

Text Analytica: cloud-based document analysis

Florian Bauer
Department of Computer Science
University of Bristol
Bristol, United Kingdom
ya18048@bristol.ac.uk

Nathalie Pett
Department of Computer Science
University of Bristol
Bristol, United Kingdom
aq18034@bristol.ac.uk

Abstract—The application can be run online at <http://textanalytics.lukaspmann.io/>. The source code is available at <https://github.com/darkcookie298/CloudComputing>.

I. INTRODUCTION

Text Analytica is a cloud-based application which aims to support the analysis of text documents. For the purpose of this coursework a prototype has been developed and deployed using different cloud computing technologies. In the remainder of this section I, the general concept behind Text Analytica is discussed as well as the limitations of the implemented prototype. The section II of this report explores the reasons behind choosing Microsofts cloud services over Oracles. Afterwards in section III the system architecture and technologies used for the project are explained in more detail. The scalability of the developed solution is addressed in section IV, including evidence of the performance of the service under load. Lastly, in section V, future improvements of the application are proposed.

A. Vision

University students are often faced with an abundance of resources regarding specific units or even certain topics within a unit. These range from lecture notes or slides to personal notes and additional scientific papers as well as e-books or extracts thereof. The first step in the exploration process is for students to familiarise themselves with these materials by identifying the documents key aspects and discovering links between different sources. This is what Text Analytica ultimately aims to facilitate.

More specifically the functionalities of Text Analytica could include tagging, keyword search, suggestions of related documents based on textual analysis and eventually the generation of short summaries, all based on user-supplied PDF documents. These functionalities render Text Analytica a useful tool for a large number of scenarios in which people are confronted with many different and possibly complex text sources, e.g. in the context of management decisions in industry or business / commerce.

B. Limitations of the submitted prototype

The focus of this coursework assignment was to deploy an application using different cloud services and explore its scalability. Therefore the functionality of the submitted Text Analytica prototype was stripped down to a minimum.

To skip the step of extracting machine readable text from PDFs by applying OCR techniques, currently users are only able to upload simple .txt documents. These files are then parsed and analysed. At this point the analysis merely tags the system entries with the three most used words in the text. As the current state of analysis functionality does not require the files to be stored within the application, they are discarded after being processed. Additionally the user account and login functionality has not yet been implemented.

- explain what we actually implemented and how it is different from the proposed application in FA2 (point to section about improvements)

II. PLATFORM CHOICE

The considerations detailed below led to choosing to work with Microsoft Azure to remain within the given time scope for the coursework.

A. Setup

- technical issues with Oracle: unintuitive UI (add contributor), service limits

At first we tried to use the Oracle Cloud and to build and maintain a kubernetes cluster we wanted to use the Terraform command line tool. This was the recommended way we got from the tutorials in class from the Oracle Team. After a long period of installing all needed tools e.g. *Terraform* itself, *Terraform Provider*, *Terraform Kubernetes Installer for Oracle Cloud* [1], we tried to create an easy example cluster. But already at this point we run into service limits. These limits are restrictions on how many instances of a specific resource you are allowed to use. To increase the limit you have to open a ticket request and then after a few days there will be an Oracle support guy helping you. As the ticket request system is very inconvenient and complicated, and the whole process takes several days this is a very strong disadvantage of the **Oracle Cloud** in comparison to **Microsoft's Azure**.

B. Documentation and support

An even bigger advantage of using one of "the big three", so Google Cloud, Amazon AWS or Microsoft Azure, is there are a lot of tutorials, answered questions on *Stackoverflow*, etc and better documentations. Respectively using Oracle leads to a very small number of tutorials, which is one of the reasons why it is more difficult for beginners. There were a lot of

errors, too, but searching for solutions of our problems with Terraform and Oracle Cloud was very unsuccessful. So after increasing the service limits and running in errors again and again, we got to a point, where we decided to switch to Azure. In retrospect this was one of our best decisions of the whole project.

- more documentation
- more community support

C. *More Advantages*

- faster scaling without the need of starting a new VM for peak times
- TODO add other more 'small' advantages

III. SYSTEM ARCHITECTURE

A. *Infrastructure*

- container and Kubernetes (IaaS)

B. *Data storage*

- CosmosDB (PaaS); similar to MongoDB: high availability, scaling, ...

C. *Microservices*

- Flask and 3rd party
- REST API between front end and back end, services (back end, analysis, front end)

D. *Infrastructure as code*

- infrastructure can be deployed from a template automatically

IV. SCALABILITY

A. *Infrastructure*

As mentioned in section III all services are managed by the Kubernetes cluster. The main advantage of using a Kubernetes cluster is scalability and loadbalancing. So if there is a high usage of the existing resources the cluster can scale up by itself. In our case this might happen at peak times when a lot of people are using the Text Analytica webservice. Then the cluster creates more (TODO: describe more carefully what exactly happens here) [2].

As we are using the Azure specific option of *Virtual Nodes*, we can achieve even faster up- and downscaling times than with the pure Kubernetes cluster itself. This is due to the fact we do not need to wait on new VMs or container booting up (TODO: check this). [3].

B. *Data storage*

- PaaS and other region

C. *Service*

- container and microservices -¿ why is this important for scalability?!

D. *Monitoring and Load Testing*

- include some evidence on how our application scales
- maybe show screenshots and explain functionality of kubernetes dashboard

V. FUTURE IMPROVEMENTS

A. *Infrastructure*

- serverless (Lambda, Azure functions), PaaS

B. *Service*

- add functionality (login, pdf, text recognition, more advanced analysis, save files and retrieve them, ...)
- better scoping of microservices
- use of more Kubernetes functionalities

C. *Monitoring*

- advanced cloud monitoring

VI. CONCLUSION

conclusion...

REFERENCES

- [1] Github terraform kubernetes installer. <https://github.com/oracle/terraform-kubernetes-installer>. Accessed: 2019-01-07.
- [2] Microsoft azure kubernetes service aks. <https://docs.microsoft.com/en-us/azure/aks/>. Accessed: 2019-01-08.
- [3] Microsoft documentation virtual node. <https://azure.microsoft.com/de-de/resources/samples/virtual-node-autoscale/>. Accessed: 2019-01-08.