Project Proposal
James Bowers - 14046475
Supervisor: Kashinath Basu
Module: U08096
Word count: 1866/(1500+25%)

Platform for deployment of user applications across a multi-cloud environment

Introduction and rationale

The popularity of the cloud has grown exponentially, but users still fear consequences regarding price and feature deprecation of being locked into a particular cloud provider. For instance, the BBC has created their own internal tool to abstractify the deployment process, ultimately to be completely cloud agnostic. However even they are still not in a position to switch cloud vendor [appendix 2, Q2]

The concern that applications running on a single cloud is a single point of failure which "fails all the time"[1,2], has prompted large organisations to move to an interconnected or cloud agnostic model to have cloud redundancy. Additionally this lets companies choose from a wider range of products[3].

Aim

To develop an intercloud deployment platform, to allow applications to use products offered by both Amazon Web Service (AWS) and the Google Cloud Platform (GCP) cloud providers.

Objectives

- To design a system architecture capable of storing the following information, and to display it using a web based UI and CLI; personal user information, application code, configuration files, and cloud access keys.
- To deploy the same "Hello World" application to GCP and AWS.
- To trigger a deployment to AWS and GCP through a CLI tool.
- To authorise and authenticate an application deployed to AWS to access one of GCP's cloud services (Google DataStore) and vice-versa.
- To deploy a single "Hello World" application to at least 2 separate regions on both AWS and GCP with latency based routing.
- A tear-down feature to remove all cloud services used by an application.

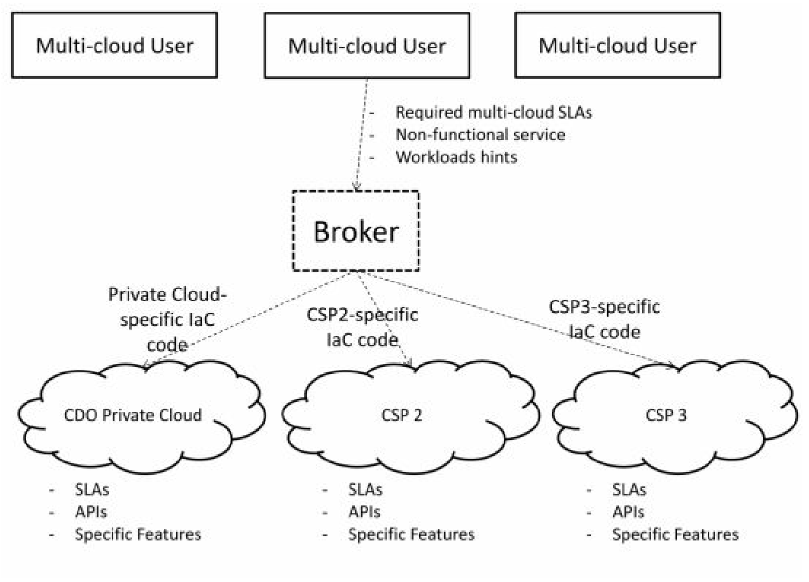
Deliverables

- A server side system to create and update multiple sets of infrastructure across clouds, and to trigger new deployments.
- A web based UI where users can manage their project and trigger deployments.
- A command line tool which offers the same functionality as the web based UI system.
- Interim and final report.

Research and Literature Review

“85 percent of enterprises have a multi-cloud strategy”[4] including private clouds, this has triggered a huge volume of research and publications in this domain in academia. One paper presents the idea of a “cloud builder” which creates infrastructure across cloud providers through a broker (Fig 1 from [5]):

The paper is directly relevant, specifying requirements of the broker and cites the supercloud project



[6]. It was authored by 5 people thereby drawing from wide experience and being written in 2017 the comments on the current situation of the cloud are up-to-date. However, as it is such a recent paper it has not yet been cited in other work and therefore not gained academic endorsement.

A paper written in 2016 [7], and cited by 3 papers so giving it academic credence, claims there is a “huge appeal” for a multi-cloud broker, but goes on to say “the market is in fact hopelessly slim”, and claims this is down to companies not wishing to ‘trust’ a broker with such a crucial task.

Cloud platform providers are competing to offer the most advanced products, which has lead to a huge expansion in the range of cloud products [8,9], including tools to aid continuous integration which narrows the division between developers and operations. Current continuous integration technologies like CircleCi [10] automates the deployment process by reacting to triggers set off by the VCS, then running tests before deploying the application.

Cloud vendor security products such as AWS’s Security Groups[11] are not compatible across cloud vendors, which creates a trust boundary between clouds that the application code has to deal with. The supercloud project[6] is an attempt to standardise multiple aspects of the cloud providers, including security.

Legally, the web UI will have to conform to the W3 accessibility standards [23], and the storage of personal information to the Data Protection Act 1998[24]. This project puts operations staff jobs at risk in the same way continuous delivery does.

Methodology and Resources

I will use a mix of Agile and Kanban style of working [appendix 1] to manage tasks, and track them on a Trello[12] board.

Design process

1. Data flow diagram of the server side architecture.
2. The NoSQL schema for storing the user's personal and application specific data.
3. Design the json config file(s) which will describe the infrastructure for the project.
4. I will use Swagger to document the RESTful web endpoints of the backend system, which can also be used for documentation.
5. Create wireframe design for each page of the web UI.
6. Feature files for the web UI which will be used for BDD.

Resources, final system specification and justification

I have chosen to use JavaScript extensively as it will let me share the library code-base easily across a nodeJS server environment, a web-browser, and as a command line tool. To achieve this I will use the transpiler Babel[13], and bundle the JS using webpack for use in the browser [14]. JavaScript is also perfect for manipulating JSON, which is the format I have chosen for the config file as it can be sent in the body of an HTTP request and unlike YAML, it does not need to be converted into another format before use. This is also why I have decided to use Google's NoSql DataStore which stores data in a JSON format.

I will use React[15] for the web UI as this is a widely used JavaScript library which combines OO and FP to manage stateless and stateful components. My familiarity with it is also advantageous over alternative frontend libraries like Angular, and as it makes use of the FP pattern, it can become more effective when used with lodash[16], another FP library. The main idea of FP is that a function's output does not depend on external state, thus making functions easier to test.

Event based server side systems are becoming common practice in industry through cloud products such as Google CloudFunctions. Events will be triggered via API calls and a user uploading their project to Google's CloudStorage. I will use Google's CloudFunctions as it supports NodeJs and separates server side processes[17].

Swagger is used in industry to specify the "RESTful contract for your API, detailing all of its resources and operations in a human and machine readable format" and will be used to specify the contract of my system's backend services. [18]

Cucumber is a BDD library which takes cases in feature files and converts them into tests which test the user's interaction with a system. When used with Selenium, tests can send input into a browser, imitating a user's interactions whilst making assertions by looking at the content on the page[19]. The BDD tests will be derived from the feature files that also make up the requirements for the system.

A centralised server system will abstractify the web UI and CLI from the cloud vendors, which means only one codebase will need to be updated to keep up with the constantly changing products cloud vendors offer.

Users will be authenticated and authorised access to a project via a certificate they install upon creating an account. Despite certificate authentication not being perfect [20], I believe 2FA would slow the development process considerably if required for every request.

Evaluation and Testing

I will be using BDD, TDD, integration and usability testing where appropriate. I will take a BDD approach when developing the front end web UI, and for the libraries used on the frontend and backend services, I will use a combination of TDD and integration tests and 5 users for usability testing as research has shown this to be the optimum number. [21]

Risk Analysis

5					
4	F			E	
3		A			
2				B	C
1				D	
Y-axis: Probability X-axis: Impact	1	2	3	4	5

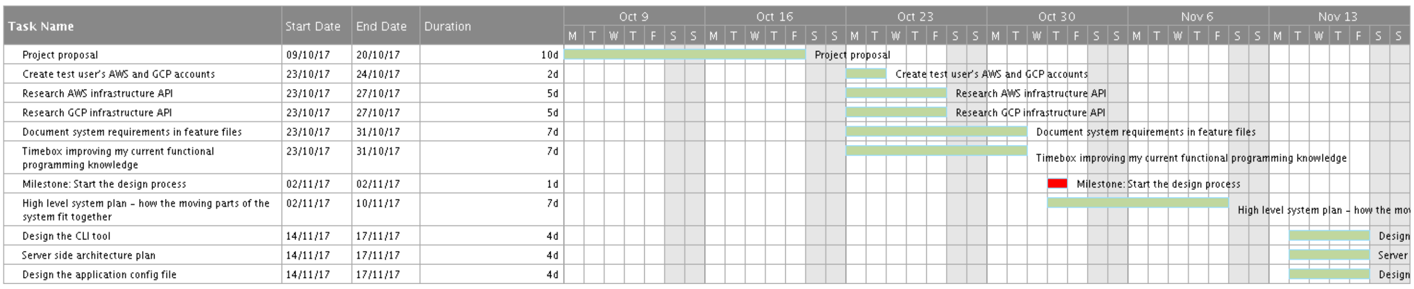
ID	Risk	Score	Prevention or mitigation
A	Cloud vendor price increase	6	Risk acceptance, this decision is made when users decide to use the cloud.
B	Cloud vendor’s product fails	8	Risk reduction, use the same product in multiple regions to failover too if a product fails.
C	Cloud budget exceeded	10	To mitigate this risk, as soon as possible, I will build a tear down feature to remove all cloud resources from a project. Therefore resources will not be paid for longer than they are needed.
D	Cloud products deprecated	4	Risk acceptance, there is nothing I can do about this apart from research alternatives that could

			be moved to if a provider was to deprecate a product.
E	Running out of time due to incorrect task length predictions	16	Risk reduction, I will prioritise the critical path tasks, and add a contingency time.
F	New requirements added during development	4	Risk reduction, I will work in a Kanban development style, which allows for new tasks to be added during Agile sprints.[22]

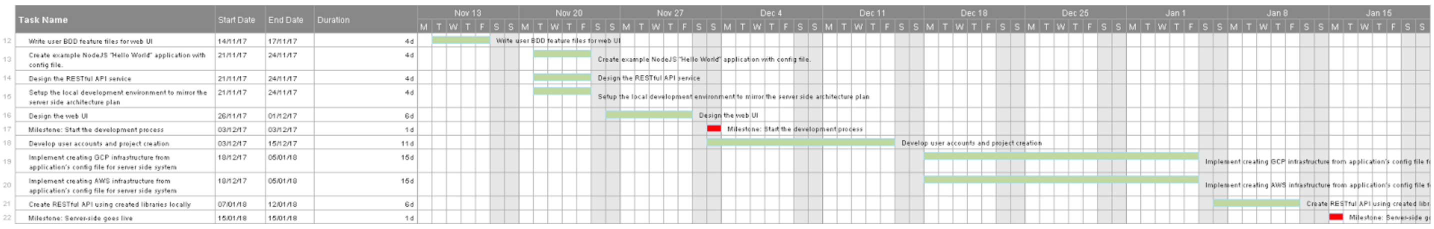
Project Plan

Task Name	**Start Date**	**End Date**	**Duration**
Project proposal	09/10/17	20/10/17	10d
Create test user’s AWS and GCP accounts	23/10/17	24/10/17	2d
Research AWS infrastructure API	23/10/17	27/10/17	5d
Research GCP infrastructure API	23/10/17	27/10/17	5d
Document system requirements in feature files	23/10/17	31/10/17	7d
Timebox improving my current functional programming knowledge	23/10/17	31/10/17	7d
Milestone: Start the design process			
High level system plan - how the moving parts of the system fit together	02/11/17	10/11/17	7d
Design the CLI tool	14/11/17	17/11/17	4d
Server side architecture plan	14/11/17	17/11/17	4d
Design the application config file	14/11/17	17/11/17	4d
Write user BDD feature files for web UI	14/11/17	17/11/17	4d
Create example NodeJS “Hello World” application with config file.	21/11/17	24/11/17	4d
Design the RESTful API service	21/11/17	24/11/17	4d
Setup the local development environment to mirror the server side architecture plan	21/11/17	24/11/17	4d

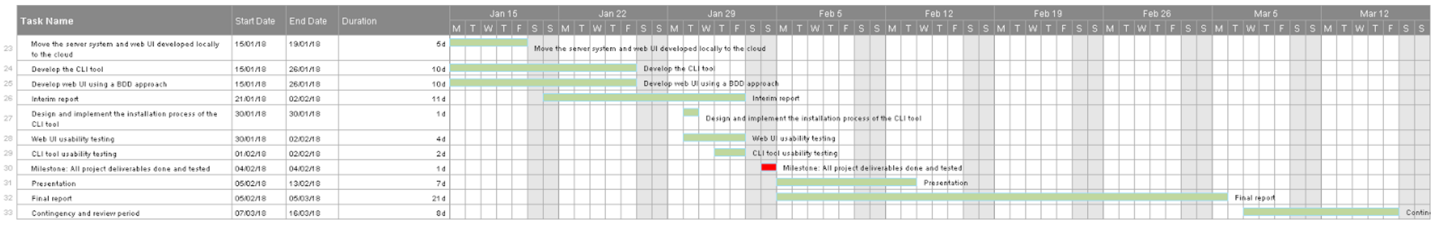
Design the web UI	26/11/17	01/12/17	6d
Milestone: Start the development process			
Develop user accounts and project creation	03/12/17	15/12/17	11d
Implement creating GCP infrastructure from application's config file for server side system	18/12/17	05/01/18	15d
Implement creating AWS infrastructure from application's config file for server side system	18/12/17	05/01/18	15d
Create RESTful API using created libraries locally	07/01/18	12/01/18	6d
Milestone: Server-side goes live			
Move the server system and web UI developed locally to the cloud	15/01/18	19/01/18	5d
Develop the CLI tool	15/01/18	26/01/18	10d
Develop web UI using a BDD approach	15/01/18	26/01/18	10d
Interim report	21/01/18	02/02/18	11d
Design and implement the installation process of the CLI tool	30/01/18	30/01/18	1d
Web UI usability testing	30/01/18	02/02/18	4d
CLI tool usability testing	01/02/18	02/02/18	2d
Milestone: All project deliverables done and tested			
Presentation	05/02/18	13/02/18	7d
Final report	05/02/18	05/03/18	21d
Contingency and review period	07/03/18	16/03/18	8d



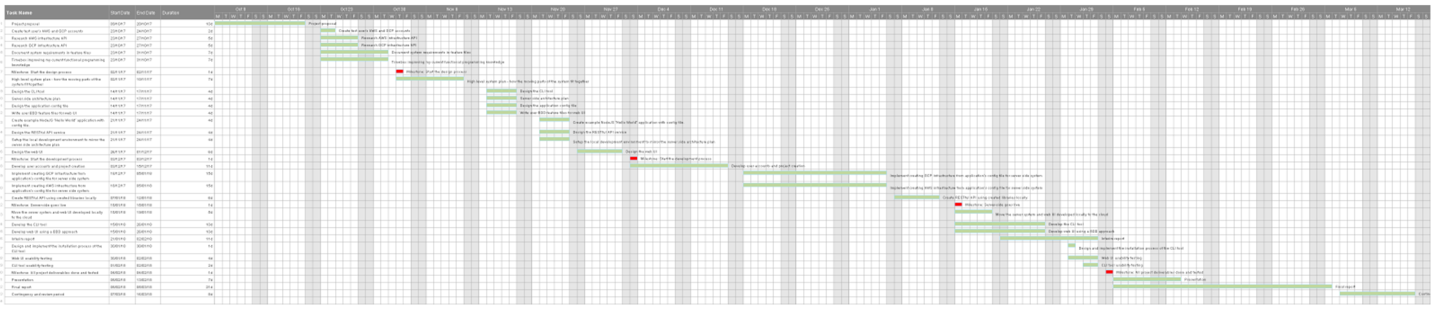
	Task Name	Start Date	End Date	Duration
1	Project proposal	09/10/17	20/10/17	10d
2	Create test user's AWS and GCP accounts	23/10/17	24/10/17	2d
3	Research AWS infrastructure API	23/10/17	27/10/17	5d
4	Research GCP infrastructure API	23/10/17	27/10/17	5d
5	Document system requirements in feature files	23/10/17	31/10/17	7d
6	Timebox improving my current functional programming knowledge	23/10/17	31/10/17	7d
7	Milestone: Start the design process	02/11/17	02/11/17	1d
8	High level system plan – how the moving parts of the system fit together	02/11/17	10/11/17	7d
9	Design the CLI tool	14/11/17	17/11/17	4d
10	Server side architecture plan	14/11/17	17/11/17	4d
11	Design the application config file	14/11/17	17/11/17	4d



	Task Name	Start Date	End Date	Duration
12	Write user BDD feature files for web UI	14/11/17	17/11/17	4d
13	Create example NodeJS "Hello World" application with config file.	21/11/17	24/11/17	4d
14	Design the RESTful API service	21/11/17	24/11/17	4d
15	Setup the local development environment to mirror the server side architecture plan	21/11/17	24/11/17	4d
16	Design the web UI	26/11/17	01/12/17	6d
17	Milestone: Start the development process	03/12/17	03/12/17	1d
18	Develop user accounts and project creation	03/12/17	15/12/17	11d
19	Implement creating GCP infrastructure from application's config file for server side system	18/12/17	05/01/18	15d
20	Implement creating AWS infrastructure from application's config file for server side system	18/12/17	05/01/18	15d
21	Create RESTful API using created libraries locally	07/01/18	12/01/18	6d
22	Milestone: Server-side goes live	15/01/18	15/01/18	1d



	Task Name	Start Date	End Date	Duration
23	Move the server system and web UI developed locally to the cloud	15/01/18	19/01/18	5d
24	Develop the CLI tool	15/01/18	26/01/18	10d
25	Develop web UI using a BDD approach	15/01/18	26/01/18	10d
26	Interim report	21/01/18	02/02/18	11d
27	Design and implement the installation process of the CLI tool	30/01/18	30/01/18	1d
28	Web UI usability testing	30/01/18	02/02/18	4d
29	CLI tool usability testing	01/02/18	02/02/18	2d
30	Milestone: All project deliverables done and tested	04/02/18	04/02/18	1d
31	Presentation	05/02/18	13/02/18	7d
32	Final report	05/02/18	05/03/18	21d
33	Contingency and review period	07/03/18	16/03/18	8d



Glossary

AWS	Amazon Web Services.
GCP	Google Cloud Platform.
Cosmos	The BBC cloud deployment platform
CLI	Command Line Interface
TDD	Test Driven Development
BDD	Behaviour Driven Development
UI	User Interface
FP	Functional Programming
OO	Object Oriented

YAML	Yet Another Markup Language
JSON	JavaScript Object Notation
2FA	Two Factor Authentication
VCS	Version Control System

Appendix

1. Agile and Kanban

The task list will provide the backlog of tickets to get through, and I will set the sprint backlog according to my progress, and the dependencies of tasks. If there is a task which needs to be prioritised quickly, I will bring it straight into the sprint, which is a more Kanban style of working than Agile.

2. Interview with Bogdan Dogaru - BBC News Principal Software Engineer for Article Pages

1. What is the thing that concerns you the most about using the cloud?

- Even though it offers scalability and good value, the public cloud can provide a single point of failure in terms of resilience and security. There are plenty of techniques to make systems highly available and distribute them geographically in a single public cloud but sometimes an entire managed service can suffer from breaches (https://www.pwc.co.uk/cyber-security/pdf/cloud-hopper-report-final-v4.pdf) or outages across different regions. Though rare, these outages can impact businesses significantly.

2. How difficult would it be to switch the products you create and maintain another cloud provider?

- Depending on the architecture of these systems and how many vendor-specific services are used, it can be more or less painful. At the very least, documents defining the infrastructure (e.g. Cloudformation templates for AWS), would need to be adapted for a different vendor.
- In reality, an enterprise-level system would require functionality such as monitoring, alerting, logging, access control, etc. which are more laborious to migrate to a different cloud provider.

3. Are you concerned that using one cloud provider is a single point of failure?

- Yes, a single cloud provider can be a single point of failure. Even though there is a degree of resiliency for managed services, they can have issues (e.g. https://mwork.io/2017/03/14/aws-route53-dns-outage-impacts-last-almost-a-full-day/).

4. Do you feel that being locked into a single cloud provider restricts you from using other cloud services offered by rival cloud platforms?

- Hybrid Public-Private cloud systems have been in use for a while in the industry, proving this can work. In practice, there are some limitations around this, which can be addressed to some degree. For example, network traffic out of a public cloud to servers on-premises is charged at a higher rate, unless a direct peerage is established (and paid for) with the public cloud provider.

5. Would you like to be able to design a system that could cherry-pick services from rival cloud providers to optimise on price/functionality?

- Absolutely, provided there are no hidden costs and there is compatibility between services of different cloud providers, I think this is a great idea!

References

[1] AWS, “Werner Vogels: ‘Everything fails all the time’”, 2008. [Online]. Available: https://thenextweb.com/2008/04/04/werner-vogels-everything-fails-all-the-time/. [Accessed: 14th October 2017]

[2] AWS, 'Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region', 2017. [Online]. Available: https://aws.amazon.com/message/41926/. [Accessed: 14th October 2017].

[3] Toosi, Adel Nadjaran and Calheiros, Rodrigo N. and Buyya, Rajkumar, “Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey” 2014 ACM Comput. Surv, New York, NY, USA, pp.7-47

[4] rightscale “Cloud Computing Trends: 2017 State of the Cloud Survey” [Online]. Available: https://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2017-state-cloud-survey [Accessed: 19th October 2017]

[5] A. Palesandro, M. Lacoste, N. Bennani, C. Ghedira-Guegan and D. Bourge, "Mantus: Putting Aspects to Work for Flexible Multi-Cloud Deployment," 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, CA, 2017, pp. 656-663.

[6] European supercloud project, “User-Centric Management Of Security And Dependability In Clouds Of Clouds” 2017 [Online]. Available: https://supercloud-project.eu/ [Accessed: 15th October 2017]

[7] Elkhatib, Yehia, “Mapping Cross-Cloud Systems: Challenges and Opportunities.”, 2016 HotCloud Lancaster University, UK

[8] aws.amazon.com “Cloud Products” 2017 [Online]. Available: https://aws.amazon.com/products/ [Accessed: 15th October 2017]

[9] cloud.google.com “Products & Services | Google Cloud Platform” 2017 [Online]. Available: https://cloud.google.com/products/ [Accessed: 15th October 2017]

[10] M. Soni, "End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery," 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), Bangalore, 2015, pp. 85-89.

[11] docs.aws.amazon.com, “Amazon EC2 Security Groups for Linux Instances”, 2017 [Online] Available: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html [Accessed: 14th October 2017]

[12] trello, “Trello” 2017 [Online]. Available: http://trello.com [Accessed: 15th October 2017]

[13] Babel, “Use next generation JavaScript, today.” 2017 [Online]. Available: https://babeljs.io/ [Accessed: 14th October 2017]

[14] Webpack, “webpack asset bundler” 2017 [Online]. Available: https://webpack.js.org/ [Accessed: 14th October 2017]

[15] React, “A JavaScript library for building user interfaces” 2017 [Online]. Available: https://reactjs.org [Accessed: 14th October 2017]

[16] lodash, “A modern JavaScript utility library delivering modularity, performance & extras.” 2017 [Online]. Available: https://lodash.com [Accessed: 14th October 2017]

[17] G. McGrath, J. Short, S. Ennis, B. Judson and P. Brenner, "Cloud Event Programming Paradigms: Applications and Analysis," 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), San Francisco, CA, 2016, pp. 400-406

[18] swagger.io, “Swagger Editor” 2017 [Online]. Available: https://swagger.io [Accessed: 14th October 2017]

[19] J. L. de Moura, A. S. Charão, J. C. D. Lima and B. de Oliveira Stein, "Test case generation from BPMN models for automated testing of Web-based BPM applications," 2017 17th International Conference on Computational Science and Its Applications (ICCSA), Trieste, 2017, pp. 1-7.

[20] W. Li, S. Xiang and S. Chen, "Improvement Method of SSL Protocol Identity Authentication Based on the Attribute Certificate," 2012 International Conference on Computer Science and Service System, Nanjing, 2012, pp. 1154-1157.

[21] nngroup.com, 'Why You Only Need to Test with 5 Users', 2000. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users. [Accessed: 14th October 2017].

[22] kanbanblog, “What is Kanban?”. 2017 [Online]. Available: http://kanbanblog.com/explained/ [Accessed: 15th October 2017]

[23] w3, ‘Accessibility’, 2016. [Online]. Available:
https://www.w3.org/standards/webdesign/accessibility [Accessed 19th October 2017]

[24] UK Government, ‘Data Protection Act’ , 1998. [Online]. Available:
https://www.legislation.gov.uk/ukpga/1998/29/contents [Accessed 19th October 2017]

[25] P. Castro, V. Ishakian, V. Muthusamy and A. Slominski, "Serverless Programming (Function as a Service)," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 2658-2659.

[26] G. McGrath and P. R. Brenner, "Serverless Computing: Design, Implementation, and Performance," 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), Atlanta, GA, 2017, pp. 405-410

[27] X. Jia, "Google Cloud Computing Platform Technology Architecture and the Impact of Its Cost," 2010 Second World Congress on Software Engineering, Wuhan, 2010, pp. 17-20.
doi: 10.1109/WCSE.2010.93

[28] T. Islam and D. Manivannan, "Predicting Application Failure in Cloud: A Machine Learning Approach," 2017 IEEE International Conference on Cognitive Computing (ICCC), Honolulu, HI, 2017, pp. 24-31.

[29] Vishwanath, Kashi Venkatesh and Nagappan, Nachiappan, “Characterizing Cloud Computing Hardware Reliability” 2010 Proceedings of the 1st ACM Symposium on Cloud Computing, Indianapolis, Indiana, USA, pp.193-204

[30] Janus, Pawel and Rzadca, Krzysztof, “SLO-aware Colocation of Data Center Tasks Based on Instantaneous Processor Requirements” 2017 Proceedings of the 2017 Symposium on Cloud Computing, Santa Clara, California, pp.256-268

[31] geekwire.com, “Amazon Web Services’ secret weapon: Its custom-made hardware and network”, 2017 [Online]. Available:

https://www.geekwire.com/2017/amazon-web-services-secret-weapon-custom-made-hardware-network/. [Accessed: 14th October 2017]

[32] cloudplatform.googleblog.com, “Titan in depth: Security in plaintext”, 2017 [Online]. Available: https://cloudplatform.googleblog.com/2017/08/Titan-in-depth-security-in-plaintext.html [Accessed: 14th October 2017]

[33] I. I. Kholod, S. V. Rodionov, K. A. Tarasov and A. V. Malov, "Using the features of functional programming for parallel building of decision trees," 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg, 2017, pp. 445-449.

[34] U. S. Premarathne, "Reliability analysis of trust based federated identity management in InterCloud: A graph coloring approach," 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2017, pp. 345-348.