UNIVERSITY OF TARTU

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Institute of Computer Science

Software Engineering

Halil Ibrahim Karaca

Migration of an On-Premise Single-Tenant Enterprise Application to the Azure Cloud: The Multi-Tenancy Case Study

Master Thesis (30 ECTS)

Supervisors: Dr. Peep Küngas, Indrek Karu

Author:	Halil Ibrahim Karaca	"……. "	May 2013
Supervisor:	Peep Küngas	"……..."	May 2013
Co-supervisor:	Indrek Karu	"……..."	May 2013
Approved for defence			
Professor:	Marlon Dumas	"……"	May 2013

TARTU, 2013

ABSTRACT

The success of cloud computing is changing the way how information technology services are developed, deployed, maintained and scaled. This makes the 'migration to the cloud' a buzzword in the industry for most of the enterprises today. Observing so many advantages of this phenomenon technology, enterprises from small to large scales are interested in migrating their software applications, database systems or infrastructures to cloud scale solutions.

Migrating existing systems to a cloud scale solution can reduce the expenses related to costs of the necessary hardware for servers, installation of the operating system environment, license costs of the operating system and database products, deployment of the database products and hiring professional staff for keeping the system up and running. However, storing the application data to a back-end that serves multiple tenants on the cloud will be also costly if the resources on the cloud platform are not shared fairly among tenants. Thus, a carefully designed multi-tenant architecture is essential for an organization that serves multiple tenants.

In this master thesis, we will describe a case study and lessons learned on the migration of an enterprise application from an on-premise deployment backend to the Azure Cloud. More specifically, the thesis describes the migration of a productivity tool specialized for legal professionals to a multi-tenant data storage back-ends on Azure Cloud. Moving an on-premise, single-tenant software backend to a multi-tenant data storage system on the cloud will also require design and implementation of authentication mechanisms.

The core focus of the work consists of the design and implementation of a secure, scalable and multi-tenant efficient data storage system and application architecture on the cloud. SQL Database (formerly SQL Azure) offers native features (SQL Federations) for the secure isolation of the data among tenants and database scalability which has been used inside the project. Furthermore, the basic application authentication mechanism is enhanced with identity providers such as Google Account and Windows Live ID by embedding native functionality of Windows Azure called Azure Access Control Service to the login mechanism.

Migration of the software backend to a cloud scale solution is expected to reduce the costs related to delivery, deployment, maintenance and operation of the software for the business. Furthermore, it will help the business to target new markets since it is a cloud based solution and requires very little initial effort to deliver the software to the new customers.

ACKNOWLEDGEMENTS

I am heartily thankful to my supervisors, Peep Küngas and Indrek Karu, whose encouragement, guidance and support from the beginning to the end enabled me to develop an understanding of the subject and the project. While Indrek supported me on the company side from the technical aspects of the project, Peep was an excellent supervisor with his guidance on the methodological point of view and help on developing a vision on the subject.

I offer my regards and blessings to my parents, Hasan Karaca and Guluzar Karaca, my brothers Enes, Omer and Selim Karaca, and my colleagues Joosep Püüa, Olgun Cakabey, Allahshukur Seyidov who supported me in any aspect during the completion of the project.

Finally and most importantly, I am grateful to my wife Agnieszka Burda-Karaca, whom I married during the master thesis writing process, on constant support, encouragement and love.

Halil Ibrahim Karaca

Table of Contents

List of Figures

List of Tables

Abbreviations and Acronyms

ACS	Access Control Service
API	Application Programming Interface
AWS	Amazon Web Services
DB	Database
EU	European Union
IaaS	Infrastructure as a Service
IDM	Identity Management
ISACA	Information Systems Audit and Control Association
IT	Information Technology
NoSQL	Not only Structured Query Language
PaaS	Platform as a Service
REST	Representational State Transfer
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SLA	Service Level Agreement
SQL	Structured Query Language
TDS	Tabular Data Stream
UML	Unified Modeling Language
XACML	Extensible Access Control Markup Language
VM	Virtual Machine

1. Introduction

Cloud computing brings wide range of computing capabilities and adds value to its actors such as consumers of the software products, businesses and application developers. According to National Institute of Standards, cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [21].

The traditional software delivery methods of the enterprise applications that the database servers are installed on the customer's premises, database instances are created for every single of them and the network connectivity is being ensured between the servers and the database instances are not automated processes and they cost extra time, effort and money. In most cases the traditional deployment process of the enterprise applications require consolidation, setup and installation on the customer site by the IT staff of the software provider for every single new customer. Furthermore if the software product is not reasonably priced, costs that are related to the time spent for deployment, setting up and consolidation of the product may even exceed the total price paid by the customer for the product. Additionally, there are also costs that are related to the hardware and database server administration to keep the system up and running.

The success of cloud computing is changing the way how information technology (IT) services are developed, deployed, maintained and scaled. This makes the 'migration to the cloud' a buzzword in the industry for most of the enterprises today. Observing so many advantages of this technology, enterprises from small to large scales are interested in migrating their software applications, database systems or infrastructures to cloud scale solutions.

Cloud computing offers a new approach to the traditional methods where applications, data, platform and the infrastructure are all stored and serviced on the ‘cloud’ and are available on demand. This approach makes it available to deliver a range of low-cost applications, anytime anywhere using a terminal (web browsers, smartphone or any other device) to the consumers of software products, delivers business applications on the moment it is needed for business employees, decreases the hardware and software costs and provides a flexible, easy to maintain environment for application developers.

More specifically, with multi-tenant database systems on the cloud, service providers are able to offer expertise for consolidation, database management and scalable environments to small companies and individuals with less experience, resources and manpower for more affordable and reasonable prices. Not only for small organizations but also for the large enterprises, multi-tenant database systems on the cloud can reduce the expenses of buying the necessity hardware for servers, installation of the operating system environment, license costs of new environment both for operating system and database products, deploying the database products and hiring professional staff to run and administer the system.

The most important value of the multi-tenant systems on the cloud is that it can help enterprises to catch the “long tail” markets [18]. Moving from on-premise software delivery to Software as a Service (SaaS) structure will enable enterprises to change the way how they sell and maintain the software; their business model, the application architecture and operational structure. As a result the minimum cost at which software can be sold will reduce significantly. The per-tenant subscription fee reduction will make it possible for the software providers to target long tail markets where with traditional on-premise methods, the market could not be able to be targeted. As revealed, worldwide software-as-a-service revenue is forecasted to reach $14.5 billion in 2012, a 17.9 percent increase from 2011 with revenue of $12.3 billion, according to Gartner, Inc [19].

Cloud computing is considered today a mature technology but migration of existing applications to the cloud platforms may be challenging in terms of choosing the most appropriate method for data storage systems. The key challenge for the migration to the cloud is keeping the right balance between tenant isolation and the cost of providing dedicated resources. The shared amount of

resources among the tenants must be carefully identified. Since the major aim of cloud computing is decreasing the computational, hardware and operational costs, the method that has been chosen for migration, must decrease the costs and ensure that the solution operates without any problem. Usually on most of the cloud platforms, if the amounts of shared objects are increased, the total cost becomes cheaper. But on the other hand, increasing the shared objects (shared database instances, shared database tables etc.) among tenants also brings complexity and may be an obstacle in terms of scalability.

Keeping the preceding in mind, in order to design an efficient multi-tenant architecture on the cloud, there are several aspects to be taken care of. Since tenants are going to use some shared objects, isolation of stored data between tenants, authentication and authorization mechanisms must be carefully designed and addressed. Furthermore, management and monitoring, provisioning tenant resources and billing are other challenges to be addressed.

On the other hand, since shared structures are harder to modify individually, in the long run the multi-tenancy can be disadvantageous for the application's extensibility where businesses need specialized versions of the enterprise applications. For instance, there can be complex applications that need high computational resources. In such cases a multi-tenant data architecture may become less attractive, and a different architectural model on the cloud should be considered or migration to the cloud should not be considered at all.

The core focus of this master thesis project is to address the issues that can be faced while migration of enterprise applications from traditional on-premise deployment to multi-tenant data storage systems on the cloud. The goal is to describe a case study and lessons learned on the migration of an industrial single-tenant application from on-premise deployment backend to Azure Cloud.

There are several architectural styles, which can be taken into consideration while designing a multi-tenant data architecture on the cloud to provide a SaaS model for the target application. From an architectural point of view, there are some key differences between a well-designed SaaS application and a poorly designed one. These key differences are defined as scalability, multi-tenant efficiency and configurability by Microsoft [18].

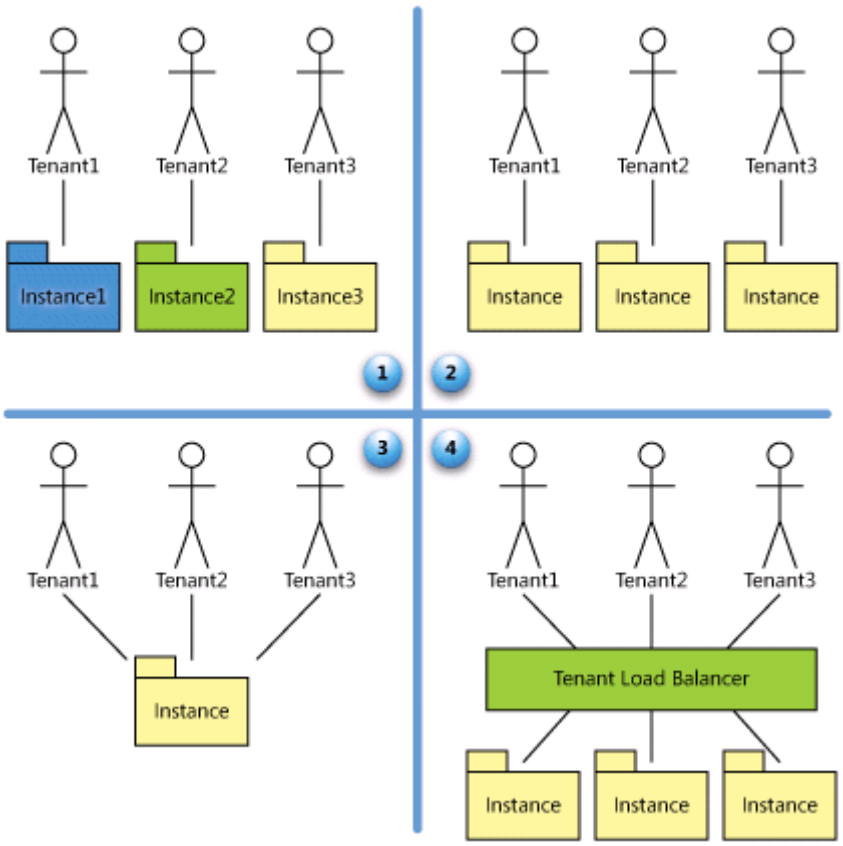


Figure 1. Maturity Levels of SaaS (from [18])

Architectural styles on the cloud platforms may be expressed using a Software as a Service maturity model with four distinct levels. Figure 1 illustrates the maturity levels of these architectural styles. Each level is distinguished with addition of one of those key differences mentioned above.

For creating data architectures to accomplish multi-tenancy on the cloud, three main approaches can be taken into consideration. These approaches are;

- Separate databases
- Shared databases, separate schemas
- Shared database, shared schema

Figure 2. Continuum between shared data and isolated data (from [22])

All of these approaches have its own advantages and disadvantages and there are technical and business factors to be taken into consideration while deciding the most appropriate approach for data storage architecture. These considerations are economic, security, tenant and regulatory considerations and will be extended in Chapter 3 of this thesis. Also each of these data storage approaches needs further attention to ensure data security, to create an extensible data model and to provide scalable data infrastructure. The common design patterns to cover these issues are going to be presented Chapter 3 as well.

The enterprise application to validate the objectives of this master project is a productivity, time tracking and billing software for legal professionals and is called LawTime [20]. The current deployment method of the product is setting up the database on the customer's premises and installing the software on customer's end-user workstations. The business model of the company for this product is to sell a lifetime and software assurances on a yearly basis for the customers who bought the license.

For a productivity software with traditional on-premise delivery methods, it is not even possible to offer a full featured 'trial version' that target customers may use and interact with before deciding to buy a license. This situation makes it quite hard to go outside the local market and target new customers for the company. Since the local market (Estonia) is considerably small for this niche product, it is already saturated after 10 years in the business. The yearly software assurances fees from the local market is not enough to keep the company alive, so migration of the product to a cloud based solution to reach new customers and new markets is vital for the company.

2. Background & Related Work

2.1. Cloud Computing

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service. The datacenter hardware and software is what is called a Cloud. When a Cloud is made available in a pay-as-you-go manner to the general public, it is called as a Public Cloud; the service being sold is Utility Computing. The term Private Cloud is to refer to internal datacenters of a business or other organization not made available to the general public. Thus, Cloud Computing is the sum of SaaS and Utility Computing, but does not include Private Clouds. People can be users or providers of SaaS, or users or providers of Utility Computing [25].

Cloud computing is emerging as a model in support of "everything-as-a-service" (XaaS). Virtualized physical resources, virtualized infrastructure, as well as virtualized middleware platforms and business applications are being provided and consumed as services in the Cloud [26]. There are many types of cloud computing but the service model of cloud computing can be grouped under 3 main topics. These topics are; Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

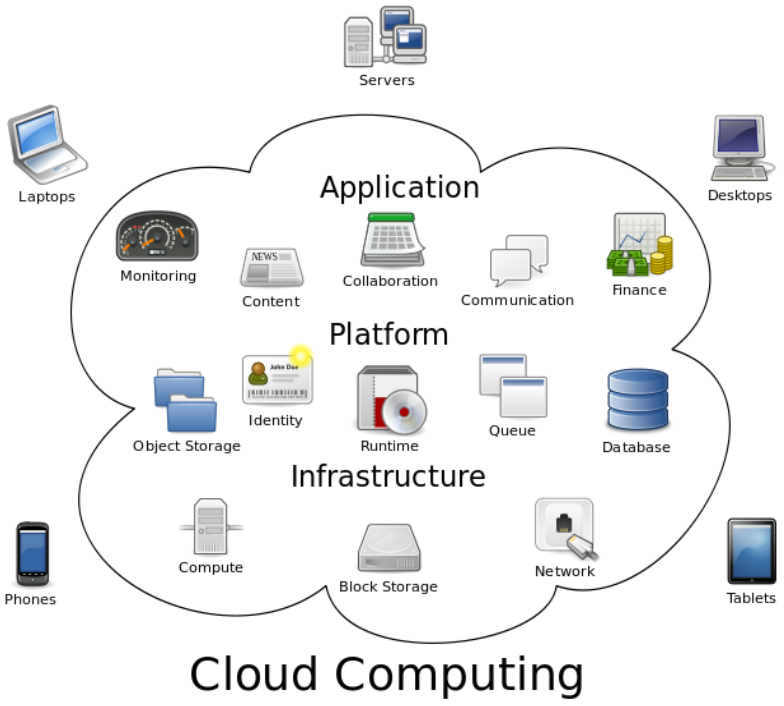


Figure 3. Cloud Computing Service Models (from [7])

SaaS focuses on separating the possession and ownership of software from its use. Delivering software's functionality as a set of distributed services that can be configured and bound at delivery time can overcome many current limitations constraining software use, deployment, and evolution. Such a model would open up new markets, both for relatively small-scale specialist-services providers and for larger organizations that provide more general services. In addition, service provision could include the dynamic creation and development of entirely new services that use existing ones [27].

All the applications that run on the Cloud and provide a direct service to the customer are located in the SaaS layer. The application developers can either use the PaaS layer to develop and run their applications or directly use the IaaS infrastructure [26].

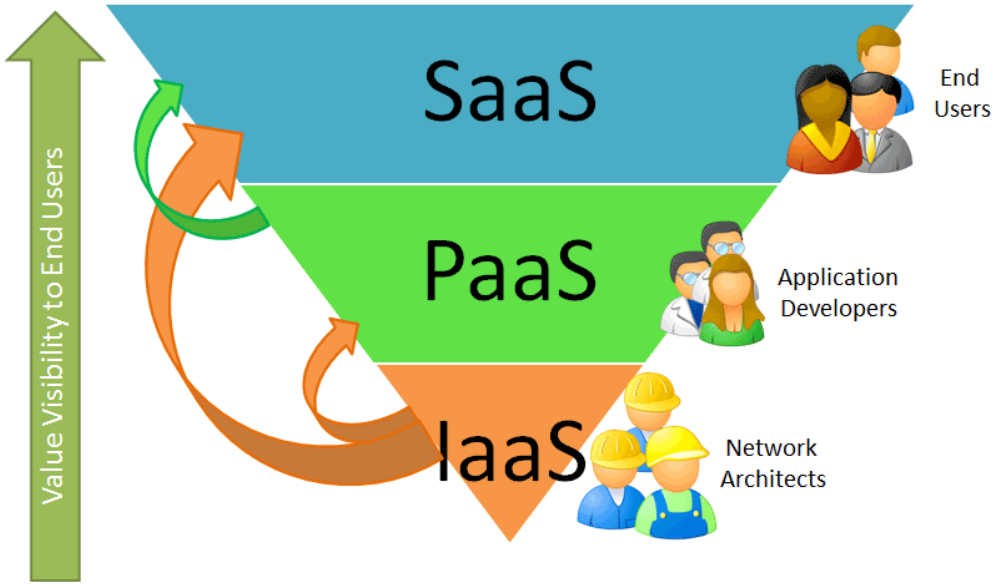


Figure 4. Services on the Cloud and Associated Actors (from [29])

PaaS is an application development and deployment platform delivered as a service to developers over the Web. It facilitates development and deployment of applications without the cost and complexity of buying and managing the underlying infrastructure, providing all of the facilities required to support the complete life cycle of building and delivering web applications and services entirely available from the Internet. This platform consists of infrastructure software, and typically includes a database, middleware and development tools [28].

Infrastructure as a Service is the delivery of hardware (server, storage and network), and associated software (operating systems virtualization technology, file system), as a service [28]. It is an evolution of traditional hosting that does not require any long term commitment and allows users to provision resources on demand.

Unlike PaaS services, the IaaS provider does very little management other than keep the data center operational and users must deploy and manage the software services themselves just the way they would in their own data center.

2.2. Cloud Service Providers

Cloud computing may be quite beneficial for all of the actors that are interacting with it. Thus, from the first emerging years of this technology, quite many of cloud service providers appeared in the business. A cloud provider is a company that offers some components of cloud computing.

Researchers on Merrill Lynch identified the companies with exposure to the cloud and some other promising cloud service providers [9]. According to mentioned report Amazon which is the first major cloud provider in the business, offers a wide range of Cloud computing services such as Elastic Compute Cloud (EC2), Simple Storage Service (S3), SimpleDB and Simple Queueing Service (SQS). Similar services are provided by other providers such as AppNexus Cloud [11], Bluelock Virtual Cloud Computing [12], ENKI Virtual Private Data Centers [13], FlexiScale Cloud
Computing [14], GoGrid Cloud Hosting [15], Joyent Accelerators [16], Rackspace Mosso Cloud Servers [17], and Terremark Infinistructure [18]. Other major cloud providers can be listed as Microsoft that offers Windows Azure as the main platform, Google, Apple, Cisco, Citrix, IBM, Rackspace and Salesforce where each of them has a different service focus.

While some of the cloud service providers aims providing all the services that is described as a service model on the cloud some others focuses on some specific model. Figure 5 illustrates the cloud service providers depending on their core focus.

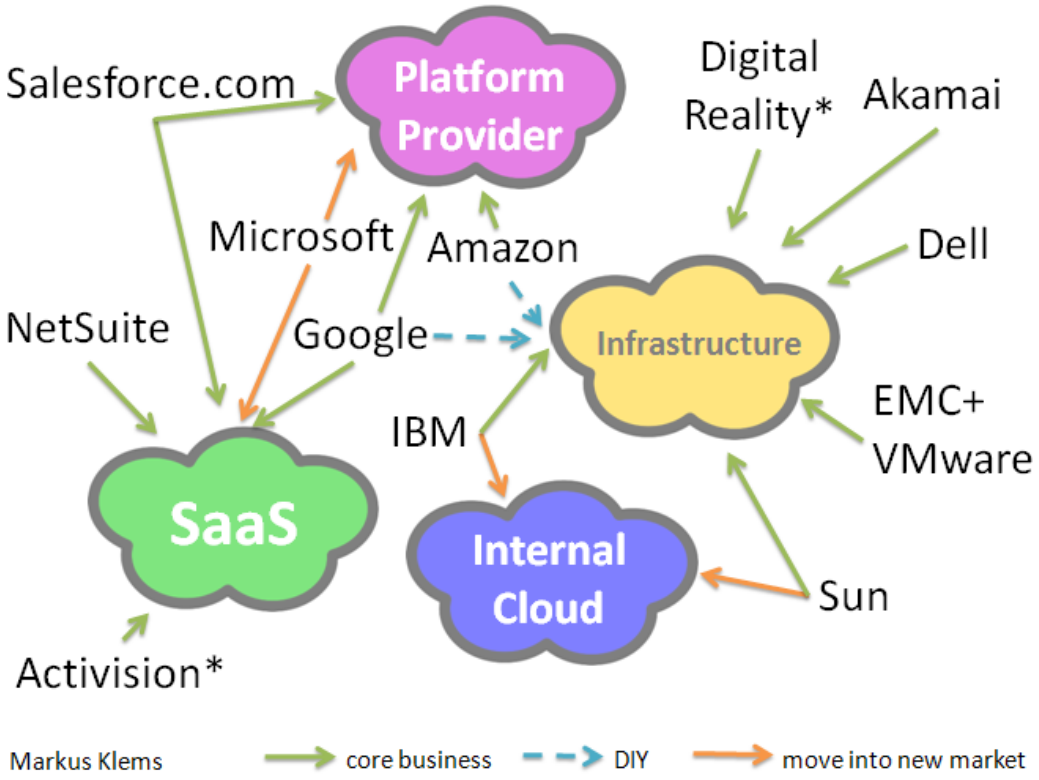


Figure 5. Cloud Service Providers (from [8])

In terms of data storage systems on the cloud the major cloud service providers can be considered as; Salesforce.com has Database.com that hosts relational database service, Microsoft offers SQL Azure Database as a standalone service, Amazon Web Services has its own Not Only Structured Query Language (NoSQL) cloud database service called SimpleDB and Google offers Google AppEngine Data Store.

Choosing the best cost effective cloud service provider for your application or service is an important issue. There are some factors while deciding appropriate cloud service provider. Cost of using the service usually based on a per-use utility model but physical location of servers may also be a very important factor. For example most of the European Union countries have established privacy protection as a national government function thus the servers that the data to be stored must physically be stored inside the European Union. In our specific case for this master project,

according to the Estonian privacy protection laws and data privacy compliance the cloud servers also should be stored inside the European Union.

Other important factors can be reliability and the security. In case of reliability, usually there is a cloud storage service level agreement between the provider and the customer where the system uptime is specified. The other important consideration is security and there are some organizations such as Cloud Security Alliance [23] that offers certification to the cloud provider that meet their criteria. If security is one the most important factor for the target application, these kind of certifications can be a selection criteria for the cloud service provider.

2.3. Windows Azure

Microsoft Windows Azure Platform is Microsoft’s cloud computing platform used to build and host cloud scale solutions through Microsoft data centers. Microsoft Azure is classified as "platform as a service" and forms part of Microsoft's cloud computing strategy. The platform consists of various on-demand services and it includes a number of features with corresponding developer services which can be used individually or together. Figure 6 illustrates the components and the capabilities of Windows Azure. The platform provides a cloud operating system that is called Windows Azure and it serves as a runtime environment for the applications. All these components and services can be grouped under 3 main core components; compute, data services and application services.

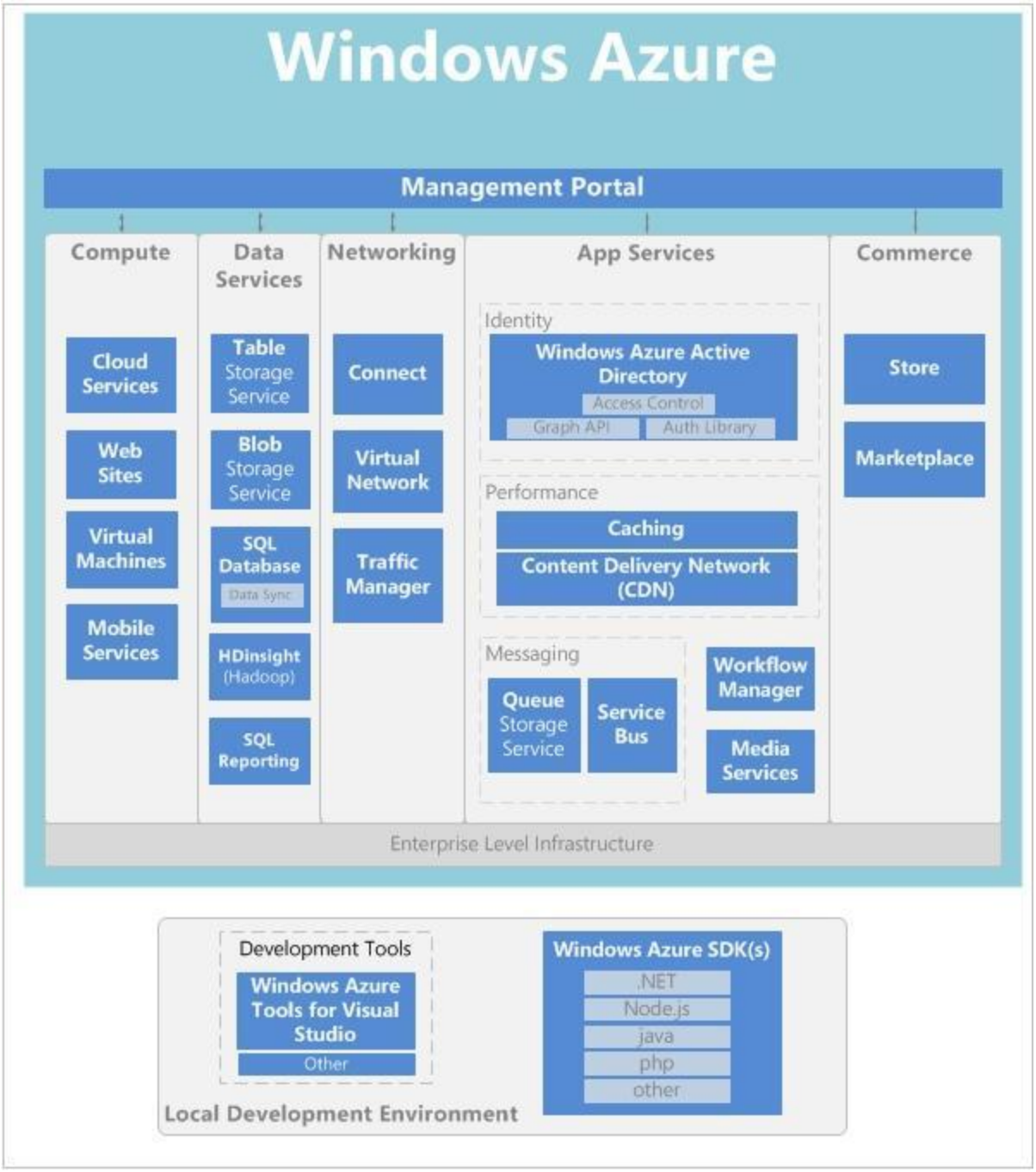


Figure 6. Windows Azure Components (from [32])

2.3.1. Compute

Compute provides a computational environment that consists of web sites, cloud services, mobile services and virtual machines.

Virtual Machines give the full control over a server in the cloud and maintain it for the business requirements. It is possible to choose between Linux and Windows Servers with various pre-installed additional applications. They are usually priced per hour. The virtual machine approach is commonly known as “infrastructure as a service”.

Windows Azure web sites offer deployment directly from the source code repository. It has a scale as you go model on the cloud platform across shared and reserved instances for better isolation and performance [32]. Windows Azure Web Sites is intended to be useful for developers. For development, it supports .NET, PHP, and Node.js, along with SQL Database and (from ClearDB, a Microsoft partner) MySQL for relational storage. The goal is to provide a low-cost, scalable, and broadly useful platform for creating web sites and web applications in the public cloud.

Cloud Services is for deployment and management of multi-tier applications where Windows Azure handles the details such as provisioning, load balancing, and health monitoring for continuous availability. It’s meant to focus on the application rather than infrastructure and offers %99.95 monthly SLA for the hosted applications [32]. It is possible to choose two roles to choose from when creating an instance of a cloud service; web role and worker role. The main difference between the two is that an instance of a web role runs IIS, while an instance of a worker role does not. Both are managed in the same way, however, and it's common for an application to use both. For example, a web role instance might accept requests from users, and then pass them to a worker role instance for processing. To scale applications up or down, it is possible to request that Windows Azure creates more instances of either role or shut down existing instances [32].

Mobile Services are designed to create functional mobile apps using Windows Azure. It is possible to streamline common development tasks like structuring storage, integrating push notifications and configuring user authentication [32].

2.3.2. Data Services

For data management Windows Azure offers multiple services on the cloud. These services are relational storage, scalable NoSQL tables, and unstructured binary storage.

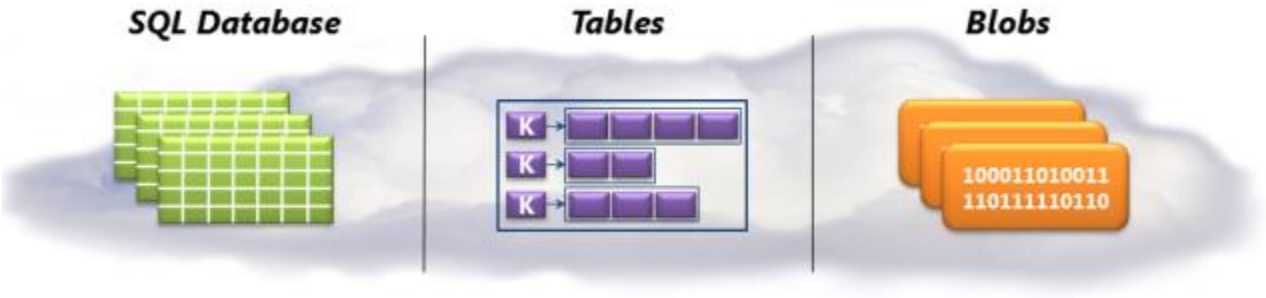


Figure 7. Windows Azure Data Storage Types (from [33])

SQL database is the actual implementation of Microsoft SQL Server on the cloud for relational storage. It is suitable for applications that require full featured database-as-a-service. Formerly called SQL Azure, SQL Database provides all of the key features of a relational database management system, including atomic transactions, concurrent data access by multiple users with data integrity, ANSI SQL queries, and a familiar programming model [33].

Tables offer NoSQL capabilities for applications that require storage of large amounts of unstructured data [33]. Tables are an ISO 27001 certified managed service which can auto scale to meet massive volume. The data is accessible from anywhere through RESTful API's.

Blobs, the third option for data management is designed to store unstructured binary data. It is suitable to store large amounts of unstructured text or binary data such as video, audio and images.

2.3.3. App Services

Windows Azure provides several services to control and optimize the applications hosted on the platform such as Identity, messaging and performance services.

Access Control Service provides an easy way of authenticating and authorizing users to gain access to the applications and services. Instead of implementing an authentication system with user accounts that are specific to the target application, it is possible to let ACS orchestrate the authentication and much of the authorization of the users. ACS integrates with standards-based identity providers, including enterprise directories such as Active Directory, and web identities such as Windows Live ID (Microsoft account), Google, Yahoo!, and Facebook [33].

Windows Azure Caching is to provision a cache in the cloud to be used from any applications or services that could benefit from caching. Caching provides several benefits to application developers. Caching increases performance by temporarily storing information from other backend sources. High performance is achieved by maintaining this cache in-memory across multiple cache servers. For a Windows Azure solution, Caching can reduce the costs and increase the scalability of other storage services such as SQL Database or Azure storage [51].

The Content Delivery Network caches Windows Azure blobs and the static content output of compute instances at strategically placed locations to provide maximum bandwidth for delivering content to users [33].

2.4. Windows Azure vs Amazon Web Services: A Comparison of Major Cloud Service Providers

Amazon AWS is the leader in the IaaS market and Windows Azure combines IaaS and PaaS together with the deep integration with on-premise Microsoft technologies with a unique value proposition. So the comparison between Amazon AWS and Windows Azure in the current cloud platforms market can make sense while choosing one instead of the other.

Among the two leaders in the cloud space, some businesses prefer Amazon Web Services, others

think Windows Azure is more convenient and when a decision is to be made, usually cost, available resources, development tools and ecosystem is taken into account. Basically the choice is between Iaas and Paas.

These two market leaders in the cloud business can be compared depending on the ease of building applications, performance, time saving, failover strategy, big data handling and costs. Table 1 summarizes the comparison results of the two cloud service providers.

AWS vs. Azure	Amazon Web Services	Windows Azure
Building Applications	Needs deep understanding of the whole platform.	Relatively easier for the average engineer.
Performance	Outperforms Azure in specialized applications.	Same as AWS when it comes to pure performance
Configuration & Time Saving	Most of the configuration must be manually done by the developer.	PaaS platform - Automatic configuration and updates of the platform.
Failover	%99.95 Availability	%99.95 Availability
Costs	Depends on the services used	Depends on the services used

Table 1. Amazon Web Services - Windows Azure Comparison

2.4.1. Building Applications

The development environment that Azure offers seems more user-friendly than Amazon Web Services. According to Craig Knighton of LiquidSpace, "It is easier for the average engineer to accomplish high availability on Azure because the tools push you down that path and make it pretty easy to build stateless applications." [34]. From a development point of view, if the migration is from a Microsoft environment where .NET is the main application framework, Azure seems beneficial for the application developers where Amazon Web Services can be beneficial for the Java and Python developers.

2.4.2. Performance

There are no data available for the comparison of the two cloud computing platforms for performance and probably both platforms are equal when it comes to pure performance. But Amazon Web Services may outperform Azure in specialized applications.

2.4.3. Configuration and Time Saving

From the rapid development point of view as a strong PaaS environment, Microsoft's Azure Cloud platform automatically configures, optimizes and updates the cloud environment while these mostly must be done manually by the developers in Amazon Web Services.

2.4.4. Failover

In the computing world it is never possible to provide a 100% uptime of availability. Both Windows Azure and Amazon Web Services offer 99.95% availability.

2.4.5. Costs

While the main purpose of migration of existing systems to the cloud is decreasing the operational and hardware related costs the cost of using the cloud platform depends what type of services are going to be used. Thus it is not possible to specify one is cheaper than the other. Both of the platforms allow users to utilize a free usage tier to test out the platform.

2.5. Multi Tenancy on the Cloud

There have been several efforts to provide and define efficient multi-tenancy on the cloud [3, 4]. Stefan et. al. [4] introduced a new schema-mapping technique for multi-tenancy called Chunk Folding, which extends the common practice that maps multiple single-tenant logical schemas in the application to one multi-tenant physical schema in the database. The weak point of this approach in terms of scalability is defined as the number of tables that a database can handle is limiting the scalability. Authors proposed a technique where certain tables shared among tenants and the certain tables mapped into fixed generic structures. This work mainly focuses on handling very large (in terms of schema level heterogeneity) databases that may contain more than 100.000 tables for each tenant. Compared to our case, where the amount of tables are around 200, this technique may be an unnecessary effort where scalability and performance is not as important as these database structures that this work focuses on.

In the process of migrating a single-tenant database structure to a multi-tenant one, there are 3 popular methods depending on the workload, sensitivity and some other parameters of the data to be stored. These methods are distinguished depending on the 'shared' objects. The shared object can be the virtual machine, database instance or the table. These methods are listed as independent database and independent database instances, independent tables and shared instances and shared tables and shared database instances [3]. All of these methods provide multi-tenancy on the cloud based services but each of them has pros and cons. In our case the storage method may be multiple where very sensitive large tenant's data stored in an independent database and independent database instance and where tenants with rather small amount of data and users, may be grouped and use shared tables and shared database instances. Figure 8 illustrates the architectures of multi-tenancy structures.

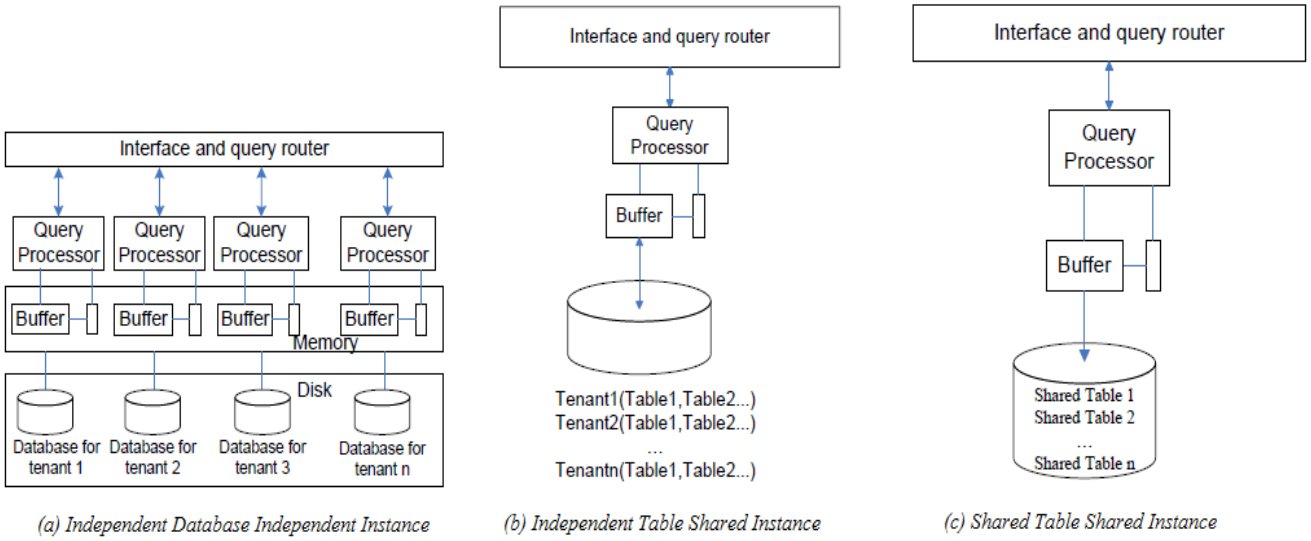


Figure 8. Types of Multi-Tenant data storage systems (from [3])

The database services such as Microsoft SQL Azure and Amazon RDS are promising in terms of establishing the market need, but currently all of these services are lacking to provide efficient multi-tenancy, scalability and database privacy [1]. There have been efforts [1] in order to address these 3 issues with providing a transactional database as a service called "Relational Cloud". The goal in multi-tenancy is to lower the hardware and maintenance costs and to meet the application level query performance. While addressing the multi-tenancy, Carlo et. al. [1] experienced packing each individual DB instance into a VM and multiple VM's on a single machine. These approach required more hardware and delivered less performance. Instead they used a single database server on each machine and hosted multiple databases and used a non-linear optimization formula while determining which database should be placed on which machine. However in this work the shared table for multiple tenants is not being considered as a method.

2.6. Architectural Choices for Multi-Tenant data

There are three main approaches that can be taken into consideration while choosing a multi-tenant data model. The optimal approach for choosing a data architecture for multi-tenant data storage systems on the cloud depends on some sort of technical and business considerations. Each of these considerations is extended in the corresponding topics.

2.6.1. Separate Databases

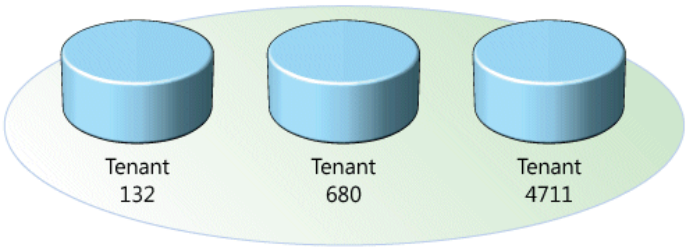


Figure 9. Separate Databases (from [22])

In separate databases approach each tenant has its own separate database. This makes it easy to extend the application's data model to meet tenant's individual needs and recovering the database in case of a required backup. But since each tenant has its own database this approach leads to higher costs.

2.6.2. Shared Database, Separate Schemas

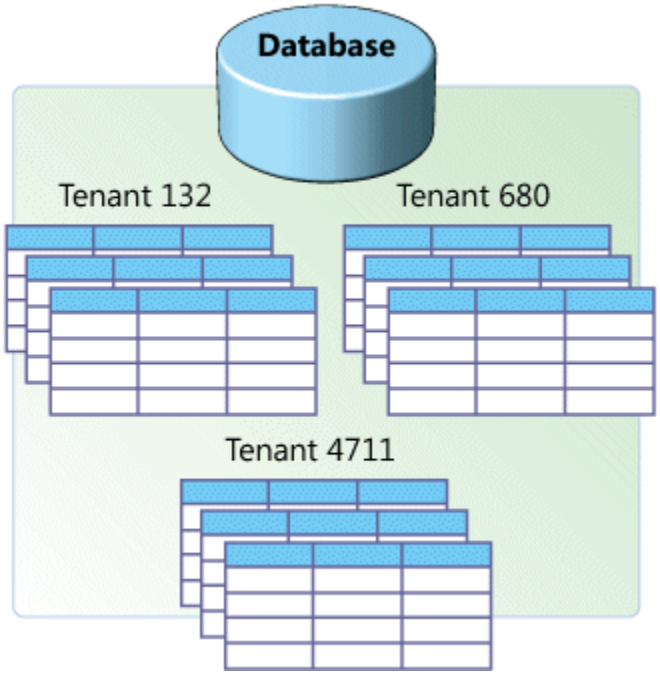


Figure 10. Shared Database, Separate Schemas (from [22])

Hosting multiple tenants in the same database, with each tenant having its own set of tables is the second approach. This approach is also relatively easy to implement compared to shared everything approach. The data model extension is also easier for each particular tenant.

As stated in the Microsoft blog [22];

> *A significant drawback of the separate-schema approach is that tenant data is harder to restore in the event of a failure. If each tenant has its own database, restoring a single tenant's data means simply restoring the database from the most recent backup. With a separate-schema application, restoring the entire database would mean overwriting the data of every tenant on the same database with backup data, regardless of whether each one has experienced any loss or not. Therefore, to restore a single customer's data, the database administrator may have to restore the database to a temporary server, and then import the customer's tables into the production server—a complicated and potentially time-consuming task.*

if this approach is being chosen for multi-tenant architecture, there should be failover strategies in place to recover from unexpected failures.

2.6.3. Shared Database, Shared Schema

The third approach is using the same database and the same tables to host multiple tenant's data. Since this approach allows serving the largest number of tenants per database instance, it has the lowest hardware and backup costs compared to the other two approaches. But it requires additional development effort to ensure the data isolation and security among tenants.

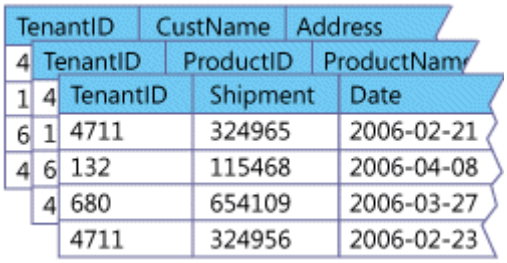


Figure 11. Shared Database, Shared Schemas (from [22])

This approach is appropriate when it is important that the application should be capable of serving a large number of tenants with a small number of database servers.

2.7. Security in Multi-Tenant Applications on Cloud Platforms

The potential benefits and business opportunities that cloud computing platforms could bring resulted as cloud migration being desirable by most of the enterprises today. However there are several concerns that may affect the decision of cloud migration [35]. Some of these factors are performance dropouts, uncertain availability, and vulnerability to network attacks or cloud service provider lock-in. The other more relevant consideration for the scope of this work is the security in the PaaS environment where multi-tenant applications are hosted.

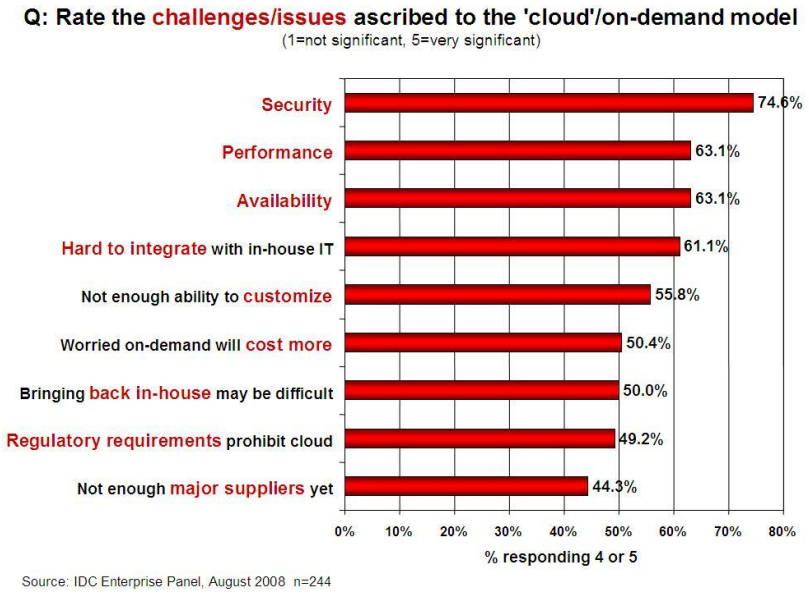


Figure 12. Rating of challenges/issues for cloud adoption (from [36])

The potential users of the clouds are skeptical if their data and the applications are going to be securely hosted. This issue has always been the main concern for the potential cloud users when it comes to cloud adoption. Figure 12 shows the result of the survey that carried out by International Data Corporation in 2008 [36] where security is the main concern for cloud. If clouds are perceived as risky environments enterprises may be unwilling to migrate their systems and host their data and applications there.

In a PaaS environment like Windows Azure, cloud components from different users can be run in the same platform. Malicious activities can try several straightforward ways to interfere with the normal execution of other components. Takabi et al. [38] specified in his work that providers are responsible for isolating components so that no user software can interfere with other users. Thus, software providers (customers of cloud service providers) are responsible for protecting the applications they build on the cloud. Cloud service providers are then responsible for isolating the customer's applications and workspaces (e.g. Virtual Machines) from one another.

There are potentials for large-scale attacks via the cloud platforms. Chow et al. [41] described the concerns on the cloud in his work by categorizing security concerns into three category; traditional concerns, availability and third party data control. Researchers of Gartner [42] published seven

security risks such as risks associated with data location and segregation - recovery and long-term viability. The European Network and Information Security Agency identified [39] a list of 35 issues related with the cloud computing in 4 categories. Organizations such as ISACA and Cloud Security Alliance published guidelines and best practices to mitigate the security issues in the cloud [43, 44].

In order to avoid confusion, it must be clarified that there are systems when cloud is used as Infrastructure as a Service and there are no shared resources that all the software instances are deployed in a unique environment per user which is hosted in non-shared machines. In these systems it is the cloud service provider who is responsible of implementing proper isolation of the virtual machines which has its own challenges mentioned by Ristenbart et al. [40]. The scope of this work does not deal with security issues such Infrastructure as a Service environment where implementation of secure isolation is handled in VM level by the cloud service provider.

2.7.1 Security Risks Associated with the Multi-Tenant Software Systems

The most important classes of cloud-specific risks relevant to the scope of this work can be listed as vendor lock-in, isolation failure and third party control which are also identified by the European Network and Information Security Agency in their report [39].

2.7.1.1 Lock-in

Cloud service providers currently do not offer in the way of tools, procedures or standard data formats or services interfaces that could guarantee data, application and service portability. This can make it difficult for the customer to migrate from one provider to another or migrating data and services back to an in-house IT environment [39]. As a result a situation can arise in which the user becomes locked-in to a particular vendor. This can be due to a difficulty in migrating data to a new vendor.

2.7.1.2 Isolation Failure

Multi-tenancy and shared resources are the most important characteristics of cloud computing. This risk identifies the failure of mechanisms separating storage, memory, routing and even reputation between different tenants [39].

2.7.1.3 Third-Party Control

Third party control is probably the prime cause of concern in the cloud. With the growing value of corporate information, third party access can lead to a potential loss of intellectual property and trade secrets. There is also the issue of a malicious insider who abuses access rights to tenant information which is mentioned by Atayero and Feyisetan [45].

2.7.2 Security Approaches

Providing a trustworthy cloud computing environment requires responsibility from both cloud service providers and the cloud users. To provide a secure, multi-tenant environment, data access and protection, application deployment and access policies must be accounted by the cloud service providers. On the other hand customers are responsible for protecting the applications they build and run on the platforms. Takabi et al. [38] identified various challenges and discussed the approaches for each of the challenges to address security and privacy requirements of cloud service providers and service integrators. The approaches to address the challenges can be examined as follows;

2.7.2.1. Authentication and Identity Management

The end users of the cloud can access their data and application from various places and they must be able to export their digital identities and securely transfer them to any kind of platforms. User centric Identity Management (IDM) has received attention for handling the private identity attributes.

In user-centric IDM approach, the system properly maintains the semantics of the context of user's identity information. In order to build a user-centric federated IDM for the clouds, the IDM services should be able to be integrated with an enterprise's existing authentication management framework [46] .

2.7.2.2. Access Control Needs

Using credential or attribute based policies such as role-based access control (RBAC) can capture various policy requirements and meet the policy integration needs because of their policy neutral natures. RBAC has been accepted as the most promising model with its simplicity, flexibility in capturing dynamic requirements, and support for the principle of least privilege and efficient privilege management [47].

2.7.2.3. Secure Interoperation

In order to build a comprehensive policy management in cloud environments several researches have been focused on multi-domain access control policies [43, 48]. To integrate access policies of different domains and define global access policies, secure integration and policy engineering mechanisms have been addressed with previous researches [49, 50].
Additionally, to ensure that the cross domain accesses are properly specified and verified there is a necessity of global specification frameworks such as Security Assertion Markup Language (SAML), Extensible Access Control Markup Language (XACML) and Web service standards [50].

2.7.2.4. Securing Database Tables via Tenant View Filter

Implementing just a secure authentication mechanism is not enough for ensuring the data security on the cloud for the application. In other words, the security among the tenants must be ensured and none of the tenant's data may interfere with another one. One approach for this problem is using SQL views that can be used to grant individual tenants access to some of the rows in a given

table, while preventing other tenants from accessing other rows [52]. Views can serve as security mechanisms by restricting the data available to users. Some data can be accessible to users for query and modification, while the rest of the table or database is invisible and inaccessible.

A view is a virtual table that is defined by the results of a SELECT query In Microsoft SQL. This virtual table than can be used and queried as if it was an actual data table. After creating SQL views via parametric SQL statements (using TenantID as filtering parameter) only rows belonging to the particular tenant are available. Each individual tenant's data access account would be granted permission to use the created view, but granted no permissions to the source table itself. This is an appropriate way to secure tenant data in a shared-schema application, in which multiple tenants share the same set of tables [51].

3. Architecture

3.1 The 4 + 1 Architectural View

4 + 1 is an architectural view model that is designed and proposed by Philippe Kruchten for describing the architecture of software-intensive systems, based on the use of multiple, concurrent views [24]. This view model aims to address the issues and concerns of the possible actors that may involve in software systems. This approach proposes a rather generic view to handle functional and non-functional requirements separately by dividing the total work into four conceptual view.

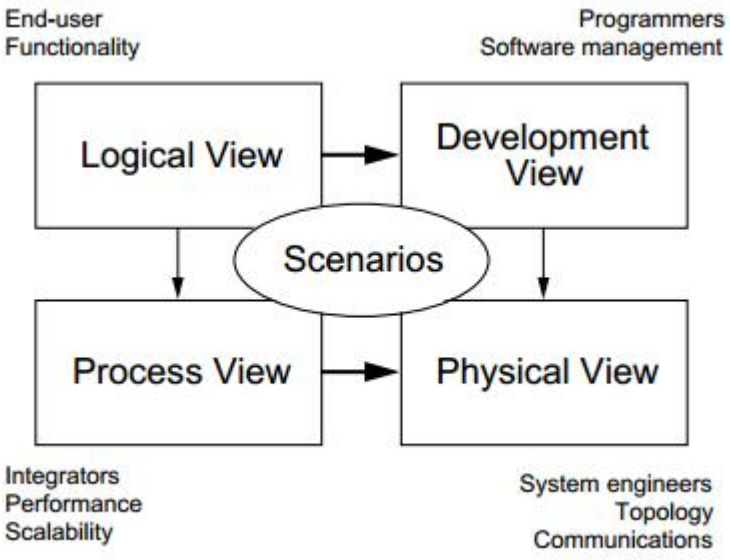


Figure 13. 4 +1 Architectural View (from [24])

Logical view defines the logical architecture of software systems to solve the functional requirements. It includes important classes and class relationships inside the system. The viewer of the system is the end user and it considers what the system should provide in terms of services to its users.

Process view describes the running processes or the instantiated objects exist in the system with using multiple level of abstraction. It covers non-functional requirements of the design such as concurrency, synchronization, performance and scalability aspects.

Physical view shows the deployment scenarios and associated hardware. It describes the physical hardware that run the software and how components are deployed onto those machines and configured in run-time. The view considers the non-functional requirements of the underlying hardware such as topology and communication.

Implementation view describes the layered structures of the software and defines the responsibilities of each layer in the system. The viewers of this part are software developers and this view considers the modular organization of the software such as the hierarchy of layers, software maintenance and the development considerations.

Finally the plus one in 4 + 1 view model defines the possible scenarios that the architecture should satisfy. These scenarios consist of the important requirements that the system should satisfy. The other four views are centered on the defined scenarios that are the most important to be solved.

Considering the 4 + 1 approach makes perfect sense in the design process of the architecture for software systems on the cloud based solutions. Moving from on premise software delivery methods to software as a service requires enterprises to shift their thinking in most of their functional areas.

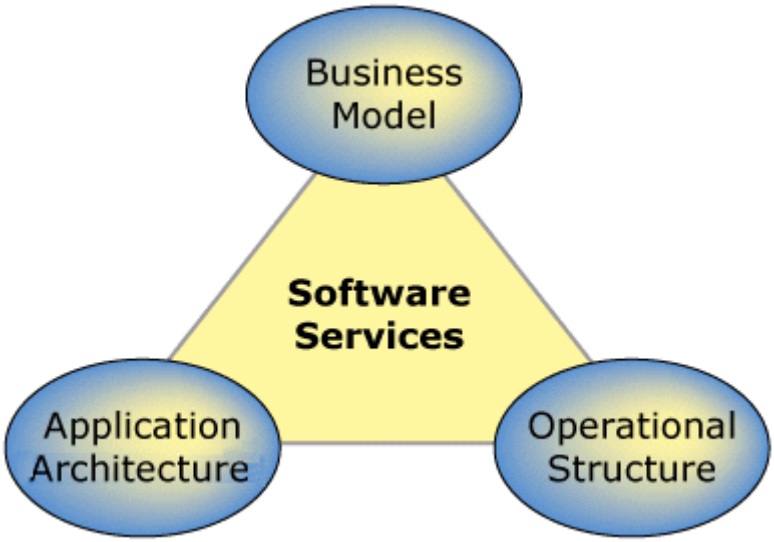


Figure 14. Areas where Software Providers should shift their thinking [18]

Figure 14 gives an overview of the areas where software providers should shift their thinking. All these areas consist of various actors from the organizational structures around the system and shifting from existing methods to cloud approach includes several functional and nonfunctional requirements for each of these areas. Separating the whole process into architectural view models will provide a better overview and decrease the complication of the migration process. Our core focus for the Cloud migration process is the application architecture.

3.2. Tailoring 4 + 1 Architectural View for Cloud Migration Process

There are some factors specific to cloud that may require further tailoring of views to fully represent a cloud system [30]. Figure 15 shows the four architectural view that is tailored for our cloud migration process. Since we use Windows Azure platform the infrastructure in the deployment view and the required processes in the process view is already provided by the cloud service provider. But logical view and the implementation view requires further exploration.

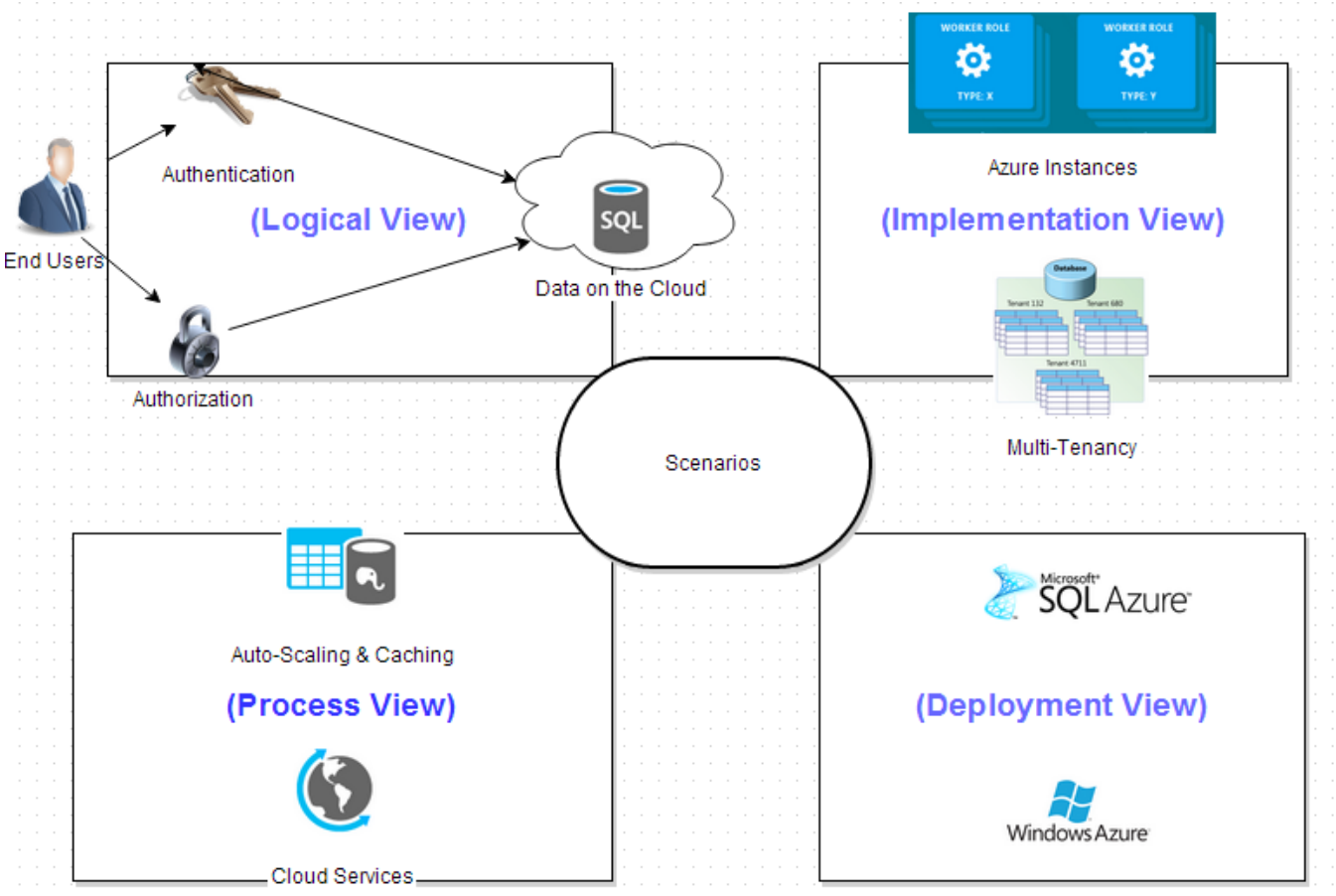


Figure 15. 4 + 1 Architectural View model for Cloud Migration Process

3.2.1. Logical View

In most cloud migration cases, the **logical view** where the functionality of the system from the perspective of end users, does not have significant changes. Usually in cloud migration process the back-end of the software -especially data storage systems- is the most important concern for software providers. In our specific case our aim is to shift the data storage mechanism from on-premise delivery method where every tenant has their own databases to cloud based data solutions with efficient multi-tenancy. Since our tenants (and the end users) are going to reach the same platform to access their data storage on the cloud, we need some sophisticated authentication and authorization mechanisms. We can define these mechanisms in the logical view, because the end users are going to have a new service where they can access their data from anywhere, anytime.

Currently end users authenticate to the application with their domain account through integrated security or by their manually created username and password inside their domain. In order to reach the application, end user must be inside the company network where the database servers are located. This current architecture does not require any further mechanism for authentication but when the data architecture migrated on the cloud there is a necessity for an extended authentication mechanism.

Figure 16 extends the logical view and presents an overview for the authentication mechanism as UML communication diagram. The cloud service provider (Windows Azure) has an Access Control Service to authenticate users from identity providers like Microsoft account, Google or Facebook.

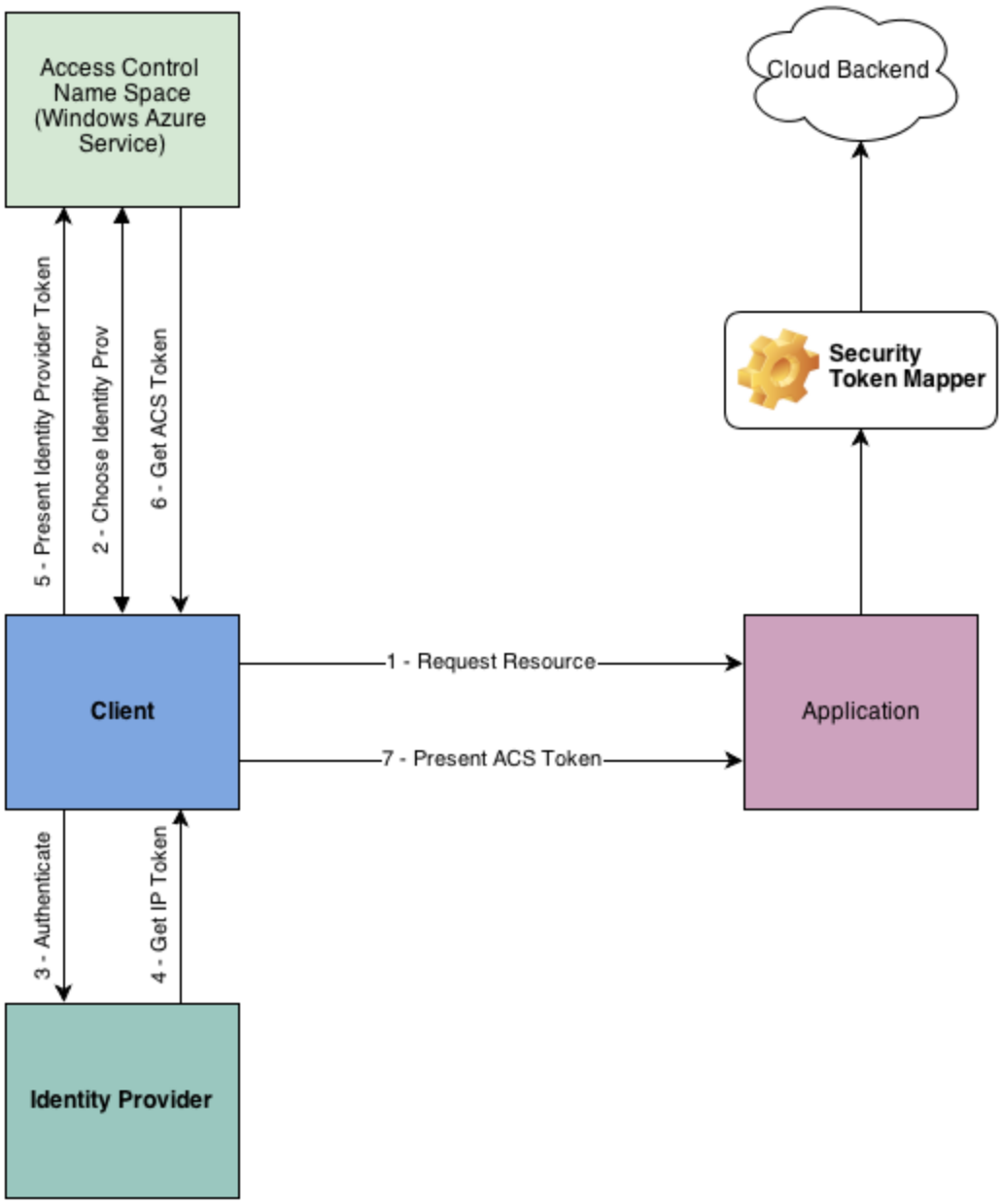


Figure 16. Logical View (Authentication Mechanism)

.

The client is the end user application that is trying to gain access to the data and the application interface in this context.

The Application is the middle-layer that is a service between the Cloud back-end the authorization mechanism to map the authenticated users with their organization ID to their data storage view in the cloud back-end.
The Identity Provider is an authority that authenticates user identities and issues security tokens. Typical examples of IPs include Microsoft account, Facebook, business user repositories (like Active Directory), and so on [31].

The authentication mechanism consists of the following steps (see Figure 10.);
1. For a successful authentication the client should make the request to reach the application to the Application middle-layer.
2. In order to authenticate the request the application redirects the user to the Access Control Service (ACS). ACS presents to the user the possible Identity Providers to choose an appropriate provider.
3. The application redirects user to the Identity Provider authentication module.
4. After the identity credentials are entered, Identity Provider issues a security token
5. Identity Provider sends this token to the Access Control Service.
6. ACS validates the security token issued by the Identity Provider, inputs the identity claims into ACS rules engine and calculates the output identity claims and issues a new security token that contains these output claims.
7. Finally client sends the new security token issued by ACS to the application middle-layer. ACS provides programmatic access to the token mechanism and the middle layer should take care of the transformation logic between the application and the cloud database backend.

3.2.2. Implementation View

The other important concern for our specific case is the implementation view where we adopt a layered structure defining 4 layers. Each of these layers contains various components. All the components have different responsibilities. Figure 17 represents the layered UML Component Diagram to reflect implementation view. For the cloud migration process all these components should be integrated, implemented from scratch or the existing methods should be modified.

In the **Application Layer**, the Authentication component represents the implementation necessities and changes in the authentication module in the actual software. Moving from on premise single tenant back-end to multi-tenant data storage on the cloud requires a different approach for authentication mechanism. In the single-tenant architecture one type of tenant's end users are connecting to their own local data store where Windows Authentication and in house authentication methods are easily applied and used securely. But in the multi-tenant data storage mechanism, there is a necessity of distinguishing each tenant and tenant's end users which is already mentioned in the logical view. For handling tenants in the cloud back-end, the proposed method is to create a separate database instance for customer data where data such as TenantId and the credentials of the tenant's users are stored to be matched from the identities that are mapped from the identity tokens. Then each end user may have a session with their accurate TenantID to view only their own data.

The Application component represents the necessary changes on the actual software. Because in the current architecture, there is no designated data access layer and the application directly queries the database. Moving to the cloud solution and using a shared data approach will require the transformation of all in application hard-coded SQL queries to have the ability to filter the tenant identifier.

The data access layer, represent the layer between user terminals and the data storage back-ends. The SQL Federations component is a native solution that is a service offered by Windows Azure on the cloud and addresses the issues about data access and secure data isolation between tenants. Furthermore it plays a vital role for horizontal scalability of data storage.
The SQL Azure Gateway handles connection pooling and acts as a proxy, forwarding the Tabular Data Stream (TDS) requests to the logical server. It also acts as a security boundary providing login validation, enforcing the firewall and protecting the instances of SQL Server behind the gateway against denial-of-service attacks.

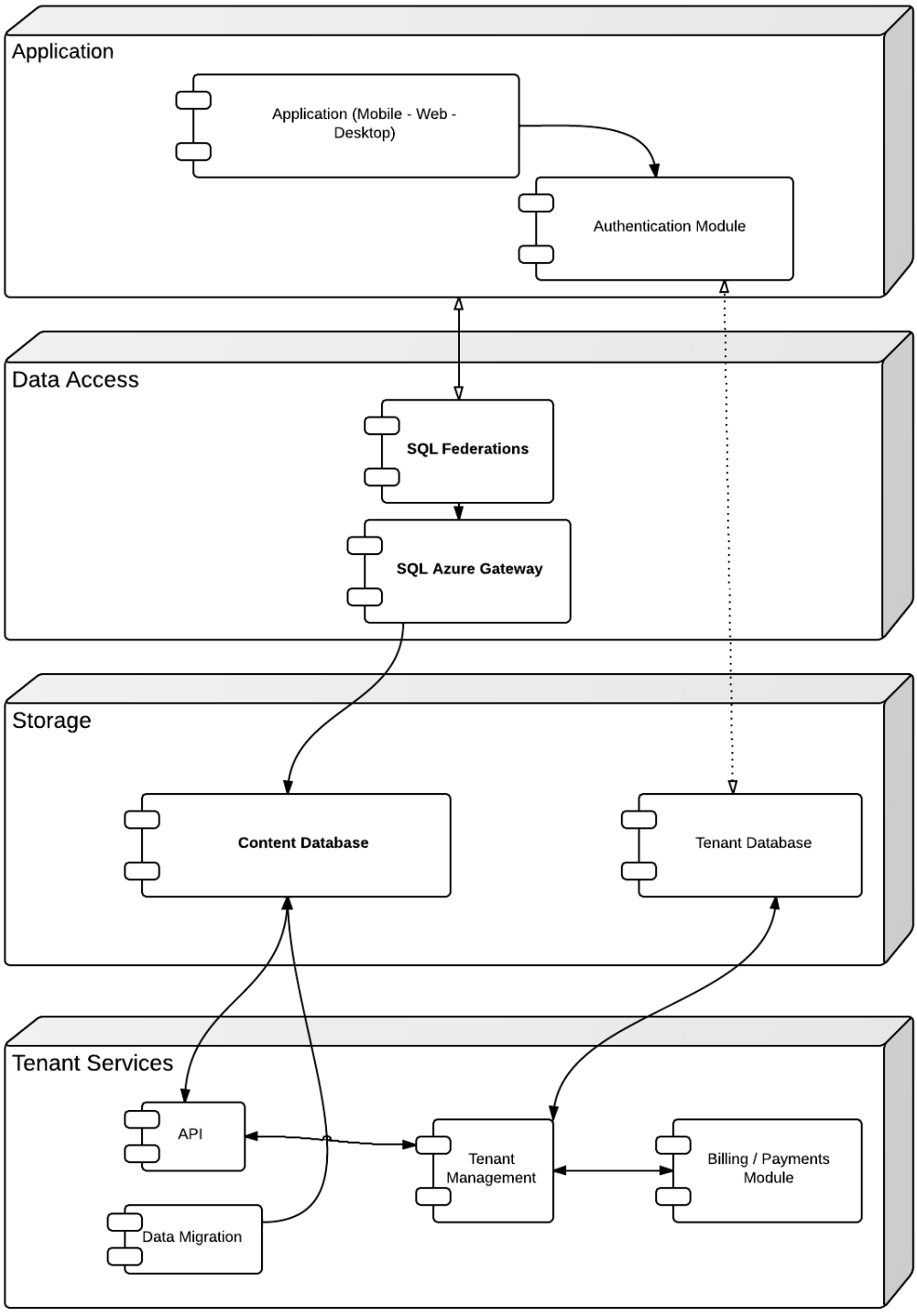


Figure 17. Implementation View - UML Component Diagram

Tenant data and the Customer data components in the **Storage** layer represent the database instances and database tables that are stored on Azure cloud. In the tenant data component, the all the database tables must be extended with a TenantID column because they are supposed to store multiple tenant data instead of a single one. The Customer data component should be created from scratch to store the information about tenants such as their TenantID, associated credentials, subscription mode, enabled status etc. The customer data will also be a back-end data structure for tenant management.

Tenant Services layer is about management and data services for tenant. This layer interacts with the storage layer and independent from other layers. The tenant management component is for controlling tenant access as a whole such as limiting the user access after the trial period or suspending the user access if the subscription is finished. The billing and payments component offers payment methods to customer and associate payments with the tenant data on the cloud for extending the subscription or notifies the customers just before their subscription is about to be terminated and must be renewed.
The API component is for letting customer to access their data programmatically via a RESTful interface. The data migration module is necessary for the first time migration of the actual data from on-site database instances to the data storage on the cloud.

3.2.3. Scenarios

3.2.3.1 Scenario 1 - End User Using a Social Identity for Authentication

In this scenario an end-user would like to use his / her computer outside the company domain e.g. at home. The steps of the scenario for authentication using a social identity (in this example it's Google) is described below. According to this scenario end user starts with presenting the credentials and ends up being authenticated.

a. Presenting the Credentials

- End user opens the application and the application automatically makes a request to the federation provider that is Windows Azure Access Control Service.
- The ACS list the Identity Provider that it trusts.
- End User selects the appropriate issuer. (Google ID)
- ACS redirects end user to the Google issuer.
- Google verifies end-user's credentials and returns a security token that includes information such as email and username.

b. Transform the claims.

- ACS converts the token issued by the identity provide and copies the claims issued by Google into the new token.

- ACS returns the new token to end user's application.

c. Map the Claims

- The application middleware applies token mapping rules to the ACS security token. These rules transform the claims into claims that the LawTime application can understand.

d. Transmit the Mapped Claims and Perform the Requested Action

- The application reads the claims and creates a session to for database table views for the end user with the associated tenant id previously. It can use identity information from the token to determine what kind of data end user can see in the application.

3.2.3.2 Scenario 2 - Billing the Customers

Moving from on premise software delivery to a cloud scale solution also requires a shift in business model. Currently software is sold with a lifetime license and owned by the customer. This model evolves from owning the software and having a lifetime license to a subscription business model. Since the software is going to be used as a service, a billing mechanism is necessary to charge the customers on a yearly or monthly basis.

In this scenario instead of selling a lifetime license for the software per customer per user, customers are going to be charged on a monthly or yearly basis per user.

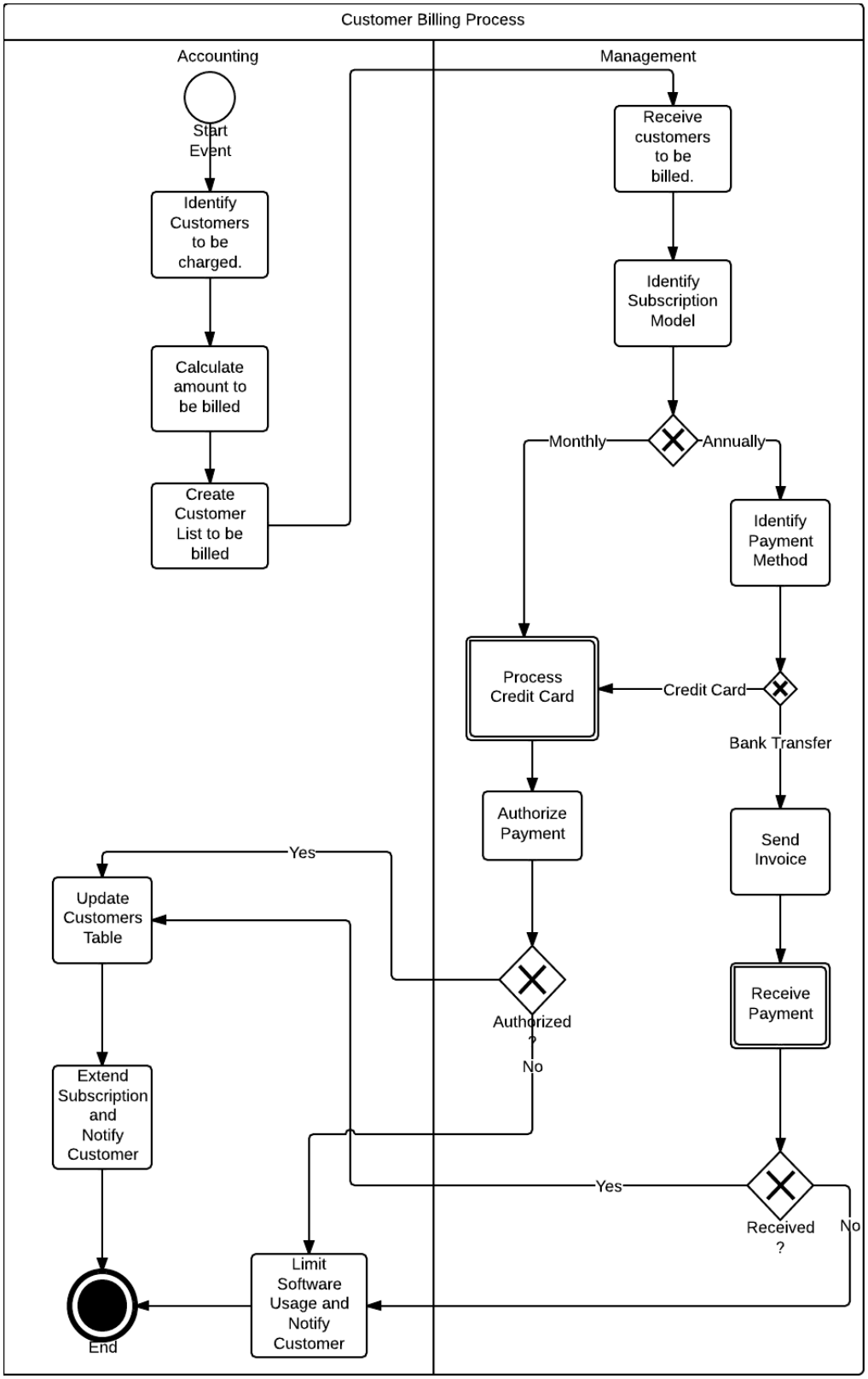


Figure 18. Customer Billing Process

Figure 18 shows the business process diagram of customer billing process. The billing process starts with identifying the customers to be billed that is a service in the management module. Afterwards, the accounting module retrieves the list of customers to be billed and depending on the subscription and payment method customers are being charged. After this process the accounting module updates the customer's subscription data on the management module.

3.3 Requirements for the Cloud Migration

There are three actors involved in the process of migrating an enterprise software to a cloud scale solution. In order to decrease complexity of requirement specification it is reasonable to separate the requirements depending on the stakeholder's point of view. These stakeholders are "user" that represents a tenant, "software provider" that aims to deliver a multi-tenant cloud application and "cloud services provider" that offers the cloud environment that application will be served. Each of these requirements are categorized as functional and non-functional requirements. Table 2 summarizes the requirements for migration to a cloud scale solution of the system.

User Requirements	
Functional	
UF1	API to access the tenant data programmatically
UF2	Tenant access control
UF3	Self-service Sign Up
Non - Functional	
UN2	Availability and reliability
Software Provider Requirements	
Functional	
SF1	Support multi-tenancy
SF2	Secure isolation of tenant data
SF3	Per tenant customization
SF5	User ID and authentication
SF6	Subscription and billing mechanisms
SF7	Tenant management and monitoring
Non - Functional	
SN1	Meet tenant service level agreements
SN2	Decrease hardware, deployment and maintenance costs
SN3	Scaling and managing the application
Cloud Provider Requirements	
Functional	
CF1	Compatibility with the legacy SQL systems
CF2	Providing a PaaS environment
Non - Functional	
CN1	Data Security and Privacy
CN2	Meet governmental regulations for cloud computing
CN3	Utilization based pricing
CN4	Storage Availability
CN5	Elastic Scalability

Table 2. Requirements for Cloud Migration

3.3.1. User Requirements

User requirements section defines the set of functional and non-functional requirements from the tenant's point of view. As the cloud migration process does not change the front end logic that the end user interacts there are only few important requirements.

3.3.1.1. Functional requirements

In the on premise software delivery method, the database systems are located psychically on the customer's premises and programmatic access is usually granted to the tenant's IT staff. Since the data architecture is single tenant, there is no need to worry about data isolation. But after the migration due to the shared database and table structure and the cloud structure itself, it is not possible to grant access to the tenant's data in a similar way. **UF1** defines an API where data isolation among tenants is handled in the API level and data access is granted for each particular tenant via a RESTful API. On the other hand if there are any customized application that is built on top of the on-premise data store, it must be transformed to consume this API.

In the on premise delivery method, software provider do not really have to address the access control mechanism to reach the application since data storage is on the customers side and usually integrated with the existing mechanisms such as Active Directory . But migration of the data back-end to a cloud environment will need a mechanism for tenants to control the authentication and access for their end users **(UF2)**. Thus an administrative interface for the tenant's admin staff should be provided.

Furthermore, since the application service model is going to be evolved in Software as a Service model, providing a self-service sign-up mechanism **(UF3)** is important to minimize the initial effort that must be spent by the software provider and for sake of application reachability to the end user.

3.3.1.2. Non-functional requirements

From the end user point of view the two non-functional requirements are the availability and the reliability of the system **(UN2)**. For availability the metric used to measure is the percentage of time that the system is capable of serving its intended function.

As stated in Microsoft service level agreements [53] SQL Database will maintain a monthly availability of 99.9% that also meets the service level agreements between software provider and customers.

Related with the high availability, reliability and as also a part of failover strategy, each SQL Database instance has three replicas residing on three different physical machines within a datacenter, one primary and 2 secondary replicas. All reads and writes go through the primary replica, and any changes are replicated to the secondary replicas asynchronously. In case of any failure in the main instance the application can be routed to one of the replicas.

3.3.2. Software Provider Requirements

In the cloud migration process of an enterprise system, the software architects and developers that belongs to the vendor, play the most important role indeed. While business people should take care of the concerns that target customers may have, the architects should carefully design the underlying infrastructure in order to have a solid system on the cloud. This section identifies the functional and non-functional requirements from the perspective of software provider.

3.3.2.1. Functional requirements

The most important aspect of migration of an enterprise system to a cloud platform is supporting true multi-tenancy **(SF1)**. The multi-tenancy models outlined in the architectural section describes the possible methods that can be implemented. Migrated system should ensure that a robust multi-tenancy model is implemented and fully supported with all aspects.

The multi-tenancy model on the cloud includes sharing the resources. If the chosen multi-tenancy architecture is a shared database or shared everything model it raises another issue to be addressed; the secure isolation of the tenant data **(SF2)**. None of the end users interaction should interfere with a different tenant's data than the tenant they belong to. Thus, instead of the security on the cloud, the application level data isolation must be carefully implemented.

Enterprise applications often require some level of customization for each individual tenant. Thus the architecture on the cloud platform where all the tenants share the same data model should be able to offer some extensibility **(SF3)**. The methods and walkthroughs should be in place in order to customize the application for any particular tenant.

User access control and authentication is another important requirement that must be in place **(SF4)**. From an end-user perspective cloud can mean reaching the application from anywhere, anytime. The system should provide a secure authentication mechanism to the end user. Since multi-tenancy requires the users to be identified to find out which tenant they belong, a mechanism to sustain user identification and authentication is important. If required, integration with identity providers should also be provided.

Changing the deployment model from on-premise delivery method to software as a service on the cloud means some important changes in the business model. Usually on premise software is sold with a lifetime license and customer owns the software mainly because of the deployment method. But hosting the application on the cloud also means that the application will be served as a service. Due to the changes in the software ecosystem, usage duration and number of users per tenant, there needs to be a certain kind of mechanism to keep a track and manage the application and produce billing information that is handy for the tenant administrators **(SF5)**.

In coordination with the billing mechanism there is also a necessity of some administrative interface to control and monitor the tenants for the software provider **(SF6)**. The interface should have functionalities such as limiting the tenant's access or updating the trial users to a professional license.

3.3.2.2. Non-Functional requirements

One of the most important motivations of the cloud migration is decreasing the costs that are related with the hardware and software maintenance **(SN2)**. Thus the migration process should include cost estimation after migration of the system. In some cases if the resources on the cloud environment are not efficiently consumed, it may result as the migrated system being more expensive. The model where each tenant has its own virtual machine is promotes as multi-tenancy but it does not exactly defines true multi tenancy. Creating a virtual machine and setting up the database backend individually will not be cost efficient and one of the main purposes of the cloud migration process may not be achieved.

The other important requirement of the cloud system is that the Service Level Agreements (SLA) between the software vendor and the customers **(SN1)**. In order to the meet the SLAs, firstly cloud service provider must be carefully chosen even though most of them guarantee %99.95 service level. Secondly the service level of the cloud platform do not guarantees the application level SLA. Thus the application itself should be implemented in a way that it will ensure the SLA's are met.

Shifting the application architecture from single-tenant to a multi-tenant architecture requires a very vital issue to be addressed beforehand; scalability **(SN3)**. Cloud platform is also meant for serving scalable applications but the system must be designed in a way that the data storage backend and the communication layers should be able to handle the requests from multiple tenants. Yet increasing number of tenant's should not affect the other tenant's performance.

3.3.3. Cloud Provider Requirements

Choosing the most efficient and appropriate cloud environment that the target application is going to be hosted needs identification of some set of functional and nonfunctional requirements. Usually the requirements in this section are software specific and mainly depend of the target applications character.

3.3.3.1. Functional requirements

Changes in the data back-end architecture for supporting the multi-tenancy require quite a lot work on the data storage mechanism. Thus the compatibility with the legacy SQL system is a very important issue in order to decrease the effort for the implementation process in the cloud migration **(CF1)**. The data storage environment should offer functionalities such as supporting application roles, logins, triggers and so on.

Furthermore in the scope of this work cloud is not the virtualization of the existing software delivery environment. The cloud service provider should provide a platform where some set of services such as identity management, access control mechanisms and database itself as a service that is called Platform as a Service **(CF2)**.

3.3.3.2. Non-Functional requirements

There are several important non-functional requirements for cloud computing that needs to be taken care of. The most important of them is the Data Security and Privacy **(CN1).** European Union has very strict regulations about data security such as geographical location of the data storage mechanisms and cloud services that are going to operate or restricting the transfer of the data outside the European Union (EU) that do not offer an adequate level of protection. So practically the cloud service provider must have data centers within EU and should offer operational services inside the EU domain. On the other hand there can be government specific regulations for cloud computing, that is should be taken care of **(CN2)**.

The other important aspect for cloud computing is the way how cloud service provider prices the services. Besides the initial installment and setting up costs, service usage should be priced in a pay as you go model **(CN3).** Some cloud service providers may charge even there is no actual usage or consuming of the services which will increase the costs. The pricing model must be utilization based.

Using cloud services for data storage will soon increase the amount of stored data for the application. The cloud service provider should offer extending the initial storage capacity **(CN4).** Furthermore the cloud platform must let the data storage system to be scalable in any time without any down time **(CN5)**. Scaling the data back-end should not affect the operation and the availability of the software.

3.4. Choosing a Multi-Tenancy Approach

The possible approaches for efficient multi-tenancy on the cloud platforms were mentioned in section 2.6 of the thesis.

Each of the three approaches described before, offers its own set of benefits and tradeoffs that make it an appropriate model to follow in some cases and not in others, as determined by a number of business and technical considerations. Some of these considerations are listed below.

3.4.1. Economic Considerations

Applications designed for a shared approach usually require a larger development effort than applications designed using an isolated approach. This results as higher initial costs. But the ongoing operational costs are going to be lower since they can serve more tenants and they can be handled together.

3.4.2. Security Considerations

Since the cloud application is going to store sensitive tenant data, customers are going to have high expectations about security which is also mentioned in the background chapter of this thesis. Most of the customers believe that only physical isolation can provide an appropriate level of security. Actually, data stored using a shared approach may also provide strong data safety. But it requires a sophisticated level of design.

3.4.3. Tenant Considerations

In order to choose a more isolated approach or a more shared approach there are 3 major parameters should be taken into account. One of these considerations is the number of prospective tenants that are going to be targeted. If there are more tenants a more shared approach will probably fit better.

The other consideration is about the amount of storage space that an average tenant data will occupy. If the tenants are going to store very large amounts of data, then probably a more isolated approach is the best. Last consideration is the number of concurrent end users that the average tenant has. The more the end users are, the more appropriate a more isolated approach.

3.4.4. Regulatory Considerations

The last of the considerations that should not be skipped is about the regulations that target customers are subject to. The regulatory environments that the target customers occupy in the markets should be investigated and decisions should be determined such as storing the data only in the European Union psychically.

4. Implementation

The implementation chapter of this thesis shows the deployment methods of the software before and after the cloud migration. The implementation steps about database redesign, changes in the business tier and data access, implementing the secure multi-tenancy and authentication mechanisms are clarified.

4.1 Current Deployment Method and Architecture

The software delivery method for this enterprise application is the traditional in house application deployment model. When a new license is purchased, the deployment process starts. At first step if the tenant do not have an existing hardware that can run Microsoft SQL server, a database server is purchased. Afterwards licenses are purchased for SQL Server and the operating system.

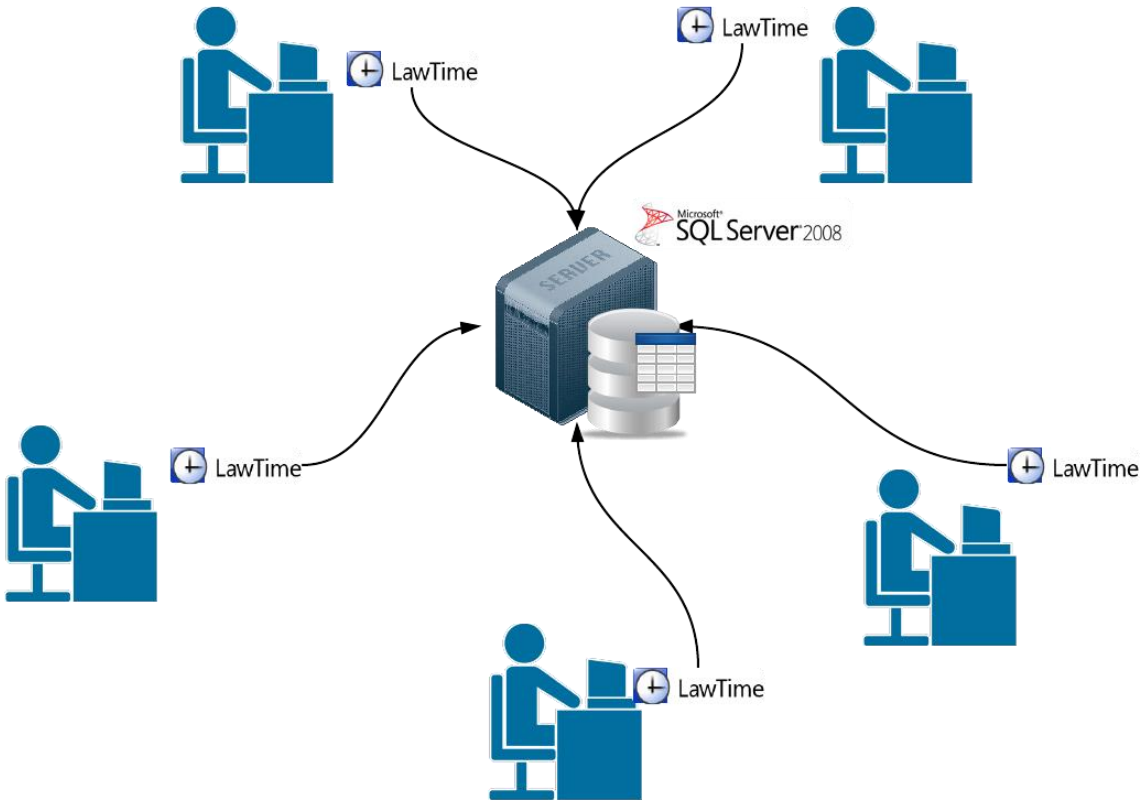


Figure 19. Current Software Deployment Model

The deployment process continues with installing the operating system and SQL server. Tests are made to see if the deployment environment is up and running. This process is followed by attaching the database instance that contains the tables, stored procedures and views. Afterwards the first initial user with admin role is created manually on the users table. The rest of the users are added either by the admin role user or the support person who installs the software. Figure 19 shows an overview that how the software is used inside the tenant's environment.

4.2 Deployment Method and Architecture on the Cloud

The migration of the on premise software architecture to the cloud scale solution removes all the required initial effort that is described above and the installment plus the licensing of the necessary software to run the product. After cloud migration any prospective customer may immediately start using the software without any help from the software provider. This process is fully automated and one of the main goals of the migration.

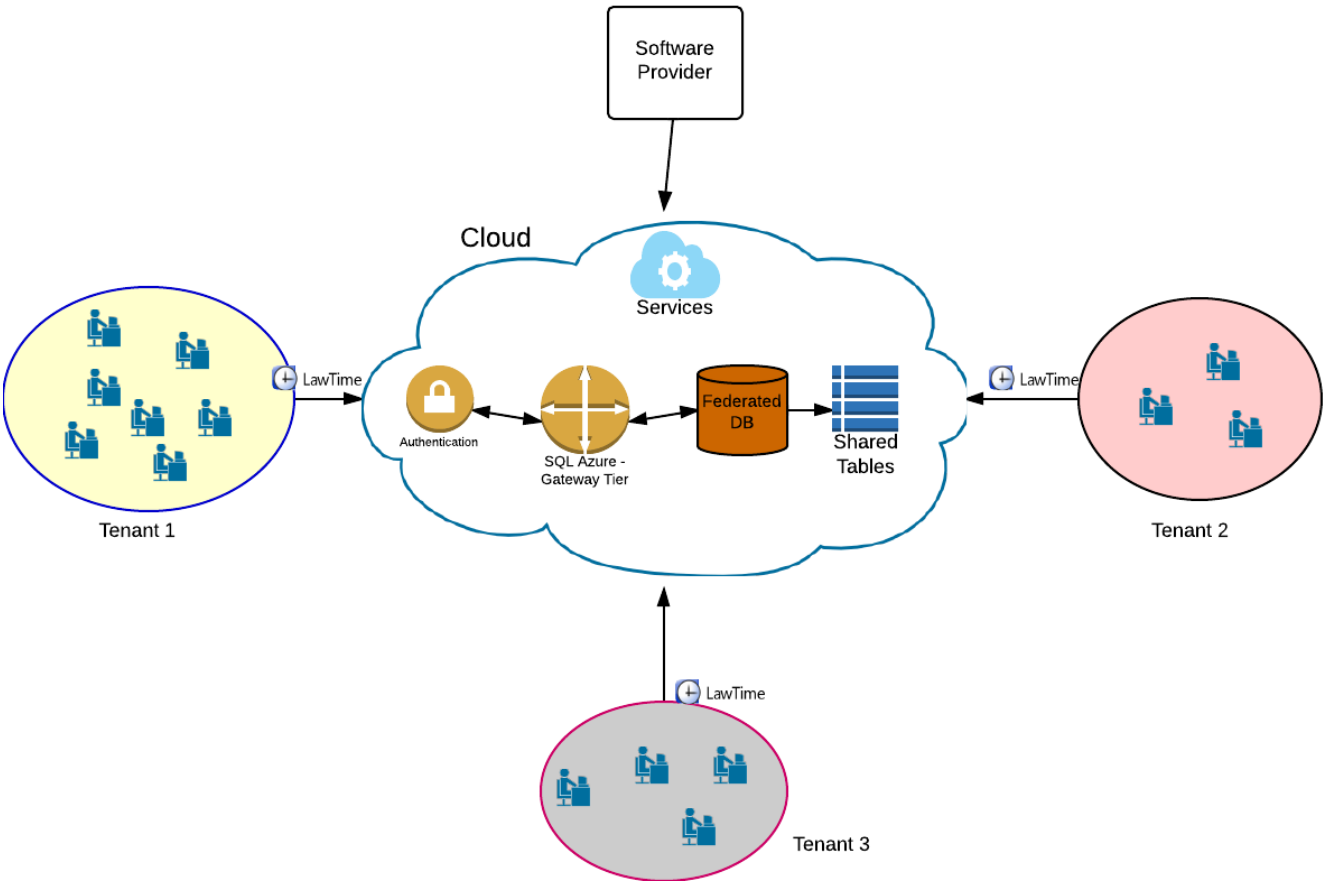


Figure 20. The Software Delivery Method after Cloud Migration

Figure 20 gives an overview how the software is delivered and used on the cloud. Comparing this architecture to the initial one, there aren't any per tenant server and deployment environment anymore. Instead there is a shared environment that is used by every tenant. On the other hand while authentication mechanisms ensure the secure access of the data, the federated databases ensures the secure isolation of the tenant data on the shared tables.

4.3 Implementation Steps

Implementation of the multi-tenant data architecture on the cloud is a rather new and complex challenge. Thus it requires a separation of modules that should be handled by a step by step approach. In this section the modules are separated into three main sections that are database redesign, security and authentication mechanisms.

4.3.1 Database Redesign and Data Access

Our proposed way of multi-tenant database structure on the cloud aims storing all of the tenant's data inside the same shared tables. In order to achieve this goal there is a method where SQL Federations is used as a data access and mapping tier. It also offers advantages in terms of data scalability.

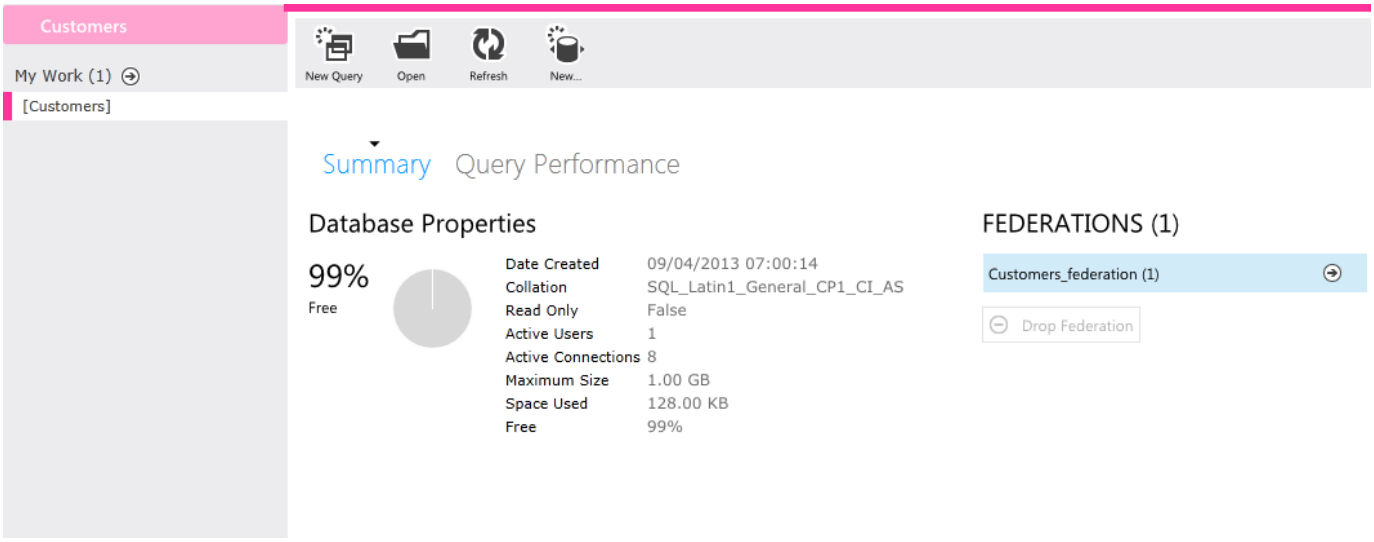


Figure 21. SQL Database Federations

First of all on SQL Database (formerly called SQL Azure) a new database must be created. On top of this new database a new federated database should be created as shown in the Figure 22. The name of the federation should be a unique name that is different from the original database, because we are actually creating another child database. The distribution name is the key value that is going to be matched with the tenant identifier (TenantID) while filtering the data. The data filtering for distinguishing tenants will be extended in the next steps.

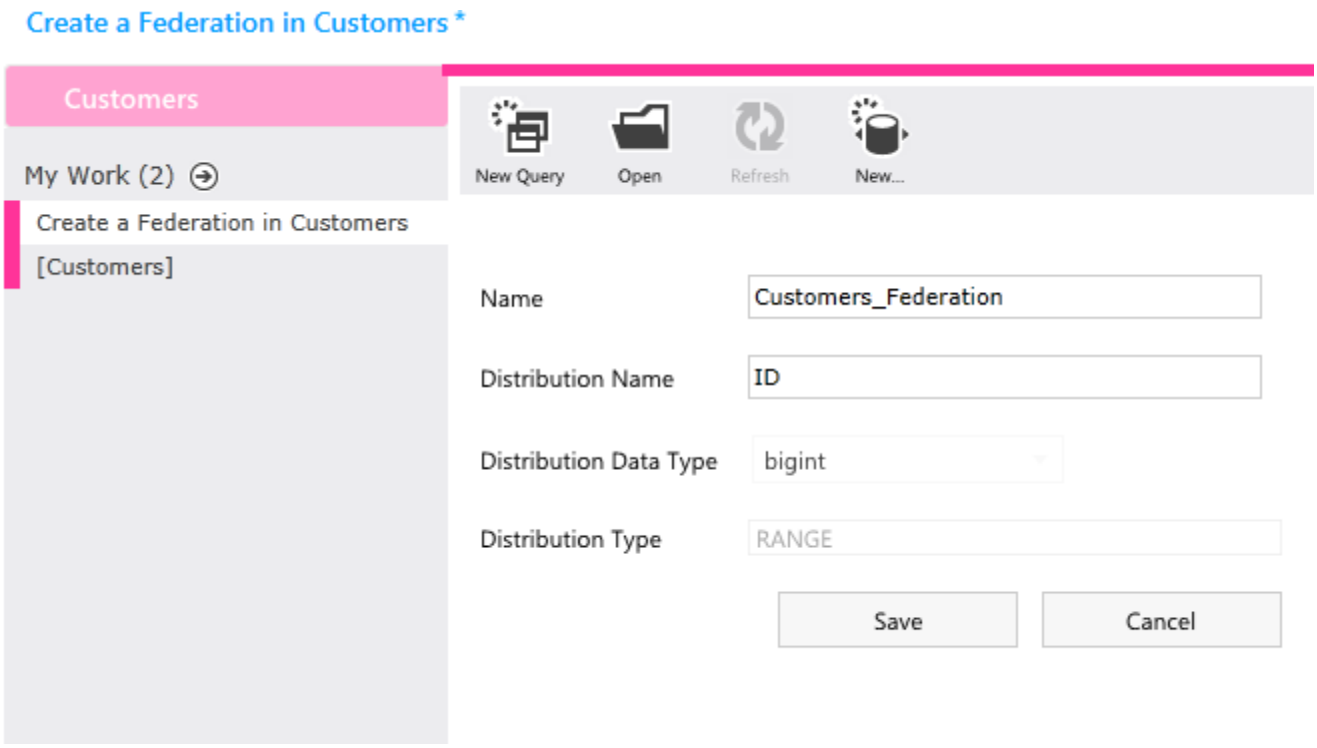


Figure 22. Creating a Federation in Database

As for the second step, in order to distinguish one tenant data from another within the tables, there is a necessity of an identifier in each table that is a key value to handle this process. This value is called 'TenantID'. All the database scripts must be updated to include TenantID.

```
USE FEDERATION [Customers_federation] ([ID] = -9223372036854775808) WITH FILTERING = OFF, RESET
GO
```

```
-- ================================================================================
-- Create a federated table in a federation member based on this example.
-- This script will run only in the context of a federation member.
-- The primary key column type needs to be of data type bigint for federation [Customers_federation].
-- ================================================================================

CREATE TABLE dbo.Invoices(
        TenantID bigint NOT NULL,
        Column1 nvarchar(50) NULL,
        Column2 nvarchar(15) NULL,
 CONSTRAINT [PK_Invoices] PRIMARY KEY CLUSTERED
 (
     TenantID ASC
 )
 ) FEDERATED ON ([ID] = TenantId)
 GO
```

Figure 23. Creating Federated Tables

Figure 23 shows how a federated table is created. 'Customers_federation' is the name of the federated database that is created earlier. With setting the filtering off, we are creating a common table for each of the tenants inside the federation. The 'FEDERATED ON' clause matches the distribution value with our custom created TenantID value.

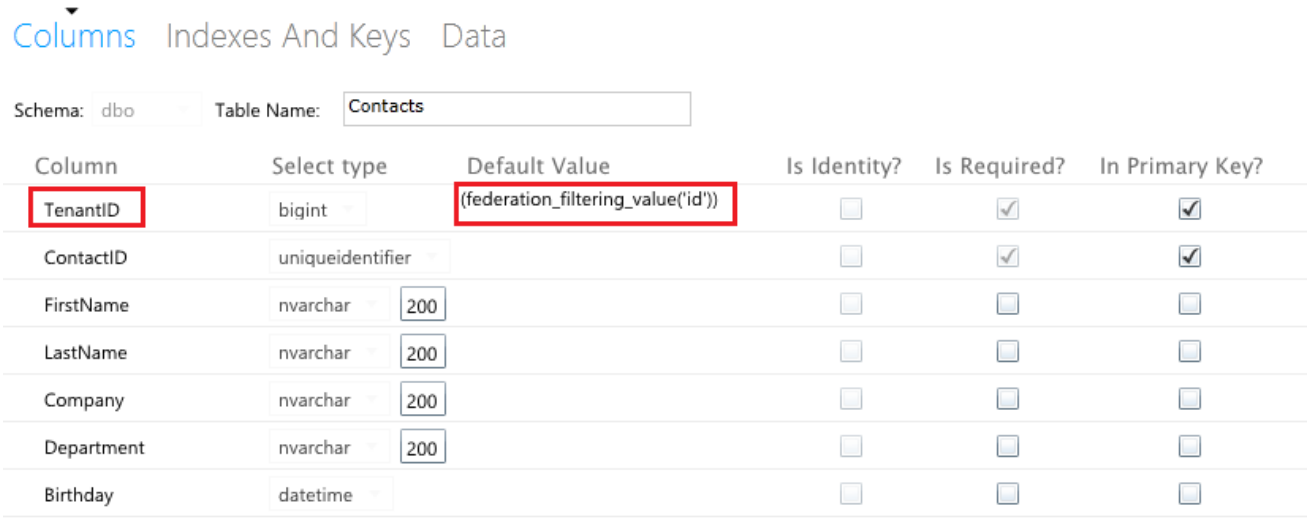


Figure 24. Federated Table Structure

After these two major steps and creating the federated tables it will be beneficial to update the indexes on all tables so that these take into account the TenantID.

The next important step is updating all database queries made at the business logic tier and include the tenant identifier. In this step even there isn't any business logic tier and all the queries are hard coded and spread around the whole project, it isn't going to be a hard task to update the queries since we are using the SQL federation's structure. Before executing any SQL command all the queries need an additional line of SQL query such as;

(USE FEDERATION Customers_federation(TenantID = 12) WITH RESET, FILTERING=ON)

After execution of the command above, the data associated with the tenant will be stored, updated or retrieved accordingly. We can simply continue to issue the "SELECT * FROM dbo.invoices" query on a filtered connection and the algebrizer in SQL Azure automatically injects "WHERE TenantID=12" into the query.

As the final step, all the stored procedures must be updated. They must all include an additional parameter similar to the one shown above to include the TenantID. The *(USE FEDERATION Customers_federation(TenantID = 12) WITH RESET, FILTERING=ON)* statement can either be executed in the business logic tier or inside the stored procedure. For example while executing a SQL INSERT INTO statement, the stored procedure will require a TenantID and it can be retrieved with a select statement from the system since we already executed necessary SQL statement to use federations and system now knows the TenantID. Otherwise the TenantID must be supplied as an additional parameter before each execution of the stored procedure.

To sum up SQL Federations address most of the issues. We have a Multi-tenant database model with federations that allow us to dynamically adjust the number of tenants per database at any moment in time. It is possible to repartition with a command like SPLIT. It allows us to perform these operations without any downtime.

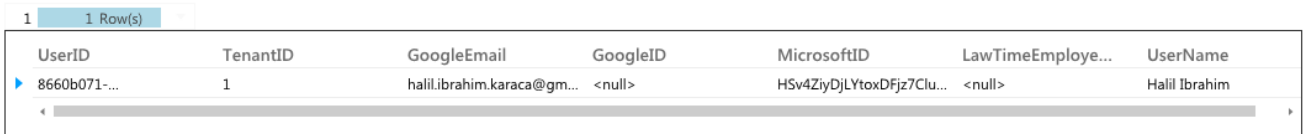
On the other hand Federation allows a cost efficient dynamic tenant model that can adjust to growing workloads in individual tenants or easily onboard new tenants.

Furthermore SQL Azure's gateway tier does the connection pooling automatically. The Connection pool efficiency is built in with Federations. It is possible to connect through a single endpoint and we don't have to worry about connection pool fragmentation for the applications regular workload.

4.3.2 Security

Microsoft Windows Azure provides a native way for handling the data isolation among tenants that we described above (Federations). This addresses the security concerns in the runtime between

tenants and we can be sure that none of the tenants interfere with another tenant's data. But there is more work to be done in the data access tier to create a fully secure environment and defensive programming approach is always safer.



1 Row(s)

UserID	TenantID	GoogleEmail	GoogleID	MicrosoftID	LawTimeEmploye...	UserName
8660b071-...	1	halil.ibrahim.karaca@gm...	<null>	HSv4ZiyDjLYtoxDFjz7Clu...	<null>	Halil Ibrahim

Figure 25. Tenant Database - User Table

In order to provide a fully secure data storage environment, we created 2 separate databases. First database is the **content database** (Customer_DB) where all the actual data content is stored and there is no direct access to this database from any end user. The structure of this database is described above. Second database is the **tenant database** (Tenant_DB) where end user authentication data (username and passwords, tenantID's, Google, Facebook and Microsoft Accounts) and encrypted connection strings are stored for the end users of tenants.

The connection string to the actual content database is never stored in the client side and it is only possible to access this database after the secure authentication to the system through the Tenant_DB. After secure authentication through the tenant_db, it is possible to retrieve encrypted connection string and create a connection to the content database.

4.3.3 Implementing the Authentication Mechanism

The idea of the authentication module for the cloud scale system and most of the main functionality of the authentication mechanism is covered in the Architecture Chapter - Logical View section of this master thesis.

The main authentication mechanism that is provided by Azure (Access Control Service) is a middle tier between the end user application and the database layer. The authentication module only interacts with the Tenant Database and not with the Content Database that is because of security reasons.

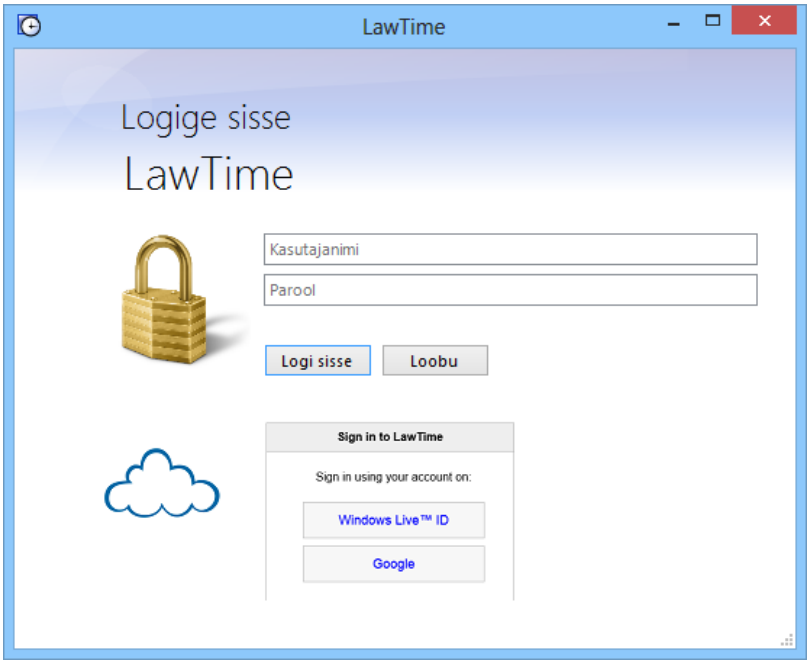


Figure 26. Login Screen

The Login mechanism that is provided by the Azure Access Control Service is embedded into the forms application login screen inside a web browser. In order to use the native Azure login mechanism, the requests should be made via https and thus a separate browser based application is needed to handle the tokens and transform the claims and map with the end user data on the database. The cloud login mechanism then interacts with the database on the cloud where the tenant data is stored. If the authentication is successful then end user is able to connect to the actual data content.

Furthermore this mechanism also provides a hybrid way for the end users. If any of the tenants is not willing to migrate their data to the cloud, then they can simply continue accessing their data on their own premises.

5. Validation

In the scope of this work we were able to create a multi-tenant efficient data storage back-end and underlying application architecture in Microsoft Windows Azure Cloud for our software product ‘LawTime’ that is an on-premise single-tenant enterprise application. However the modules and the necessity components to fully support cloud operability is not being completed yet. Thus, currently migration of a real customer is not possible. The functionalities and the modules such as subscription and billing mechanisms, access control for end users of tenants, and self-service sign-up should be in place. Furthermore an interface for tenant management and monitoring should be implemented. As it is possible that these implementations can introduce new performance penalties, it is not possible to formally evaluate the performance and correctness of the implemented functionalities.

Table 3 shows a list of requirements that were completed, not completed or partially completed. These results are with respect to the initial requirements as part of validation of product functionality. In functional user requirements section, the partially completed requirement UF2 - Tenant access control, needs an interface for the tenant’s administrative staff to control the end user access. The availability and the reliability of the system should be also observed and tested after migration of the real customers.

Using Microsoft Azure’s Federations, securely isolated, multi-tenant efficient and per tenant customizable application architecture on the cloud was created. The authentication mechanisms that supports identity providers such as Google and Microsoft Live ID is implemented and embedded to the system using Azure’s native product Access Control Service. The Windows Azure platform provides a native interface for monitoring the database processes. However a sophisticated monitoring interface per tenant should be implemented. Furthermore a tenant management interface should also be in place to control tenant activity.

The correctness of the implementation is verified manually for the functionalities listed in the validation table shown in the Table 3. But still there is a necessity for automated testing in this

context. Investigating the possibilities of defining a test methodology for multi-tenant software systems may be a future work.

User Requirements	
Functional	
UF1	API to access the tenant data programmatically
UF2	Tenant access control
UF3	Self-service Sign Up
Non - Functional	
UN2	Availability and reliability
Software Provider Requirements	
Functional	
SF1	Support multi-tenancy
SF2	Secure isolation of tenant data
SF3	Per tenant customization
SF5	User ID and authentication
SF6	Subscription and billing mechanisms
SF7	Tenant management and monitoring
Non - Functional	
SN1	Meet tenant service level agreements
SN2	Decrease hardware, deployment and maintenance costs
SN3	Scaling and managing the application
Cloud Provider Requirements	
Functional	
CF1	Compatibility with the legacy SQL systems
CF2	Providing a PaaS environment
Non - Functional	
CN1	Data Security and Privacy
CN2	Low Latency
CN3	Meet governmental regulations for cloud computing
CN4	Utilization based pricing
CN5	Storage Availability
CN6	Elastic Scalability

Completed	Partially Completed	Not Completed

Table 3. Validation

6. Conclusions and Future Work

Observing so many advantages and success of cloud computing, the term ‘migration to the cloud’ became a buzzword in the industry for most of the enterprises today. In this effort, we described a case study and lessons learned on the migration of an industrial single-tenant application from on-premise deployment backend to Azure Cloud. We start with introducing our motivation, scope and objectives and continue with defining the current trends for the cloud computing, approaches and design choices for multi-tenant data architecture on the cloud.
As a result we have developed a secure, multi-tenant efficient and scalable application architecture that also supports application extensibility and customization on the cloud.

Having a Microsoft based development environment, compatibility issues with the legacy software and native functionalities of the cloud platform lead us choosing Microsoft’s cloud platform, Windows Azure. In the development and implementation process, using some of the native functionalities of the cloud platform such as SQL Federations for data isolation among tenants and embedding Azure Access Control Service to our existing mechanism for handling authentication made our life easier in the design and implementation process of the secure, multi-tenant efficient and scalable application environment on the cloud. Furthermore we have seen that, if we have had a nicely layered application architecture and a data access layer, the transformation of single-tenant minded database queries to the multi-tenant one would have been way much easier and less time consuming.

Eliciting and analyzing the requirements took significant effort since migration of a single-tenant system to a multi-tenant one is a complex project. We took 4 + 1 architectural view model for describing our multi-tenant application architecture that made it easier to describe the big picture. Some of the required functionalities for migration to the cloud such as an API to access the tenant data programmatically, subscription and billing mechanisms , self-service sign up, tenant access control and tenant management mechanisms were not be able to implemented due to time limitations. But since these are independent modules they do not require all developers working on the project to be trained in multi-tenancy.

Although there are many benefits of multi-tenancy, still there are some set of challenges that must be considered before implementing a cloud scale solution. Since all the tenants use the same hardware, a problem that is caused by one tenant may affect all the others. In this point the cloud service provider should be chosen carefully where disaster recovery options are offered in such cases for business critical systems. Furthermore the enterprise application may require scalability with zero-downtime which may also be offered by the cloud service provider as Windows Azure offers as part of the functionality in SQL Federations.

Development of an automated test methodology, the open issues in the validation table such as the API for accessing the tenant data programmatically, sophisticated subscription and billing mechanisms and a real time monitoring mechanism might be seen as a part of the future work for this project.

Asutuse sisene ainu-tarbijaga ettevõtte tarkvara andmekihi migratsioon Azure pilve keskkonda: Mitme -tarbijaga andmekihi juhtumiuuring

Magistritöö (30 EAP)

Autor: Halil Ibrahim Karaca

Juhendajad: Dr. Peep Küngas, Indrek Karu

Resümee

Kokkuvõte

Pilvearvutuse edu muudab radikaalselt tavasid kuidas edaspidi infotehnoloogia teenuseid arendatakse, juurutatakse ja hallatakse. Sellest tulenevalt on sõnakõlks „pilve migratsioon“ vägagi aktuaalne paljudes ettevõtetes. Tänu sellele tehnoloogiale on paljud suured ja väikesed ettevõtted huvitatud enda tarkvara, andmebaasi süsteemide ja infrastruktuuri üleviimisest pilve keskkonda.

Olemasolevate süsteemide migreerimine pilve võib vähendada kulutusi, mis on seotud vajamineva riistvara, tarkvara paigaldamise ning litsentseerimisega ja samuti selle kõige haldamiseks vajaminevate inimeste palkamisega. Rakenduse ja selle andmete hoidmine pilves, mis teenindab mitmeid üürnike (ik. tenants) võib osutuda kalliks kui ei kasutada jagatud lähenemist üürnike vahel. Sellest tulenevalt on teadlikult disainitud rakenduse ning andme arhitektuur äärmiselt oluline organisatsioonile, mis kasutab mitme-üürniku (ik. multi-tenant) lähenemist.

Käesolevas magistritöös kirjeldatakse juhtumiuuringut (ik. case study) ning saadud kogemusi eraldiseiseva majasiseselt paigaldatava rakenduse migreerimisel Azure pilve keskkonda. Töö kirjeldab juristidele mõeldud tootlikkuse mõõtmise tarkvara andmekihi migreerimist Azure pilvekeskkonda. Majasisese ühe tarbijaga tarkvara andmekihi üleviimine efektiivsele mitme-üürniku andmekandja süsteemi pilve keskkonnas nõuab lisaks ka kõrgetasemelise autentimis-mehhanismi disainimist ning realiseerimist.

Töö põhirõhk on turvalise skaleeruva ning mitme-üürniku efektiivse andmekandja süsteemi arhitektuuri disainimine ning realiseerimine pilve-keskkonda. Projektis kasutatakse SQL

Database'i (endine SQL Azure) poolt pakutavat sisse ehitatud võimekust (SQL Federations) selleks, et tagada turvaline andmete eraldatus erinevate üürnike vahel ja andmebaasi skaleeruvus.

Tarkvara andmekihi migreerimine pilve keskkonda toob kaasa kulude vähenemis, mis on seotud tarkvara tarnimisega, paigaldamise ning haldamisega. Lisaks aitab see ettevõttel laieneda uutele turgudele, mis enne migreerimist oli takistatud kohapeal teostava tarkvara paigaldamisega. Tänu pilves olevale andmekihile nõuab uuele kliendile süsteemi paigaldamine väga väikest kulutust.

References

[1] Carlo Curino, Evan Jones, Yang Zhang, Eugene Wu, and Samuel Madden.
Relational Cloud: The Case for a Database Service. Conference on Innovative Database Research, 2011.

[2] Matthias Brantner, Daniela Florescu, David Graf, Donald Kossmann, Tim Kraska
Building a Database in the Cloud. SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, June 2008.

[3] Mei Hui, Dawei Jiang, Guoliang Li, Yuan Zhou, Supporting Database Applications As A Service. IEEE International Conference on Data Engineering, 2009.

[4] Stefan Aulbach, Torsten Grust, Dean Jacobs, Alfons Kemper, Jan Rittinger
Multi-Tenant Databases for Software as a Service: Schema-Mapping Techniques. SIGMOD '08 Proceedings of the 2008 ACM SIGMOD international conference on Management of data Pages 1195-1206, 2008.

[5] Carlo Curino, Evan P. C. Jones, Raluca Ada Popa, Nirmesh Malviya, Eugene Wu, Sam Madden, Hari Balakrishnan, Nickolai Zeldovich
Relational Cloud: A Database-as-a-Service for the Cloud. 5th Biennial Conference on Innovative Data Systems Research, January 2011.

[6] Farah Habib Chanchary, Samiul Islam.
Data Migration: Connecting Databases in the Cloud. Proceedings of the 1st International Conference on Computing and Information Technology (ICCIT), pp. 450 – 455, 12-14 March, Taibah University, Madinah, Saudi Arabia, 2012. Available at: http://www.taibahu.edu.sa/iccit/allICCITpapers/pdf/p450-chanchary.pdf

[7] Cloud Computing. Wikipedia.
Available at: http://en.wikipedia.org/wiki/Cloud_computing,
Seen: 2013-02-18, 2013.

[8] Merill in the cloud. Markus Klems homepage.
Available at: http://markusklems.wordpress.com/2008/07/05/merill-in-the-cloud/,
Seen: 2013-02-18, 2013.

[9] K. Rangan, A. Cooke, M. Dhruv, J. Post, N. Schindler, J. Fidacaro, W. Mohan, G. A. B. III, and J. Vleeschhouwer. The cloud wars: $100+ billion at stake. Technical report, Merrill Lynch, 2008.

[10] Amazon Web Services. Amazon web services homepage.
Available at: http://aws.amazon.com, Seen: 2013-02-18, 2013.

[11] AppNexus. Appnexus cloud.
Available at: http://www.appnexus.com, Seen: 2013-02-19, 2013.

[12] ENKI. Virtual private datacenters.
Available at: http://www.enkiconsulting.net/virtual-private-data-centers/, Seen: 2013-02-18, 2013.

[13] FlexiScale. Flexiscale cloud computing.
Available at: http://www.flexiscale.com, Seen: 2013-02-18, 2013.

[14] GoGrid. Gogrid cloud hosting.
Available at: http://www.gogrid.com, Seen: 2013-02-18, 2013.

[15] Joyent. Reasonably smart.
Available at: http://www.joyent.com, Seen: 2013-02-18, 2013.

[16] Rackspace. Rackspace managed hosting.
Available at: http://www.rackspace.com, Seen: 2013-02-18, 2013.

[17] Terremark. Infinistructure.
Available at: http://www.terremark.com/services/managed-hosting.aspx, Seen: 2013-02-18, 2013.

[18] Architecture strategies for catching the long tail.
Available at: http://msdn.microsoft.com/en-us/library/aa479069.aspx/

[19] Gartner Newsroom.
Available at: http://www.gartner.com/newsroom/id/1963815

[20] LawTime – Time and Billing Software for Legal Proffesionals
Available at: http://zoig.ee/lawtime/tutvustus.htm

[21] The NIST Definition of Cloud Computing. NIST.
Available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[22] Multi-tenant data architecture by Microsoft.
Available at: http://msdn.microsoft.com/en-us/library/aa479086.aspx#mlttntda_tvf

[23] Cloud Security Alliance.
Available at: https://cloudsecurityalliance.org/

[24] Architectural Blueprints - The "4+1" View Model of Software Architecture
Available at: http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

[25] Above the Clouds: A Berkeley View of Cloud Computing
Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz,
Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia.
EECS Department. University of California, Berkeley. Technical Report No. UCB/EECS-2009-
28 February 10, 2009

[26] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai
What's Inside the Cloud? An Architectural Map of the Cloud Landscape.
Available at: http://www.di.ufpe.br/~redis/intranet/bibliography/middleware/lenk-what-2009.pdf

[27] Ho-Fung Leung, Dickson K. W. Chiu, Patrick C. K. Hung
Service Intelligence and Service Science: Evolutionary Technologies and Challenges

[28]Sushil Bhardwaj, Leena Jain, Sandeep Jain
CLOUD COMPUTING: A STUDY OF INFRASTRUCTURE AS A SERVICE (IAAS)
Available at:
http://ijeit.org/index_files/vol2no1/CLOUD%20COMPUTING%20A%20STUDY%20OF.pdf

[29] Demystifying the cloud
Available at: http://www.saasblogs.com/saas/demystifying-the-cloud-where-do-saas-paas-and-other-acronyms-fit-in/

[30] SRINIVASAN SUNDARA RAJAN, Cloud Architecture Diagramming Standards
Available at: http://cloudcomputing.sys-con.com/node/1660757

[31] How to Authenticate Web Users with Windows Azure Active Directory Access Control
Available at: http://www.windowsazure.com/en-us/develop/net/how-to-guides/access-control/

[32] Windows Azure documentation
Available at: http://msdn.microsoft.com/en-us/library/windowsazure/dd163896.aspx

[33] Introduction to Windows Azure
Available at: http://www.windowsazure.com/en-us/develop/net/fundamentals/intro-to-windows-azure/

[34] Windows Azure vs Amazon Web Services
Available at: http://gigaom.com/2011/09/04/the-great-debate-windows-azure-vs-amazon-web-services/

[35] Leavitt, N., January 2009. Is cloud computing really ready for prime time?
Computer 42, pp. 1520.

[36] IT Cloud Services User Survey, pt.2: Top Benefits & Challenges. Retrieved April 2, 2013 from http://blogs.idc.com/ie/?p=210

[37] New IDC IT Cloud Services Survey: Top Benefits and Challenges. Retrieved April 2, 2013 from http://blogs.idc.com/ie/?p=730

[38] Takabi, H., Joshi, J. B. D., Ahn, G.-J., December 2010. Security and privacy challenges in cloud computing environments. IEEE Security and Privacy 8, pp. 2431.

[39] Catteddu, D. and Hogben, G. Cloud Computing: benefits, risks and recommendations for information security. Technical Report. European Network and Information Security Agency, 2009.

[40] Ristenpart, T., Tromer,,, E., Shacham, H., Savage, S., November 2009.Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09), pp. 199212.

[41] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina. Controlling data in the cloud: Outsourcing computation without outsourcing control. In ACM Workshop on Cloud Computing Security, 2009

[42] J. Brodkin. Gartner: Seven cloud-computing security risks.
Available at: http://www.infoworld.com/d/security-central/gartnerseven-cloud-computing-security-risks-853

[43] Cloud Computing: Business Benefits With Security,
Governance and Assurance Perspectives. White Paper. Information Systems Audit and Control Association, 2009.

[44] Brunette, G. and Mogull, R. Security Guidance for Critical Areas of Focus in Cloud Computing V2.1. Technical Report. Cloud Security Alliance, 2009.

[45] Aderemi A Atayero and Oluwaseyi Feyisetan, \Security issues in cloud computing: The potentials of homomorphic encryption," Journal of Emerging Trends in Computing and Information Sciences, vol. 2, no. 10, pp. 546{552, October 2011, CSI Journal

[46] M. Ko, G.-J. Ahn, and M. Shehab "Privacy-Enhanced User-Centric Identity Management," Proc. IEEE Int'l Conf. Communications, IEEE Press, 2009, pp. 998–1002.

[47] J. Joshi et al., "Access Control Language for Multidomain Environments," IEEE Internet Computing, vol. 8, no. 6, 2004, pp. 40–50.

[48] E. Bertino, F. Paci, and R. Ferrini, "Privacy-Preserving Digital Identity Management for Cloud Computing," IEEE Computer Society Data Engineering Bulletin, Mar. 2009, pp. 1–4

[49] M. Blaze et al., "Dynamic Trust Management," Computer, vol. 42, no. 2, 2009, pp. 44–52

[50] Y. Zhang and J. Joshi, "Access Control and Trust Management for Emerging Multidomain Environments," Annals of Emerging Research in Information Assurance, Security and Privacy Services, S. Upadhyaya and R.O. Rao, eds., Emerald Group Publishing, 2009, pp. 421–452.

[51] Caching in Windows Azure.
Available at: http://msdn.microsoft.com/en-us/library/windowsazure/gg278356.aspx

[52] Microsoft - MSDN , Multi-Tenant Data Architecture.
Available at: http://msdn.microsoft.com/en-us/library/aa479086.aspx#mlttntda_tvf

[53] Microsoft Windows Azure - Service Level Agreements.
Available at: http://www.windowsazure.com/en-us/support/legal/sla/

Non-exclusive licence to reproduce thesis and make thesis public

I, Halil Ibrahim KARACA (date of birth: 22.07.1986),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Migration of an On-Premise Single-Tenant Enterprise Application to the Azure Cloud: The Multi-Tenancy Case Study, supervised by Peep Küngas,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **20.05.2013**