieved from links found in the Cloud Foundry Liberty Profile “readme” page, when the licenses are gathered they need to be included into the application’s “manifest.yml” (As discussed previously in “4.3.2 Websphere Liberty Profile”, the “manifest.yml” is required for such customization).

```
env:
    IBM_JVM_LICENSE: <jvm license code>
    IBM_LIBERTY_LICENSE: <liberty license code>
```

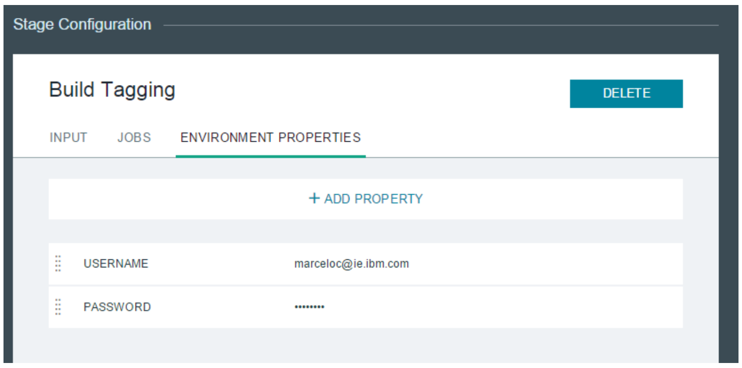
Code listing D.7: License parameters that need to be added to manifest.yml.

The GUI Test stage connects to a Selenium Hub that is hosted in a Bluemix Virtual Machine, the Maven command associated with this stage is similar to the one used in the Build stage, it just specifies the test classes under the Selenium folder.

```
mvn -Dtest=com.calmlywriting.selenium.** test
```

Code listing D.8: Maven command to run Unit Tests and build the application package.

If the GUI Test stage returns a positive result, the build must be tagged, so the next stage was created to address this need, in the “Build Tagging” stage, the configuration was performed with two steps: Configuring the “Username” and “Password” variables, which is done in the Environment Properties tab of the stage configuration (Figure D.17) and the addition of git commands to tag the build, the credentials variables are appended to the command and the number of the build (represented by the $BUILD_NUMBER variable) is appended to the name of the tag.


Stage Configuration

Build Tagging

USERNAME	marceloc@ie.ibm.com
PASSWORD	••••••••

Figure D.17. Setting variables to store Git credentials.

```
#!/bin/bash
git tag -f Good_build_$BUILD_NUMBER
git remote set-url origin
https://$USERNAME:$PASSWORD@hub.jazz.net/git/marceloc/CalmlyWritingApp
git push --tags origin
```

Code listing D.9: Git commands to tag build within the pipeline.

With the Deployment Pipeline in place, the code changes are verified through Unit Tests, packaged into a deployment artifact and deployed to Bluemix's Liberty Profile, then the application's interface is automatically verified and the build is tagged in the version control system in case the changes are successfully validated.

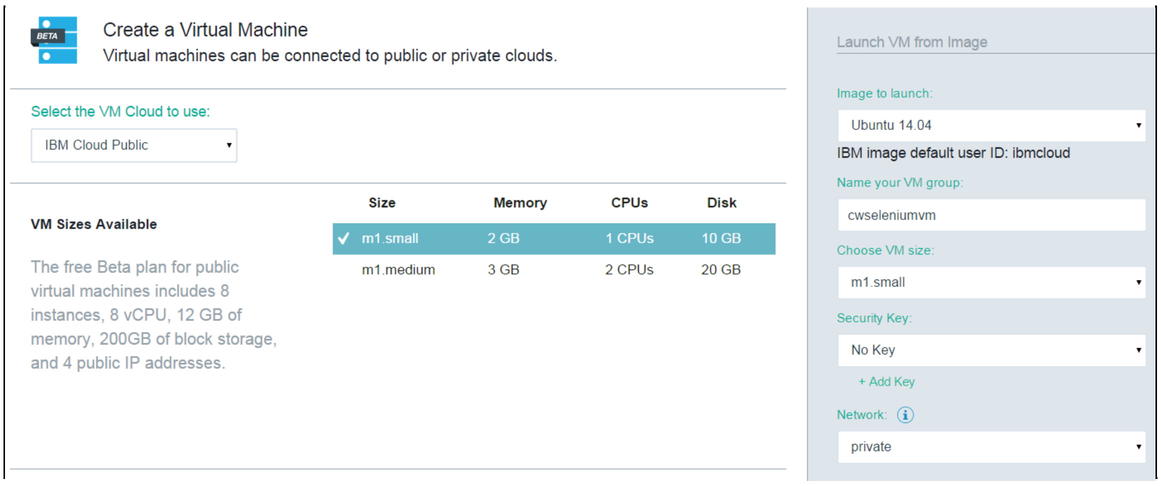
D.6 Creation of VM in SoftLayer Infrastructure

In order to execute the GUI Test through the Deployment Pipeline, the author had to follow an approach similar to the one applied to Google Cloud Platform, which means, to create a Virtual Machine and install the Selenium Hub on it. At the time of this research, the author could not identify any option in Bluemix's service catalogue that offers GUI Tests capabilities, so the VM was created to receive requests from the Maven job running within the "GUI Test" stage in the Build and Deploy Pipeline.

To create a Virtual Machine in Bluemix, the user needs to navigate to the main dashboard, find the "Virtual Machines" section and click on the "Create Virtual Machines" button, currently the Virtual Machines and the "IBM Containers" offerings are in BETA mode so the user might need to wait until the access is enabled. Once the access to Virtual Machines is enabled, a new interface presents a form (Figure D.18) so the user can specify the required data to create the VM, that includes:

- The Image to launch
- Name of the VM

- VM Size
- Security Key
- Network



Create a Virtual Machine

Virtual machines can be connected to public or private clouds.

Select the VM Cloud to use: IBM Cloud Public

VM Sizes Available

The free Beta plan for public virtual machines includes 8 instances, 8 vCPU, 12 GB of memory, 200GB of block storage, and 4 public IP addresses.

Size	Memory	CPUs	Disk
✓ m1.small	2 GB	1 CPUs	10 GB
m1.medium	3 GB	2 CPUs	20 GB

Launch VM from Image

Image to launch: Ubuntu 14.04

IBM image default user ID: ibmcloud

Name your VM group: cwseleniumvm

Choose VM size: m1.small

Security Key: No Key

+ Add Key

Network: private

Figure D.18. Creating VM in Bluemix.

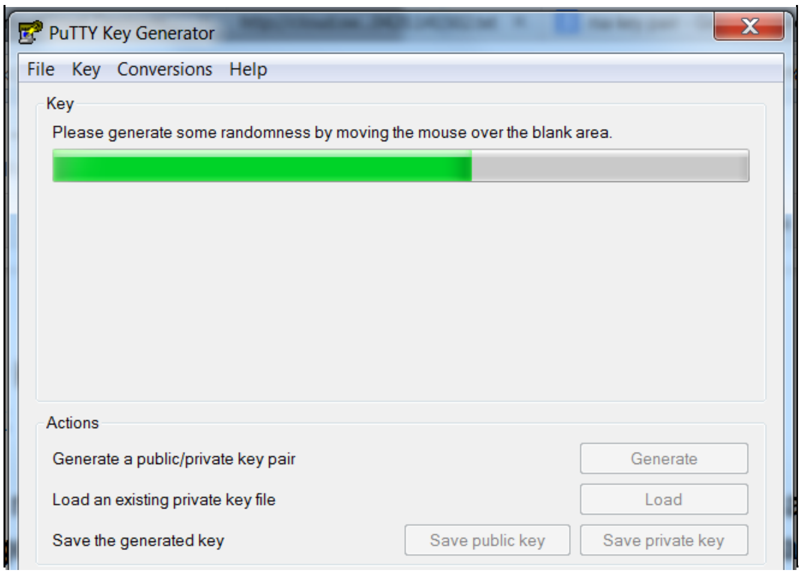
According to the Bluemix documentation, these are the images available for Virtual Machines:

- Red Hat Enterprise Linux 6.5 x86_64
- Red Hat Enterprise Linux 7.0 x86_64
- CentOS 6.5 x86_64
- CentOS 7.0 x86_64
- Ubuntu Server 12.04 LTS x86_64
- Ubuntu Server 14.04 LTS x86_64
- SUSE Linux Enterprise Server 11 SP3 x86_64
- Windows Server 2008 R2 SP1 Standard
- Windows Server 2008 R2 SP1
- Enterprise Windows Server 2008 R2 SP1
- Datacenter Windows Server 2012 R2
- Standard Windows Server 2012 R2 Datacenter

In BETA mode only the Linux images can be selected.

The VM name defined for this research was “cwseleniumvm”. The VM size represents the resources (CPU, RAM, Disk) allocated for this particular image, in BETA mode the images sizes are limited to basic options, 2 or 3 gigabytes of

RAM and 1 or 2 CPUs. There is only one option available for the “Network” field, which is the “private” network. The Security Key is the Public Key from an RSA key pair that is used to SSH into the VM, in this research, the keys were generated by using a tool called “Puttygen” (Figure D.19), this tool creates a pair of unique RSA keys (Public and Private keys). The user can configure the public key prior to the creation of the VM and be able to connect to it through Putty using the private key.



Please generate some randomness by moving the mouse over the blank area.

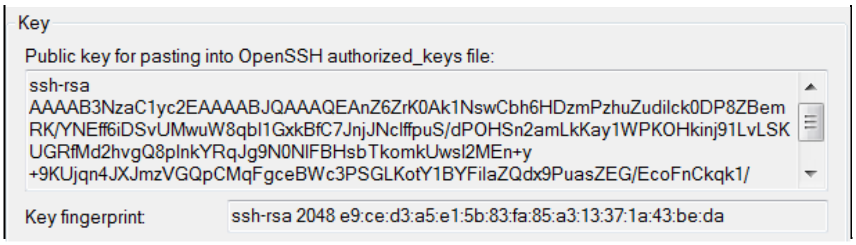
Generate a public/private key pair

Load an existing private key file

Save the generated key

Figure D.19. Creating RSA Key Pair through Puttygen.

Once the keys are created, save both the public key and the private key in the local disk and load the public key with Puttygen once more by using the “Load existing private key file”, when the private key is loaded it presents the public key in a non-editable text area (Figure D.20), the contents of this field must be copied and pasted into the dialog that is presented in the “Create a Virtual Machine” page, the key is associated with the VM so the user can proceed with the creation of the VM by pressing “Next”.



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAnZ6ZrK0Ak1NswCbh6HDzmPzhuZudilck0DP8ZBem
RK/YNEff6iDSvUMwuW8qbl1GxkBfC7JnjJNclffpuS/dPOHSn2amLkKay1WPKOHkinj91LvLSK
UGRfMd2hvgQ8pInkYRqJg9N0NIFBHsbTkomkUwsl2MEn+y
+9KUjqn4JXJmzVGQpCMqFgceBWc3PSGLKotY1BYFilaZQdx9PuasZEG/EcoFnCkqk1/
```

Key fingerprint: ssh-rsa 2048 e9:ce:d3:a5:e1:5b:83:fa:85:a3:13:37:1a:43:be:da

Figure D.20. Copying Public key from Puttygen.

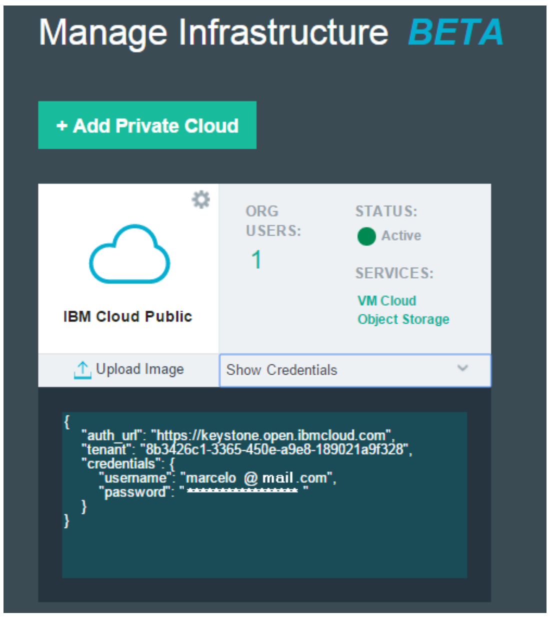
The VM should be provisioned automatically in a few minutes and update the main dashboard with a new Virtual Machine instance, the VM details (the IP

Address) can be found by clicking on the VM. To connect to the VM and start the configuration of the Selenium Hub, the author had to create a new Putty session with the VM IP Address and place the private key under the “Connection > SSH > Auth”, more specifically, in the “private key for authentication” field. While connecting with Putty, it prompts for the username, according to the official documentation, the default user for the VMs is “ibmcloud”, which does not have a password, in order to facilitate the configuration within the VM, the author redefined the password for this user with the command “sudo passwd ibmcloud”. With the connection properly set, the following commands were used to configure the required tools.

```
sudo apt-get update
sudo apt-get install --no-install-recommends ubuntu-desktop
sudo shutdown -r now
sudo apt-get install firefox
sudo apt-get install xfce4-terminal
sudo apt-get install java7-jdk
# download selenium jar using the browser
java -jar selenium-server-standalone-2.45.0.jar -role hub
sudo java -jar selenium-server-standalone-2.45.0.jar -role node -hub
http://localhost:4444/grid/register
```

Code listing D.10: Commands executed to prepare the VM to host the Selenium Hub.

The initial command updates the package management system, then it installs the “ubuntu-desktop” so the VM has a proper interface to facilitate the visualization of the tests that should be triggered by the Selenium Test Cases, after the machine is restarted, the VM desktop interface can be visualized through the “Horizon Dashboard”, this is a special IaaS Management Console built on top of OpenStack (Open-source cloud computing management platform) that allows the user to manage the VM instances created within Bluemix. To access the “Horizon Dashboard”, the user needs to navigate to the “Manage Organizations” page (by clicking on the icon on the top-left corner of the Dashboard page) and click on “Manage Infrastructure BETA”, the credentials can be found in the JSON-formatted data that is presented when the user clicks on the “Show Credentials” link of the instance of the IBM Cloud (Figure D.21), then the user needs to click on the configuration icon to reveal the link to “Launch Horizon Dashboard”.



```
{
    "auth_url": "https://keystone.open.ibmcloud.com",
    "tenant": "8b3426c1-3365-450e-a9e8-189021a9f328",
    "credentials": {
        "username": "marcelo @ mail .com",
        "password": "****************** "
    }
}
```

Figure D.21: Finding credentials to access the Horizon Dashboard.

The "Horizon Dashboard", which is also known as OpenStack console, exposes some configuration of the IaaS elements that are provisioned through Bluemix, the menu is divided in 4 categories: Compute, Network, Object Store and Orchestration (Figure D.22).

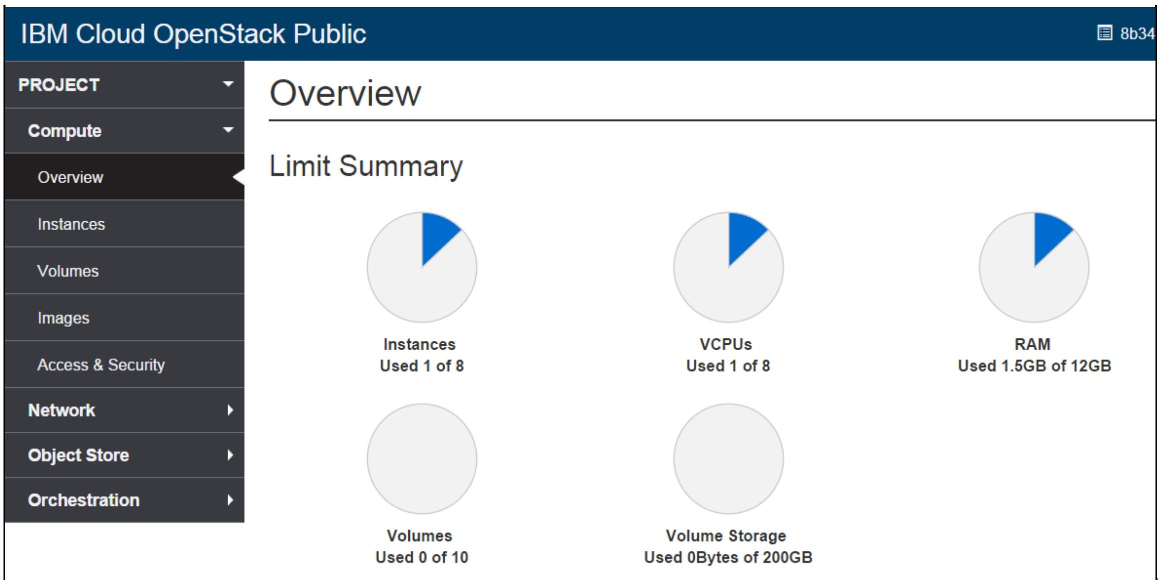
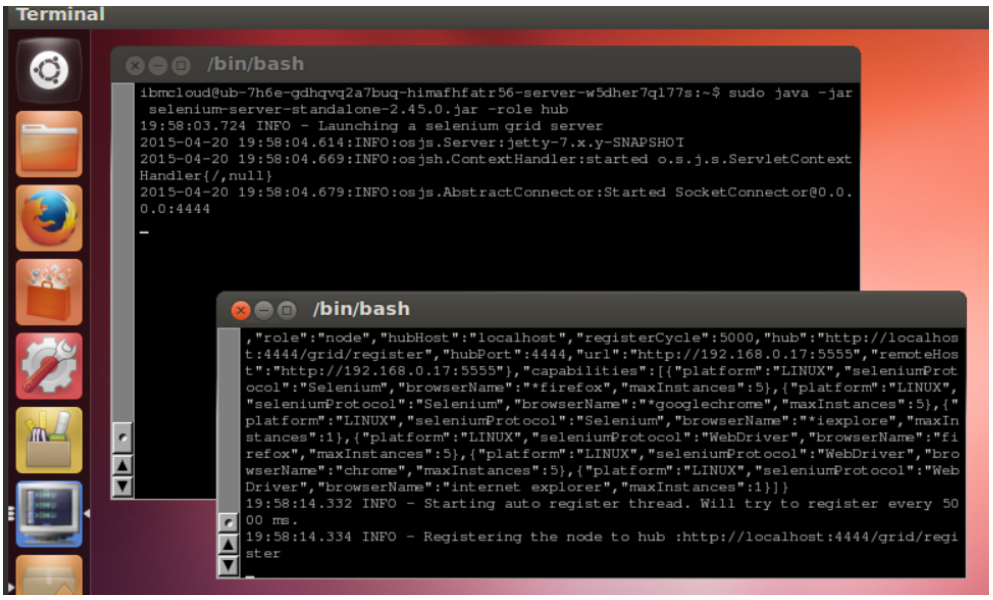


Figure D.22. Bluemix's Horizon Dashboard (OpenStack).

The most relevant section for this research is found in the "Instances" page, by clicking on the VM that was recently provisioned and changing to the "Console" tab, the browser renders a remote desktop interface that facilitates the access to the actual OS Desktop component. The remaining commands from Code Listing D.10 can be used to start the Selenium Hub (Figure D.23).



```
ibmcloud@ub-7h6e-gdhqvq2a7buq-himafhfatr56-server-w5dher7ql77s:~$ sudo java -jar
 selenium-server-standalone-2.45.0.jar -role hub
19:58:03.724 INFO - Launching a selenium grid server
2015-04-20 19:58:04.614:INFO:osjs.Server:jetty-7.x.y-SNAPSHOT
2015-04-20 19:58:04.669:INFO:osjsh.ContextHandler:started o.s.j.s.ServletContext
Handler{/,null}
2015-04-20 19:58:04.679:INFO:osjs.AbstractConnector:Started SocketConnector@0.0.
0.0:4444
_
```

```
,"role":"node","hubHost":"localhost","registerCycle":5000,"hub":"http://localhos
t:4444/grid/register","hubPort":4444,"url":"http://192.168.0.17:5555","remoteHos
t":"http://192.168.0.17:5555"},"capabilities":[{"platform":"LINUX","seleniumProt
ocol":"Selenium","browserName":"*firefox","maxInstances":5},{"platform":"LINUX",
"seleniumProtocol":"Selenium","browserName":"*googlechrome","maxInstances":5},{"
platform":"LINUX","seleniumProtocol":"Selenium","browserName":"*iexplore","maxIn
stances":1},{"platform":"LINUX","seleniumProtocol":"WebDriver","browserName":"fi
refox","maxInstances":5},{"platform":"LINUX","seleniumProtocol":"WebDriver","bro
wserName":"chrome","maxInstances":5},{"platform":"LINUX","seleniumProtocol":"Web
Driver","browserName":"internet explorer","maxInstances":1}]}
19:58:14.332 INFO - Starting auto register thread. Will try to register every 50
00 ms.
19:58:14.334 INFO - Registering the node to hub :http://localhost:4444/grid/regi
ster
```

Figure D.23. Ubuntu VM running Selenium Hub.

Due to the same connectivity limitations that impacted the Unit Tests, it was not possible to run the Selenium Test Cases against the Selenium Hub installed in the VM, the lack of authorized outbound connections in the build machines brings a negative impact to the automation within the Deployment Pipeline.

Appendix E - Combined DOMICloud Summary ScoreCard

Here is a combined DOMICloud summary scorecard comparing both Google Cloud and IBM Bluemix, it is based on the evaluation of the interactions with the DOMICloud capabilities and their respective weighting:

Interaction	Google Score	IBM Score
Service subscription	5.86	8.22
Management Dashboard	7.06	7.36
Client-side Utilities (Application server, Command-Line Interface tool and Plugins)	8.11	5.34
PaaS J2EE Container	6.73	8.19
Services configuration and binding	6.52	7.47
Deployment Pipeline (Build, Tests and Deployment)	6.06	8.22
Virtual Machines	6.71	7.15
Additional features (Monitoring, Logging and Bug Tracking)	6.62	7.97
TOTAL	53.67	58.92

Table E.1. *Combined DOMICloud summary scorecard.*