Developing in the Cloud

Aftab Iqbal Michael Hausenblas
Stefan Decker

DERI Technical Report 2012-10-06
October 2012

DERI Galway
IDA Business Park
Lower Dangan
Galway, Ireland
http://www.deri.ie/

DERI Technical Report
DERI Technical Report 2012-10-06, October 2012

Developing in the Cloud

Aftab Iqbal[1] Michael Hausenblas[1] Stefan Decker[1]

Abstract. With cloud computing as an infrastructure model maturing, more and more efforts around cloud-based software development are emerging. In this model, a cloud-based environment enables developers to edit, test and deploy Web applications through a Web browser. We consider this tool stack to be disruptive in its nature, changing the way software development is being carried out and opening up new opportunities for service providers as well as (teams of) developers. In this report, we review the state of the art of cloud-based software development environments, analyse their shortcomings and provide a roadmap for the next generation of cloud-based development tools.

Keywords: cloud computing, agile software development, collaborative software development, integrated development environment.

[1]DERI, National University of Ireland, Galway, Ireland. firstname.lastname@deri.org

Acknowledgements: This work has been carried out in the Linked Data Research Centre (LiDRC) and has been supported by Science Foundation Ireland under Lion 2 and the European Commission's FP7 Support Action LOD-Around-The-Clock (LATC), project no. 256975, Intelligent Information Management (ICT-2009.4.3).

Contents

1 Introduction

Nowadays, Web applications are not launched in the classical sense anymore—they are continuously released; with every refresh in the browser one has potentially a new version of the application at hand. On the Web, we continuously integrate and constantly iterate, take, for example, the roll-out of new features in Google's Office suite, Twitter's Web front-end or the changes to the Facebook UI. Software developers implement new features, debug errors, push changes to the repositories, update necessary documentation and finally deploy the application. Hence, developers quite often switch between development and deployment phase of the project and interact with many underlying software tools as shown in Fig. 1:

1. **Development Phase** – is associated with carrying out software development tasks at ease. Software developers use a variety of tools to manage their projects. They use source control repositories to manage their source code, bug tracking systems fix issues, and testing frameworks to test the project in different environments, etc.

2. **Runtime Phase** – is associated with deployment of the project in a real-world settings so that it can be used by the end-users of the project. A working copy of the project might be installed on a production server in a production environment or in a test/development environment for testing purposes.

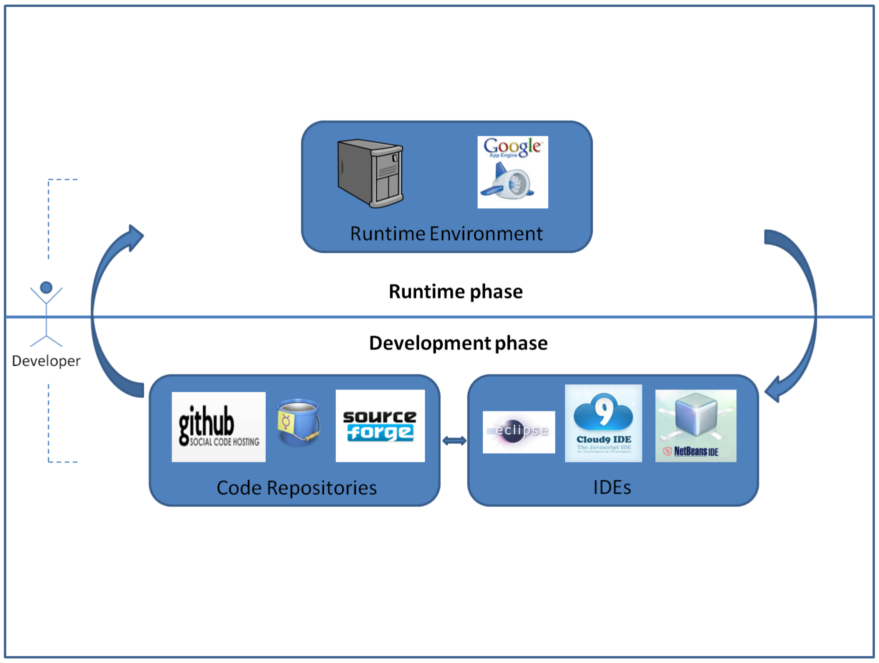


Figure 1: Continuous Integration and Deployment of Software Applications.

Besides hosting of the project's source code and its related artifacts on the Web, the typical developer still carries out the actual development on her own machine using desktop-based environments. Imagine a developer who would like to work on multiple projects using different programming and database environments. She is required to configure the necessary desktop-based software tools such as an IDE before starting to contribute to a particular project. A developer who joins the team is also required to install and configure all the necessary software tools before being a productive member to the project. Last but not least, the transition to a new developer machine requires again installation and configuration of IDEs or development frameworks, which can turn out to be time-consuming. With the emergence of cloud infrastructure services, we have seen tremendous growth in adopting cloud services by leading IT organizations as their development and deployment infrastructure. The services offered by different providers covers almost all of the software tools which are required by the developers to carry out software development tasks.

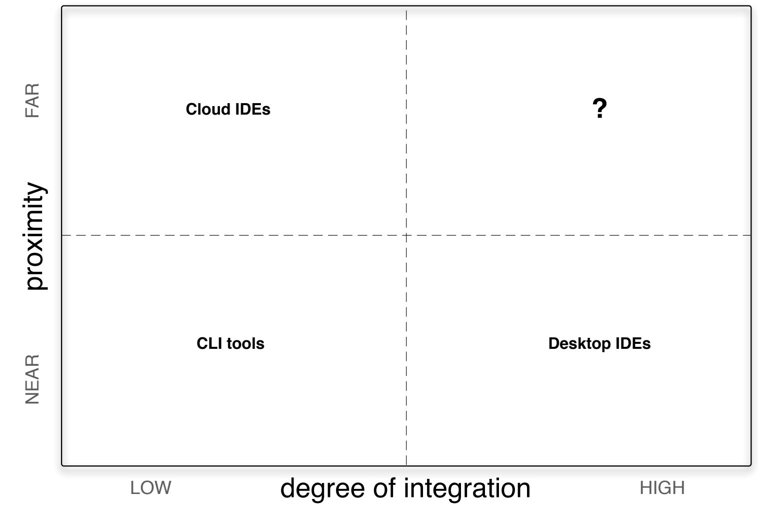


Figure 2: Categorization of software development tools.

In Fig. 2 we provide a categorization of software development tools along two dimensions:

1. **Proximity**—indicates how much administrative control one has over the software tools: “near” means that the software tools are available locally (e.g., command-line tools, vi editor etc.), while “far” means the software tools are accessible via browser-based interface.

2. **Degree of integration**—indicates how well the various software tools are integrated with each other: “low” means that there is little or no integration among software tools, while “high” means that the software tools are well-integrated with each other and support automation of tasks. For example, the Eclipse IDE supports integration with deployment tools such as Apache Tomcat [tom]) and build tools (e.g., Apache Ant [ant]).

In this report, we first review and compare existing cloud-based development tools and IDEs in Section 2. Then, in Section 3, we outline a roadmap of cloud-based software development by identifying challenges and opportunities and propose a set of requirements for the next generation of cloud-based development tools. Finally, we conclude our work in Section 4.

2 Cloud-based Software Development Stack

With the emergence of cloud computing infrastructure, small and large corporations alike have started to move towards hosting their data, software applications, operational communication net-works etc., on the large-scale server farms. Time and cost benefits are the leading motivating factors and measures for any business accessing tools and services via the cloud [Wil].

Popular cloud computing services fit into one of the following categories: *Infrastructure, Platform* or *Software*. Software as a Service (SaaS)[1] are offered by many service providers, including online project management applications, customer relation management, online office applications and many more. Following the successful adoption of cloud services by the enterprises, different cloud service providers started to offer data access, software, hardware and data storage services. Among them, few efforts takes "as a service" to the software development field where we get *Development as a service (DaaS)*. DaaS is a suite of tools that allows to use traditional development practices for creating on-demand applications. DaaS expands the cloud computing development process to encompass external tools such as integrated development environments, source control systems and collaborative tools to facilitate development and deployment[2] with the aim to facilitate and manage software tools and infrastructure for an enterprise, especially with development teams geographically distributed. By leveraging a cloud platform, an enterprise can start using software tools instantly, cost effectively and without managing any development infrastructure [Wil]. According to [AFG$^+$10], an enterprise can utilize cloud development infrastructure to:

1. Scale on demand.

2. Spend more resources on time-to-market by relieving resources from infrastructure management responsibilities.

3. Achieve higher cost efficiencies.

4. Configure development infrastructure in minutes rather than spending whole day on an installed infrastructure.

Recently, service providers have started to offer browser-based development environment, allowing to edit, test, debug, deploy and manage applications using the browser without the need of installing or configuring any tool on a local machine. It enables developers to harness the cloud computing power: just as Platform-as-a-Service[3] (PaaS) enables an enterprise to run applications in the hosted platforms, DaaS provides a new software development stack[4] giving developers the power to write, deploy and manage software applications in the cloud as depicted in Fig. 3:

The cloud-based software development stack typically comprises the following components (exemplary coverage overlaid in Fig. 3):

1. **Editor**—A browser-based code editor, allowing to edit, debug and test source code, for example, Cloud9 IDE [cloa].

2. **Deployment**—Deployment of an end-user application in the cloud through a PaaS provider, such as Heroku [Her].

3. **Data Store**—A cloud-based data storage system (RDBMS or NoSQL) used by an application to store and query application data, for example CouchDB [cou].

[1]`http://en.wikipedia.org/wiki/Software_as_a_service`

[2]`http://wiki.developerforce.com/page/An_Introduction_to_Metadata_and_Development_as_a_Service`

[3]`http://en.wikipedia.org/wiki/Platform_as_a_service`

[4]Kudos to Mike Amundsen for coining the term "cloud-stack" programming in his blog post available via: `http://www.amundsen.com/blog/archives/1116`

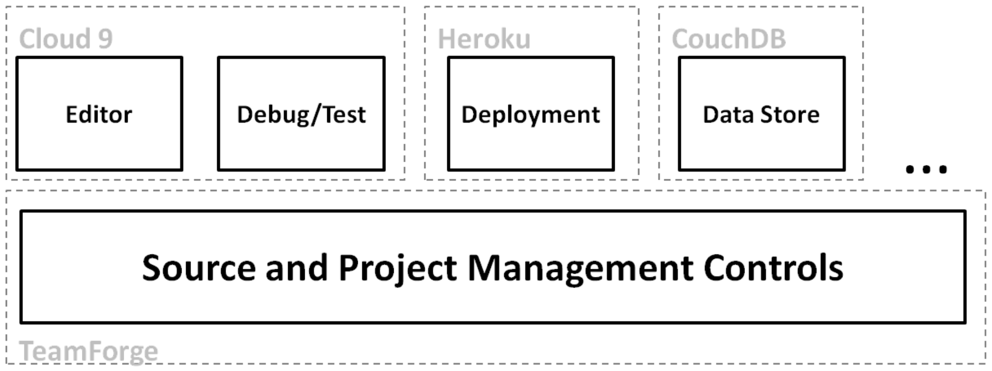


Figure 3: Cloud-based Software Development Stack.

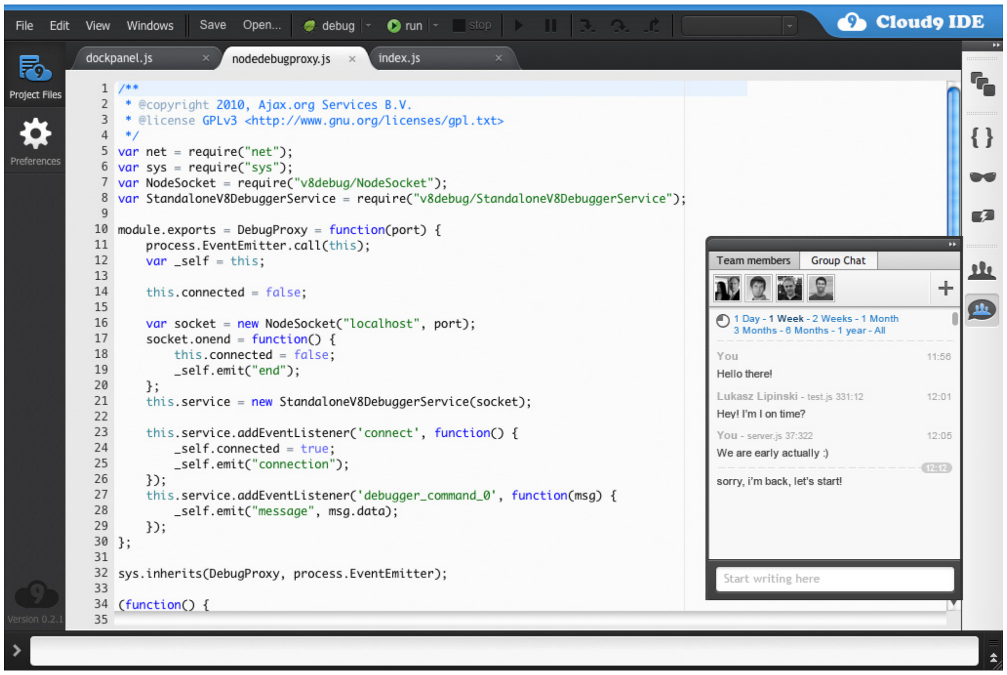
4. **Source and Project Management Controls**—Managing and sharing software repositories through cloud-based collaborative software development architecture, such as TeamForge [tea].

In the following we will review each component of the cloud-based software development stack and outline their features and limitations.

2.1 Browser-based IDE

From the software development perspective, the ability to write, build and deploy a software application using only a browser is a promising approach. This shift of writing code directly in browser-based IDEs is getting more and more attention recently with some trailblazing examples such as Ace [ace], Cloud9, Orion [ori], SourceKit [sou], BrainEngine [bra] and CodeMirror [coda] making impressive advancements[5]. Service providers offer single component or a combination of components as shown in the cloud-based software development stack (Fig. 3). In the beginning, browser-based IDEs were considered nothing more than a text editor with syntax highlighting but these online IDEs are more than that, nowadays: they offer a hosted development environment where developers can develop their Web applications in JavaScript, Java, PHP, Python, Ruby, etc. The platform runs in the browser and lives in the cloud, allowing development teams run, debug and deploy applications from anywhere, anytime. It enables developers to easily start projects behind a single URL, share their code, and collaborate with co-developers all over the world without having to install anything on the client. It also supports syntax highlighting for most popular languages as well as parses and analyze the code in the background and points out any errors in the code if exists. Besides allowing developers to write programs in different programming languages, browser-based IDEs also supports agile and collaborative software development processes. It facilitates collaboration by allowing developers to chat and collaborate with other fellow developers without leaving the browser-based IDE (see Fig. 4). Few browser-based IDEs also supports integration with code forges (e.g., GitHub) to support versioning, maintenance and sharing source code online.

[5]`http://blogs.developerforce.com/developer-relations/2011/03/browser-based-ides.html`



```
/**
 * @copyright 2010, Ajax.org Services B.V.
 * @license GPLv3 <http://www.gnu.org/licenses/gpl.txt>
 */
var net = require("net");
var sys = require("sys");
var NodeSocket = require("v8debug/NodeSocket");
var StandaloneV8DebuggerService = require("v8debug/StandaloneV8DebuggerService");

module.exports = DebugProxy = function(port) {
    process.EventEmitter.call(this);
    var _self = this;

    this.connected = false;

    var socket = new NodeSocket("localhost", port);
    socket.onend = function() {
        this.connected = false;
        _self.emit("end");
    };
    this.service = new StandaloneV8DebuggerService(socket);

    this.service.addEventListener('connect', function() {
        _self.connected = true;
        _self.emit("connection");
    });
    this.service.addEventListener('debugger_command_0', function(msg) {
        _self.emit("message", msg.data);
    });
};

sys.inherits(DebugProxy, process.EventEmitter);

(function() {
```

You 11:56
Hello there!
Lukasz Lipinski - test.js 331:12 12:01
Hey! I'm I on time?
You - server.js 37:322 12:05
We are early actually :)
12:12
sorry, i'm back, let's start!

Figure 4: Cloud9 IDE taken from [cloa].

2.1.1 Comparison of browser-based IDEs

In the following we compare browser-based IDEs as example of cloud-based software development components. The browser-based IDEs are selected without any preference and are listed in no particular order (cf. Table 1). We compare them in terms of programming language support, source control integration, project collaboration features as well as deployment options. Developing code directly in the cloud should enable developers to deploy the application into the cloud, we will see which browser-based IDEs supports application deployment. Quite often, a software project is developed by a team of developers who are geographically distributed; we will see which browser-based IDEs currently supports the social collaboration aspect.

Product	**Language**	**Source Control Integration**	**Project Collaboration**	**Deployment**
CodeRun [codb]	C#/PHP/JavaScript/HTML	No	Yes	Yes
Cloud9	Ruby/PHP/JavaScript/HTML	Yes	Yes	Yes
eXo Cloud	Java/PHP/Ruby/JavaScript/HTML	Yes	Yes	Yes
Bespin [bes]	JavaScript/HTML/CSS	No	Yes	No
Kodingen [kod]	PHP/Perl/Python	Yes	Yes	No
Bungee Connect [bun]	custom written language	Yes	No	Yes
codeanywhere [codc]	PHP/CSS/JavaScript/HTML	No	No	No
ECCO [ecc]	PHP/Java/JavaScript/HTML	No	No	No
WonderFL [won]	Action Script3	No	Yes	No

Table 1: Comparison between browser-based IDE(s).

As shown in Table 1, all browser-based IDEs support Web standard scripting languages although few of them also support object oriented languages such as Java. Version tracking on source code is an important asset in software development but not all browser-based IDEs currently support it. Regarding social collaboration aspect, few browser-based IDEs support project collaboration allowing developer to share his/her workspace with other fellow developers in order to collaborate with each other.

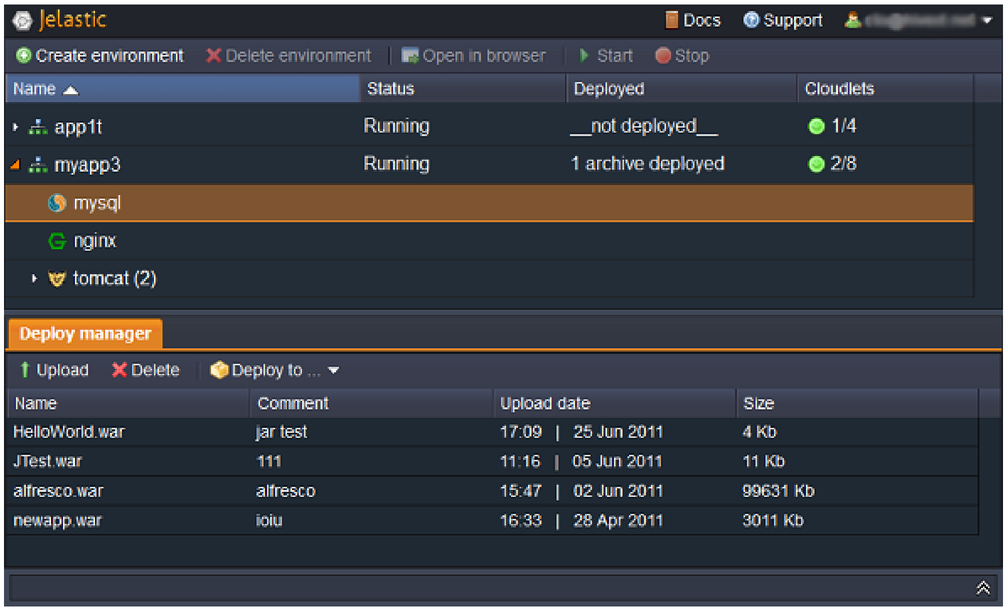
Comparing the current state of features offered by browser-based IDEs with Fig. 1, we can conclude that browser-based IDEs still require improvements in their design and features. Only few browser-based IDEs support integration with code repositories. Moreover, they do not support all programming-languages. For example, IDEs like `Cloud9`, `eXo Cloud` etc., fit best into the continuous development and deployment architecture (cf. Fig. 1) but it does not suit Python developers due to no support for Python programming-language. Contrary to that, `Kodingen` does support Python programming-language but does not support integration with code repositories and deployment of applications in the cloud. Hence, Python developers are unable to get the best from using browser-based IDEs under consideration.

With browser-based IDEs, developing and deploying applications become much more streamlined and enables cloud platforms easily accessible to the developers. These Web-based programming environments can (nearly) replace our desktop IDEs and code editors. They support most of the programming languages available to date as well as allows developers to start connecting to each other into the cloud and build up teams in a way that they couldn't be able to do before, hence giving them more freedom to organize their team development. Apart from team building and programming language support, it offers convenience, pace – the ability to ramp up a project in the order of minutes rather than days or weeks which is a huge benefit. Contrary to the benefits which browser-based IDEs has brought into software development field, there is still room for improvement in these IDEs before it is widely adopted. In this section, we only highlighted some limitations of browser-based IDEs and further compared their features against each other as shown in Table 1.

2.2 Deployment in the Cloud

Cloud-based infrastructure has changed the way applications were built and deployed. Platform-as-a-Service (PaaS) offer services with a primary focus on simplifying and accelerating the entire software application development lifecycle, making it easier and robust than ever to build, run and deploy applications. It provides a browser-based interface to allow application providers to select the software stack, relational databases, application frameworks etc., without installing or configuring anything as shown in Fig. 5. Using PaaS, application deployment is only a matter of uploading the application package (e.g. *.war file) to the selected environment with no extra coding or configurations required. PaaS providers also provides versioning on deployments so that the application provider can deploy an older version of the application, if necessary. In short, building and deploying applications in the cloud brings a non-exhaustive list of benefits as follows:

1. Scalability
2. Faster development and deployment
3. Reliability
4. Managing applications via single console



Jelastic · Docs · Support · [redacted]

Create environment · Delete environment · Open in browser · Start · Stop

Name	Status	Deployed	Cloudlets
app1t	Running	__not deployed__	● 1/4
myapp3	Running	1 archive deployed	● 2/8
mysql			
nginx			
tomcat (2)			

Deploy manager

Upload · Delete · Deploy to ...

Name	Comment	Upload date	Size
HelloWorld.war	jar test	17:09 | 25 Jun 2011	4 Kb
JTest.war	111	11:16 | 05 Jun 2011	11 Kb
alfresco.war	alfresco	15:47 | 02 Jun 2011	99631 Kb
newapp.war	ioiu	16:33 | 28 Apr 2011	3011 Kb

Figure 5: Jelastic Environment Manager taken from [jela].

2.2.1 Comparison of PaaS providers

In the following we take into consideration few PaaS providers as example of cloud-based software development components (Table 2): we compare them in terms of programming-language support, continuous integration tools as well as data storage services because we believe that these tools/frameworks are important assets of software development. Build integration is an important tool for building large and complex software systems which is the reason we selected this particular feature for comparing different PaaS providers. Most applications deal with some sort of storage medium to store and later query data from it. We list down which storage formats are supported by different PaaS providers under consideration. The PaaS providers in Table 2 are selected without any preference and are listed in no particular order.

Product	**Language**	**Build/Continuous Integration**	**Data Storage**
CloudBees [Clob]	Java/Spring/JRuby/Grails/Groovy/Scala	Yes	SQL/NoSQL
OpenShift [Ope]	PHP/Python/Ruby/Perl/Node.js/JBoss	Yes	SQL/NoSQL
Heroku [Her]	Ruby/Node.js/Closure/Java/Python/Scala	Yes	SQL/NoSQL
Jelastic [jelb]	Java/JRuby/Groovy/Scala/ColdFusion	No	SQL/NoSQL
Google App Engine [goo]	Java/Python/JRuby/Scala/Django	No	SQL/Proprietary
dotCloud [dot]	PHP/Python/Java/Perl/Node.js/Ruby	No	SQL/NoSQL
Azure [azu]	.Net/Node.js/Java/PHP	No	SQL/NoSQL
CloudFoundary [Cloc]	Spring/Ruby/Grails/Node.js/Scala	No	SQL/NoSQL/Key-Value

Table 2: Comparison between PaaS providers.

As depicted in Table 2, most PaaS providers offers support for many different programming and scripting languages while few of them only supports JVM-based languages (e.g., CloudBees, Jelastic). Many PaaS providers under consideration is not supporting build integration services currently which we believe is the downside of these PaaS providers. The reason is we are focusing on a desktop-less software development environment where we consider browser-based IDEs as our code editor for writing applications. Hence, we believe that build integration tools and frameworks must be provided by the PaaS providers. Most PaaS providers under consideration supports both SQL and NoSQL storage format with the exception of `Google App Engine` which also provides its own proprietary storage format (i.e., Big Table). By comparing the set of features currently offered by PaaS providers with Fig. 1, we can conclude that the services offered by PaaS providers are pretty much aligned with the needs of enterprises and also make them free from the hassle of managing infrastructure requirements.

2.3 Cloud-based Project Management Tools

With the growing interest in the usage of cloud-based infrastructure services, many service providers have started to offer cloud-based developer-related services specially for enterprises having distributed teams across the globe, hence freeing them from the hassle of deploying and hosting project management tools and frameworks internally. Developer-related services essentially covers the Application Lifecycle Management[6] (ALM) which radically simplifies management of software projects. Using cloud-based infrastructure, provisioning of project management tools can be done in minutes and scale easily without having to invest time and energy on managing the infrastructure.

2.3.1 Comparison of Cloud-based Project Management Tools

In the following we take into account few project management tool providers as example of cloud-based software development components (Table 3): we compare them in terms of support for different project management tools which developers interact with in their day to day software development tasks. The service providers are selected without any preference and are listed in no particular order.

Product	**Version Control**	**Build/Continuous Integration**	**Social Collaboration**	**Project Planning/-Tracking**
TeamForge	Yes	Yes	Yes	Yes
OnDemand [atl]	Yes	Yes	Yes	Yes
JazzHub [jaz]	Yes	No	Yes	Yes
AccuRev [acc]	Yes	Yes	Yes	Yes

Table 3: Comparison between cloud-based project management tools.

As shown in Table 3, service providers under consideration covers all major project management tools with the exception of JazzHub which does not support build integration. Although it is customized specifically for academic research and classroom projects which might be the reason for exclusion of certain features comparing to other service providers. Comparing the features of these cloud-based development tools with Fig. 1, we can conclude that it pretty much covers everything required for continuous integration and development infrastructure. Though, cloud-based project management services are designed primarily to offer services to enterprises (which mostly works on closed-source projects), that could be the reason we don't see integration of such project management services with code repositories such as GitHub.

In the next section, we will discuss challenges concerning cloud-based tools and IDEs along with potential features which they could take into consideration to gain a competitive advantage.

[6]`http://en.wikipedia.org/wiki/Application_lifecycle_management`

3 Towards a Future Generation of Cloud Development

In traditional in-house software development, we deal with different software repositories which surround a particular project. These software repositories are necessary to maintain and execute a project in an structured way. As we have discussed the emerging trends of cloud-based infrastructures to support software development (cf. section 2) and the different browser-based development tools/frameworks and IDEs which are available to date (cf. Table 1, Table 2, Table 3), we foresee the development of software projects in the cloud as depicted in Fig. 6.

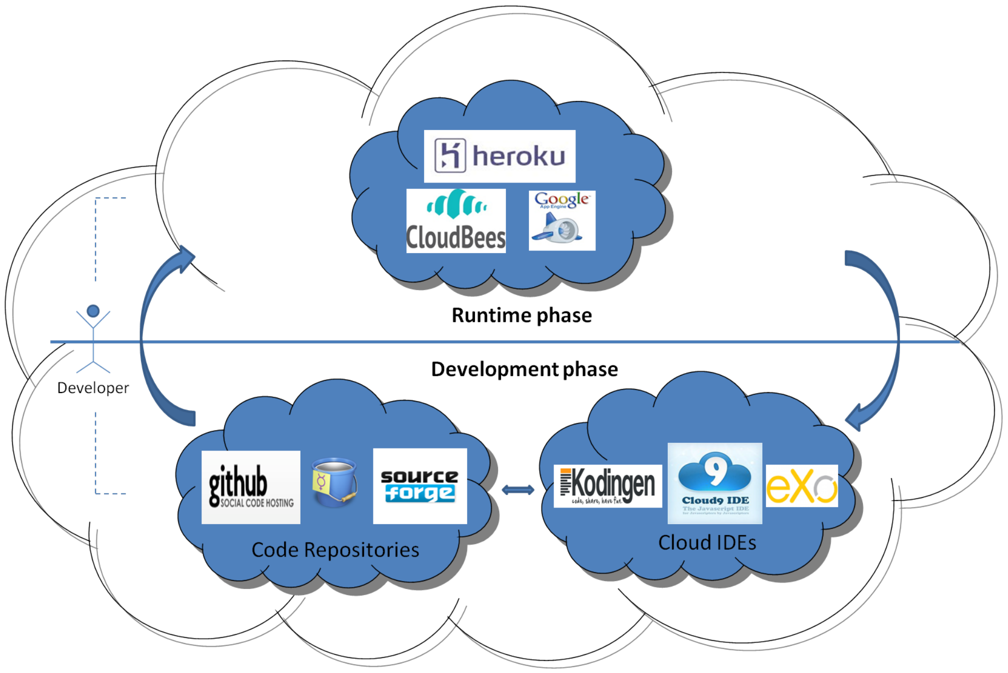


Figure 6: Continuous Integration and Deployment of Software Applications in the Cloud.

3.1 Challenges and Opportunities

These browser-based IDEs do have limitations: most focus on Web application development using only HTML, JavaScript etc., others support Web application development using Java etc. Further, support for accessing other software repositories is lacking, which is quite often necessary to manage large projects. In the following we discuss challenges or areas of improvement which can foster the adaption of browser-based software development:

1. **Dealing with the agile software development life-cycle.** In agile software development, immediate feedback is essential and it is generated through frequent commits, builds, testing and continuous integration. As the cloud-based development infrastructure offers support for these software tools as a single integrated platform (cf. Table 3), it is worth investigating if the online browser-based IDE adds more efficiency, support and transparency than developing the code in a desktop-based environment. Currently, cloud-based software development tools offer integration with desktop-based development environments but such type of integration

is missing with browser-based IDEs. In order to fully support agile software development in the cloud we need to support integration between different cloud-based development tools and infrastructures.

2. **Automation of tasks.** Cloud-based infrastructure makes it straight-forward to export and share data from software tools (i.e., coding, testing, build logs, integration tests etc.). Real time capturing of data from these tools will enable enterprises as well as developers to measure performance, tracking activities of a developer and monitoring the overall progress on the project. In order to achieve this, service providers must provide plugins/tools which can publish and consume data from different cloud-based development tools.

3. **Integration with code forges.** Hundreds of thousands of projects are hosted on different code forges. Only few browser-based IDEs do support integration with code forges (e.g., Cloud9 supports GitHub integration). Integration with code forges will enable developers to easily work on software projects which are already hosted on these code forges, using browser-based IDE. As code forges hosts a variety of project management tools (i.e., bug tracking systems, mailing lists, discussion forums), it will further assist developers to reuse the existing project management tools hosted on these code forges. For example, a bug could be fixed by modifying the source code in a browser-based IDE, compiling, building, testing and later pushing the changes back to the code forge along with changing the status of a bug on the code forge.

4. **Bridging the "off-line gap".** Browser-based IDEs should allow developers to import or export projects directly from their desktop IDEs to the cloud. A developer should be able to code on his local machine and a single push button would allow him to push the changes to the cloud so that other team members can access or review the code changes using browser-based IDE. In the same direction the question has to be answered how (local or remote) backup are supported.

3.2 Requirements

Based on the previous section we have derived a set of requirements we consider pivotal for the next generation of cloud-based software development tools.

1. The integration between PaaS and cloud-based development tools is essential; the interfaces should be standardised, both on the data format level (for example, JSON) as well as on the protocol level (RESTful).

2. Cloud-based development tools (such as build integration, project planning etc.) must support integration with browser-based IDEs.

3. Browser-based IDEs should support interfaces allowing real-time capturing of a developer's activity in a project to promote awareness among co-developers.

4. Browser-based IDEs must provide interfaces to enable integration with code forges to support open source software development in the cloud.

5. Integration between desktop-based and browser-based IDEs should be supported. This can include import and export of projects as well as configuration settings.

4 Conclusion

In this report, we have established a cloud-based development stack and reviewed browser-based IDEs that can bring a significant change in the way software development is carried out. With the existing browser-based IDEs, *small applications* can be easily written and deploy, however we believe that they are not ready yet for prime time to handle large software projects. We have identified a number of challenges in the cloud-based development stack, especially concerning the browser-based IDEs, which might hamper their take-up and proposed a roadmap towards a new generation of cloud-based software development tools.

References

[acc] `https://accurev.com/`. Last accessed on: 10/06/2012.

[ace] `http://ajaxorg.github.com/ace/`. Last accessed on: 10/06/2012.

[AFG+10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53:50–58, 2010.

[ant] `http://ant.apache.org/`. Last accessed on: 10/06/2012.

[atl] `http://www.atlassian.com/software/ondemand/overview`. Last accessed on: 10/06/2012.

[azu] `http://www.windowsazure.com/`. Last accessed on: 10/06/2012.

[bes] `http://mozillalabs.com/blog/2009/02/introducing-bespin/`. Last accessed on: 10/06/2012.

[bra] `http://www.brainengine.net/`. Last accessed on: 10/06/2012.

[bun] `http://www.bungeelabs.com/`. Last accessed on: 10/06/2012.

[cloa] `http://cloud9ide.com/`. Last accessed on: 10/06/2012.

[Clob] `http://www.cloudbees.com/`. Last accessed on: 10/06/2012.

[Cloc] `http://www.cloudfoundry.com/`. Last accessed on: 10/06/2012.

[coda] `http://codemirror.net/`. Last accessed on: 10/06/2012.

[codb] `http://www.coderun.com/studio/`. Last accessed on: 06/21/2012.

[codc] `https://codeanywhere.net/`. Last accessed on: 10/06/2012.

[cou] `http://couchdb.apache.org/`. Last accessed on: 10/06/2012.

[dot] `https://www.dotcloud.com/`. Last accessed on: 10/06/2012.

[ecc] `http://ecco.sourceforge.net/`. Last accessed on: 10/06/2012.

[goo] https://developers.google.com/appengine/. Last accessed on: 10/06/2012.

[Her] http://www.heroku.com/. Last accessed on: 10/06/2012.

[jaz] https://hub.jazz.net/. Last accessed on: 10/06/2012.

[jela] http://jelastic.com/docs/setting-up-environment. Last accessed on: 10/06/2012.

[jelb] http://jelastic.com/. Last accessed on: 10/06/2012.

[kod] https://kodingen.com/. Last accessed on: 10/06/2012.

[Ope] https://openshift.redhat.com/app/. Last accessed on: 10/06/2012.

[ori] http://wiki.eclipse.org/Orion. Last accessed on: 10/06/2012.

[sou] https://chrome.google.com/webstore/detail/iieeldjdihkpoapgipfkeoddjckopgjg. Last accessed on: 10/06/2012.

[tea] http://www.collab.net/products/teamforge. Last accessed on: 10/06/2012.

[tom] http://tomcat.apache.org/. Last accessed on: 10/06/2012.

[Wil] Wang Willie. Reinforcing Agile Software Development in the Cloud. White paper.

[won] http://wonderfl.net/. Last accessed on: 10/06/2012.