An Evaluation of Cloud-based Platforms for Web-Application Development

Jens Albrecht and Kai Wadlinger
Fakultät Informatik, Technische Hochschule Nürnberg, Germany

Keywords: Cloud Computing, Platform-as-a-Service, PaaS, App Engine, Bluemix, Azure, Heroku.

Abstract: A wide variety of service models and options is being offered by cloud solution providers, ranging from simple infrastructure to complex business applications. While the use of cloud-based infrastructure and software services has become common in many enterprises, the Platform-as-a-Service model has yet to take off. Platform providers have invested heavily in their offerings in the last years. The result is a big toolbox consisting of cloud-based components for everything that is needed to implement, deploy and run custom software applications. The developer's expectation is that these components just have to be configured and plugged together to get scalable multi-tiered applications. In our research, we practically evaluated major cloud development platforms on the basis of the requirements of a typical web-based business application.

1 INTRODUCTION

The ever growing importance of information technology in today's fast moving economy comes at the risk of hardly manageable complexity and cost. Thus, in the past decade there has been a huge shift towards commodity hardware, virtualization/containerization and agile development processes. One way to reduce complexity, gain flexibility and maintain cost is the use of cloud-based services. Major cloud-offerings started just about 10 years ago, e.g. AWS S3 and EC2 in 2006, Googles App Engine in 2008, or Microsoft Azure in 2010 (Ragupathi, 2011). According to a 2016 survey by IDG, today about 70% of organizations run at least one application in the cloud (IDG, 2016).

Cloud services can generally be structured into at least three layers (Mell and Grance, 2010; IBM, 2014): *Infrastructure-as-a-Service (IaaS)* comprises servers, network and storage resources, *Platform-as-a-Service (PaaS)* provides components to develop, deploy and run custom applications, and *Software-as-a-Service (SaaS)* allows the use of standard applications supplied by the service provider. While IaaS and SaaS have been broadly used by enterprises in the last years, the adoption of PaaS is still lagging behind. However, the segment is growing now with over 40% per year (Gartner, 2016a).

Several trends are leading towards rapidly growing PaaS utilization in the near future. Most importantly, many software vendors are switching to implement cloud-first SaaS solutions (IDC, 2016), and PaaS provides the development and operations infrastructure for that. An example is Microsoft with a focus on Office 365. Additionally, PaaS will be a major component in IoT projects (Gartner, 2016b). IoT systems by nature are event-driven and need a scalable, highly available backend for message processing. All this can be provided by the PaaS model. Another factor is the open-source movement. Scalable and mature open-source databases and processing frameworks are a necessary prerequisite to keep operating costs low. Open source tools therefore are an essential part even of most commercial cloud platforms. And last but not least, PaaS offerings have grown and matured significantly in the last two or three years.

PaaS can be distinguished into Integration Platform-as-a-Service (iPaaS) and Application Platform as a Service (aPaaS) 0(Paul et al., 2016). iPaaS offerings provide functionalities to connect and integrate cloud and on-premise data sources, applications and services and can be interpreted as cloud-based enterprise application integration (EAI) platforms. aPaaS consist of services and tools for application development and deployment in the cloud.

In this paper we take a closer look at major aPaaS offerings and how they can be used to build custom web and mobile applications. Thus, our focus is on the developer's perspective, and in contrast to theoretical overviews like (Paul et al., 2016) and (Varma and

Albrecht, J. and Wadlinger, K.
An Evaluation of Cloud-based Platforms for Web-Application Development .
DOI: 10.5220/0006427503020309
In *Proceedings of the 12th International Conference on Software Technologies (ICSOFT 2017)*, pages 302-309
ISBN: 978-989-758-262-2

Choi, 2016) we include a practical evaluation. The sample application we used for our analysis is a cloud-based task list which can be shared by different users. It includes many components which are common to most business applications like user authentication, database backend, email notification, and a chat function. The backend of the application was prototypically implemented on four platforms: Google App Engine, IBM Bluemix, Microsoft Azure and Salesforce Heroku. All platforms have been evaluated on a set of common criteria like programmer-friendliness, usability, and cost. As a result, this article gives not only an overview on the current state-of-the art, but can also be helpful to support enterprises in their decision making on cloud-based or on premise platforms.

The paper is structured as follows: After an overview on the examined PaaS offerings in the next section, we describe the requirements of the sample application in section three. Section four contains the evaluation of the four platforms. It is followed by a short summary.

2 PAAS OFFERINGS

The PaaS layer uses IaaS infrastructure and provides middleware components, like database, messaging and security services for the backend of SaaS applications. Thus, it is mainly focusing on a simplification of software development and operations and directly supporting the DevOps model.

For our analysis, we focused on leading PaaS suppliers. Our selection was based on Gartner's 2016 Magic Quadrant (Paul et al., 2016), identifying Salesforce Heroku and Microsoft Azure as market leaders. We also included Google AppEngine (challenger) and IBM Bluemix (visionary) as established providers supplying all components necessary to implement web-applications. We will briefly introduce these four providers. More details on these and other market players can be found in (Paul et al., 2016), (Varma and Choi 2016), and (Li, Zhang and Li, 2017).

2.1 Google App Engine (GAE)

The App Engine is part of Google's cloud platform, which was launched in 2008. Google tries to simplify the development of web applications by focusing on easy development and scalability (Srivastava et al., 2012; Shabani, Kovaci and Dika, 2014). The applications are hosted on the same servers and with using the same technology as Google Apps like Google Docs, Calendar, and Gmail.

GAE supports Java, Node.js, Python, Ruby, Go and PHP. As databases Google provides Cloud SQL, Google's cloud based version of MySQL, and Cloud Datastore, a scalable NoSQL database with a programming and a SQL-like API.

2.2 IBM Bluemix

The youngest platform in this survey is from IBM and was launched in 2014 (Kobylinski et al., 2014). Bluemix is based on the Cloud Foundry framework, an open source multi-language PaaS maintained by Pivotal, a subsidiary of VMWare and EMC (Bernstein, 2014). The platform supplies many starter packs for web, mobile and IoT development.

Bluemix inherits the language support from Cloud Foundry for Java, Node.js, Python, Ruby, Go, PHP, Swift, and .Net. Bluemix supports many different databases including Cloudant as completely managed NoSQL database and DB2 on Cloud for relational storage. Besides that, dozens of third party services for data store, messaging, analytics and much more are offered.

2.3 Microsoft Azure

Microsoft Azure subsumes a number of cloud services ranging from infrastructure via development tools to business software. Microsoft Azure App Service denotes its cloud-based application development platform. Microsoft provides most of its development tools in specific cloud versions, ranging from Visual Studio to SQL Azure. The Microsoft cloud offering also comprises tools for resource management, scheduling, log analytics and more.

Azure App Service supports ASP.NET, Node.js, Java, PHP and Python as languages. Azure SQL is the primary choice for relational data, DocumentDB is Microsoft's version of a scalable document store and supports SQL- and JavaScript-like expressions for queries.

2.4 Salesforce Heroku

Salesforce Heroku is a pure PaaS solution which does not supply IaaS, but instead runs physically on Amazon AWS. Heroku started independently in 2007 as one of the first cloud platforms and was acquired by Salesforce in 2010. Besides Heroku, a general platform for custom web-development, Salesforce clients can also use Force.com to develop applications based on building blocks with a strong connection to the Salesforce environment.

Initially starting with Ruby as its only programming language, it now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. Customer Applications run in virtual containers, called Dynos, which are also the units for scaling. The main database for Heroku is PostgreSQL for relational data. Redis and MongoDB (externally hosted by mLab) can be used as NoSQL databases (Salesforce, 2017).

3 DEVELOPMENT REQUIREMENTS FOR PAAS APPS

The basis for our evaluation of the suitability and maturity of these PaaS offerings is a demo-application for a cloud-based task list, which provides users the capability to manage tasks. Figure 1 shows sample use cases. Tasks are organized into projects, which can be shared among users. In this case a chat function should enable real-time communication between the members of a project. Moreover, each day an email should be sent to every user with his open tasks.

These use cases require components and functionalities which are typical to many web applications (see figure 2):

- authentication with credentials from an existing provider like Google, Facebook, etc.
- user management
- a database to persist and retrieve heterogeneous objects
- a scheduler for task disposition
- an email delivery service
- real-time communication for the chat function

A PaaS provider should supply components and services which can be easily accessed by and integrated to custom applications. Functions like authentication, scheduling etc. should only need to be configured, but not developed. Thus, development effort should decrease, and setup as well as maintenance of these services should not be necessary. For developers it is important that applications can be developed and tested on a local developer machine. Deployment to the cloud should be an easy process.

Very important is the dynamic allocation of resources (resource pooling). All shared services should scale automatically in case of high system load. Manual scaling should be possible for dedicated resources.

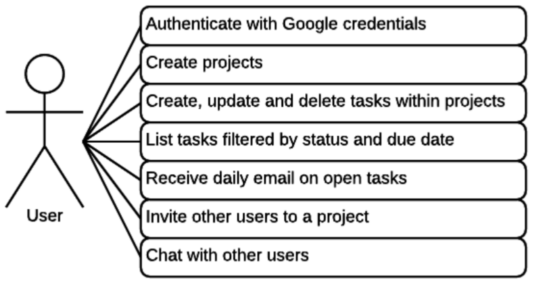
User
Authenticate with Google credentials
Create projects
Create, update and delete tasks within projects
List tasks filtered by status and due date
Receive daily email on open tasks
Invite other users to a project
Chat with other users

Figure 1: Sample use cases for the task-list.

4 EVALUATION

In order to evaluate and to compare the different PaaS offerings, we implemented the backend of our application on all four selected platforms in Java. In each case application was recoded in order to use the application server, database and other services supplied by the platform provider. The frontend is based on JavaScript combined with AngularJs and Bootstrap. The communication was handled via REST (JAX-RS and jQuery). The application was successfully deployed on each platform after implementation. The evaluation criteria have been grouped into the following six categories (see also table 1):

- General: Programming languages, documentation and support
- Runtime: Pricing, scaling and availability
- Development: Plug-ins and deployment
- Authentication: Sources, cost, functionality
- Database: Pricing, scaling, data and query model
- Job Scheduling: Functionality and ease of use

The first three categories characterize the platform as a whole. Authentication, database and job scheduling describe specific functionalities required for the application. The evaluation is based on our own tests and information from the platform documentations (Google, 2017; IBM, 2017; Microsoft, 2017; Salesforce, 2017).

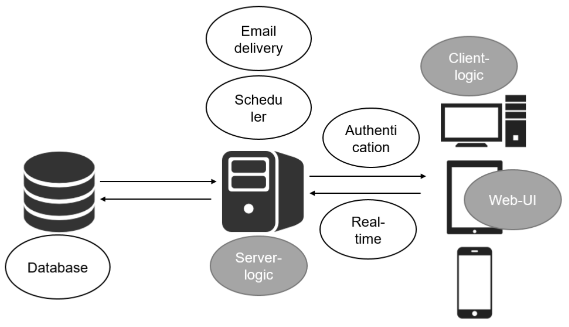

Figure 2: Typical components of web-applications.

4.1 General Criteria

Programming Languages and Build Packs

Java, Node.js, Python and PHP are supported by each provider, and .Net (Core) by all but Google.

Bluemix and Heroku offer the possibility to develop and use own or a third party buildpacks to expand the number of supported programming languages. On Bluemix, a wide variety of Cloud Foundry community buildpacks can be used. Go, PHP, Python, Ruby and Tomcat are examples and directly included in Bluemix. More can be found on GitHub. On Heroku, each officially supported buildpack is open-source and available on GitHub. Additionally, the Heroku marketplace offers more than 3000 third party buildpacks, which are not officially supported.

Google offers a "flexible" and a "standard" environment for application hosting (Google, 2017). The flexible environment supports custom languages and software packages in Docker containers.

Microsoft Azure does not offer buildpacks or similar functionalities.

E-Mail-Support

All providers offer email support as an option which can be purchased. Salesforce Heroku also provides free technical support. IBM states that employees monitor and answer questions on Stack Overflow.

Language of UI and Documentation

The platform UI and the documentation is generally available in English. Besides that, Microsoft and IBM offer localized or partly localized versions of their UIs. Documentation is available mostly in English. Microsoft provides also some local language documentation and tutorials.

4.2 Runtime

The term "runtime" in the context of PaaS generally denotes the infrastructure that the web application is running on, i.e. the virtual machine (hardware) and the application server (software) hosting the web application. Additional services like the database are not a part of it.

Pricing

All providers offer a pay-per-use model based on the usage of virtual servers. Servers generally have a certain performance class with respect to RAM and/or CPU power. The price is depending on the usage time measured in hours (App Engine, Bluemix), minutes (Azure) or seconds (Heroku). In addition, monthly flat-rates are offered with discount.

Automated and Manual Scaling

Generally, there are two options for scaling: to change the number of running instances (horizontal scaling), or to change the instance type, i.e. upgrade or downgrade the hardware (vertical scaling). Moreover, scaling can either be done manually or automatic (auto-scaling) based on system load or SLA parameters.

Automated scaling always affects the number of running server instances and is based on certain load parameters, which depend on the platform.

Google offers instance types with different CPU speeds (up to 4.8 GHz) and RAM configurations (up to 1GB) for automated horizontal scaling in the "standard" environment. Scaling can be based on the response time and the amount of concurrent connections.

Auto-scaling on Bluemix is possible via an add-on. The scaling parameters are the response time, amount of queries/second, JVM Heap size and RAM usage and custom parameters. Sizing and scaling on Bluemix is based on RAM size, which can be extended up to 512 GB. The number cores is adjusted internally. However, that information is not visible on the platform.

Azure offers servers with up to 8 cores and 14GB RAM. Auto-scaling can be based on schedule or workload (CPU, RAM, http queue length, I/O and custom counters).

Heroku provides instances with up to 14GB RAM and scales automatically based on the response time. No information about the number of CPU cores or speed is given. As with IBM CPU scaling is done internally. The costs are prorated to the second. Scaling and pricing at Heroku is based on lightweight containers called "dynos".

The definition of CPUs/cores is generally virtual, physical performance specifications are not supplied (except clock speed at Google). Thus, an accurate comparison in terms of price vs. performance is not possible without actual measurements.

Availability SLAs

Google, IBM and Microsoft guarantee 99.95% availability of the platform resp. application. Neglecting the SLA leads to discounts with several stages. See table 1 for examples. Heroku has not published a general SLA.

Selectable Region

In many countries, especially within the European Union, exist legal obligations that an application and especially the data must be hosted within the political region. Thus, the region for the servers should be selectable, preferably a certain country.

Google implements a regional model where rough geographic regions can be selected, but data can be stored and moved by Google between all data centers in that region.

IBM offers four regions: Germany, Australia, United Kingdom and United States (South). Only the latter provides all services. In Germany, there are just about 16 services out of 110 usable. The single-sign-on service for our web application is only available in the United States (South).

Microsoft Azure is generally available in 34 regions divided into three top regions America, Europe and Asia Pacific. Specific locations can be chosen within the region, e.g. in Europe Frankfurt, Magdeburg, Ireland, London, Netherlands and Cardiff. Not all services are available in all locations.

Heroku allows to choose between two models. If the application is deployed to a common (shared) runtime, only Europe and the U.S. are available regions. If it is deployed to a "private space", i.e. an isolated network, there are five regions selectable (Oregon, Virginia, Sydney, Frankfurt, Tokyo).

4.3 Development

For development, IDE-support is very important to get quickly started. Thus, every platform provides an Eclipse plug-in for easy deployment to and testing in the cloud.

Google's App Engine additionally supports jump-starting the development with a maven archetype. Maven is a tool for amongst other things managing the dependencies and the build of a software project. The archetype includes the App Engine SDK, the Maven plugin and the application server. Starting the app locally can be done with a single Maven command. Another command deploys the project to the cloud. Moreover, Google provides the unique feature, that the Cloud Datastore can be emulated locally. This allows completely local development and is especially useful for unit testing.

Starting with Bluemix isn't as easy as with App Engine. The application server WebSphere must be installed and configured manually. Azure lets the user choose between Jersey and Tomcat application server for Java. Both have to be installed and configured as at Bluemix. There is no special Maven archetype and plugin. An Eclipse plug-in is available for deployment and monitoring of the app.

Heroku doesn't provide a special Maven archetype and plugin. The app can be locally deployed with the tool Heroku Toolbelt, but without emulated database.

4.4 Authentication

Google App Engine and Salesforce Heroku provide an OAuth (Hardt, 2012) solution to authenticate users. Accessing the credentials is similar on both platforms. A prefabricated button starts the authentication process. After successful authentication, a specified JavaScript method is invoked, where the user information and a verifiable token are accessible. The information is sent to the backend via a REST interface.

Google Sign-In is easy to use, but works only with Google accounts. Heroku supports also Auth0 as an add-on (called OAuth0), which allows to choose between more than 30 identity providers that can be activated at the same time, for example Amazon Web Services, Dropbox and Evernote. Furthermore, Auth0 provides its own database for storing users and a multifactor authentication.

IBM Bluemix and Microsoft Azure support Single-Sign-On with one identity provider activated at a time. The complete list of available providers is shown in table 1. There is no login button available. Instead, each unauthenticated access to the web application will be forwarded to the log in page. On Azure, user credentials can be accessed by the HTTP-Header on a HTTP request. Each source has its own named header. On Bluemix the access is done by WebSphere credentials, which can't be accessed easily: Our implementation needed about 300 lines of code.

Comparing OAuth (Google, Heroku) and Single Sign-On (Microsoft, IBM) on our web app, OAuth is more suitable and provides better functionality, because it can be integrated into the web application and used for UI control. Single-Sign-On is more static and suitable for services within companies. Access to user credentials in Bluemix awkward, which is a real disadvantage.

4.5 Database

All providers offer database services for relational and NoSQL databases. We focused our evaluation on NoSQL databases, because they offer greater scalability and flexibility for cloud applications (Burtica e.a., 2012). Besides hosting (shared or dedicated) and pricing, we evaluated the supplied functionality in terms of accessibility, data structures and query interface. The data model for our application is based on the entity types User, Project, Task and Message. If possible, projects and tasks should be stored as nested objects (e.g. JSON documents) to prevent the necessity of joins.

IBM Cloudant, Microsoft DocumentDB and MongoDB fulfill this requirement natively, because all are JSON-based document stores. The data model of Google's Cloud Datastore is based on the concept of (plain) entities. Each entity can have a certain kind (type) and a variable list of properties with one or more values, but nesting is not supported. Therefore, each task belonging to a project is stored as a separate object, and each project holds a list of task ids.

Persisting and retrieving objects can easily be done with each database. The systems differ in the handling of complex queries, because the query languages are quite different. DocumentDB and Datastore (GQL) support besides Java APIs SQL-like query languages, which are easy understandable and writeable. In contrast, queries for Cloudant and MongoDB must be specified as JSON documents, which aren't as readable as SQL.

The functionality in all these NoSQL databases is limited, as joins are generally not supported. DocumentDB allows disjunctive and conjunctive predicates, negation and simple aggregations. Datastore does not support aggregate queries and OR-predicates, but IN-lists. Cloudant, which is based on CouchDB, supports aggregations via so called view aggregation and complex queries via a map operation. MongoDB has a rather powerful query interface supporting arbitrary complex predicates and aggregation.

Cloudant has a nice additional feature: It supports multiversion concurrency control. Object must be stored with the newest revision number. For this reason, the revision number has to be sent to the client after updating an object. Conflicts can be resolved with a Git-like merge-function.

Google's Datastore has the unique feature, that it can be locally emulated with one command. But it is also the only database missing a web UI for data discovery and querying.

Cloudant and mLab MongoDB must be installed and configured manually. DocumentDB can only be used with remote access.

To access a database in the cloud, credentials are needed. On Google, they are bound to the account. Consequently, they don't need to be manually configured. On Bluemix and Heroku the credentials can be accessed during runtime in Java. Only Azure requires manual configuration.

Looking at the hardware, shared clusters are provided on all platforms. Dedicated clusters are additionally offered on Bluemix and Heroku. On Heroku there are only ten different hardware configurations available, each of which having a fixed size and price. Sizes range from 1GB (shared) via 8GB (shared) to 700GB (dedicated). All other systems provide a pay-per-use model. Pricing on Azure is based on database size used per hour and data throughput (request units/second). Bluemix uses storage and data throughput for accounting. The latter is defined by lookups/second, writes/second and queries/second. Google charges based on size and the number of reads, writes and deletes.

4.6 Job Scheduling

Many applications need the possibility to run specific tasks periodically or at a certain point in time. Therefore, we included the functionality to inform a user about is open tasks on a daily basis in our application. All platforms include scheduling mechanisms to invoke some kind of time-based actions.

Google's App Engine provides a simple, cost-free and easy to use scheduler. It has no Web UI and is configured by an xml document. The periodicity can be specified on a very granular level. The scheduler works as a REST client and invokes an http GET request on a specified resource.

Microsoft Azure and IBM Bluemix offer more sophisticated schedulers with a Web UI. The periodicity can be specified accurately and the range of possible actions is high, including for example Database operations, REST calls or interaction with other services on the platform. On IBM Bluemix only the first 50 Jobs of a month are free. Microsoft Azure provides 3600 jobs per month without cost.

Heroku's Scheduler can run script files stored on the platform daily, hourly or every 10 minutes. Thus, the REST call has to be written in one of the supported languages of the runtime.

4.7 Communication

E-mail delivery is handled on all platforms via SendGrid, a cloud-based transactional e-mail delivery service (sendgrid.com). With its simple Web API, SendGrid is easy very to use. SMTP can be used alternatively.

The offerings of the providers differ in the number of cost-free e-mails per month. Google and Heroku provide 12000, Bluemix none and Azure 25000 E-Mails per month without costs.

Real-time communication in contrast to e-mail might seem expendable in many applications. However, we need real-time communication for the chat functionality of our application. Google is the only platform supporting this functionality. Its channel API offers an easy way to implement real-time communication in a web application.

5 SUMMARY

The evaluation clearly shows that scalable web applications can be developed and deployed on all evaluated cloud platforms. The differences only become visible in the details.
The idea behind PaaS is to have a runtime provided, which is runnable out of the box. This is actually done by all platforms. But for the combination of local and offline development isn't. Only Google AppEngine provides a full local emulation out of the box with a database. The Maven support is excellent. On Azure and Bluemix the local runtime environment must be manually installed and configured. There might be differences on the local environment configuration in contrast to the platform that can lead to bugs on the online application. For experts, this is not an obstacle. Furthermore, if a NoSQL Database is used, it's necessary to have the same product for local developing. That can't be done on Azure. Heroku provides a local runtime out of the box without a database. Deploying an application is easy on all platforms.
To sum up, the number of available services is high on each platform, even though the portfolios are different. But it's still a long way off to a one-click local developing sandbox with all services available…

REFERENCES

Bernstein, D., 2014: Cloud Foundry Aims to Become the OpenStack of PaaS. *IEEE Cloud Computing 1(2)*

Burtica, R., e.a. 2012. Practical application and evaluation of no-SQL databases in Cloud Computing. *2012 IEEE International Systems Conference (SysCon 2012)*. Vancouver, BC.

Gartner, 2016a: *Gartner Says Market Leaders Failed to Capitalize on PaaS Growth in 2015*. Gartner Press Release. Available at http://www.gartner.com/newsroom /id/3283217

Gartner, 2016b: *Gartner Says IoT Adoption Is Driving the Use of Platform as a Service*. Gartner Press Release. Available at http://www.gartner.com/newsroom/id/ 3241817

Google, 2017: Google App Engine Documention. Available at https://cloud.google.com/appengine/docs

Hardt, D., 2012: *RFC 6749: The OAuth 2.0 Authorization Framework*. IETF, October 1st, 2012

IBM, 2014. *Cloud Computing Reference Architecture (CCRA) 4.0 - Overview*.

IBM, 2017. IBM Bluemix Documentation. Available at: https://console.ng.bluemix.net/docs

IDC, 2016. *Worldwide Semiannual Public Cloud Services Spending Guide*. IDC Market Guide.

IDG, 2016. *Cloud Computing Survey (Executive Summary)*. Research Report. Available at http://www.idgenterprise.com/re-source/research/ 2016-cloud-computing-executive-summary

Kobylinski, K., Bennett, J., Seto, N., Lo, G., Tucci, F.: Enterprise Application Development in the Cloud with IBM Bluemix, 2014. *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering (pp. 276–279)*. Markham, Ontario, Canada, 2014.

Li, Z., Zhang, Y., Li, Y., 2017: Towards a full-stack devops environment (platform-as-a-service) for cloud-hosted applications. *Tsinghua Science and Technology*, 22(1), pp. 1–9

Microsoft, 2017. Microsoft Azure Documentation. Available at: https://docs.microsoft.com/azure/

Mell, P. M., Grance, T., 2010. *The NIST definition of cloud computing*. NIST Special Publication 800-145.

Paul, V., Yefim, N., Kimihiko, I., Thomas, A., Duni, R., Driver, M., 2016: *Magic Quadrant for Enterprise Application Platform as a Service*. Worldwide. Gartner, 2016.

Raghupathi, K.: *5 Key Events in the history of Cloud Computing*. DZone/Cloud Zone Blog, 2011. Available at https://dzone.com/articles/5-key-events-history-cloud

Salesforce, 2017: Heroku Documentation. Available at: https://developer.salesforce.com/platform/heroku

Shabani, I., Kovaçi A., and Dika, A., 2014. Possibilities Offered by Google App Engine for Developing Distributed Applications Using Datastore. *Sixth International Conference on Computational Intelligence, Communication Systems and Networks*. Tetova, Macedonia.

Srivastava, S., Trehan, V., Yadav, P., Manga, N., Gupta, S., 2012: Google App Engine. *International Journal of Engineering and Innovative Technology (IJEIT)*, (3), pp. 163–165

Varma, M.K., Choi, E., 2016: Comparative Study of Various Platform As A Service Frameworks. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 6(1), 2016.

Table 1: Comparison of evaluated Cloud PaaS Offerings.

	Criteria	Google App Engine	IBM Bluemix	Microsoft Azure	Salesforce Heroku
General	Programming languages	Java, Node.js, Python, Ruby, Go, PHP	Java, Node.js, Python, Ruby, Go, PHP, Swift, ASP.Net Core,	Java, Node.js, Python, PHP, .Net	Java, Node.js, Python, Ruby, Go, PHP, Play, Scala, Clojure, ASP.NET Core
	Buildpacks	No	Yes	No	Yes
	Free techn. E-Mail support	No	No	No	Yes
	Language UI	English	English, partly local	English, local	English
	Language documentation	English	English	English, partly local	English
Runtime	Pricing	Instance-Hours*Instance-Costs+Outgoing Network Traffic	Instance-Hours*RAM	Instance-Minutes*Instance-Costs	Instance-Seconds*Instance-Costs
	Automatic horizontal scaling	response time, coincident connections	response time, queries/s, JVM heap, RAM usage	CPU, RAM, http queue, data in, data out	response time
	Min Hardware (CPU, RAM)	0.6 GHz, 128MB	-, 64MB	1 Core, 1GB	-, 0,5GB
	Max. Hardware (CPU, RAM)	4.8 GHz, 1GB	-, 512GB	8 Cores, 14GB	-, 14GB
	SLA	10% discount, if < 99.95% 25% discount, if < 99% 50% discount, if < 50%	10% discount, if < 99.95% 25% discount, if < 99,5%	10% discount, if < 99.95% 25% discount, if < 99%	-
	Selectable Region / Europe / Specific Country	Yes / Yes / No	Yes / Yes / No	Yes / Yes / Yes	Yes / Yes / Yes
Development	Maven Plugin	Yes	Yes	No	No
	Maven Archetype	Yes	Yes	No	No
	Eclipse Plugin	Yes	Yes	Yes	Yes
	Local run one Click/Command	Yes	No	No	Yes
Authentication	Service Name	Google-Sign-In	Single-Sign-On	Integrated	OAuth0
	Type	OAuth	Single-Sign-On	Single-Sign-On	Both
	Sources	Google	Google, Facebook, Github, SAML Enterprise, Cloud Directory	Twitter, Google, Facebook, Microsoft, Azure Active Directory	Google, Facebook, Microsoft, Soundcloud +31 other (OAuth)
	Several Source at same time	Yes	No	No	Yes
	Cost-free	Yes	No	No	Yes
	Credentials Access	Client JavaScript	Backend	HTTP-Header	Client JavaScript
	Usable in local mode	Yes	No	No	Yes
	Pros & Cons	+ Easy to use	+ No Code on Client - User-Data access complicated	+ No code on client, easy credentials access - Different source => different header name	+ Large source selection, User management, Register process - Elaborate Client implementation
Database	Service Name	Datastore + Objectify	Cloudant NoSQL DB	DocumentDB	mLab MongoDB + Morphia
	Hosting	shared cluster	shared + dedicated cluster	shared cluster	shared + dedicated cluster
	Pricing	database size + reads + writes + deletes	database size + API calls (shared) or hardware (dedic.)	Size + data throughput	size
	Nested Objects	No	Yes	Yes	Yes
	Query Interface	easy to use, but limited functionality (post-processing in application)	complex query syntax	SQL-like and simple to use, great functionality	query syntax not as easy as SQL, great functionality
	Web-UI existing	no	yes	yes	yes
	Usable in local mode	yes (automatically)	yes (after installation)	no (only remote access)	yes (after installation)
	Credentials retrievable	not necessary (implicit)	yes	no	yes
Job Scheduler	Service Name	Cron Jobs	Workload Scheduler	Scheduler	Scheduler
	Configurable by	XML	Web-UI	Web-IU	Web-IU
	Periodicity	accurate	very accurate	very accurate	not accurate
	Cost-Free	Yes	55 jobs/month	3600 jobs/month only hourly	Yes
	REST call implemented	Yes	Yes	Yes	No
	Pros & Cons	+ easy to use	- only 50 jobs/month cost-free		- Job definition with Rake
Communication	Number of cost-free emails per month in SendGrid	12.000	0	25.000	12.000
	Real-time-communication API	Yes	No	No	No