

Addressing the challenges of Cloud Computing adoption in an enterprise environment.

Use case for encouragement and raising awareness among the staff, development of secure and compliant components and analysis of application performance on different Microsoft Azure Cloud Services within the Cloud Competence Center in Rabobank

Stefan Stojkovski



Delft University of Technology

Addressing the challenges of Cloud Computing adoption in an enterprise environment.

Use case for encouragement and raising awareness among the staff, development of secure and compliant components and analysis of application performance on different Microsoft Azure Cloud Services within the Cloud Competence Center in Rabobank

Master's Thesis in Computer Science

Parallel and Distributed Systems group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

Stefan Stojkovski

13th November 2018

Author

Stefan Stojkovski

Title

Addressing the challenges of Cloud Computing adoption in an enterprise environment.

MSc presentation

4th December 2018

Graduation Committee

Prof. Dr. Dick Epema	Delft University of Technology
Dr. Jan S. Rellermeyer	Delft University of Technology
Dr. Georgios Gousios	Delft University of Technology
Erik Jongsma	Rabobank

Abstract

Rabobank is currently planning a complete transition of its services to the public cloud. Currently there are around 400 DevOps teams that need to make the transition from deployment on traditional on-premise infrastructure, to deploying their products to the public cloud. The thesis project investigates what are some of the biggest challenges in adopting cloud computing technologies in an enterprise. Some of these are encouraging the staff to adopt the technology, how to embed security and compliance to the cloud computing infrastructure and which services to choose when migrating an on-premise application to Microsoft Azure Cloud. Investigation has been done on how to better encourage and inform the staff about the cloud adoption. This is achieved by improving the Cloud Awareness session (where the employees are informed for the cloud initiatives in the bank) through implementation of complete CI/CD (Continuous Integration / Continuous Deployment) pipeline of a .NET Core 2.0 application with modern HTML5 responsive layout that deploys the web application on Microsoft Azure Public Cloud and gives recommendations for testing and monitoring. Moreover, the project investigates what is needed to develop the secure and compliant feature in a huge enterprise like Rabobank with an example of development an Azure Cosmos DB feature delivered as a VSTS (Visual Studio Team Services) extension to be used by the DevOps teams in their CI/CD pipeline. Finally, an analysis is done on the performance, cost and lifecycle management of the same .NET Core 2.0 application deployed on different service offerings by Microsoft Azure Public Cloud, including Windows Server virtual machine, Azure Web App Service and Azure Kubernetes Service.

Preface

MOTIVATION FOR RESEARCH TOPIC I am writing my master thesis project as part of an internship at the Cloud Competence Center within Rabobank, Dutch multinational banking and financial services company headquartered in Utrecht, Netherlands, enterprise which employs more than 40.000 people worldwide. Additionally, the thesis is part of the EIT Digital Master Programme in Cloud Computing and Services at Delft University of Technology. Rabobank is currently going through a process of complete transition of its services to the public cloud. This means that around 400 DevOps teams will be on-boarded to use the Cloud Computing technologies, primarily Microsoft Azure, and Amazon Web Services (AWS) in the later stage.

In order for this to happen there are many things that need to be arranged in advance, so the use of the technologies is done in a controlled environment which is safe and compliant to the banking industry standards. Many of the applications are processing very sensitive data that must be suitably protected. For this purpose the Cloud Competence Center was created within Rabobank.

My task was to address some of the challenges of cloud adoption in a big enterprise such as Rabobank, with the focus on raising awareness and encouragement of the affected staff, and dealing with security and compliance by developing out of the box secure and compliant features to be used by the teams. To address the former challenge, an investigation has been done on how to better present the advantages of using the Microsoft Azure Cloud to the DevOps teams on the Cloud Awareness Sessions held within Rabobank on regular intervals. The current demos in place were a simple deployment of a single component to the Microsoft Azure Cloud and did not fully showcase the full potential of the Cloud that can be used by the teams. For this I have created a demonstration of complete CI/CD (Continuous Integration / Continuous Deployment) pipeline of a .NET Core 2.0 application with modern HTML5 responsive layout.

To address the latter challenge, the thesis investigates what is needed to develop the secure and compliant feature in a huge enterprise like Rabobank through an example of development of a feature used by the DevOps teams in a form of VSTS (Visual Studio Team Services) extension. This feature is Azure Cosmos DB with the MongoDB API.

Finally, in the final part of my thesis, an analysis is done on the performance, costs and life cycle management of the sample NET Core 2.0 application deployed on

different service offerings by Microsoft Azure Public Cloud, including Windows Server virtual machine, Azure Web App Service and Azure Kubernetes Service. This has benefits for the bank as it would give overview of the different services offered, their pros and cons, and will help the decision making process for the applications that are migrated from on-premise to the cloud without being completely re-architected.

ACKNOWLEDGEMENTS I would first like to thank my thesis advisor Prof. Dr. Jan S. Rellermeyer, of Parallel and Distributed Systems group, of the Faculty Electrical Engineering, Mathematics, and Computer Science (EEMCS) of Delft University of Technology. . The door to Prof. Rellermeyer office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge M.Sc. Erik Jongsma of Rabobank as my mentor and supervisor within the team in the Cloud Competence Center in Rabobank, and I am gratefully indebted to him for his very valuable comments on my work and on this thesis.

Stefan Stojkovski

Delft, The Netherlands
13th November 2018

Contents

Preface	v
1 Introduction	1
1.1 Problem statement	2
2 Background and concepts	5
2.1 Cloud Computing	5
2.2 DevOps Way of Working	6
2.3 Microsoft Azure	8
3 Literature survey on challenges of cloud adoption for banks	11
3.1 Introduction	11
3.2 Technological context	12
3.3 Organizational context	13
3.4 Environmental context	14
3.5 Perceived risks and benefits	15
3.6 Conclusions	15
4 Implementation of demonstration prototype	19
4.1 PaaS (Platform as a Service)	20
4.1.1 Direct deployment from Visual Studio	20
4.1.2 Deployment with CI/CD pipeline in VSTS (Visual Studio Team Services)	21
4.2 Infrastructure as a Service	26
4.2.1 Azure Automation Runbook Deployment	27
4.2.2 Azure Automation with DSC (Desired State Configuration)	27
4.2.3 Deployment of Azure resources with ARM templates . . .	29
4.3 Testing	33
4.3.1 Testing in production	33
4.3.2 Coded UI Test using Selenium in Visual Studio	33
4.4 Application monitoring	34
4.4.1 User Telemetry and Perf Monitoring with App Insights . .	35
4.4.2 Creating Custom Telemetry Events	36
4.4.3 Feature flag implementation	37

4.5	Conclusions	38
5	Qualitative and quantitative comparison of an application application performance on various cloud services	43
5.1	Testing environment and conditions	44
5.2	Windows Server Virtual Machine	44
5.2.1	Ease of deployment	44
5.2.2	Lifecycle management	44
5.2.3	Cost	45
5.2.4	Performance	45
5.3	Azure Kubernetes Service	47
5.3.1	Ease of deployment	47
5.3.2	Lifecycle management	47
5.3.3	Cost	47
5.3.4	Performance	48
5.4	Azure Application Service - Web Application	50
5.4.1	Ease of deployment	50
5.4.2	Lifecycle management	50
5.4.3	Cost	50
5.4.4	Performance	50
5.5	Overall performance comparison	53
5.5.1	Weekday vs. weekend days testing, different times of day testing	53
5.5.2	Performance under load of 300 users	54
5.5.3	Performance under load of 1000 users	55
5.5.4	Performance under load of 300 and 1000 users with experimental conditions	58
5.5.5	Coded User Interface testing with selenium	59
5.6	Conclusions	60
6	Security and Compliance: Developing out of the box secure and compliant cloud components. Use case of Azure Cosmos DB	63
6.1	Description of Azure CosmosDB	64
6.2	Planning and design decisions	64
6.3	Proof of Concept	67
6.4	Implementation details	68
6.5	Requirements for Security, Service Management Controls and separation of responsibilities	69
6.5.1	Security requirements and separation of responsibilities	69
6.5.2	Service management controls requirements and separation of responsibilities	70
6.6	Testing and evaluation	70
6.6.1	Extension build, release and publishing	71

6.6.2	Installing the extension and deployment in different environments with different configurations	71
6.6.3	Manual test of the desired configuration after the resources are deployed	72
6.7	Conclusions	74
7	Conclusions and Future Work	85
7.1	Conclusions	85
7.2	Future Work	87

Chapter 1

Introduction

The IT Infrastructure department at Rabobank is currently transitioning towards a public cloud solution for some of its services using Microsoft Azure Cloud platform. This transition needs to be carried out very carefully on the one hand and fast enough on the other hand so that the company could maintain its technology leadership by utilizing the most recent Cloud Computing technology offerings and speed up its delivery cycle. This means that around 400 DevOps are affected of this restructuring and will be on-boarded to use the Cloud Computing technologies, primarily Microsoft Azure, and Amazon Web Services (AWS) in the later stage. This has a huge impact on the whole organization, starting from spreading awareness of the capabilities and benefits of the new technology, to implementation of the whole on-boarding process in a consistent, safe and compliant manner. The focus of this thesis is to investigate how to raise awareness and encourage the staff to adopt the cloud technologies by improving the Cloud Awareness Sessions, that are delivered to the potential clients among the DevOps teams, so they reflect the benefits and the capabilities of the Microsoft Azure Public Cloud technology. Moreover, an analysis is done on the performance, costs and life cycle management of the sample NET Core 2.0 application deployed on different service offerings by Microsoft Azure Public Cloud, including Windows Server virtual machine, Azure Web App Service and Azure Kubernetes Service. This has benefits for the bank as it would give overview of the different services offered and their pros and cons, and will help the decision making process for the applications that are migrated from on-premise to the cloud without being completely re-architected. Finally, the project investigates how to implement and deliver safe and compliant service Microsoft Cosmos DB, globally distributed database service, in a form of VSTS (Visual Studio Team Services) extension ready to use by the DevOps teams in a release pipeline.

1.1 Problem statement

The product deployment process in the public cloud is different than the traditional one. Traditionally, development and operation teams were siloed which meant a clear separation of responsibilities. Development teams are responsible for development, integration, testing and releasing of the product while the operation teams are responsible for deployment and infrastructure management. With the DevOps way of working, development and operations teams are no longer siloed. Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function. DevOps aims at shorter development cycles, increased deployment frequency, and more dependable releases, in close alignment with business objectives. Traditionally, the assumption is that all systems are long-lived. Furthermore, it is assumed that someone is responsible for the management of the systems (have control what is installed on it). These days, with Cloud Computing arising, these statements are no longer true in many cases where deployment happens daily or even few times a day. The focus of the project is to investigate how to raise awareness and encourage the staff, affected from the change, by improving the Cloud Awareness Sessions to reflect the benefits and the capabilities of the Cloud Computing Technology. The Cloud Awareness sessions are sessions held in regular intervals in the bank, with the focus to inform the employees, that are affected of the cloud adoption, of current developments around Cloud Computing within the bank. The improvement of this session is done by creating a complete CI/CD pipeline for an .NET Core 2.0 which is an example eCommerce website site based for training purposes described in chapters 31-35 of The Phoenix Project, by Gene Kim, Kevin Behr and George Spafford. This application is part of Microsoft Professional Program (MPP) with DevOps series of online courses available on edX. Moreover, an analysis is done on the performance, costs and life cycle management of the sample NET Core 2.0 application deployed on different service offerings by Microsoft Azure Public Cloud, including Windows Server virtual machine, Azure Web App Service and Azure Kubernetes Service. This has benefits for the bank as it would give overview of the different services offered and their pros and cons, and will help the decision making process for the applications that are migrated from on-premise to the cloud without being completely re-architected. Finally, an investigation is done on how to deliver a safe and compliant Cosmos DB VSTS extension to be used by the DevOps teams that already have access to the Microsoft Azure platform within Rabobank.

Research questions

1. How to raise awareness and encourage the affected staff from Cloud adoption by demonstration of the capabilities of Microsoft Azure Cloud technology and Visual Studio Team Services to speed up development and operations processes.

2. How to conduct a qualitative and quantitative comparison of the performance of an sample application deployed on different service offerings by Microsoft Azure Public Cloud.
3. How to enforce security and compliance controls on using Microsoft Azure Components. Use case of development of a Visual Studio Team Services extension, so it can be used in a safe and compliant manner in a CI/CD pipeline.

The rest of this thesis is organized in the following way:

- Context of the cloud computing technologies, DevOps way of working and Microsoft Azure Public Cloud technologies
- Literature survey for the challenges of adoption of Cloud Computing technologies in an enterprise
- Description of the implementation of the CI/CD pipeline for a .NET Core 2.0 web application
- Description of comparison of the performance, costs and life cycle management of the sample NET Core 2.0 application deployed on different service offerings by Microsoft Azure Public Cloud, including Windows Server virtual machine, Azure Web App Service and Azure Kubernetes Service.
- Description of the implementation of the Microsoft Azure Cosmos DB VSTS(Visual Studio Team Services) extension
- Conclusion and future work recommendations

Chapter 2

Background and concepts

This chapter describes concepts of Cloud Computing, DevOps way of working and Microsoft Azure Public Cloud that are used throughout this thesis.

2.1 Cloud Computing

The National Institute of Standards and Technology (NIST), which is the official US-based standards and technology definitions body, has defined Cloud Computing as the following

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

- The NIST Definition of Cloud Computing by Peter Mell and Timothy Grance [30]

The model has five essential characteristics:

- On-demand self-service: The ability for companies to allocate required resources by themselves, without involvement from the cloud service provider.
- Broad network access: Being accessible through standard network access mechanisms, without the need for any specialized infrastructure.
- Resource pooling: The pooling of the various resources to be allocated from and returned to as needed.
- Rapid elasticity: The pooling of the various resources to be allocated from and returned to as needed.
- Measured service: The ability to measure exactly what resources are being used, to monitor and control those services, and to be able to present that data to the service provider or end-user.

The model has three service models:

- **Software as a Service (SaaS):** SaaS services use the internet to deliver applications as cloud-based services. The users can subscribe to the service and use it directly through the web browser, and do not require any downloads or installations on the client side. The users do not have to do any maintenance since that is responsibility to the service provider.
- **Platform as a Service (PaaS):** PaaS services provide cloud-based services that provide resources on which developers can build their own solutions. These serve as a framework for developers that they can build upon and use to create customized applications. The service provider manages the fundamental operating system (OS) capabilities, servers, storage, and networking while the developers can maintain management of the applications.
- **Infrastructure as a Service (IaaS):** IaaS services provide on-demand self-service highly scalable and automated compute resources like compute, network, and storage infrastructure components. IaaS facilities are managed in a similar way to on-premises infrastructure and provide an easy migration path for moving existing applications to the cloud. 2.1 [35]

The model has four deployment models:

- **Private cloud:** Consists of infrastructure used exclusively by one organization and can be used by multiple consumers (business units). The private cloud can be physically located at your organizations on-site datacenter, or it can be hosted by a third-party service provider.
- **Community cloud:** Consists of infrastructure used by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It can be owned, managed, and operated by one or more of the organizations in the community, and it may exist on or off premises.
- **Public cloud:** Consists of infrastructure owned and operated by third-party cloud service providers and delivered over the Internet. Microsoft Azure is an example of a public cloud. It exists on the premises of the cloud provider.
- **Hybrid cloud:** Consists of infrastructure that is a composition of two or more distinct cloud infrastructures, so organizations can reap the advantages of both. In a hybrid cloud, data and applications can move between private and public clouds for greater flexibility and more deployment options. [30]

2.2 DevOps Way of Working

"DevOps is the union of people, process, and products to enable continuous delivery of value to our end users."

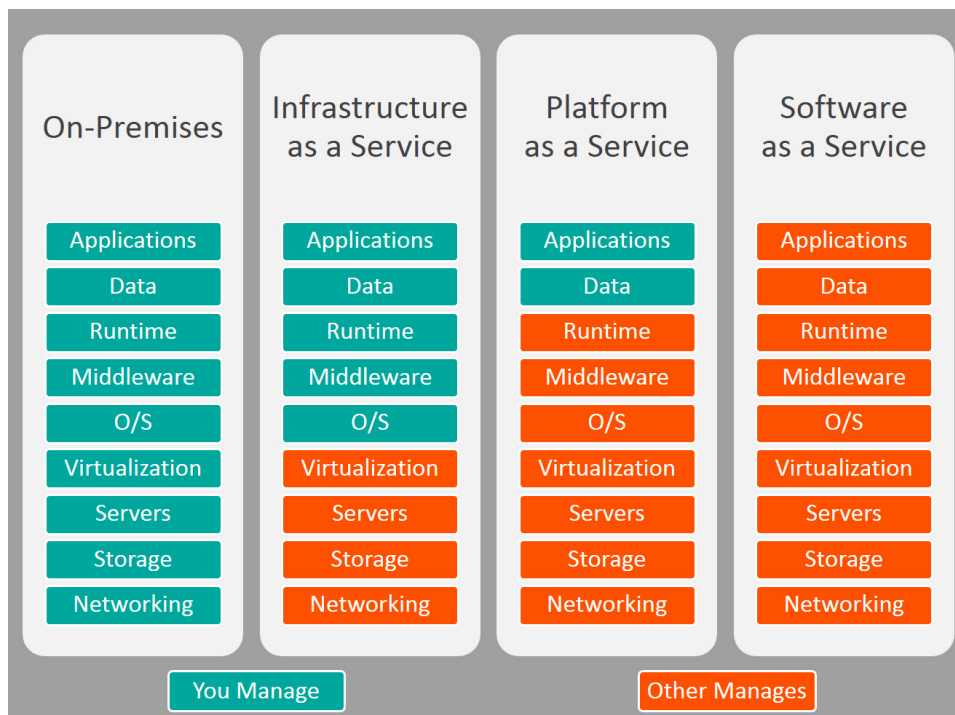


Figure 2.1: Summary of key differences between the three service models of cloud computing

You cannot buy DevOps and install it. DevOps is not just automation or infrastructure as code. DevOps is people following a process enabled by products to deliver value to end users."

Donovan Brown, Microsoft DevOps Program Manager

One of the core values of DevOps is a shortened release cycle. On the figure 2.2, the left chart shows the rate of change for a slow delivery cadence with a long time between deliveries, with a massive amount of change, including many lines of code, lots of features, and lots of things that can go wrong.

The goal of DevOps is to create fast delivery cycle, in which there are very frequent deployments, with less and less change, to ensure that what goes into production has limited impact and can be tested very specifically, get early feedback learn and repair quickly. Within a DevOps culture, all team members who are involved in creating, delivering, and monitoring software, work together closely to deliver value to consumers at increasing frequencies. There are seven key DevOps practices:

- Configuration management: management of the configuration of all environments for an application that is version controlled.
- Continuous integration: the process of automating the build and testing of

code every time a team member commits changes to version control [18]

- Continuous deployment: the ability to use the output from CI and then automatically deploy the tested build to different environments [18]
- Release management: the maturation of continuous deployment from just one environment to the rest of the environments in a pipeline. [18]
- Infrastructure as Code: enables the automation and validation of the creation and teardown of infrastructure, such as networks and virtual machines [21]
- Test automation: the process of executing automated tests as part of the delivery pipeline, so we can get an early feedback on the software quality
- Application performance monitoring: measuring the performance and availability of software to improve stability, detect compliance and risk.

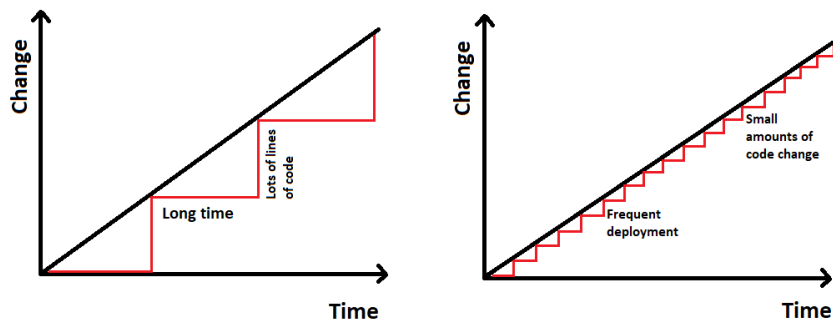


Figure 2.2: Figure showing the transition from slow delivery cycles to fast delivery cycles

2.3 Microsoft Azure

Microsoft Azure is Microsoft's public cloud offering which enables clients to create, deploy, and operate cloud-based applications and infrastructure services. The platform offers IaaS, PaaS, and SaaS services to clients worldwide. [22]

Microsoft Azure provides various cloud services across the IT spectrum which can be organized in several broad categories. 2.3



Figure 2.3: Most popular Microsoft Azure services organized in several broad categories

Chapter 3

Literature survey on challenges of cloud adoption for banks

3.1 Introduction

After the economic crisis of 2008 and the invention of the smartphone, the banks needed new customer-centric business model since the customer was in control of which financial institution to choose more than ever before. One of the ways to achieve this is through the use of cloud computing (CC) technology. Cloud computing can help banks to deliver innovative customers experiences, have more effective collaboration and enhance the speed to market, among the other benefits. [12]

Cloud computing has arrived as a novel IT paradigm that promises to revolutionize the way IT services are provisioned and consumed. [36] Kiran Kawatra and al. state that cloud computing offers a number of benefits to the financial institutions which are stated in section 3.5 3.5

But despite the obvious benefits, the adoption of the cloud technologies was not as fast as initially expected given the banking industry's traditional heavy dependence on information technology[10]. The purpose of this literature survey is to investigate and summarize the factors that affect cloud adoption in banks and other big enterprises. The analysis is based on the TOE-Framework of Tornatzky and Fleischer (1990) [34] Most of the analyzed literature either used this model or their findings fit into the proposed categories of the framework[12] [10] [2] [31] [28] [27] [38] [33] [5].

The TOE framework identifies three context groups: technological, organizational, and environmental. The technological context includes the internal and external technologies that are relevant to the firm which may include both equipment as well as processes. The organizational context refers to the properties of the organization, such as the size of the firm, degree of centralization, need formalization, managerial structure, human resources quality etc. Finally, the environmental context refers to the industrial environment, such as the competitors, the govern-

ment policy, the size and structure of the industry[8].

Reiger et al. [10] conducted expert interviews with senior decision makers from nine large German banks and summarized their findings in the following figure 3.1

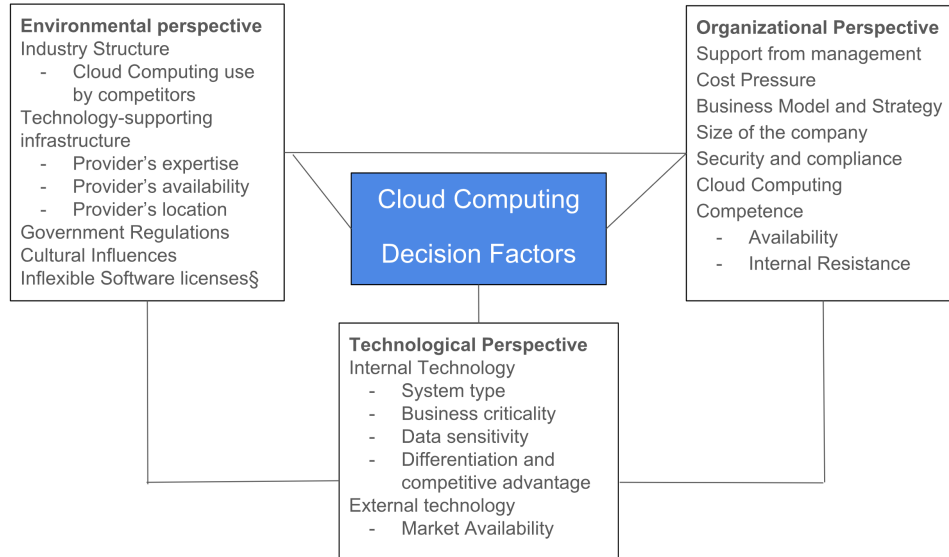


Figure 3.1: Empirical Results of Framework of CC Decision Factors by Rieger et al.

This is used as a base for the discussion that follows in the next sections where the arguments are supported with the findings of several other related studies.

3.2 Technological context

One of the most common motivations to adopt a cloud computing solution is the improvement of the speed of business value delivery. This is achieved through shifted responsibility of maintenance and operation of the systems which shortens the delivery cycle and enables teams to create more business value according to the customer needs. [12] [2] [10]

Another important reason to adopt cloud solution was the interoperability and portability of information between private clouds and public clouds [28], though another study [38] defined the integration and interoperability more as a challenge than an opportunity.

Moreover, banks are more eager to first adopt cloud computing solution to less critical support systems, rather than core business applications in order to avoid risk and storing sensitive data in the cloud [10].

Complexity and compatibility, unexpectedly, were found not to be significant discriminators. This is primarily due to the fact that most cloud vendors provide ex-

pertise in forms of consultancy and most cloud adopters deploy hybrid team model which is a team composed of in-house employees and external consultants [5] [12] [10] [31]. However, this was seen as a threat by the research of Raza et al. [27] who interviewed employees facing cloud adoption, if they saw the cloud adoption as a threat to their job and if they were eager to postpone it, while some [28] do not have strong opinion for either side and state that it depends on the company culture.

Low et al. [5] observed that relative advantage had a significantly negative influence on cloud computing adoption in the high-tech industry. They argue the reason for this may be that the adoption is seen as too complex, and the price for the gained relative advantage was too high, observed through the cost of implementation, gaining the know-how and the organizational change.

3.3 Organizational context

One of the most important factors for cloud computing adoption is the management support which is critical for creating a supportive climate and for providing adequate resources for the adoption of cloud computing technologies. The decisions are often driven by the incentive of cost reduction and the pressure to offer competitively priced products to their customers. This is discussed and supported in most of the studies analyzed [10] [5] [31] [28] [33].

Another factor that influences cloud adoption is the size of the company. It is observed that bigger company size has a positive correlation with the adoption of the cloud. This is due to the fact that larger companies have more resources and are more eager to try out the technologies. Moreover, can afford to invest in developing in-house knowledge. On the other hand, small companies do not have the resources to afford a technological switch if they do not use the cloud, but are more likely to adopt cloud technologies if they are in the start-up phase due to the lower capital expenditures [10] [31] [5].

Other important components are security and compliance requirements. The banks are subject to strong regulations and have strong security and compliance requirements. For example, data-security and data-protection regulation requirements are some of the most important factors influencing the decision for cloud adoption. Banks must ensure reliability during disaster recovery and to have clearly set backup plans. Banks must know where their data resides in the cloud. Distributed data storage makes the control more challenging and is perceived as a risk for customers. This is due to introduction of more complicated synchronization algorithms that must persistently hold the application state, and dealing with failure as something that happens consistently. In case of a failure, one wants the overall system performance and state to not be affected and remain consistent. The security is affected as well as now both the storage sites and infrastructure communication must be protected [7]. Moreover, banks need to provide evidence of the state changes for auditing purposes [10] [2].

Business strategy and business model of the bank also influence the motivation for cloud adoption. Rieger et al.[10] found that highly specialized businesses are less likely to adopt cloud solutions.

Finally, the competence of the employees is also seen as an influential factor . The cloud computing technologies allows the banks to use the expertise of the providers employees for many of the tasks that were done by the current employees. This can have a negative impact on the enthusiasm of the employees for cloud adoption and lead to resistance. [10] [38]

3.4 Environmental context

In the environmental context, competitive pressure and trading partner power are considered as the most influential factors that lead to cloud adoption. Independent studies [10] [5] state that the banks are more likely to adopt CC technologies if their competition is also adopting it and learn from their experience which serves as a reference. The trading partners' power can also have influence which can be either convincing or compulsory in some cases [5]. For example a strategic partner may announce new technology strategy that affects the compatibility with your internal systems that are using the partners' technology, so you may be forced to update you technology or

Government regulations plays a huge role in the adoption of cloud computing technologies. Both the government, and the banks must have a clear perception how the laws should be applied in a cloud environment regarding sensitive issues such as which data and where the data is stored, privacy settings for the consumers, confidentiality requirements, and other legal consequences [2]. In the European Union, this is regulated by the General Data Protection Regulation (GDPR) which is a regulation in EU law on data protection and privacy for all individuals within the European Union (EU) and the European Economic Area (EEA). It also states that personal data must be stored within the region of EU and EEA [29].

Another major factor is the technology supporting infrastructure. The expertise of the provider can have a great influence of the decision for adoption. The provider must understand the banks' business needs, have a good image, build trust with handling sensitive data. Another thing that is considered is the location where the data is being stored and processed, which is strongly regulated in the European Union(EU) and requires data to reside inside the borders of the EU [10].

The certificates and provider references also play a crucial role in the decision for adoption of CC since they are of great importance for the bank in terms of compliance to the laws and security requirements, so the quality and implementation of the certificates is taken very seriously. [10]

Unclear and inflexible software licenses can also hinder the adoption of cloud computing. This is because the existing licenses are usually focused on an internal IT and are not valid for the resources in the cloud which leads to uncertainty of their usability. [10]

Finally, there are other factors that can influence the adoption of the cloud computing technologies such as the external cultural factors and subjective perceptions. Another point was the different language between the bank and the provider which can lead to misaligned perceptions by both parties and influence the decision. [10] [27]

3.5 Perceived risks and benefits

In this section the perceived risks and benefits of using cloud computing technologies are summarized. The findings are similar in all of the studies analyzed and can be summarized as follows. 3.1 3.2

3.6 Conclusions

In conclusion, it is very important to clarify the motivations behind the decision to adopt cloud computing technology. Identification of key concerns for the adoption which also helps the development of the Microsoft Azure Cosmos DB feature described in chapter 5.

Government regulations are an important factor that has a great influence on the decision of adopting cloud computing technology. This regulations are highly interdependent with other factors like security and compliance and data location.

Another major factor is the perceived financial benefits, which drive the adoption of CC technologies. This is a great incentive for financial institutions such as banks, which are profit oriented.

There are also other concerns for the adoption. The major concern is around security and the data location. Moreover, strategic dependence was viewed as another major concern, since it implies less strategic flexibility for the future.

On the other side, there are many benefits, and the cloud computing technology enables new faster development cycle, encourages innovation and is constantly improving and addressing all of the above mentioned concerns and becomes more and more mature every day. These benefits encourage massive adoption from both small and large companies, such as the case with Rabobank.

Perceived benefit	Explanation
Cost Reduction	Switching from capital expenditure to operational expenditure and pay only for the resources you actually use
Business continuity	Improved availability by SLAs and managed disaster recovery
Business Agility and Focus	Less effort needed for operational tasks and the focus is shifted to delivering more business value by improved velocity
Green IT	Lower carbon footprint by using the resources only when they are needed
Scalability	Easy to scale as pool of (almost) infinite resources is available on demand and the scaling can be automated
Development and Testing	Quick and cost-efficient reaction to less-predictable events and changing customer requirements
Compute Infrastructure	No need for deep analysis of compute needs as (almost) infinite amount of computing resources is available on demand
Managed Backup	The backup and disaster recovery is directly managed by the cloud provider and controlled by SLA
Independence of employees and reduction of workload	Employees are less dependent of the infrastructure as this is handled by the cloud provider, resulting in reduction of workload from operational tasks
Contractually determined service quality (SLA)	Service level agreements are determined in advance and the cloud provider is responsible to comply to this
Possibility to use technology and know-how from the provider	The cloud provider can provide experts that can train your stuff and share knowledge with your employees

Table 3.1: Perceived benefits of using cloud computing technology.
[12] [10] [2] [28] [38]

Perceived risks	Explanation
Performance failure and poor performance	No control over the the performance of the whole system since the control is shifted to the provider
Inappropriate software licensing and pricing models	Licensing and pricing models are tailor to cloud components and may introduce challenge to calculate overall costs for licensing
Limited and standardized functionalities	The functionalities are limited to the offer of the particular cloud provider
Loss of unique selling proposition	The unique selling proposition that comes from the old infrastructure design may be lost due to standardized offer from the cloud provider
Strategic dependence from the provider	The companies become dependent to the cloud provider where their technology implementation rests
Lack of strategic flexibility	The strategic flexibility is limited to the services that the cloud providers offer
Security risks through employees	Both internal and cloud provider employees having some form of access to the technology you are using may introduce additional security risk
Dependence of security implementation by the provider	The security implementation is dependent of the cloud provider and you do not have control over some security policies
Data Segregation and Privileged User Access	New challenges arise when designing data segregation strategy and restricting access privileges to users
Violation of government requirements	No control if the cloud provider violates some of the government requirements or challenges of implementing some requirements in a cloud environment
Loss of benefits	Loss of benefits when purchasing in big quantities from some suppliers (ex. licensing fees)
Loss of image	Perceived loss of image as the stuff is unable to develop the needed technologies by themselves
Interoperability	Concerns about interoperability of internal systems with the system provided by the cloud provider
Changes in the IT Organization	Change of mindset of the employees needed and change in the processes within the organization

Table 3.2: Perceived risks of using cloud computing technology.
[10] [2] [28] [38]

Chapter 4

Implementation of demonstration prototype

This chapter describes some general possible use cases that can be used as a guideline to the DevOps teams within Rabobank. This is done in order to improve the Cloud Awareness Sessions which are held within Rabobank, in which the Cloud Competence Center presents the capabilities of the Cloud Computing technologies, explains the role of the team, and demonstrates some of the capabilities and how can this technology can be leveraged. However, the demos only demonstrated very simple use cases where a single component was deployed in a single environment in the Microsoft Azure cloud. One of the purposes of this thesis is to investigate how can this sessions be improved, and in order to do that a better demonstration use cases were developed. This was needed to improve the perception and demonstrate the capabilities of the technology better by showcasing complex applications deployed to the Microsoft Azure Cloud.

The following sections provide descriptions of the use cases of the Microsoft Azure offerings for PaaS (Platform as a Service), IaaS (Infrastructure as a Service), Containers, Testing, Application monitoring and feedback loops and the integration of these services with Visual Studio Team Services (VSTS) which is the CI/CD (Continuous Integration / Continuous Delivery) tool developed by Microsoft.

The examples were implemented in reference to the instructions in the labs in the Microsoft Professional Program (MPP) for DevOps.

Prerequisites The following examples are based on a sample application called Parts Unlimited which is a .NET Core + SQL Azure example eCommerce website site for training purposes for DevOps scenarios described in chapters 31-35 of The Phoenix Project, by Gene Kim, Kevin Behr and George Spafford. [11]

Technologies used in the examples include

- Visual Studio 2017
- Git

- Microsoft PowerShell Azure module
- NodeJS v6.12.3
- Docker
- Visual Studio Team Services
- Microsoft Azure subscription

4.1 PaaS (Platform as a Service)

This section describes the use of Microsoft Azure PaaS offering, App Service Web Apps. This is a service that enables you to build and host web applications in many programming languages without the need to manage the infrastructure. The infrastructure is managed by Microsoft and it is not visible to the end user. It offers auto-scaling and high availability, on both Windows and Linux and enables automated deployments from GitHub, Visual Studio Team Services, or any Git repo. [25]

Two ways of deploying the application to the Web Apps service are demonstrated.

- Directly from Visual Studio
- Through CI/CD pipeline with Visual Studio Team Services (VSTS)

The preferred and recommended way for Rabobank is automated deployments through VSTS pipeline for compliance and auditing purposes. Automated deployments help auditing because every action is logged, stored and can be traced back. If there is some problem, these logs can be used to determine the cause and the initiator of the problem by tracing back the actions taken on a resource. The retention period of the logs is determined by the regulator. For example if some database gets deleted, the auditor is able to query the logs and trace back the delete action and the person who initiated it, because for each action, the initiator is also stored as you are not allowed to do any actions without previous authentication and authorization on the platform.

The infrastructure needed in Microsoft Azure in both cases is provisioned by using the Azure Resource Manager (ARM) templates, which is the Microsoft way of doing Infrastructure as Code. ARM templates are JSON files that define the resources you need to deploy for your solution. [19]

4.1.1 Direct deployment from Visual Studio

One of the options is to deploy the application directly from Visual Studio. To do this, access to a valid Microsoft Azure subscription is needed as well as proper configuration of the input parameters.

This way of deployment takes advantage of the ARM templates, but the deployment must be initiated manually, changes to configuration can't be reversed easily and is in general much more error prone since it is not fully automated. The things that are needed for this type of deployment are:

1. Microsoft Azure subscription
2. Deploy the PartsUnlimitedEnv project
3. Select the resource group and region for deployment
4. Select the fullenvironmentssetupmerged.json ARM template file
5. Select and edit the parameters file fullenvironmentssetupmerged.param.json
6. Deploy the environment to Microsoft Azure
7. Publish the web application to one or more of the deployment slots

Pointers for choosing values for the parameter can be found in the online manual. [24]

The deployment creates resources for a Web App with 3 deployment slots, Dev, Staging and Prod. This can be seen in the Azure Portal, when exploring the resource group specified during the deployment. After deploying the ARM template, the environment is created, but publishing an application to some of the deployment slots of the Web App is still needed. To do this, publishing of the project PartsUnlimitedWebsite which contains the web application is done. Publishing the application as an existing Microsoft Azure App Service is needed, and during the configuration of the publishing, a specification of the target deployment slot for the application needs to be configured.

Another useful option is swapping deployment slots which swaps the application from one slot to another without any need for configuration changes like changing the connection strings. This is useful when the application is deployed and tested in Staging and one wants to push it to production.

4.1.2 Deployment with CI/CD pipeline in VSTS (Visual Studio Team Services)

In this subsection a description is made on how to create CI/CD pipeline for the application PartsUnlimited described in the beginning of this chapter.

Continuous Integration

Continuous integration (CI) is the process of automating the build and testing of code after every commit done in your version control repository. [18]

In order to enable Continuous Integration pipeline, one needs to do few things:

1. Import Source Code into your VSTS Account with Git

2. Create Continuous Integration Build definition
3. Test the CI Trigger in Visual Studio Team Services

Once VSTS account and a project for the application are created, one is able to create or import an existing code repository by using either the GUI in VSTS or git command line.

Next, a CI build definition needs to be created. This is done by navigating to the VSTS project Build hub, and then creating new empty build definition. After this, in the process heading one needs to choose a build agent queue. Next, one needs to specify the source repository for the build definition to be the VSTS repository with the master branch. 4.1

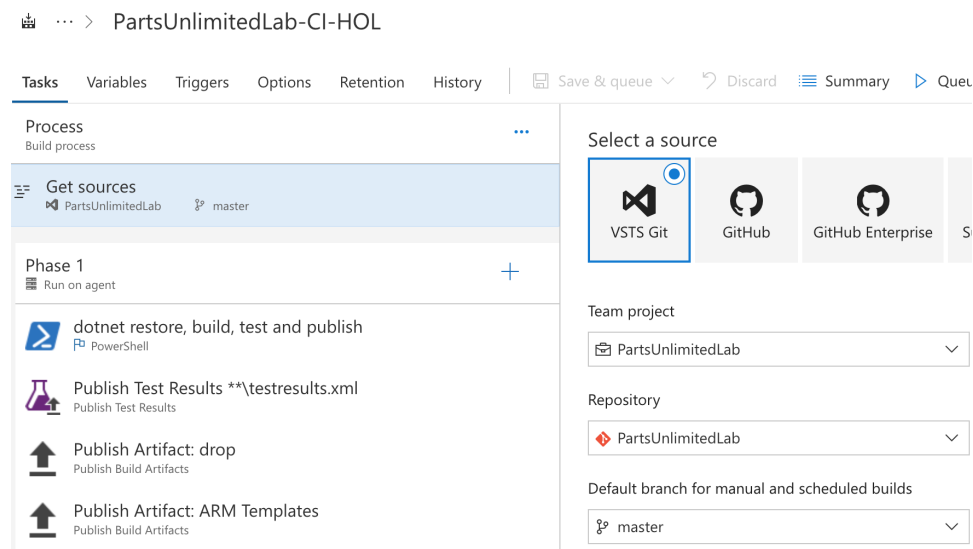


Figure 4.1: Configuration of the source for the Continuous Integration Build Pipeline

Finally one needs to add and configure the build tasks. In order to do this one needs to add a PowerShell task which will build the solution, Publish Test Results task which executes the tests in the project, and two Publish artifacts tasks, one for publishing the website artifact that will later be used to deploy the web app to Azure, and the other for publishing the ARM templates which are needed to provision the infrastructure in the release definition.

The build script in the powershell task does the following: restore, build, test, publish, and produce an MSDeploy zip package. 4.2

The Publish test results task is configured as in the figure. 4.3

Finally, in the publish tasks we publish the deployment artifacts of the application executable and the templates for the infrastructure deployment.

Preferred way to configure the inputs in the tasks is through variables because in

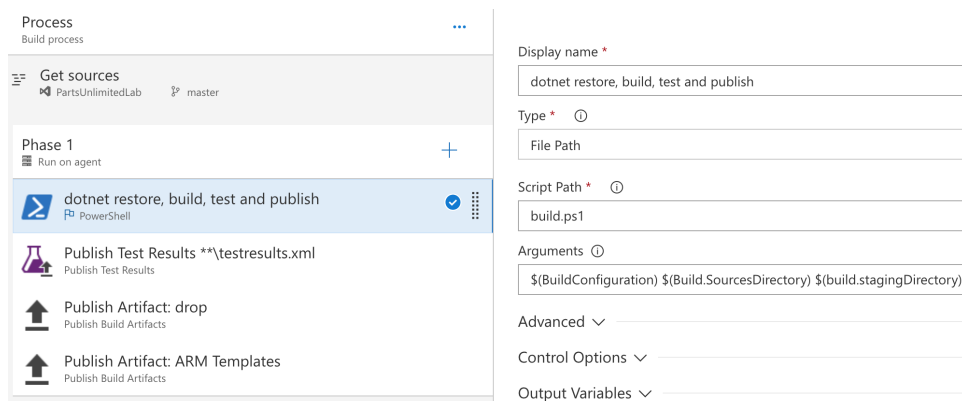


Figure 4.2: Configuration of the PowerShell task for the Continuous Integration Pipeline

this way one can keep track and change the configurations in one common place and reuse common configurations. After this is done in the Triggers tab one should check the Enable continuous integration (CI) option and set it to trigger on the master branch of the repository.

To check if the setup of the pipeline is successful one can make a new commit and see if the build succeeds.

Continuous Deployment

Continuous deployment, which typically occurs after continuous integration (CI), triggers a deployment by using the artifacts of the tested build to different environments. The goal is to enable the developers just to develop and as soon as they commit the code, that code is built, tested, packaged, and deployed into an environment.[18] In order to enable Continuous Integration pipeline, few things need to be done:

- Create a Service Endpoint from VSTS to an Azure Account
- Create a Release Pipeline for the Parts Unlimited Website
- Trigger a Release

To enable our pipeline to interact with Microsoft Azure, a service endpoint needs to be created in our VSTS account which includes the authentication information required to deploy to Azure. There are more ways to create Service Endpoint, also called Service Principal.

After successfully setting up the Service Endpoint, one needs to create the release definition that will deploy the build artifacts to different environments. An Environment is simply a logical grouping of tasks - it may or may not correspond

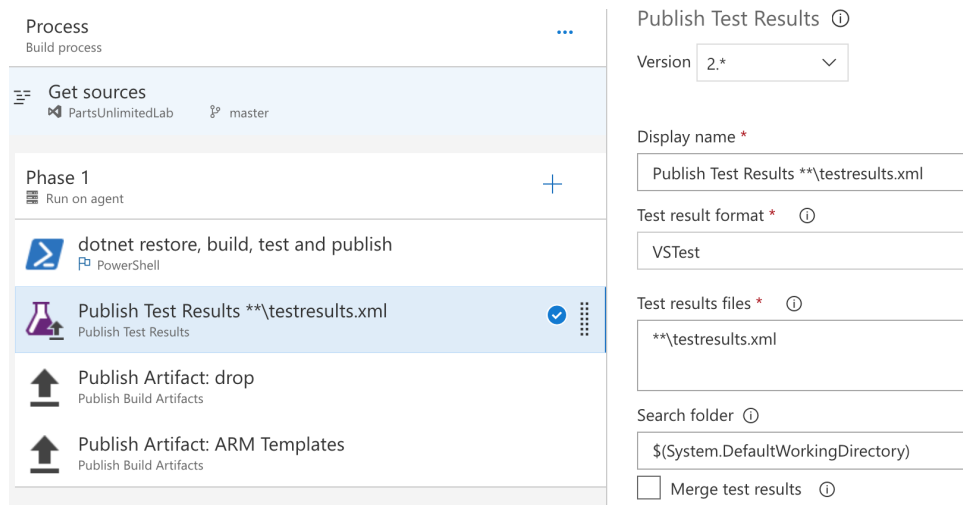


Figure 4.3: Configuration of the Publish Test Results task for the Continuous Integration Build Pipeline

to a set of machines. In the Release Definition 3 environments are created, Dev, Staging and Production corresponding to the deployment slots in the Azure Web App defined in the ARM template. The ARM Template will be invoked during the deployment in the Dev Environment before deploying the website to Dev. It will not be necessary to run any infrastructure tasks during Staging or Production deployments in this case. [18]

The first thing that needs to be done is to connect the release definition to build artifact from the build definition that was created before. 4.4

Next 3 environments are defined: Dev, Staging and Production. The Dev environment consists of two tasks, one to create the needed infrastructure in Azure from the ARM templates and another task for deploying to Azure Web App.

In the Dev environment a Azure Resource Group Deployment task is added first. This ensures that the infrastructure needed for the web app deployment is created in Azure. The most important things to configure are to set Create or Update Resource Group Action, set \$(ResourceGroupName) variable for the Resource Group Name, for Template the FullEnvironmentSetupMerged.json file is chosen from the ARMTemplates build artifacts, and for Template Parameters the FullEnvironmentSetupMerged.param.json file is selected from the ARMTemplates build artifacts. Finally, one needs to override the template parameters and have all variables stored in the Variables section in the project in VSTS. To do this one needs to insert the following in the Override Template Parameters field:

4.5

Next, in the Variables tab, one needs to set the variables used in the task for creating the infrastructure in Azure. Next, one needs to add the task Azure App

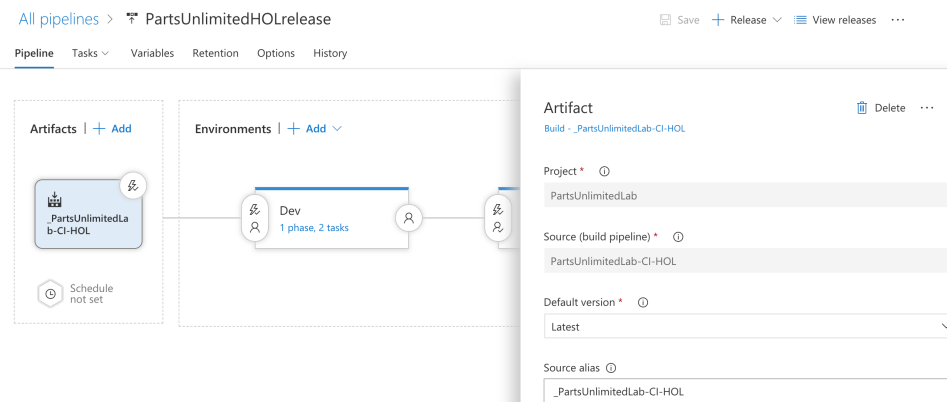


Figure 4.4: Configuration of Artifacts source for the Continuous Deployment Release Pipeline

Service Deploy in order to deploy the web app to Azure on the infrastructure created by the previous task. For App Type, Web App is chosen, for App Service Name the previously defined variable \$(WebsiteName) is set, the Deploy to Slot box is checked, for Resource group name the variable \$(ResourceGroupName) is set, for Slot Dev is entered, for Package or Folder one needs to navigate to the drop artifact from the build and select partsunlimited.zip. Moreover, the box for Take App Offline in Additional Deployment Options is checked, which stops the website for deployment period and takes it back online afterwards. All the other fields are left at their default values.4.6

In the Staging and Production environments only the task Azure App Service Deploy to deploy the app in the other slots is needed by selectin the respective deployment slots. Another difference is that pre-deployment and post-deployment approvals are set for the Staging and Production environment in order to have a manual approver who double checks the deployments in the Dev, Staging and Production Environments in order to avoid any potential issues. This can be set up by clicking the lightning bolt and user icon and enable the Pre-deployment approvals, and set a user who is responsible for this. In the same way one can set post-deployment approvals by selecting a user responsible for post-deployment approval. 4.7

Finally, to test the whole pipeline one can create a new commit to the code, trigger a build, which will trigger deployment to the Dev Environment, which if successful will create a trigger to send a notification email for the approver to approve the deployments in Staging and later in Production Environments.

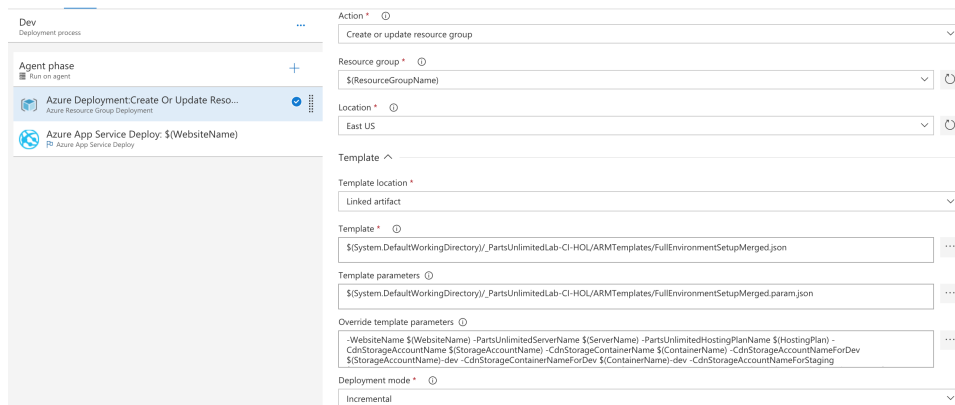


Figure 4.5: Configuration of the task for creation of the needed infrastructure in Azure for the Continuous Deployment Release Pipeline

4.2 Infrastructure as a Service

This section describes the use of Microsoft Azure IaaS offering. The use of Infrastructure as Code(IaC) paradigm is explored, which is a practice that enables the automation and validation of the creation and teardown of infrastructure, such as networks and virtual machines, to help deliver secure, stable application hosting platforms. [21] This is demonstrated through use of:

- Azure Automation Runbook deployments
- Azure Automation with DSC (Desired State Configuration)
- Deployment of Azure resources with ARM templates

Microsoft Azure has been built to support Infrastructure as Code principles. This is done with the use of Resource Manager which is the central orchestrator which works with Azure Resource Providers such as Azure Compute, Azure Storage, or Azure Network which are responsible for creating the resources orchestrated by the Resource Manager. This is all done through API (Application Programming Interface) calls as every Azure component exposes an API. The APIs can also be invoked cross platform Azure Command Line Interface or Azure PowerShell. After the creation of the resources deeper integration is done through Azure Automation and Desired State Configuration (DSC). [21] Infrastructure as Code can also be used for deploying PaaS services, as described in section 3.1 which uses ARM templates for provisioning the Web App environment. 4.1

In the following subsections, a description is made for how to create and configure 2 virtual machines (VMs) behind a load balancer and the associated resources required by the VMs and the load balancer.

The screenshot shows the configuration page for the 'Azure App Service Deploy' task in an Azure DevOps pipeline. The task is part of a pipeline named 'PartsUnlimitedHOLrelease'. The configuration fields are as follows:

- Azure subscription:** Free Trial (f4db7015-2635-485f-ade3-9b800f26d0f3)
- App type:** Web App
- App Service name:** \${WebsiteName}
- Deploy to slot:** ☒ (checked)
- Resource group:** \${ResourceGroupName}
- Slot:** Dev
- Virtual application:** (empty)
- Package or folder:** \${System.DefaultWorkingDirectory}/_PartsUnlimitedLab-CI-HOL/drop/partsunlimited.zip
- File Transforms & Variable Substitution Options:** (expanded, showing 'Take App Offline' checked and 'Publish using Web Deploy' unchecked)

Figure 4.6: Configuration of the task for deployment of the web app in Azure for the Continuous Deployment Release Pipeline

4.2.1 Azure Automation Runbook Deployment

Azure Automation is an Azure service that provides a way for users to automate the manual, long-running, error-prone, and frequently repeated tasks that are commonly performed in a cloud and enterprise environment. [21]

In order to use Azure Automation, one must have an Automation Account which is the container where automation artifacts are stored. This can be done through the Azure Portal.

After this is done one needs to configure the automation assets. First, an upgrade the Azure Modules is needed, which are part of the Automation account, then one needs to browse the modules gallery and import the AzureRM.Network module. Then, one can configure the variables which are persistent values that are available to all runbooks and DSC configurations that can also be encrypted. The variables are configured for the virtual machines, resource group name, user name, user password, and location which will be later used by the runbook. 4.8

Next, one needs to create or upload the existing runbook .ps1 file for provisioning the environment from the GitHub page of PartsUnlimited . After doing this, one can start the runbook which provisions the needed resources. 4.9

4.2.2 Azure Automation with DSC (Desired State Configuration)

One of the principles of cloud applications development is a strict separation of configuration from code as described in the 12 factor app which has become one

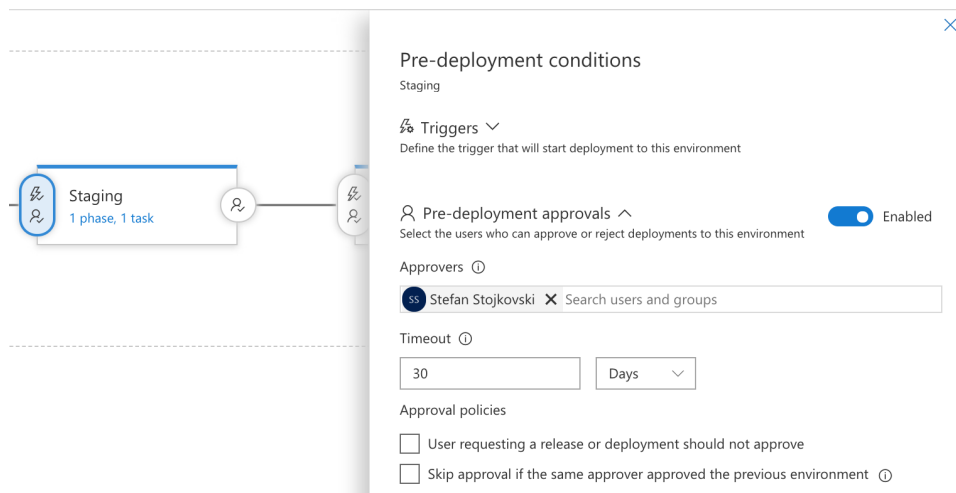


Figure 4.7: Configuration of pre-deployment approvals for the task for deployment of the web app Staging Environment in Azure for the Continuous Deployment Release Pipeline

of the most common guidelines for cloud app development principles. [37] What is described in this subsection is the Microsoft Azure approach to enable this principle of configuration as code, which is called Desired State Configuration (DSC). With DSC, one only needs to describe the desired state of the environment with a simple declarative syntax that has been added into the PowerShell language and then distribute it to each of the target nodes in the environment. This way the configuration drift is solved by constant checks of the current configuration with the desired configuration declared in DSC code. DSC provides idempotency, enabling to reach the desired state by applying the whole configuration, regardless of the starting state.[21]

Automation DSC in Azure has a simple workflow that consists of:

1. Create a DSC script
2. Upload it to an Automation Account in Azure and compile the script into a Managed Object Format (MOF) file.
3. Define the nodes that will use the configuration

In our example, a configuration is done so the VMs have IIS (Internet Information Services) enabled on them. In order to do this a simple DSC script is created. Then, upload of the script to the Automation Account is done, under DSC configurations blade using the Azure Portal. After doing this, next one needs to compile the script into MOF file through the Azure Portal. 4.10

The final step needed to be done is to connect the virtual machines to the auto-

+ Add a variable
Refresh

NAME	TYPE	VALUE	LAST MODIFIED
Location	String	eastus	6/6/2018, 4:26 PM
Password	Unknown (encrypted)	*****	6/6/2018, 4:26 PM
ResourceGroupName	String	rg1	6/6/2018, 4:25 PM
UserName	String	Student	6/6/2018, 4:26 PM
VM1Name	String	vm1	6/6/2018, 4:24 PM
VM2Name	String	vm2	6/6/2018, 4:24 PM

Figure 4.8: Configuration of variable assets needed to deploy the desired infrastructure

mation account and specify the configuration settings for each. After this is done the VMs pull the configuration settings from the Automation Account, apply them, and regularly check to compare the current state with the desired state and report the compliance. 4.11

In order to verify if the configuration has been applied, one can retrieve the IP address of the load balancer 4.12 from the Azure Portal and check if IIS has already been installed. 4.13

4.2.3 Deployment of Azure resources with ARM templates

In this subsection, it is described how the teams can use ARM templates from different sources and with different deployment options, to deploy and configure virtual machines and infrastructure to Azure. This helps the DevOps teams to get familiar with ARM templates and choose the preferred option for their use case.

The following ways to deploy an ARM template are described:

1. Deploy a QuickStart ARM Template from GitHub
2. Generate an ARM template based on an existing resource group via the Portal
3. Deploy a template using Powershell that removes all resources in a resource group
4. Edit and Deploy template via the Azure Portal
5. Deploy ARM Templates using Azure CLI 2.0

	rg1-avset1	Availability set
	rg1lb	Load balancer
	rg1lb-pip1	Public IP address
	rg1vm134433854	Storage account
	rg1vm1-nic1	Network interface
	rg1vm1-pip1	Public IP address
	rg1vm251196646	Storage account
	rg1vm2-nic1	Network interface
	rg1vm2-pip1	Public IP address
	rg1-vnet1	Virtual network
	vm1	Virtual machine
	BGInfo (vm1/BGInfo)	Microsoft.Compute/virtualMachines/extensi...
	vm2	Virtual machine
	BGInfo (vm2/BGInfo)	Microsoft.Compute/virtualMachines/extensi...

Figure 4.9: Provisioned resources after automated deployment of the runbook. The figure shows 2 virtual machines, 1 load balancer and other required resources

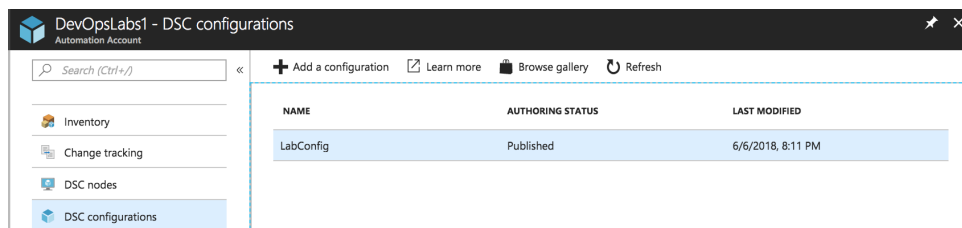


Figure 4.10: Compiled and published DSC script in Automation Account ready to be used by VMs

Deploy a QuickStart ARM Template from GitHub

One of the ways to start and deploy ARM template is to simply start exploring the ready made QuickStart templates. Once a desired template is chosen, one can click the button Deploy to Azure and fill in the required parameters according to the resource that needs to be created. For this one needs an active Azure subscription, and one can get more information about the other required parameters by reading the README files provided for each template. In order to verify if the deployment is successful, one can go to the portal and check if the desired resources have been created.

DSC nodes	Status	Node configuration name	VM DSC extension version > 2.70
Search nodes...	7 selected	2 selected	2 selected

NAME	STATUS	NODE CONFIGURATION	LAST SEEN	VM DSC EXTENSION VERSION > 2.70
vm1	✓ Compliant	LabConfig.WebServer	6/6/2018, 8:36 PM	Yes
vm2	✓ Compliant	LabConfig.WebServer	6/6/2018, 8:39 PM	Yes

Figure 4.11: VMs that have applied the Desired State Configuration and with a compliant state

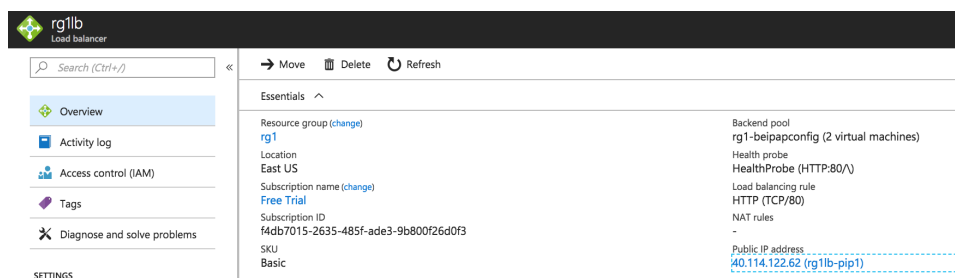


Figure 4.12: The IP address of the Load Balancer

Generate an ARM template based on an existing resource group via the Portal

For every resource that is created, there is the possibility to generate an ARM template out of it. In order to do this, one should navigate to the targeted resource and find the Automation Script blade. From there an ARM template is generated which can either be downloaded or added to library and stored in the Azure account for future use. The library can be found in the Templates service on the portal.

Deploy a template using Powershell

Another way of deploying an ARM template is through PowerShell. First, one needs to create a template file. Then run the following script in PowerShell on the PC:

```
#cmdlet to sign in to you azure subscription
Add-AzureRmAccount
```

```
#define a variable to point to your template file. You can run
#the command in either line 9 or line 12. If you placed it
#in 'mydocuments' folder, or some other system defined folder you
#can use the 'getfolderpath' method as below,
```

```
$template = [environment]::getfolderpath( mydocuments )
+"\\EmptyTemplate.json"
```

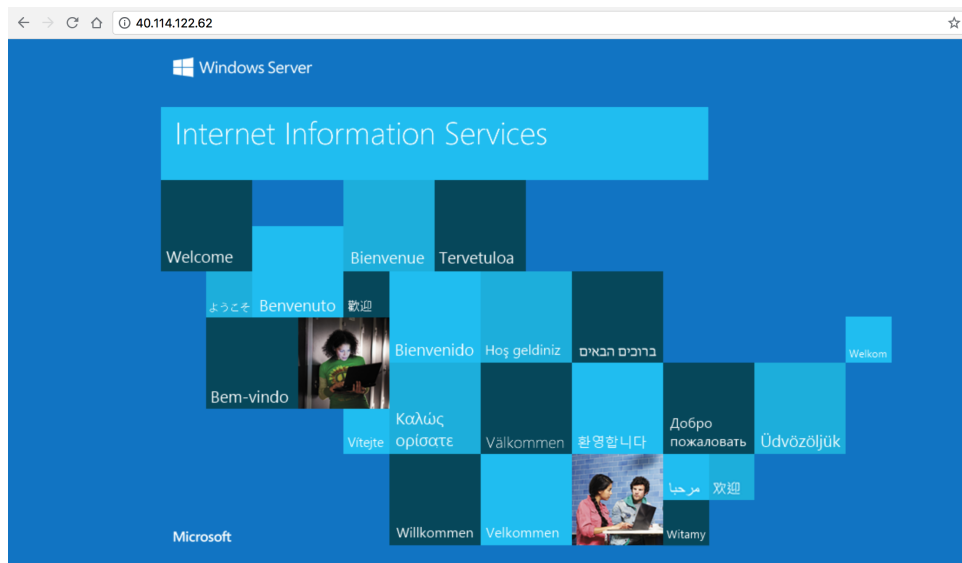


Figure 4.13: The state after successfully installing IIS, notice the IP address is same with the IP address of the Load Balancer

```
# Command to deploy the ARM template
# Replace YOUR-RG-NAME with the name of your resource group
New-AzureRmResourceGroupDeployment -ResourceGroupName 'YOUR-RG-NAME'
-Mode Complete -TemplateFile $template -Force
```

During the script execution there is a prompt to enter Azure credentials, and after the execution the outcome is that the resource group is still in place, but the resources are removed.

Edit and Deploy template via the Azure Portal

Another option for deploying ARM templates is via the Azure Portal by using the Templates service, which contains the templates library associated with the profile. From here one can view, edit and deploy your templates. When certain template is chosen for deployment, a custom blade appears with the required parameters defined in the template.

Deploy ARM Templates using Azure CLI 2.0

The advantage of using the Azure CLI is that it can be used by any platform (Windows, Linux or macOS). There are two ways to deploy an ARM template with the CLI:

- using remote source

- from local source using separate parameters file

4.3 Testing

In this section it is described how the web app can be tested in production by using deployment slots and writing automated UI Tests using Selenium. Automated testing helps developers run tests often, to verify the software quality continuously. This has various benefits such as obtaining early feedback, shortening the release cycles, reducing cost, measuring quality continuously, avoiding regressions, shipping high-quality product, and finally, making their customers happy. [20] Unit testing in the CI/CD pipeline is demonstrated for which there were tests already present in the project code and are executed as a step for the build. In this section it is described how to create automated UI tests by using Selenium and also adding a deployment slot to an Azure Website for deploying new features or enhancements to the website, as well as adding routing rules to direct traffic to the test site with PowerShell.

4.3.1 Testing in production

In this subsection it is described how one can test new feature and expose it gradually to users, so one can make a better decision if one wants to release the feature to all customers. To demonstrate this the PartsUnlimited app is used which the CI/CD deployment is described in 4.1. To do this, one first needs to add a new deployment slot in our Web App Service resource. One can do this by navigating to the App Service in our resource group, selecting the Deployment Slots tile and adding the new slot.

The change that is made and tested is a change in the interval of the jumbotron-carousel which switches the images on the home page by increasing it from 5 to 20 seconds. We do this in the Index.cshtml file in the Home folder under the Views folder in the PartsUnlimited solution. 4.14

After this change the application can be published directly from Visual Studio by creating deployment profile for the new slot, then publishing it to Azure App Service.

After the successful deployment of the app with the new feature to the Test slot, one wants to change the traffic rules to Direct Traffic to the Test Site with the help of PowerShell. [20]

4.3.2 Coded UI Test using Selenium in Visual Studio

In this subsection it is demonstrated how to use automated tests conducted through the user interface to test components together in scenarios. Coded UI tests typically drive the application through its user interface (UI) and include functional testing of the UI controls. Coded UI test was created by using The Selenium automation tools which speaks a language known as Selenese. Rather than issuing Selenese



```

1 @model PartsUnlimited.ViewModels.HomeViewModel
2
3 @using PartsUnlimited
4
5 @{
6     ViewBag.Title = "Home Page";
7 }
8
9 @section scripts {
10     <script>
11         $(function() {
12             $('.carousel').carousel()
13         })
14     </script>
15 }
16
17 @section prebody {
18     <div class="hero-display">
19         <div id="jumbotron-carousel" class="carousel slide" data-ride="carousel" data-interval="20000">
20             <!-- Indicators -->
21             <ol class="carousel-indicators">
22                 <li data-target="#jumbotron-carousel" data-slide-to="0" class="active"></li>
23                 <li data-target="#jumbotron-carousel" data-slide-to="1"></li>
24                 <li data-target="#jumbotron-carousel" data-slide-to="2"></li>
25             </ol>
26         </div>
27     </div>
28 }

```

Figure 4.14: Changing the interval of the carousel feature from 5 to 20 seconds

directly to the different types of browsers, web drivers are used to perform the translation.

The Selenium tests are written in Visual Studio 2017. One needs to create a Unit Test Project and then we need to install several NuGet packages so one can be able to use Selenium:

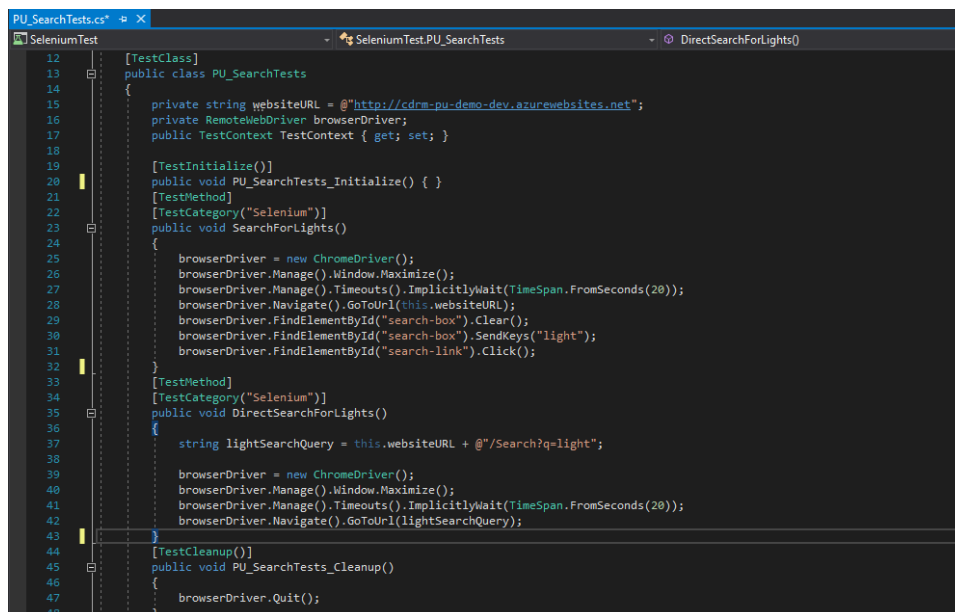
- Selenium.WebDriver
- Selenium.WebDriver.ChromeDriver
- Selenium.WebDriver.IEDriver
- Selenium.Firefox.WebDriver
- Selenium.WebDriver.PhantomJS.Xplatform

Then, one needs to write the code for the coded UI test. What Selenium is doing is finding the HTML elements and doing different actions with them. The easiest way to find out the IDs of the UI elements subject to test is by using developer tools on our browser, hover over the targeted element and the ID is displayed. 4.15

To verify the test, one needs to build the solution, select the tests to run, and click Run Selected Tests to execute the tests.

4.4 Application monitoring

In this section it is described how one can use out-of-box telemetry for Application Insights to gain further insight into how users are behaving towards their web application, and drill down into performance monitoring data. After release of the code, one wants to monitor its quality. No matter how much confidence one has



```

12 [TestClass]
13 public class PU_SearchTests
14 {
15     private string websiteURL = @"http://cdm-pu-demo.dev.azurewebsites.net";
16     private RemoteWebDriver browserDriver;
17     public TestContext TestContext { get; set; }
18
19     [TestInitialize()]
20     public void PU_SearchTests_Initialize() { }
21
22     [TestMethod]
23     [TestCategory("Selenium")]
24     public void SearchForLights()
25     {
26         browserDriver = new ChromeDriver();
27         browserDriver.Manage().Window.Maximize();
28         browserDriver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(20));
29         browserDriver.Navigate().GoToUrl(this.websiteURL);
30         browserDriver.FindElementById("search-box").Clear();
31         browserDriver.FindElementById("search-box").SendKeys("light");
32         browserDriver.FindElementById("search-link").Click();
33     }
34
35     [TestMethod]
36     [TestCategory("Selenium")]
37     public void DirectSearchForLights()
38     {
39         string lightSearchQuery = this.websiteURL + @"/Search?q=light";
40
41         browserDriver = new ChromeDriver();
42         browserDriver.Manage().Window.Maximize();
43         browserDriver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(20));
44         browserDriver.Navigate().GoToUrl(lightSearchQuery);
45     }
46
47     [TestCleanup()]
48     public void PU_SearchTests_Cleanup()
49     {
50         browserDriver.Quit();
51     }
52 }

```

Figure 4.15: The code for the Selenium Test which is a coded UI test

in the code, production is the best place to monitor the performance of their application since test data does not always represent the production data with high accuracy. Moreover, the system loads are different, the user base (operating systems, browsers, or local software) and network conditions also tend to have great variance and contain factors out of our control. This is why one should implement a strong feedback loop that can help them deliver better value to their users in all aspects such as performance, exceptions, usage, user behavior, and user experience. [14]

In the subsequent subsection it is described how one can take advantage of the Azure Application Insights to monitor user telemetry, performance and usage analysis and this chapter is finished with description on how one can implement feature flags.

4.4.1 User Telemetry and Perf Monitoring with App Insights

In this subsection it is described how, by using the out-of-box telemetry for Application Insights, the teams are able to find out how people use the application and gain insights into the goals that they need to achieve.

To do this, one first needs to set up Azure Application Insights for the PartsUnlimited website backend and frontend which is documented here.

After successfully enabling Application Insights, some scripts were created to generate requests on the website which can be found in the appendix. Finally, one can navigate to the Application Insights resource in the Azure Portal and monitor the

different metrics that it offers:

- Server response time 4.16
- Page View load time
- Number of server requests over time 4.17
- Ajax call performance

Each of these metrics can be explored even deeper to investigate possible issues and to react fast to fix them.

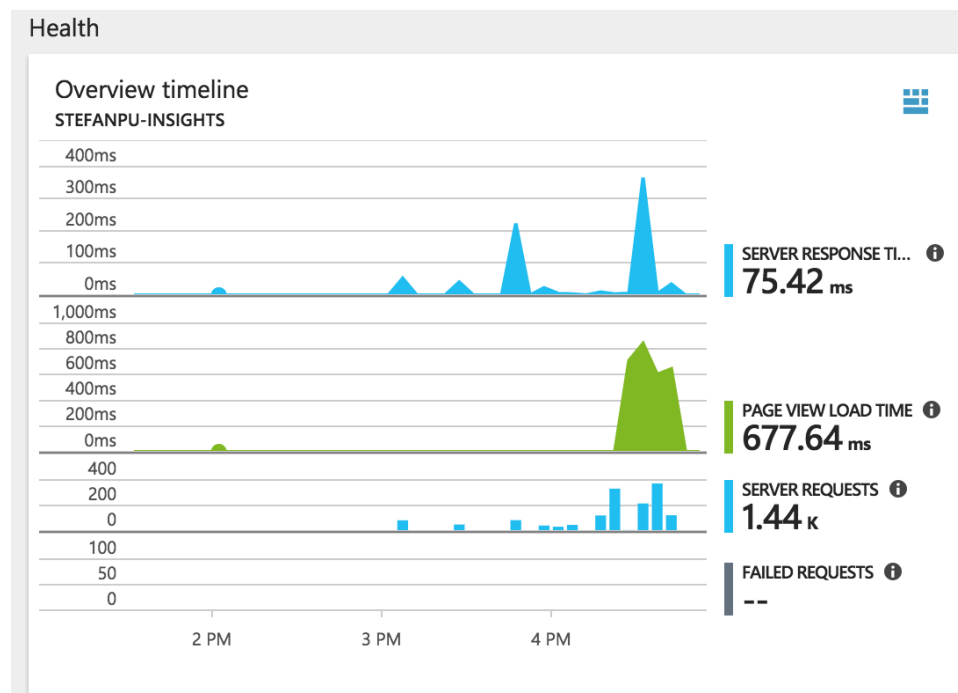


Figure 4.16: Azure Application Insights overview of the performance metrics of the web application PartsUnlimited

4.4.2 Creating Custom Telemetry Events

In this subsection it is described how one can create custom telemetry events both on client side and server side. This types of events can help them understand better how their users are interacting with the app and what kind of issues they are facing. To do this one needs to do some changes in the code and then query the logs from this events to gain deeper insights.

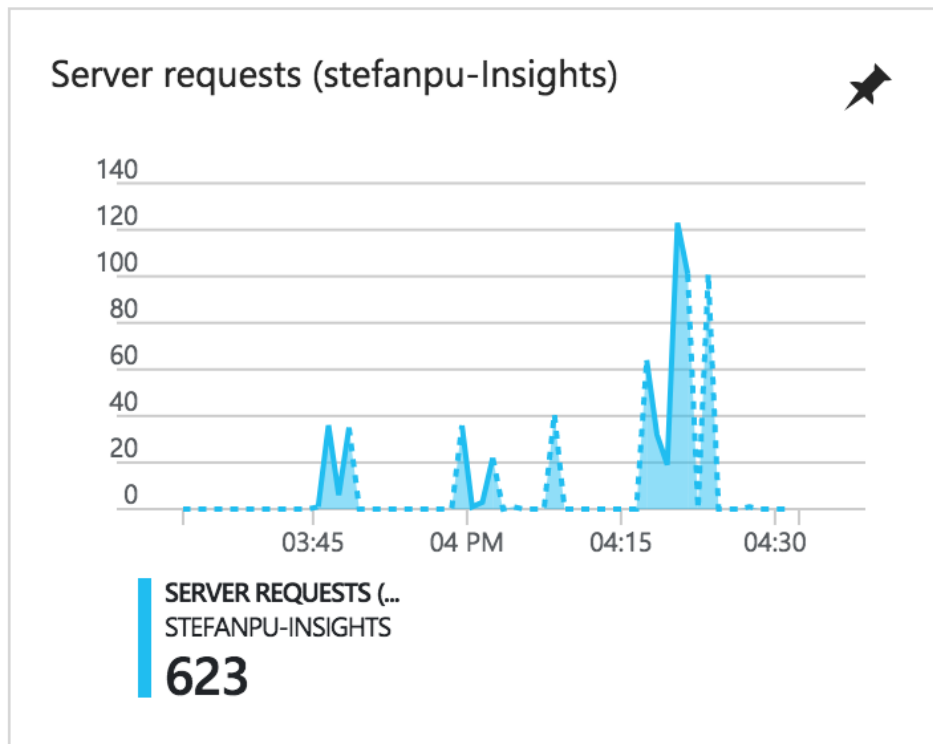


Figure 4.17: Azure Application Insights overview of the number of server requests for the web application PartsUnlimited

After data has been generated for the custom events, one can write custom queries to view the performance metrics and data. This is done from the Azure Portal in the Analytics part of the Application Insights resource. One can check the events by simply typing `customEvents` in the query window. One can also write more complex queries like:

- Number of custom events in the last couple hours on a line chart 4.18
- Top 10 custom events of our application shown in a bar chart 4.19
- Any other relevant information that we can think of

4.4.3 Feature flag implementation

In this subsection it is described how to implement feature flags. Feature flags give us the ability to turn features on or off without deploying new code. This enables us to test our features in production, getting early feedback from subset of our users,

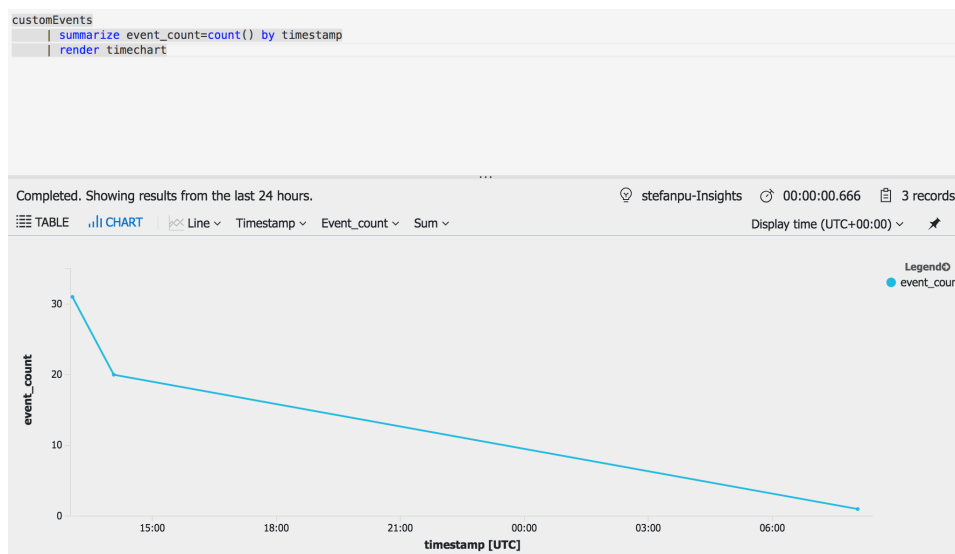


Figure 4.18: Azure Application Insights query for number of custom events in the last couple hours on a line chart

and incrementally enable it for everyone if it is successful. In our project we implement a simple feature flag for phone number validation.

To verify that the feature flag is working, one needs to rerun the website locally, log in with the credentials in the config.json file, navigate to the new page created under Profile, View Features and see the active feature flags displayed in a list which can be turned on or off. 4.20

Before turning the feature on, one needs to navigate to Profile, Manage Account, Add Phone Number and enter an invalid string and click Submit to trigger a validation error that only displays after the button is pressed. 4.21

After the feature is turned on, if one navigates to the same page, they can notice that the placeholder is different to the one before. 4.22

Finally when one types something incorrect, one sees a red border appearing around the input and if one clicks submit, a popup appears asking the user to enter the correct format. 4.23

4.5 Conclusions

This chapter addresses the challenge of how to raise awareness and encourage the affected staff from cloud adoption. This is done by demonstration of the capabilities of Microsoft Azure Cloud technology and Visual Studio Team Services to speed up development and operations processes. The purpose is to showcase how Cloud Computing technology integrates into the whole development cycle and speed up

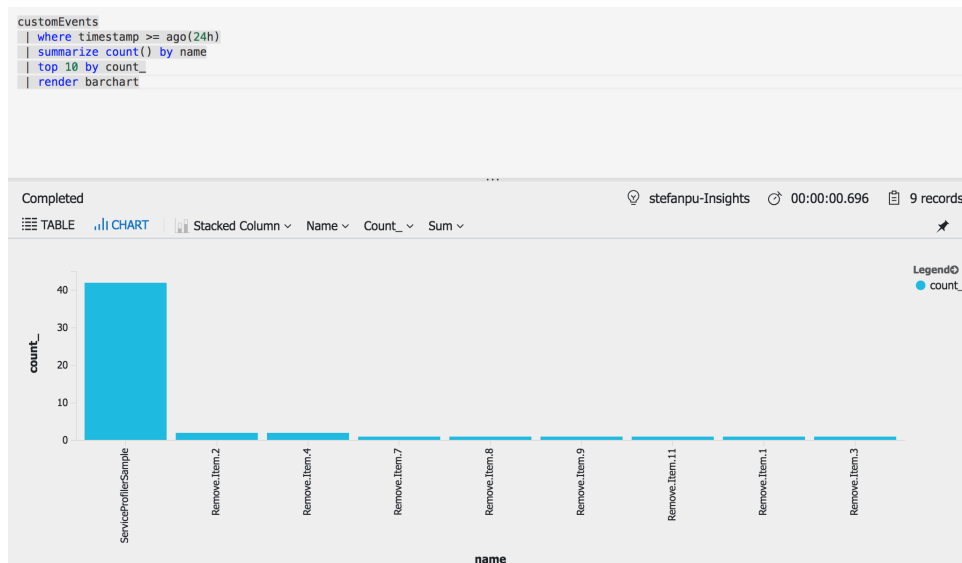


Figure 4.19: Azure Application Insights query for top 10 custom events of our application shown in a bar chart

delivery time. Moreover, it is demonstrated how many of the processes can be automated which makes the development process stable, predictable, repeatable. This also encourages experimentation since the risk in this way of developing is lower because changes can be reverted very easily and there are also many testing tools to test the outcome of the changes in a similar or same environment before it is deployed to production. Additionally, Microsoft Azure provides one platform for the whole lifecycle management from checking in the code, to deployment of infrastructure, publishing applications and later monitoring and management of the applications. This enables teams to be more independent and be in complete control of their products, since once they are on-boarded to the platform, everything further is self-service. By developing scenarios that fully demonstrate the capabilities of the Microsoft Azure platform, the potential customer teams can get better insight of all the benefits of the future change that they will face and be more encouraged to embrace the technology as early as possible.

However, the scenarios developed cover only a small fraction of the full capabilities of Microsoft Azure and are not a fit for all the workloads developed by the teams in Rabobank. The idea behind the presentation of the particular use cases is to familiarize the staff with the way of working, cloud development principles, and the change of mindset by getting full control and ownership of your product development, compared with the dependencies and shared responsibility with the on-premise datacenter team.

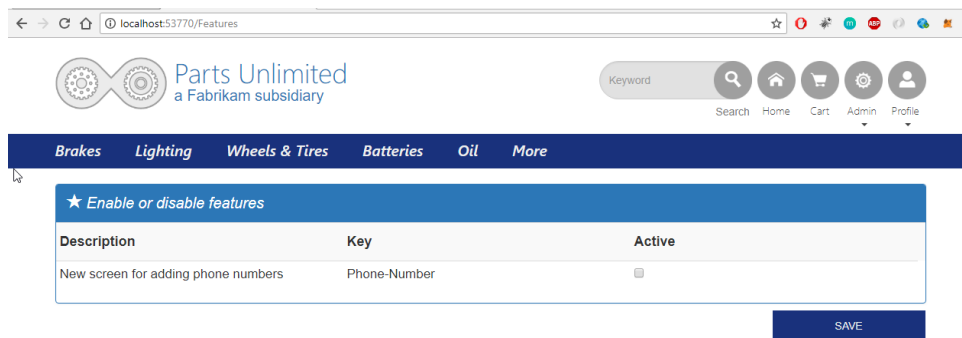


Figure 4.20: List of active feature flags displayed in the newly created view

The screenshot shows a form titled 'Phone Number' with the heading 'Add a phone number'. Below the heading is a red error message: 'The Phone Number field is not a valid phone number.' The input field contains the text 'this will error' and has a red border. To the right of the input field is a user icon. A blue 'SUBMIT' button is at the bottom.

Figure 4.21: Validation before the feature flag is turned on. It only triggers after the button is pressed

The screenshot shows the same 'Phone Number' form, but the input field now contains the placeholder text '555-555-5555' and has a clean white border. The 'SUBMIT' button remains at the bottom.

Figure 4.22: The look of the new placeholder after we enable the feature flag

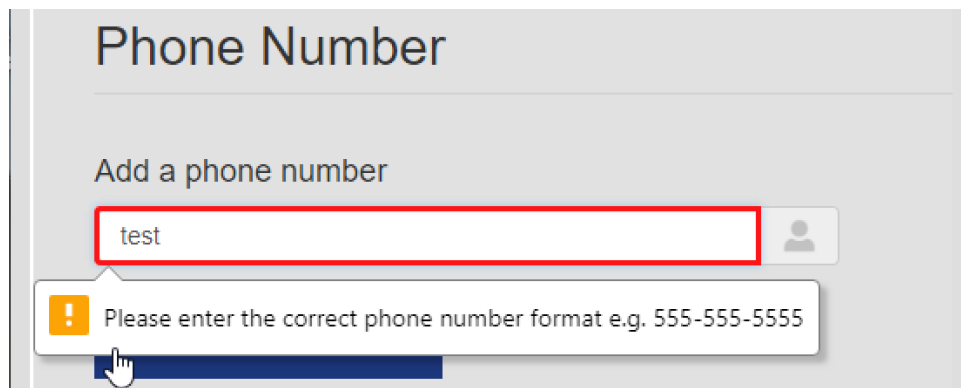


Figure 4.23: Validation after the feature flag is turned on. It triggers a popup asking the user to enter the correct format after the button is pressed

Chapter 5

Qualitative and quantitative comparison of an application application performance on various cloud services

This chapter describes the qualitative and quantitative comparison of the application deployed on various cloud services including:

- Windows Server Virtual Machine
- Azure Kubernetes Service
- Azure Application Service - Web Application

The motivation behind this comparison is to gain insight into the performance, cost, ease of deployment and life cycle management of an application deployed on different cloud services. This insight is used to aid the decision process for application migrations within Rabobank that would not be rearchitected and will be migrated in "lift and shift" style. The current Microsoft Azure offerings allow identical Microsoft Windows Server migration on a virtual machine which is IaaS offering and everything is managed by the end user. Another option is to use PaaS service and deploy the application on Azure Application Service and have the underlying application managed by Microsoft. The final option is to containerize the application and have it deployed on Azure Kubernetes Service which is the managed Kubernetes offering from Microsoft Azure. The benefits and drawbacks of each of these options are further discussed in this chapter.

5.1 Testing environment and conditions

The deployed application is designed to serve a load of 250 users. The testing was started with incremental increase of the user load and while testing with different values, it was concluded that user load of 300 users is stable and can be served as good reference point to compare the performance of the 3 different deployments. Moreover, to test the application in a more extreme conditions, performance tests were done with a user load of 1000 users to monitor the difference in the performance. The Windows Server was deployed on a virtual machine with 4 virtual cores and 8 GBs of RAM, with SSD (Solid state drive) storage device. The App Service Web app was deployed on a virtual machine with 4 virtual cores and 7 GBs of RAM, with SSD storage device. The Azure Kubernetes Service containers were deployed each on Azure Container instances with 1 virtual core and 1.5 GBs of RAM, so a similar infrastructure is used for fair comparison.

To get objective results each test was conducted in different times of the day: morning, afternoon and evening. Moreover tests were conducted both on days during the week and on days during the weekends. Each test was executed at least 3 times under each condition (ex. Day during week, morning, 300 users load). Moreover the tests were conducted in more than one day both, during the week and during the weekend. In the beginning, the tests under each condition were conducted up to 7 times, but it was concluded that there was no notable difference in the results if conducted 3 or 7 times, so it was decided to conduct each test under same conditions for 3 times. If the results between testings under same conditions were variable, then more tests were planned to be conducted, but this was not the case.

5.2 Windows Server Virtual Machine

Azure Virtual Machines (VM) is one of several types of on-demand, scalable computing resources that Azure offers. The typical choice for VM is when one needs more control over the computing environment compared to the other offers from Azure. [23]

5.2.1 Ease of deployment

The deployment of the application on Windows Server requires most effort, since the person responsible for this first needs to make sure that the proper elements are installed on the machine and configure and manage everything by themselves. The responsible person should also create all the virtual networks, network security groups, storage and OS versions.

5.2.2 Lifecycle management

In terms of life cycle management, again the Windows Server virtual machine requires most effort since this is Infrastructure as a Service offering. This means

that someone is responsible for updating the operating system with the latest software, rotation of the keys and everything else that needs to be configured. Some of these things can be automated, but the automation scripts still need to be created, managed and supervised by the staff. For application updates to be without downtime, one needs to have multiple virtual machines deployed behind a load balancer and make sure that the traffic is balanced correctly from the old to the newer version of the application. This is all manual work that slows down the delivery process and can be error prone.

5.2.3 Cost

To run the application on a virtual machine we need the following resources:

- Windows Server 2016 virtual machine with size DS2v2 which costs 183 dollars per month
- Public IP address which costs 2.628 dollars per month
- Virtual network which is charged 0.01 per GB, we assume 200GB transfer of data for 2 dollars
- Managed SSD OS disk 19.71 per month
- Network interface - free of charge
- Network Security Group - free of charge

The estimated total costs are 207.33 dollars per month.

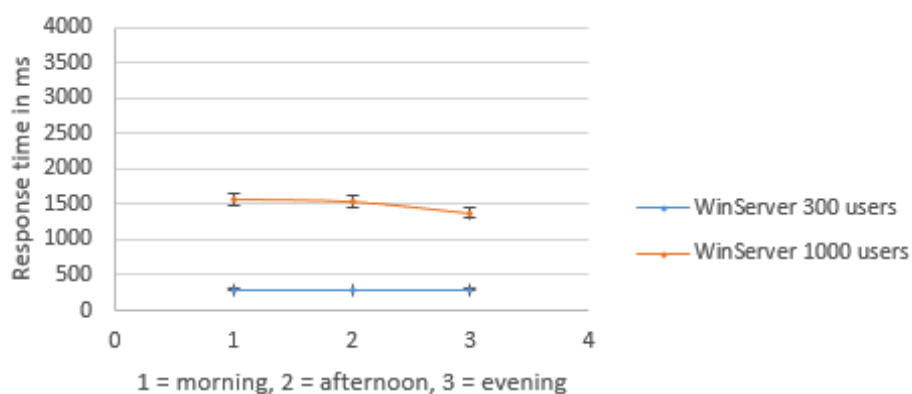
5.2.4 Performance

For this deployment a virtual machine with Windows Server 2016 with 4 virtual cores and 8 GBs of RAM was used, with SSD storage device. The application was deployed on IIS (Internet Information Services) server.

The application under the target load of 300 users displayed a stable performance between 270 and 300 ms response time during weekdays, and between 250 and 320 ms during the weekend days. On average it served around 350000 successful requests with single digit failed requests.

Under target load of 1000 users, the average response time increased to around 1500 ms. However the number of successful requests remained stable with around 350000 successful requests, but the number of failed requests increased significantly with around 100000 failed requests on average per each test. 5.1

Response time, Weekday measurement, 300 vs 1000 users. Windows Server VM



Successful requests. Weekday measurement. Windows Server VM. User load: 1000

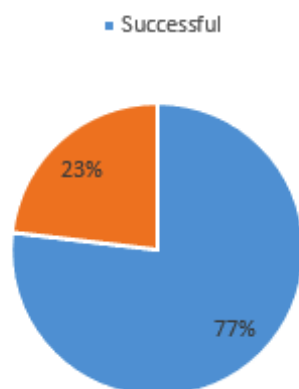


Figure 5.1: Performance comparison of Windows Server VM. User load of 300 users and 1000 users. Percentage of successful requests under user load of 1000 users.

5.3 Azure Kubernetes Service

Azure Kubernetes Service (AKS) makes it simple to deploy a managed Kubernetes cluster in Azure. It takes care of the complexity and operational overhead of managing Kubernetes. Azure handles critical tasks like health monitoring and maintenance. The Kubernetes masters are managed by Azure. As a managed Kubernetes service, AKS is free - you only pay for the agent nodes within your clusters, not for the masters. [17] Azure Container Instances is a serverless way to run a container in Azure, without having to manage any virtual machines and without having to adopt a higher-level service. [15]

5.3.1 Ease of deployment

Since Azure Kubernetes service is Platform as a Service offering, the development team does not have to manage the machines in the cluster or anything related to infrastructure components of the deployed cluster. The cluster is deployed once and then used as is out of the box. The only configuration that is done is managing the scaling policies. This can be done manually or by enabling autoscaling functionality based on a certain metric, such as CPU average utilization over certain period of time. Moreover the Kubernetes engine version update is also managed by Microsoft and does not cause any downtime for the cluster as it is done incrementally on the nodes that are part of the cluster.

5.3.2 Lifecycle management

The nodes and all the other infrastructure components related to the cluster are managed by Microsoft, so there is no need for manual interventions in this term. The team which uses the cluster is able to trigger update of the version of the Kubernetes engine and that is done incrementally without downtime. Moreover, when new application version is released everything can be automated in a CI/CD pipeline, from building the container to deployment on the cluster in a desired way (at once or incrementally).

5.3.3 Cost

To deploy an application on Azure Kubernetes service, the following resources are needed:

- AKS cluster - free of charge
- 2 to 4 container instances which are serverless instances which cost 178.85 dollars per month
- Load balancer - free of charge
- Managed SSD disk which costs 9.60 dollars per month

- One node machine which costs 14.60 dollars per month

This totals of 203,05 dollars per month

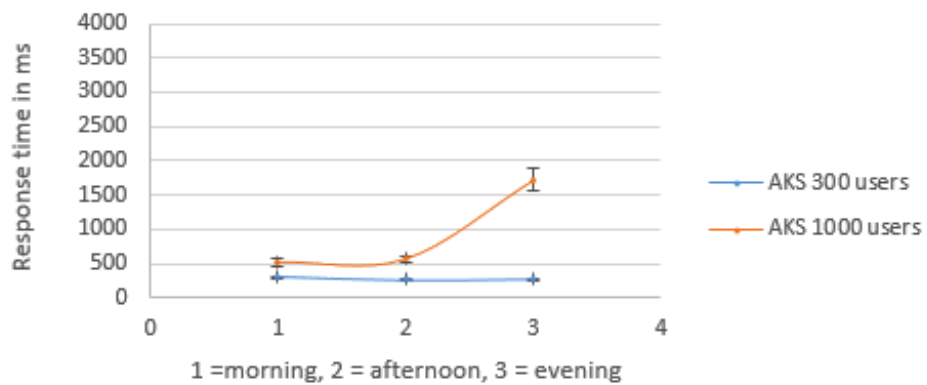
5.3.4 Performance

For the deployment on Azure Kubernetes Service, 4 Windows containers on the service Azure Container Instances were used, each with 1 virtual core and 1.5 GBs of RAM.

The application under the target load of 300 users had a response time of around 260 to 300 ms on average on weekdays. On weekends the performance was variable with 330 ms average in the morning, 230 ms average in the afternoon and 280 ms average in the evening. This variety may be explained due to the serverless architecture of Azure Container instances, and the different time it takes to spin up an instance of the container. On average it served around 400000 successful requests with single digit failed requests.

Under target load of 1000 users, the average response time increased to around 500 ms during the morning and afternoon on the weekdays and during the whole day in the weekend. The exception was one evening during the weekdays when the average response time increased to around 1700 ms on average. The number of successful requests increased with around 750000 successful requests, and the number of failed requests increased insignificantly with around 350 failed requests on average per each test, with exception of the evening of the weekday test with 6000 failed requests on average. 5.2 The reasons for this behavior are unknown, but one reason may be the noisy neighbor effect which is the effect when a co-tenant may monopolize bandwidth, CPU, disk IO, and other resources, and can negatively affect other users cloud performance. [32]

Response time 300 vs 1000 users, Weekday
measurement
Azure Kubernetes Service



Successful requests AKS, Weekday measurement.
User load: 1000



Figure 5.2: Performance comparison of Azure Kubernetes Service. User load of 300 users and 1000 users. Percentage of successful requests under user load of 1000 users.

5.4 Azure Application Service - Web Application

Azure App Service Web Apps (or just Web Apps) is a service for hosting web applications, REST APIs, and mobile back ends. One can deploy applications developed in their favorite language such as .NET, .NET Core, Java, Ruby, Node.js, PHP, or Python. The applications run and scale on Windows-based environments. Web Apps service adds the power of Azure such as security, load balancing, auto-scaling, and automated management. With App Service, you pay for the Azure compute resources you use. [26]

5.4.1 Ease of deployment

Since Azure Application Service is Platform as a Service offering, the development team does not have to manage the machines in the cluster or anything related to infrastructure components of the deployed cluster. The cluster is deployed once and then used as is out of the box. The only configuration that is done is managing the scaling policies. This can be done manually or by enabling autoscaling functionality based on a certain metric, such as CPU average utilization over certain period of time.

5.4.2 Lifecycle management

The nodes and all the other infrastructure components related to the cluster are managed by Microsoft, so there is no need for manual interventions in this term. Moreover, when new application version is released everything can be automated in a CI/CD pipeline, from building the code to publishing it to the Web App Service. Web App Service also offers deployment slots which serve as testing environments with their own hostnames. After publishing the application to a deployment slot and you are sure that everything is working correctly, it is easy to swap it with production. This eliminates downtime when you deploy your app.

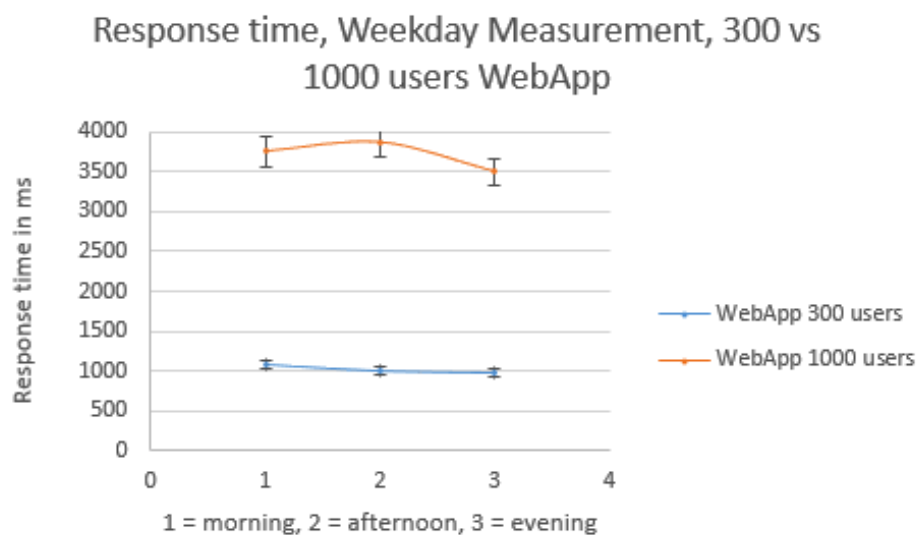
5.4.3 Cost

To deploy the application to Azure App Service we only need App Service Plan and an App Service. The cost for Azure Application Service - Web Application is 292.00 dollars per month.

5.4.4 Performance

For this deployment a Web App running on Windows was used, with 1 instance that has 4 virtual cores and 7 GBs of RAM , with SSD storage device. The application under the target load of 300 users displayed a stable performance between 950 and 1000 ms response time both during weekdays and weekend days. On average it served around 200000 successful requests with around 700 failed requests.

Under target load of 1000 users, the average response time increased to around 3700 ms. The number of successful requests decreased to around 170000 successful requests, but the number of failed requests increased significantly with around 18000 failed requests on average per each test. 5.3



Successful requests. Weekday measurement.
Web App. User load: 1000

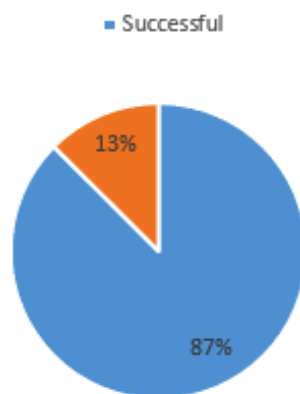


Figure 5.3: Performance comparison of Azure App Service: Web App. User load of 300 users and 1000 users. Percentage of successful requests under user load of 1000 users.

5.5 Overall performance comparison

In this section side to side analysis is done on all the services in scope of this chapter. Performance is described under load of 300 users and 1000 users under comparable underlying infrastructure. Finally some tests were performed with different underlying infrastructure for experimentation purposes.

5.5.1 Weekday vs. weekend days testing, different times of day testing

The testing was done both on days during the week and during the weekends. There were only 2 notable differences in the performance, and both were connected to the performance of Azure Kubernetes Service. The response time on weekend morning was around 200ms worse than the rest of the day on the weekend and the response time on one evening during the weekdays was around 1100 ms worse than the measurements taken during different times of that day. These performance varieties were not noticed when testing was conducted in the same time periods under same conditions on other days, but it is left as a reference that Azure Kubernetes service may have more unpredictable behavior. In the rest of the chapter, performance is described during the weekday measurements because of more tests conducted during the weekdays, and no other notable differences compared with weekend days tests. 5.4

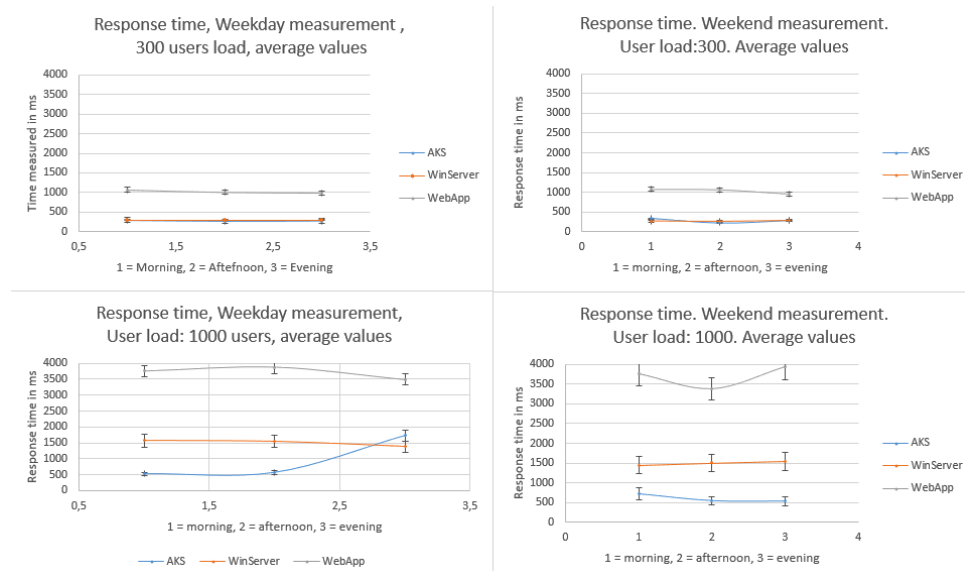


Figure 5.4: Comparison of number of successful requests under user load of 300 and 1000 users done in the weekend vs weekday.

5.5.2 Performance under load of 300 users

The deployed application is designed to serve a load of around 250 users. The load testing was started with incremental increase of the user load and while testing with different values, it was concluded that user load of 300 users is stable, the infrastructure is fully utilized and can be served as good reference point to compare the performance of the 3 different deployments.

The application deployed on Windows Server Virtual Machine and on Azure Kubernetes service displayed very similar results in terms of response time with response times between 250 and 300 ms. The application deployed on App Service : Web Apps demonstrated much worse performance with response times around 1000ms. 5.5

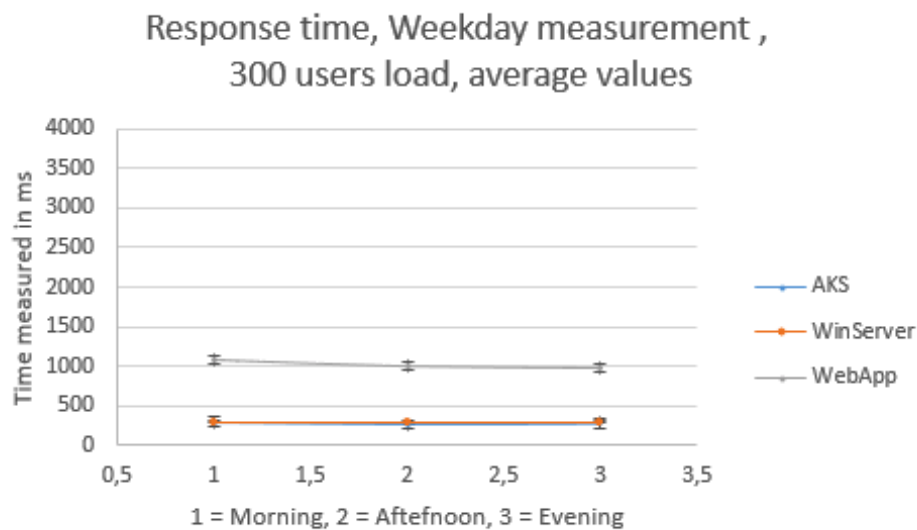


Figure 5.5: Response time comparison under user load of 300 users.

When it comes to number of successful requests the best performance is achieved with AKS with around 400000 successful requests on average, followed by Windows Server with around 350000 successful requests on average. The worst performance, was again demonstrated by the Web App service with only around 200000 successful requests or 50% worse performance than AKS. 5.6

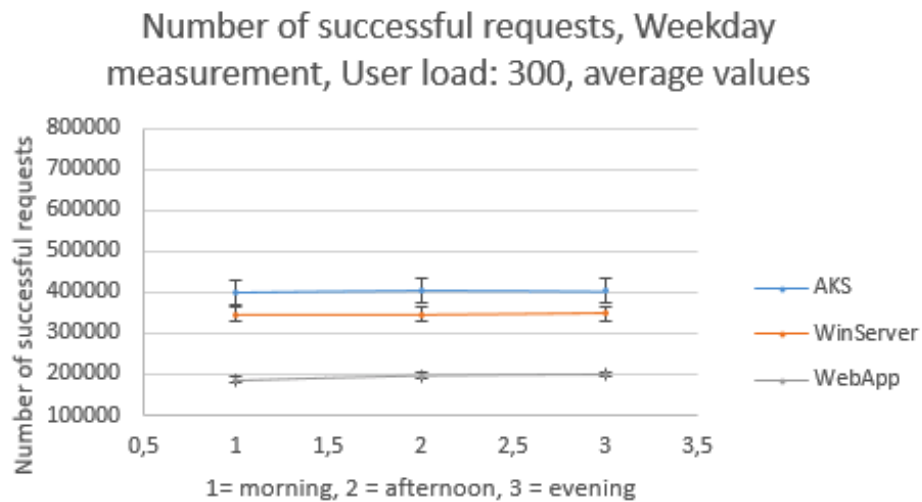


Figure 5.6: Comparison of number of successful requests under user load of 300 users.

The number of failed requests under the load of 300 users was insignificant as less than 1 percent of the requests failed on each deployment type.

The results demonstrate that while the response times are similar with AKS and Windows Server VM, the AKS deployment was able to serve 14% (50k) more requests than the Windows Server VM. The worst performance measured both in response time and number of successful requests was demonstrated by the WebApp with 50% less requests served compared to AKS and 3 times worse response times than AKS and Windows Server VM.

5.5.3 Performance under load of 1000 users

This test was conducted to serve as a reference how do the deployments on different services behave under increased load of 1000 users.

The application deployed on AKS demonstrated best performance in terms of response time with response times between 500 and 550 ms. During the testing there were 2 deployments that varied significantly and this was one evening during the weekdays with average response times of 1700 ms and one morning during the weekend with response times of 700 ms. The reasons for this much variation are not clear since the testing conditions were the same in all of the conducted tests. The application deployed on App Service : Web Apps demonstrated much worse performance with response times around 3600 ms. 5.7

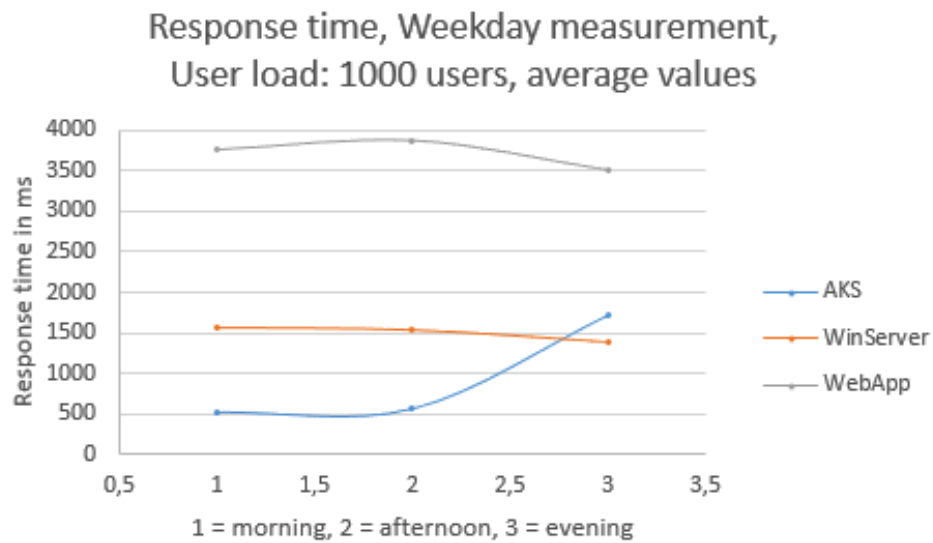


Figure 5.7: Response time comparison under user load of 1000 users.

When it comes to number of successful requests the best performance is achieved with AKS with around 750000 successful requests on average, followed by Windows Server with around 350000 successful requests on average. The worst performance, was again demonstrated by the Web App service with only around 170000 successful requests. 5.8

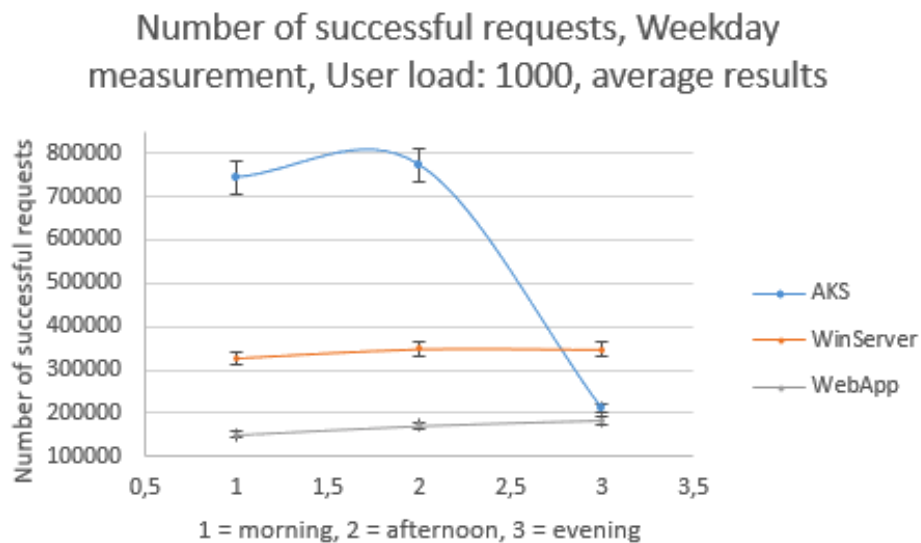


Figure 5.8: Comparison of number of successful requests under user load of 1000 users.

The number of failed requests for AKS was around 350, while there was a big increase in failed requests for the Windows Server VM with around 100000 failed requests on average. The deployment on Web App service was again with worse performance with around 170000 failed requests. 5.9

The results demonstrate that the increased load of users effects the response time of all deployments that results in longer response times. The best reaction to the increased load was demonstrated by AKS with only 200 to 250 ms increased response time, followed by Windows Server VM with around 1200 ms increase in response time, followed by WebApp with 2600 ms increase of response time. In terms of number of successful requests AKS demonstrated increase of this metric from around 400k to 750k which has a huge significance, while the increase of the failed requests was insignificant and less than 1% of total requests.

5.7 The number of successful requests was not affected on the Windows Server VM, with the deployment serving around 350k successful requests under the increased load. The only change was in the increase of failed requests, which means that not all users will be served, but the capacity of successfully served users remained stable. ??

The deployment on Web App service demonstrated worst performance with largest increase of response time and decrease of the number of successful requests. However, the increase of the failed requests was around 10% of total requests, less than the percentage of total requests for the Windows Server VM which was around 20%. 5.9

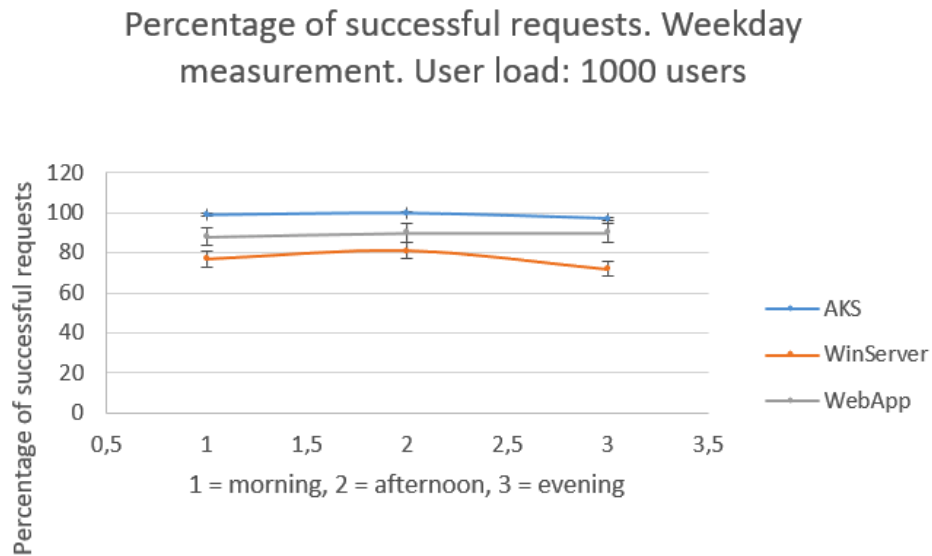


Figure 5.9: Comparison of percentage of successful requests under user load of 1000 users.

5.5.4 Performance under load of 300 and 1000 users with experimental conditions

In the final phase of the testing, the conditions were changed with a scenario with 2 or 3 instances of Web App service and other deployments under equal conditions. And finally a test with only 1 container instance of AKS, same conditions for Windows Server VM and 3 instances of Web App service was conducted. The results are displayed in 5.10. The results show that Web App even with 2 deployed instances is still not a match to the response times of the Windows Server VM or AKS (case 1 at figure 5.10). To match the response times of AKS and Windows server VM, 3 instances of Web App service were needed. (case 3 at figure 5.10). Moreover the experiment with 1 container instance of AKS deployment demonstrated much worse performance with response time of around 2500 ms, which was expected. The final experimental condition was test of user load of 1000 users in which the AKS cluster demonstrated the best performance again, even though the Web App service was scaled to 3 instances. The Web App response time decreased to 720 ms, but it was still better than the response of the Windows Server VM at 1440 ms(case 2 at figure 5.10).

The results demonstrate that the WebApp requires 3 times the resources to have the same response time as the AKS cluster and Windows Server VM. Moreover only 1 container instance on the cluster could not compare with the performance of the Web App or the Windows Server VM.

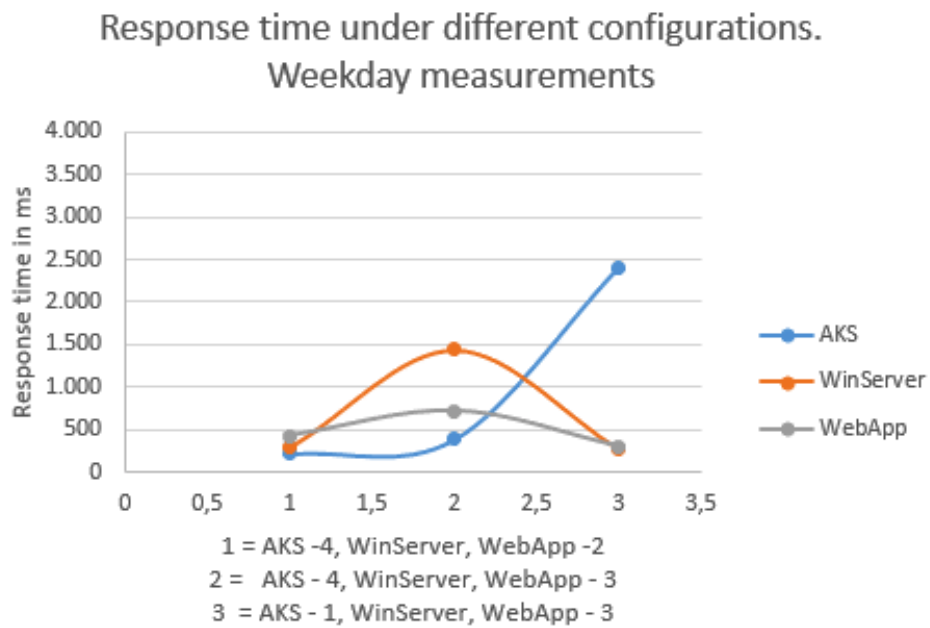


Figure 5.10: Response time comparison under user load of 300 users under experimental deployment conditions.

5.5.5 Coded User Interface testing with selenium

Coded UI tests typically drive the application through its user interface (UI) and include functional testing of the UI controls. Coded UI test was created by using the Selenium automation tools suite. The results of the tests demonstrated that all three application deployments performed equally. This is displayed in figure 5.11

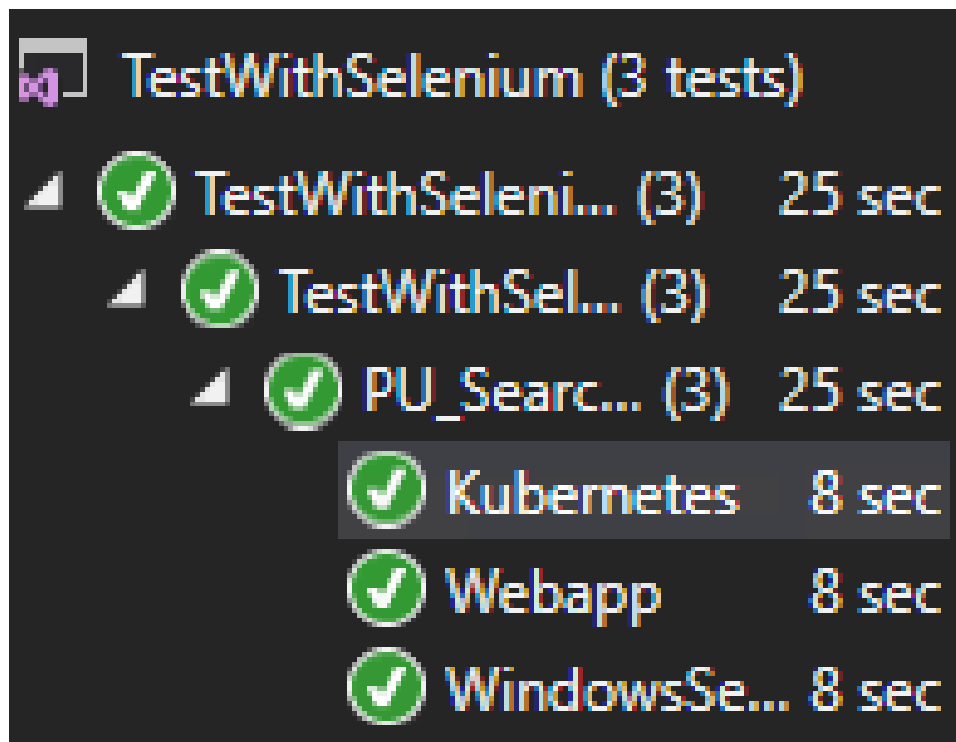


Figure 5.11: Results of coded UI testing using Selenium.

5.6 Conclusions

This chapter described the performance comparison of a sample .NET Core e-commerce application deployed on 3 different service offerings of Microsoft Azure. The motivation behind this comparison is to gain insight into the performance, cost, ease of deployment and life cycle management of an application deployed on different cloud services.

The results showed that Azure Kubernetes service has the best performance both in terms of response time and number of successful requests. Moreover the increase of user load had the least effect in the increase of response time and increase of failed requests, while the number of successful requests also increased. The experimental deployment with only 1 container instance performed poorly compared to the other services.

The results for the Windows Server VM showed that the deployment performs the same as AKS in terms of response time under load of 300 users, while the response time and number of failed requests increases much more when the load is increased to 1000 users. The load increase, however, did not affect the number of successful request which remained stable.

The results for the Web App showed that the deployment performs worse among

the services compared. It has largest response time, with lowest number of successful requests served. The experimental testing showed that to achieve the performance of AKS or the Windows Server VM, the Web App needs to be scaled out to 3 instances. Moreover, the increase of load affected the Web App the most by increasing the response time and decreasing the number of successful requests.

In terms of costs comparison Web App service is the most expensive with costs of 292 dollars per month, followed by Windows Server VM with costs of 207 dollars per month, followed by AKS with 203 dollars per month.

The ease of deployment and lifecycle management of AKS and Web App is comparable and requires much less effort as it is Platform as a Service offering, compared to Windows Server VM which is Infrastructure as a Service offering and requires much more management by the end user.

The conclusion would be to choose AKS if one wants partially managed service as it demonstrated much better performance compared to the other PaaS offer (WebApp). Moreover, it is worth noting that Windows Server VM demonstrated comparable performance to that of AKS for 300 users load and worse performance for the load of 1000 users. If the application needs to scale out the recommendation would be to choose either AKS or Web App since these services are much easier to scale. For updates without downtime, the recommendation is to choose one of the PaaS offerings (AKS or WebApp).

Finally, the shortcomings of these performance comparison need to be mentioned. The underlying infrastructure used was not identical, but comparable, and this may have influence on the final results. Moreover, the deployed application is just one example of an application and not a general rule for performance of all applications. The applications may vary a lot, from the programming language that they were written in, to applications that are more compute, memory or I/O operations intensive and can be optimized to perform best with other underlying infrastructure. This performance analysis should serve as a reference if you are deploying a similar type of application.

In the future, more tests can be conducted with various types of applications and various types of underlying infrastructure to obtain even more detailed results, but that was not in scope of this master thesis.

Chapter 6

Security and Compliance: Developing out of the box secure and compliant cloud components. Use case of Azure Cosmos DB

The adoption of the Cloud Computing technologies in an financial institution like Rabobank has to be done in a very safe, controlled and compliant way. To achieve this, Rabobank formed the Cloud Competence Center. The team enables DevOps teams to deliver business value faster using Public Cloud in a safe, controlled and compliant manner. The team takes care of everything related to management of the platform like portals, identity, monitoring, governance, networking, automation, key management and the delivery of Microsoft Azure Services to be used by the DevOps teams. The services are delivered as Pre-Certified Azure Services (PreCAS features) in form of a VSTS (Visual Studio Team Services) extensions to be used in release pipelines. These extensions are:

- *Pre-approved*: The development and release of the feature is approved by the stakeholders and the security officer
- *Compliant out-of-the box*: The feature is compliant to the security and legal requirements for technology use at the bank
- *Reusable at Rabobank*: The feature is released as Visual Studio Team Services extension, which can be reused by all the teams that work in the bank
- *Provide efficiency across the organization*: Developing and certifying the features enables teams to focus on their application development and not focus on making the infrastructure components compliant, which is done only once when developing the features.

This thesis investigates how to enforce security and compliance controls on using Microsoft Azure Components which is explained through the implementation

of the feature Microsoft Azure Cosmos DB with the MongoDB API. To deliver this the following steps were conducted:

- *Planning and design decisions* - Investigate the capabilities of the feature and decide about the design
- *Proof of Concept* - Make a proof of concept with the feature on the Azure Portal in a manual way
- *Implementation* - Implement the feature design as a VSTS extension
- *Testing and evaluation* - Test the feature with different parameters and in different environments

6.1 Description of Azure CosmosDB

Azure Cosmos DB is a Microsoft database offering that provides turnkey global distribution, elastic scaling of throughput and storage worldwide, single-digit millisecond latencies at the 99th percentile, five consistency models, and guaranteed high availability. Azure Cosmos DB automatically indexes all your data without requiring you to deal with schema or index management. Cosmos DB is a multi-model service and supports document, key-value, graph, and column-family data models. [16] The interest of our project is only the MongoDB API. The reason behind this is that this is the only API that was requested from the DevOps teams that they would like to use. The features of the CosmosDB service can be divided in two categories: Control plane and Data plane. Our team is responsible only for the security and configurations that can be done in the control plane and this is what is delivered as a final product in form of a VSTS extension. We differentiate the control plane features as features that are connected to the CosmosDB account and its configuration, such as account creation, consistency levels etc., while the data plane features are all the configurations related to data management such as database and collection creation, sharding, partitioning etc.

6.2 Planning and design decisions

The development process begins with a research of the capabilities of the Cosmos DB Service. For this the documentation of the Cosmos DB Service was read and a white board session was organized where the capabilities of Cosmos DB were presented, explaining what options can be configured with the purpose to made decisions about the design and what should the final extension enable by default. The design decisions made and the reasoning behind them are explained by category.

Security - Cosmos DB offers default encryption at rest (where the data resides) with AES and TLS or SSL encryption at transit. Access in the control plane is restricted with Role Based Access Control which is a common approach in computer systems security where each user is granted certain access rights that they

can use after they completed their authorization [9]. This prevents unauthorized access, as well as it prevents access to resources by people in your organization that should not have access to it. The communication with the data plane is done with Hmac(hash-based message authentication code) signatures which protect the data from attackers. This is done through hashing each request with the secret account key, and the subsequent base-64 encoded hash is sent with each call to Azure Cosmos DB. Validation of the request is done by the service which uses the correct secret key and properties to generate a hash which value is compared to the value of the request. If the two values are equal, the operation is authorized, or otherwise it fails[3][16]. What is configurable regarding security is firewall IP whitelisting and access from Azure virtual networks via Service Endpoints. Together with the team consisting of the developers, security officer, architect and stakeholder representing the customers needs, it was decided that this needs to be included as input in the extension so the DevOps teams may be able to configure this according to the needs of their application, infrastructure and security assessment. The reason behind this is that there is no universal IP address (ex. general application gateway), or a pre-defined virtual network that can be preconfigured and applicable to all of the users of the feature. Therefore, the users of the feature can configure this according to their security assessment. Moreover, access from other Azure Public Services can be enabled or disabled and this is also an input to the extension. The reasoning behind this is the same as for the IP whitelisting, there are multiple use cases for the customers using the extension, so no single choice should be preconfigured. Regarding the security and storage of the keys and connection strings it was decided to use the Azure Key Vault service modified by Rabobank to do automated key rotation to enhance security. One of the reasons for using key rotation is because it is mandated by regulation in the payment card industry data security standard (PCI DSS) that dictates how credit card data must be secured [6]. Moreover key rotation can also be used to revoke old keys that are comprised, or to effect data access revocation[1]. Finally, it was decided to only enable read access from the Azure portal, as a general Rabobank policy for all the features. This is done to prevent manual interventions, and every modification is done either through CI/CD pipeline or the application which is logged and monitored, and can be audited. The final outcome is allowing input for network access by either IP whitelisting or integration with vnet service endpoint or allowing access from Azure Services, input for Key Vault for mandatory storing of keys and connection strings in Rabo compliant way, and restricting access to the Azure portal with only read permissions.

Consistency and replication - Azure Cosmos DB provides five consistency levels:

- *Strong*: Linearizability. Reads are guaranteed to return the most recent version of an item.
- *Bounded-staleness*: Consistent Prefix. Reads lag behind writes by at most k prefixes or t interval
- *Session*: Consistent Prefix. Monotonic reads, monotonic writes, read-your-

writes, write-follows-reads

- *Consistent prefix*: Updates returned are some prefix of all the updates, with no gaps
- *eventual*: Out of order reads

It was decided to enable this as an input because the teams need to decide it according to the business needs of their application. Replication is limited to two Azure Regions: West Europe and North Europe. The reason for this is that the data the bank processes should not leave the EU [29] and the general policy of Rabobank is to deploy everything primarily in West Europe and replicate and do backups in North Europe. Moreover, it was decided that automatic failover should be enabled by default in production in order to prevent data loss in case of disasters. Furthermore, it was decided to note in documentation that the replication implies additional costs as twice the amount of resources are needed.

Monitoring performance - Through the Azure Portal, there is the possibility to monitor the performance of the database. The metrics that are being monitored are: Throughput, Storage, Availability, Latency and Consistency. It was decided that the teams can have read access to this metrics as they will be useful to determine the performance of their database. Moreover, there is an option of Data Explorer in the Portal which allows the user to read the data through the Azure Portal. It was decided that the access to the Data Explorer in the portal should also be configurable. The reason behind this is that different teams have different needs for read access for their application and this access may affect the CIA(Confidentiality, Integrity and Availability) rating on the application. Confidentiality refers to protecting the information from disclosure to unauthorized users, integrity of information refers to protecting information from being modified by unauthorized users and availability of information refers to ensuring that authorized users are able to access the information when this is needed[4].

Logging diagnostic logs - Azure Cosmos DB has the possibility to push diagnostic logs either to an Azure Storage Account, Event Hub or to Log Analytics. After a discussion with the team responsible for the logging policies of the platform, it was determined that the logging was already automated by the team and stored in OMS(Operations Management Suite) Log Analytics. The reason why this is done centrally is the access restriction to the management subscription and the fact that there is no RBAC(Role Based Access Control) implemented for Log Analytics which poses security risks. OMS Log Analytics is the central body where logs from all of the used components are being stored and processed. This means that if a user has access to this service, the same user has access not only to the logs of the components that he is the owner, but to all the logs in the Azure subscription which poses a security risk.

Backup and restore - Azure Cosmos DB, by default, has 2 available backups, and snapshots are taken approximately every 4 hours with a 30 days retention period. This is not configurable at the moment, so no decisions had to be made

for this. In the documentation of the feature it is noted that there are no incremental backups and no point in time restore are available, so the teams can know this in advance when making decisions about using the feature. The implications of this are that there is a risk that some data may be lost, and backup can be done only to a point of time defined by Microsoft, and not customizable by the user. There are some backup alternatives like using the Azure data migration tool, export to JSON through the portal or using Mongo scripts which is mentioned in the documentation. This has the downside of using a lot of RUs(Resource Units) that are the measure for the throughput for Cosmos DB, and also a unit for cost charging. That means that the more RUs are consumed the bigger the costs.

Data plane considerations - Configurations regarding the data plane are not in the scope of the feature, but it is the responsibility of the team using the feature. In order to help the teams make better decisions about configurations regarding the data plane, it was decided that links to relevant information are part of the Rabobank documentation of the feature. Links to resources regarding partition keys, scaling throughput and storage, indexing policies, cost and performance optimizations are included.

Other limitations - Access from on-premise development machines and servers has to be requested separately through the regular process in the bank. This means that a firewall rule change needs to be requested. Multi-master mode, which is a configuration where users can write to master nodes in multiple Azure regions, is disabled by default since it is in Preview mode. This means it is not yet officially published by Microsoft, but it is published for beta testing by users.

6.3 Proof of Concept

The proof of concept phase involved implementation of the above mentioned design by manual deployment of resources through the Azure Portal. The purpose of the PoC was to test one use case and do modifications to it, so that most usage cases are covered. For this a Cosmos DB account with the Mongo API enabled was created. The security of the control plane access through the Azure Portal was tested by giving a user contributor rights and successfully making changes from that user, while trying to access and make changes to the Cosmos DB from a second, unauthorized user failed. In order to test the network access policies a database was created, and a collection which was populated with sample data by a NodeJS application. Then a modification to the firewall rules was made to allow IP white listing to a certain machines, and an attempt to connect with a NodeJS app from the allowed and restricted machines was done. The connection was successful only on the allowed machines, as expected. The access from other Azure services was tested by deploying the same NodeJS app to an Azure Web App Service and successfully connecting only when this was enabled and failing when this option was disabled. The virtual network integration was also tested by adding firewall rule to allow access to one virtual network, then an attempt to connect to the Cosmos DB was done

from machines from both allowed and other virtual networks. The connection was successful only from the machine residing in the allowed network, as expected.

The encryption at transit was tested by monitoring the traffic with the Wireshark tool and no unencrypted data was detected.

The consistency was tested by enabling all of the consistency levels. The replication was tested by enabling it in West and North Europe regions in the Azure portal. The proof of the consistency levels was not tested since it was out of scope for the proof of concept and there are reliable sources on the Internet have already made some tests.

Performance monitoring was tested by confirming the access to the metrics in the portal and stressing the database until it had failed to respond to some requests and noted that this was shown in the Portal.

Regarding the diagnostic logs, it was confirmed that there is automated creation of logging account that pushes the logs to Log Analytics in the management Azure subscription.

The backup and restore functionalities were not tested and were out of scope for the proof of concept for the same reasons as not testing the consistency levels.

With this, the design decisions were confirmed and the development of the actual VSTS extension was started.

6.4 Implementation details

In order to create a VSTS extension release task multiple steps need to be taken:

- Step 1: Create the task metadata file
- Step 2: Create the extension manifest file
- Step 3: Package the extension
- Step 4: Publish the extension
- Step 5: Install and test the extension [13]

Moreover, specific to Cosmos DB several things need to be created:

- ARM(Azure Resource Manager) template that describes the resources
- Main script that is called from the extension
- Script for creating the Cosmos DB and related resources
- Script for removing the Cosmos DB and related resources

The technical details of the implementation may be found in the appendix.

The final outcome is a VSTS extension that automates creation 6.2 and removal of Cosmos DB and its associated resources 6.3 The network topology is described in 6.1 The allowed traffic flows are:

- From Azure Services (public) to Azure Cosmos DB (configurable)
- From Azure Private to Cosmos DB using the virtual network service endpoint. Routing is controlled by the service endpoint routing.
- From on-premises to Cosmos DB over Internet. The consumer needs to request on-prem firewall rules to connect from an on-prem server to an Cosmos DB on port 10255. Also it is recommended to enable IP whitelisting and add the machine IP to the IP white list.

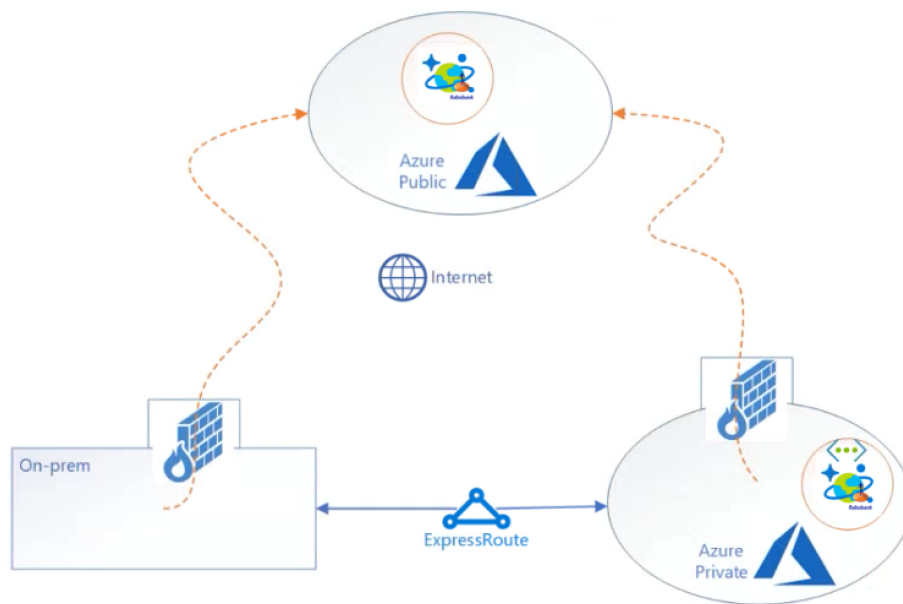


Figure 6.1: Network topology of the Azure and on-premise environment for using Cosmos DB

6.5 Requirements for Security, Service Management Controls and separation of responsibilities

This section describes the internal requirements of Rabobank for each cloud component that is developed which are derived either by regulator requirements or internal company preferred practices.

6.5.1 Security requirements and separation of responsibilities

In the subsequent tables, a summary of the security controls requirements is described, including how they apply to the Cosmos DB feature and who is responsible

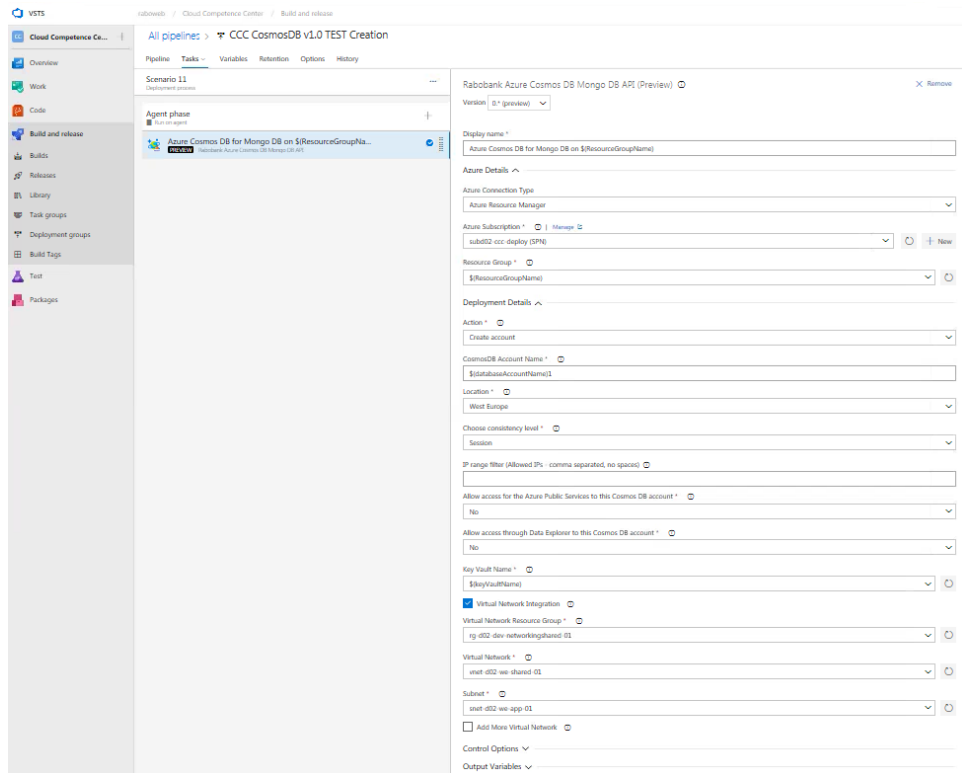


Figure 6.2: Final VSTS extension release task for creating Cosmos DB

for each. The controls are divided in Foundation controls 6.1, Standard controls for CIA(Confidentiality, Integrity, Availability) rated workloads 6.2 and Complementary advanced controls 6.3.

6.5.2 Service management controls requirements and separation of responsibilities

In the subsequent tables, a summary of the service management control requirements is described, including how they apply to the Cosmos DB feature and who is responsible for each. The controls are divided in Foundation controls 6.4 6.5, and production controls 6.6.

6.6 Testing and evaluation

Few types of tests were conducted for the extension.

- Extension build, release and publishing

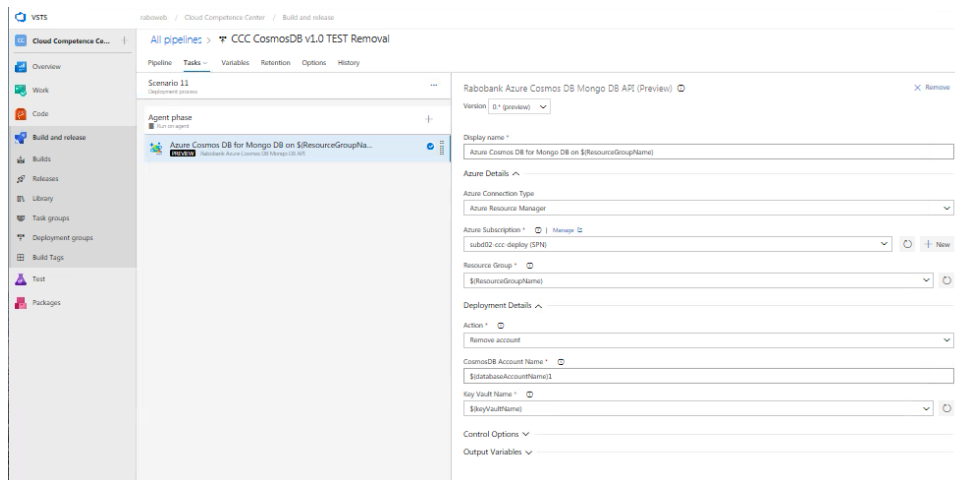


Figure 6.3: Final VSTS extension release task for removing Cosmos DB

- Installing the extension and deployment in different environments with different configurations
- Manual test of the desired configuration after the resources are allocated

6.6.1 Extension build, release and publishing

After development is finished, the build, release and publish tasks are automated using gulp.js and PowerShell script. If the tasks succeed, this means there are no syntactic errors and the extension is successfully published and shared with Rabobank VSTS environment.

6.6.2 Installing the extension and deployment in different environments with different configurations

After the extension is published and shared with the Rabobank VSTS environment, the extension is installed. In order to verify that the customizable configurations work, several configuration scenarios were deployed automatically through VSTS release pipeline using the newly created extension and then manually tested. The customizable configurations that were tested were the combinations of the following:

- No firewall rules: All connections allowed
- Firewall rules with enabled or disabled IP whitelisting, with one or more IP addresses as input, or input in CIDR format
- Firewall rules with vnet integration enabled or disabled, with one or more vnets as input

- Firewall rules with access from Azure Public Services enabled or disabled
- Firewall rules with access to the Data Explorer in the Azure Portal enabled or disabled
- Write locations: West Europe or North Europe
- Consistency levels: strong, bounded staleness, session, consistent prefix, and eventual.

Moreover, apart from the configurable configuration, the configuration enabled by default was tested as well:

- Replication (Geo redundancy) enabled by default
- Automatic Failover enabled by default
- Diagnostic logs automatically pushed to OMS Log Analytics
- Multi Master write mode disabled by default
- Primary Master and Primary Read Only keys are stored in Azure Key Vault

The outcome of the designed testing scenarios is described in 6.7

In order to create resources for the desired scenarios a VSTS pipeline which is using the created task was created. 6.4

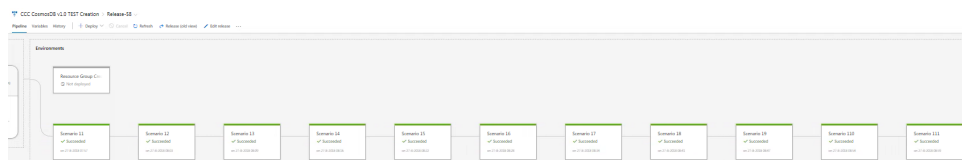


Figure 6.4: VSTS pipeline for creation of resources to test different deployment scenarios

6.6.3 Manual test of the desired configuration after the resources are deployed

After the resources have been deployed, they are subject to manual testing.6.5

In order to test the firewall settings for IP whitelisting, connection is attempted to the Cosmos DB from both allowed IPs, that are listed separately or are part of network listed in CIDR format, and blocked IP addresses. It was confirmed that the connection was successful from the authorized IPs, and rejected due to lack of authorization from the other machines as expected.

Subscription (change) [rabo-d02-ccc-consumers](#) Subscription ID [d8519368-a500-405c-b694-76765063969](#) Deployments [4 Failed 52 Succeeded](#)

Tags (change) [CreationDate : 8/16/2018 11:59:05 AM](#) [AssetName : cosmosdburg](#) [CostCenter : P482220](#)

Filter by name... All types All locations No grouping

20 items ☐ Show hidden types

NAME	TYPE	LOCATION
rgd02devcosmosdbkvr0560	Storage account	West Europe
stefancosmosdb1	Azure Cosmos DB account	West Europe
stefancosmosdb11	Azure Cosmos DB account	West Europe
stefancosmosdb110	Azure Cosmos DB account	West Europe
stefancosmosdb111	Azure Cosmos DB account	North Europe
stefancosmosdb12	Azure Cosmos DB account	North Europe
stefancosmosdb13	Azure Cosmos DB account	West Europe
stefancosmosdb14	Azure Cosmos DB account	West Europe
stefancosmosdb15	Azure Cosmos DB account	West Europe
stefancosmosdb16	Azure Cosmos DB account	West Europe
stefancosmosdb17	Azure Cosmos DB account	West Europe
stefancosmosdb18	Azure Cosmos DB account	West Europe
stefancosmosdb19	Azure Cosmos DB account	West Europe
stefancosmosdb199	Azure Cosmos DB account	West Europe
stefancosmosdb1demo	Azure Cosmos DB account	West Europe
stefanvkcosmosdb1	Key vault	West Europe

Figure 6.5: The resulting resources created from the VSTS pipeline

To test the connection from allowed virtual networks, virtual machines were created in both allowed and blocked vnets. The connection to Cosmos DB was successful from the machines in the allowed vnets, while the connection was rejected from the machines in the other vnets or machines residing in the public Internet that were not in the IP white list.

To test the connection from Azure Public services, a simple Azure Web App written in Node.js was deployed and connection was tested from the application that had the right configuration for the connection string.

To test the access to the Data Explorer, a manual test was done by locating the desired resource in the Azure Portal and confirming that the access to the Data Explorer is enabled or disabled accordingly.

To test the write location, the replication configuration and the automatic failover configuration, a manual test was done by locating the desired resource in the Azure Portal and confirming that the write region is correct, data is replicated in one more region depending on the writing region and automatic failover is on by default.

To test the consistency level configuration, a manual test was done by locating the desired resource in the Azure Portal and confirming that the desired consistency level is configured. 6.6

To test that diagnostic logs are automatically pushed to OMS Log Analytics account, a manual test was done by locating the desired resource in the Azure Portal and confirming that an account for sending the diagnostics is created and configured to push the diagnostic logs to the right place. Then, some events were triggered and later queried in the OMS Log Analytics Portal to confirm that logs are being stored.

To test that the primary master and primary read only keys are stored as secrets in the Azure Key Vault, a manual test was done by locating the desired resource in the

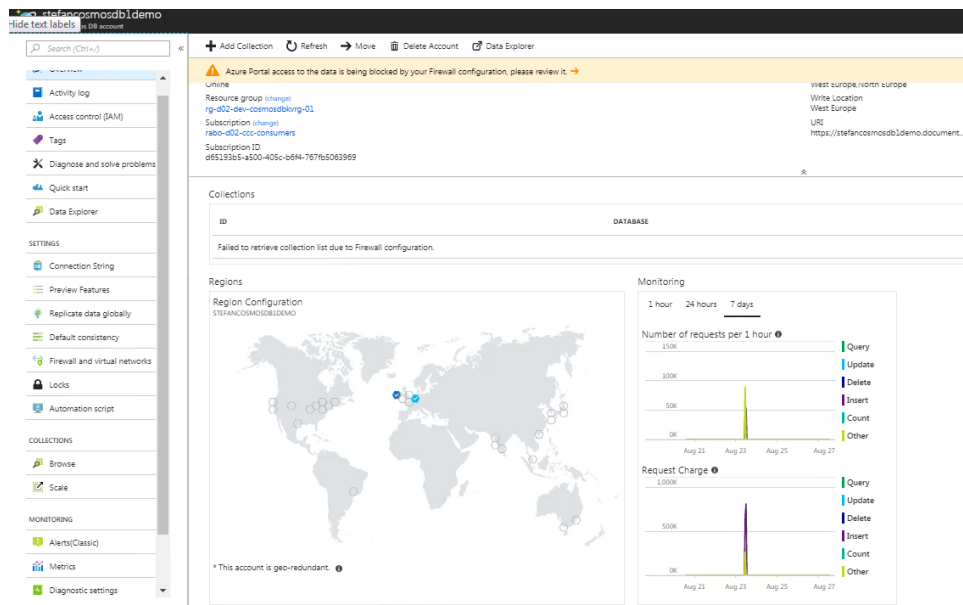


Figure 6.6: Manual testing of the desired configuration through the Azure Portal

Azure Portal and making sure that the keys are present as secrets in the Key Vault. To confirm that the key rotation is working, the keys were checked in different periods of time and compared if they values changed and successfully connecting to the Cosmos DB with the new keys. 6.7

To test that multi master write mode is disabled, a manual test was done by locating the desired resource in the Azure Portal and confirming that there is only one write region configured.

To test that the IAM (Identity and Access Management) is working properly an attempt to access the Cosmos DB through the Azure Portal was done by both authorized and unauthorized users and it was confirmed that only the authorized user could access the resource.

To test that the appropriate tags are created in each resource, a manual test was done by locating the desired resource in the Azure Portal and confirming that the desired tags are present in the resource.

6.7 Conclusions

In conclusion, the above mentioned implementation of Microsoft Azure CosmosDB as a VSTS extension gives example and answers the question of how to enforce security and compliance controls in an enterprise environment. This is done by standardizing the components by:

NAME	TYPE	STATUS
stefancosmosdb11-key-primaryMaster		✓ Enabled
stefancosmosdb11-key-primaryReadOnly		✓ Enabled
stefancosmosdb110-key-primaryMaster		✓ Enabled
stefancosmosdb110-key-primaryReadOnly		✓ Enabled
stefancosmosdb111-key-primaryMaster		✓ Enabled
stefancosmosdb111-key-primaryReadOnly		✓ Enabled
stefancosmosdb12-key-primaryMaster		✓ Enabled
stefancosmosdb12-key-primaryReadOnly		✓ Enabled
stefancosmosdb13-key-primaryMaster		✓ Enabled
stefancosmosdb13-key-primaryReadOnly		✓ Enabled
stefancosmosdb14-key-primaryMaster		✓ Enabled
stefancosmosdb14-key-primaryReadOnly		✓ Enabled
stefancosmosdb15-key-primaryMaster		✓ Enabled
stefancosmosdb15-key-primaryReadOnly		✓ Enabled
stefancosmosdb16-key-primaryMaster		✓ Enabled
stefancosmosdb16-key-primaryReadOnly		✓ Enabled
stefancosmosdb17-key-primaryMaster		✓ Enabled
stefancosmosdb17-key-primaryReadOnly		✓ Enabled
stefancosmosdb18-key-primaryMaster		✓ Enabled
stefancosmosdb18-key-primaryReadOnly		✓ Enabled
stefancosmosdb19-key-primaryMaster		✓ Enabled
stefancosmosdb19-key-primaryReadOnly		✓ Enabled
stefancosmosdb199-key-primaryMaster		✓ Enabled

Figure 6.7: Created secrets in Azure Key Vault after deployment

- restricting certain configurations that may introduce risks
- delivering a component that has predictable behavior
- developed according to the security and compliance rules of both Rabobank and the Dutch government
- it is reusable in VSTS CI/CD pipeline
- can be automated to speed up delivery cycle, providing efficiency across the organization

Even though this is an example for just one component of the Microsoft Azure platform, the same research, design and development principles can be easily adopted for developing different Cloud Computing components according to the business and security requirements in the enterprise. The example implementation can be used as a reference for both structure and design decisions and the consequences of those decisions, since many configurations are similar across the spectrum of Microsoft Azure Cloud Computing platform.

However, one should also be aware that there is no one size fits all solution to a complex question like Cloud security and compliance, so every use case has to be assessed separately and designed and developed according to the requirements that

apply for that case. The implemented methodology should be used as a reference, but not as a general rule for delivering Cloud components.

What	How	Responsible
Azure provided Key Management	The keys are stored, protected and rotated in Azure Key Vault by default	CCC and Cloud Provider
Owner of resource must be known	Owner tag is set on resource group level and enforced to all resources under a resource group during deployment.	CCC
Security and Event Monitoring and event correlation	Control Plane and Data Plane events are logged and sent to log analytics.	CCC and Cloud Provider
Platform Security Monitoring	Activity logs are available and are sent to a central Log Analytics workspace. Security Center periodically analyzes the security state of the Cosmos DB and gives recommendations.	Cloud Provider
IAM on all accounts	Customers are responsible for requesting the Active Directory groups via IAM (Identity and Access Management) and assigning the users to these groups.	Consumer and CCC
MFA on all user accounts	Federated accounts are used to access Cosmos DB. MFA is not enabled for all these account yet. This should be handled as residual risk.	IAM
Platform Activity Logs	Activity logs are available and are sent to a central Log Analytics workspace.	Cloud Provider
IAM on all resources	Technical groups are created in Azure Active Directory and mapped to organizational groups in on-prem Active Directory.	CCC
Periodically assess updated Cloud Service Terms and Audit Reports	The Azure provided services, service terms and audit reports are periodically assessed for new developments.	CCC
Identity Threat Protection	Azure Active Directory groups are used to access Cosmos DB. Azure AD Identity Protection functionality of Azure provides Identity Threat Protection. The Azure Identity Protection is connected to a central Log Analytics workspace.	Cloud Provider and CCC
Terms of use for playground environments and cloud features	Terms of use for Cosmos DB is an implicit part of this feature description which is reviewed and approved by GISO. Customers using Cosmos DB have the responsibilities for using it in accordance with Rabobank policies, rules and regulations.	CCC and Consumer

Table 6.1: Implementation of Security ⁷⁷Foundation controls and separation of responsibilities

CCC = Cloud Competence Center

What	How	Responsible
Only approved and certified cloud services can be deployed	Cosmos DB is an approved and certified Rabobank Azure Service.	CCC
No direct inbound traffic from Internet to Azure Private (Perimeter Defense)	When Virtual Network Service Endpoints are used, direct inbound traffic is not allowed and Cosmos DB becomes part of Azure private	Consumer
Infrastructure as Code (central repository, code review and approval, SDL, etc.)	All ARM templates and scripts used in deployment of Cosmos DB and the dependent Azure resources are stored and managed in VSTS. The deployment is done using VSTS release management.	CCC
Vulnerability scanning	Since Azure Cosmos DB is a PaaS, Microsoft is responsible for vulnerability management.	Cloud Provider
OS Threat Protection and Anti Virus/Malware scanning	Since Cosmos DB is PaaS, Microsoft is responsible for implementing this control. The consumer is responsible for the data that is stored in Cosmos DB thus responsible for protecting data against Viruses/Malware that are uploaded to Cosmos DB.	Cloud Provider and Consumer
Implementation of security functionality must be validated	The implemented (security) functionality of the delivered feature is tested by CCC through multiple scenarios including automated tests that run every night and connection tests to the Cosmos DB from customer perspective.	CCC
Declarative configuration management	The configuration of Cosmos DB is done using an ARM template, i.e. in a declarative way. All configuration changes on Cosmos DB are done using this template and no manual changes are possible due to limited role based access (RBAC) rights given on the resource.	CCC
Connectivity controls on public end-points	Inbound traffic flows to Cosmos DB are controlled through white listing allowed traffic via the firewall on Azure Cosmos DB. When Cosmos DB is deployed with virtual network integration, Virtual Network Service Endpoints are used for connectivity control.	CCC
Dynamic scaling and/or limits should be facilitated	Scaling is done in the data plane and is responsibility of the consumer.	Consumer

Table 6.2: Implementation of Security Standard controls for CIA rated workloads and separation of responsibilities

CCC = Cloud Competence Center

What	How	Responsible
Encrypt data in transit over public and private interconnections	All client-to-service Azure Cosmos DB interactions are SSL/TLS 1.2 capable. Also, all intra datacenter and cross datacenter replication is SSL/TLS 1.2 enforced.	Cloud Provider
Network segmentation on application tier level	Cosmos DB supports Virtual Network integration. Consumer is able to configure at maximum three Virtual Networks during deployment.	Consumer
Encrypt data at rest for all public services with Service Level encryption	Cosmos DB is encrypted at rest by default.	Cloud Provider

Table 6.3: Implementation of Security Complementary Advanced Controls and separation of responsibilities

CCC = Cloud Competence Center

What	How	Responsible
Cloud adoptions guidance is provided and available for (potential) Cloud Consumers	There is document that provides guidance and links to adopt Cosmos DB in Azure. Also, hands on guidance can be provided by CCC .	CCC
Consumption reporting is available to support showback or cross charging	Consumption and cost reports are shown in PowerBI for showback. These are updated automatically once a day.	CCC
Subscription Management is based on capacity forecasting, limit monitoring and pre-defined adjustments	Subscription limits are checked weekly and if needed a request to raise the subscription limit is done through the built-in request on Azure Portal.	CCC
Cloud Service Terms and Conditions are documented and accepted (incl. responsibilities)	Terms of use for Cosmos DB is an implicit part of the feature description which is reviewed and approved by GISO. Customers using Cosmos DB have certain responsibilities for using it in accordance with Rabobank policies, rules and regulations as described in the feature description.	CCC and Consumer
IAM controls for the Control Plane are applied and managed	All changes to the feature (deployment, configuration) is performed through VSTS which has access to Azure using a Service Principle account. No other role assignments are done on the Cosmos DB resource on control plane.	CCC
All resources are tagged (e.g. to provide reporting and applying automation)	Cosmos DB is tagged with Cost Center and Owner tags during deployment. This is enforced through Resource Policies on Resource Group level.	CCC
The operations is compliant to the Cloud Security Control Guideline	During the Cosmos DB feature design, the Cloud Security Control Guideline is consulted for the security measures. GISO does a security assessment of the feature based on the Cloud Security Control Guideline and approve the Cosmos DB feature description and the implementation whenever a new (version of a) feature is released. The security assessment of the application is the responsibility of the consumer.	CCC and Consumer

Table 6.4: Implementation of Foundation Service Management Controls and separation of responsibilities part 1

CCC = Cloud Competence Center

What	How	Responsible
Security alerts are automatically generated, correlated and managed based on the security incident response process	Activity logs are sent to a central log analytics workspace. Policies and alerts can be defined in Security Center.	CCC
Support is available on registered incidents. Alignment to the interfacing processes on the Cloud Vendor is established.	In case the incident is related to the Azure Cosmos DB, the consumer can contact CCC using email. Cloud Competence Center team has a direct line to the support team at Microsoft.	Consumer and CCC
Provisioning of platform components is managed through a Release Pipeline	Cosmos DB deployment is done through a VSTS release pipeline.	Consumer
LCM(Life cycle management) changes on platform and features is applied and pro-actively managed according to the roadmap of the Cloud Vendor	Cloud Competence Center team will monitor the changes to the Azure Cosmos DB service from the Azure Updates. In case Azure Cosmos DB functionality changes that impacts the released feature, CCC will create a new version. Consumers using the Cosmos DB will need to follow the supported feature releases of the CCC Cosmos DB and adapt their application accordingly.	CCC and Consumer
Performance levels are monitored and under/over utilized is alerted upon	Consumers using Cosmos DB are responsible for performance tuning.	Consumer

Table 6.5: Implementation of Foundation Service Management Controls and separation of responsibilities part 2

CCC = Cloud Competence Center

What	How	Responsible
A Knowledge Management environment is managed for platform and product documentation	Visual Studio Team Services Wiki is used to store the feature description of Cosmos DB. The implementation of Cosmos DB is documented in templates, scripts and readme files, which are stored in the GIT of Visual Studio Team Services .	CCC
Health alerts are automatically generated, correlated and managed based on event monitoring.	Resource health changes of Cosmos DB are reported in Log Analytics. Consumer is responsible for the health of the application that uses the Cosmos DB.	CCC and Consumer
RTO / RPO requirements are achieved for backup and recovery	The consumer is responsible for the backup and recovery of data.	Consumer
Disaster Recovery scenarios are applied and frequently tested	In case disaster recovery is needed, the consumer can use the capabilities that Azure Cosmos DB provides for business continuity and disaster recovery.	Consumer
(automated) Testing is applied on functional and technical level	Deployment and configuration code for Cosmos DB is tested by the CCC team. Consumer is responsible for testing the application using the Cosmos DB.	CCC and Consumer
IAM controls for the Data Plane are applied and managed	Azure Active Directory technical groups are granted permissions on data plane.	CCC
Policies are defined and deviations are monitored and alerted/reacted upon	Policies on subscription level are defined and the deviations are monitored. These are reacted upon on best effort.	CCC
Certified Features will be deployed initiated by consumer requests through automation and accessible for authorized users	Provisioning of Cosmos DB is fully automated and offered as self service to the consumers. The onboarded consumers get a VSTS service endpoint which uses a service principal that is authorized to do the provisioning.	CCC
Service Requests (request fulfillment) on certified products are delivered through automation.	Provisioning and deprovisioning of Cosmos DB are fully automated and offered as self service to the consumers.	CCC
Deployed certified products can be fully deleted using automation (e.g. to avoid left over AD items and firewall rules)	Deprovisioning of Cosmos DB is fully automated. This task removes also the keys (Master and Read Only) of the Cosmos DB from the key vault leaving no trace of the Cosmos DB account behind.	CCC

Table 6.6: Implementation of Production Service Management Controls and separation of responsibilities

CCC = Cloud Competence Center

#	IP Whitelist	Vnet Integration	Write Location	Access to Azure Public	Access to Data explorer	Consistency Level
1	Single IP	Multiple Vnets	W.E.	Yes	Yes	Session
2	None	Single Vnet	W.E.	No	No	Session
3	None	Single Vnet	N.E.	Yes	Yes	Session
4	None	Single Vnet	W.E.	No	No	Session
5	Multiple IPs	Multiple Vnets	W.E.	No	No	Consistent Prefix
6	Multiple IPs	Multiple Vnets	W.E.	Yes	Yes	Strong
7	Multiple IPs	None	W.E.	No	Yes	Bounded Staleness
8	Single IP	None	W.E.	Yes	Yes	Strong
9	None	None	W.E.	No	No	Eventual
10	Single IP	None	W.E.	Yes	Yes	Bounded Staleness
11	IPs in CIDR format	Multiple Vnets	W.E.	Yes	Yes	Consistent Prefix
12	Multiple IPs / IPs in CIDR format	Multiple Vnets	N.E.	No	Yes	Eventual

Table 6.7: Different deployment scenarios used for testing .
W.E. = West Europe, N.E. = North Europe

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The aim of this thesis was to investigate which are the challenges of Cloud Computing adoption in an enterprise environment. After conducting the literature survey, the focus was narrowed down to 3 main scopes of challenges:

- Raising awareness and encouragement of the affected staff from the Cloud adoption
- Conducting qualitative and quantitative comparison of the performance of an sample application deployed on different service offerings by Microsoft Azure Public Cloud which helps the decision process for choosing services when migrating existing applications
- Enforcing security and compliance controls of the Microsoft Azure Cloud components in a centralized and reusable way

The thesis was conducted as part of an graduation internship in Rabobank, a Dutch multinational banking and financial services company, so these general challenges were addressed according to a use cases within the Cloud Competence Center in Rabobank.

The challenge of raising awareness was addressed by improving the demonstration of the capabilities of Microsoft Azure Cloud technology and Visual Studio Team Services to the other employees of the bank outside of the Cloud Competence Center. The purpose was to demonstrate how Cloud Computing technology integrates into the whole development cycle and can speed up delivery time. Furthermore this demonstration includes the automation of many processes while making the development process stable, predictable, repeatable. This, in turn, encourages experimentation since the risk in this way of developing is lower because changes can be reverted very easily and there are also many testing tools to test the outcome

of the changes in a similar or same environment before it is deployed to production enabling the teams to be more independent and be in complete control of their products.

The challenge of aiding the decision process for choosing services when migrating existing applications was addressed by conducting qualitative and quantitative comparison of the performance of an sample application deployed on different service offerings by Microsoft Azure Public Cloud. This was done by deploying of a sample .NET Core e-commerce application on 3 different service offerings of Microsoft Azure. The analysis was focused on the performance, cost, ease of deployment and life cycle management of the application.

The results showed that Azure Kubernetes service has the best performance both in terms of response time and number of successful requests. Moreover the increase of user load had the least effect in the increase of response time and increase of failed requests, while the number of successful requests also increased.

The results for the Windows Server VM showed that the deployment performs the same as AKS in terms of response time under load of 300 users, while the response time and number of failed requests increases much more when the load is increased to 1000 users. The load increase, however, did not affect the number of successful request which remained stable.

The results for the Web App showed that the deployment performs worse among the services compared. It has largest response time, with lowest number of successful requests served. The experimental testing showed that to achieve the performance of AKS or the Windows Server VM, the Web App needs to be scaled out to 3 instances. Moreover, the increase of load affected the Web App the most by increasing the response time and decreasing the number of successful requests.

In terms of costs comparison Web App service is the most expensive with costs of 292 dollars per month, followed by Windows Server VM with costs of 207 dollars per month, followed by AKS with 203 dollars per month.

The ease of deployment and lifecycle management of AKS and Web App is comparable and requires much less effort as it is Platform as a Service offering, compared to Windows Server VM which is Infrastructure as a Service offering and requires much more management by the end user.

The conclusion would be to choose AKS if one wants partially managed service as it demonstrated much better performance compared to the other PaaS offer (WebApp). Moreover, it is worth noting that Windows Server VM demonstrated comparable performance to that of AKS for 300 users load and worse performance for the load of 1000 users. If the application needs to scale out the recommendation would be to choose either AKS or Web App since these services are much easier to scale. For updates without downtime, the recommendation is to choose one of the PaaS offerings (AKS or WebApp).

The challenge of enforcing security and compliance controls of the Microsoft

Azure Cloud components in a centralized and reusable way was addressed by use case of implementation of Microsoft Azure CosmosDB as a VSTS extension. This was done by standardizing the components by:

- restricting certain configurations that may introduce risks
- delivering a component that has predictable behavior
- developed according to the security and compliance rules of both Rabobank and the Dutch government
- it is reusable in VSTS CI/CD pipeline
- can be automated to speed up delivery cycle, providing efficiency across the organization

Even though this is an example for just one component of the Microsoft Azure platform, the same research, design and development principles can be easily adopted for developing different Cloud Computing components according to the business and security requirements in the enterprise. The example implementation can be used as a reference for both structure and design decisions and the consequences of those decisions, since many configurations are similar across the spectrum of Microsoft Azure Cloud Computing platform.

7.2 Future Work

The scope of this thesis was narrowed down to three challenges of cloud adoption with use cases in the Rabobank enterprise environment. However, one should be aware that there is no universal solution to all of the challenges and different approaches may be used in different environments. The described approaches could serve as a starting point and a reference for solving similar challenges in different enterprise environments.

For the challenge of raising awareness, the scenarios developed cover only a small fraction of the full capabilities of Microsoft Azure and are not a fit for all the workloads developed by the teams in Rabobank. The idea behind the presentation of the particular use cases is to familiarize the staff with the way of working, cloud development principles, and the change of mindset by getting full control and ownership of your product development, compared with the dependencies and shared responsibility with the on-premise datacenter team.

Future work addressing this challenge could be to design more possible scenarios according to the needs of the development teams and try to identify general patterns that can be reused in other environments.

The performance analysis has its own shortcomings that need to be mentioned. The underlying infrastructure used was not identical, but comparable, and this may have influence on the final results. Moreover, the deployed application is just one

example of an application and not a general rule for performance of all applications. The applications may vary a lot, from the programming language that they were written in, to applications that are more compute, memory or I/O operations intensive and can be optimized to perform best with other underlying infrastructure. This performance analysis should serve as a reference if you are deploying a similar type of application.

Future work may include more tests that can be conducted with various types of applications and various types of underlying infrastructure to obtain even more detailed results, but that was not in scope of this master thesis. The applications may be generalized by type and testing can be done to optimize the performance of the application type on a certain infrastructure. For example there are Virtual Machines that are standard purpose, compute optimized, I/O operations optimized and experimentation with these configurations may give better insight on improving performance and costs savings.

Finally, the challenge of enforcing security and compliance controls was addressed through the example of CosmosDB implementation. However, one should also be aware that there is no one size fits all solution to a complex question like Cloud security and compliance, so every use case has to be assessed separately and designed and developed according to the requirements that apply for that case. The implemented methodology should be used as a reference, but not as a general rule for delivering Cloud components.

Future work in this problem domain may include analysis of more Cloud components delivered in this way and identifying patterns that can be generalized for use in different enterprise environments.

Bibliography

- [1] Thomas Ristenpart Sam Scott Adam Everspaugh, Kenneth Paterson. Key rotation for authenticated encryption, 2017.
- [2] Anurag Bejj. Cloud computing for banking and investment services. In *Advances in Economics and Business Management (AEBM)*, volume 1. Krishi Sanskriti Publications, November 2014.
- [3] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. pages 1–15. Springer-Verlag, 1996.
- [4] Terry Chia. Confidentiality, integrity, availability: The three components of the cia triad, 2012.
- [5] Mingchang Wu Chinyao Low, Ychsueh Chen. Understanding the determinants of cloud computing adoption. In *Industrial Management and Data Systems*, volume 111, pages 34–40. Emerald Group Publishing Limited, November 2011.
- [6] PCI Security Standards Council. Requirements and security assessment procedures, 2018.
- [7] Dr.S.B.Kishor D.S.Hiremath. Distributed database problem areas and approaches. In *IOSR Journal of Computer Engineering (IOSR-JCE)*, pages 15–18, 2016.
- [8] EduTechWiki. Technology-organization-environment framework, 2018.
- [9] Kuhn D.R. Ferraiolo, D.F. Role-based access controls. In *15th National Computer Security Conference (NCSC)*, page 554563, 1992.
- [10] Bernd Schumacher Heiko Gewald, Philipp Rieger. Cloud-computing in banking influential factors, benefits and risks from a decision maker’s perspective completed research paper. In *Nineteenth Americas Conference on Information Systems*, Chicago, Illinois, August 2013.
- [11] Gene Kin Kevin Behr, George Spafford. *e Phoenix Project*. IT Revoluton Press, 2013.
- [12] Vikas Kumar Kiran Kawatra. Benefits of cloud for banking sector. volume 5, pages 15–19. IJCST, IJCST, November 2014.
- [13] Microsoft. Add a build or release task, 2018.
- [14] Microsoft. Application monitoring and feedback loops, 2018.
- [15] Microsoft. Azure container instances, 2018.
- [16] Microsoft. Azure cosmos db documentation, 2018.
- [17] Microsoft. Azure kubernetes service (aks), 2018.
- [18] Microsoft. Continuous integration and continuous deployment, 2018.
- [19] Microsoft. Define resources in azure resource manager templates, 2018.
- [20] Microsoft. Devops testing, 2018.
- [21] Microsoft. Infrastructure as code, 2018.
- [22] Microsoft. Microsoft professional orientation : Cloud administration, 2018.
- [23] Microsoft. Overview of windows virtual machines in azure, 2018.
- [24] Microsoft. Partsunlimited lab, 2018.

- [25] Microsoft. Web apps documentation, 2018.
- [26] Microsoft. Web apps overview, 2018.
- [27] Ali Nafarieh William Robertson Muhammad H. Raza, Adewale Femi Adenola. The slow adoption of cloud computing and it workforce. In *3rd International Workshop on Survivable and Robust Optical Networks (IWSRON)*, page 1114–1119. Elsevier, 2015.
- [28] Maricela-Georgiana Avram (Olaru). Advantages and challenges of adopting cloud computing from an enterprise perspective. In *The 7th International Conference Interdisciplinarity in Engineering*, page 529–534. Elsevier Ltd, 2013.
- [29] The European Parliament and the council of the European Union. General data protection regulation, 2016.
- [30] Timothy Grance Peter Mell. The nist definition of cloud computing, 2011.
- [31] John Rudolph Raj Prashant Gupta, A. Seetharaman. The usage and adoption of cloud computing by small and medium businesses. In *International Journal of Information Management*, page 861874. Elsevier Ltd, August 2013.
- [32] Margaret Rouse. Noisy neighbor (cloud computing performance), 2014.
- [33] Arka Mondal Subhas Chandra Misra. Identification of a companys suitability for the adoption of cloud computing and modelling its corresponding return on investment. In *Mathematical and Computer Modelling*. Elsevier Ltd, March 2010.
- [34] Fleischer M. Tornatzky, L. G. *The processes of technological innovation*. Lexington Books, Lexington, MA, 1990.
- [35] STEPHEN WATTS. Saas vs paas vs iaas: Whats the difference and how to choose, 2017.
- [36] Wichmann Verlag. *Cloud Computing: The Next Revolution in IT*, Photogrammetric Week, Stuttgart, Germany, September 2009.
- [37] Adam Wiggins. The twelve-factor app, 2017.
- [38] Eunseok Lee Sungyoung Lee Won Kim¹, Soo Dong Kim. Adoption issues for cloud computing. In *iiWAS200*, 2009.