Jurnal Pendidikan dan Teknologi Indonesia (JPTI)
Vol. 3, No. 6, Juni 2023, Hal. 233-240
DOI: https://doi.org/10.52436/1.jpti.293
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Peningkatan Keamanan Database Pada Layanan Azure Melalui Metode Multi-Tenant Dengan Pendekatan Separate Database

Samuel Nugraha[*1], Dian W. Chandra[2]

[1,2]Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Indonesia
Email: [1]samuelnugraha154@gmail.com, [2]dian.chandra@uksw.edu

Abstrak

Untuk semakin menghemat biaya dalam penggunaan layanan *cloud* di Azure, maka dapat menggunakan arsitektur *multi-tenant*. Arsitektur *multi-tenant* memungkinkan pengguna untuk menggunakan layanan *cloud* secara bersama-sama. Namun ada hal yang perlu diperhatikan saat menggunakan layanan *multi-tenant*, salah satunya adalah keamanan database antar pengguna. Salah satu solusi untuk mengatasi hal tersebut adalah dengan melakukan pendekatan *separate database*. Penelitian ini bertujuan untuk meningkatkan keamanan data pada metode *multi tenant* dengan pendekatan *separate database* pada layanan Microsoft Azure. Metode yang dilakukan pada penelitian ini adalah metode *multi tenant* dengan pendekatan arsitektur database yang digunakan adalah *separate database*, dan jenis layanan *cloud* yang digunakan adalah Paas (*Platform as a Service*) yang mana menggunakan Azure App Service dan Azure SQL Database dalam melakukan proses migrasi. Hasil dari penelitian ini adalah melakukan migrasi database dengan pendekatan *separate database* dari masing-masing pengguna layanan ke dalam layanan Azure. Hal ini dapat meningkatkan keamanan data antar pengguna sehingga tidak terjadi kebocoran data karena masing-masing pengguna memiliki databasenya sendiri.Desain arsitektur *multi tenant separated database* memiliki manfaat yaitu keamanan data yang lebih baik, karena tiap *tenant* memiliki database tersendiri sehingga data dari tiap *tenant* tidak bercampur menjadi satu. Sementara dari segi biaya layanan Azure App lebih hemat dibandingkan dengan layanan Azure Virtual Machine.

Kata kunci: *arsitektur multi-tenant, keamanan cloud, keamanan data, layanan Azure, separate database*

Improved Database Security on Azure Services Through Multi-tenant Methods With a Separate Database Approach

Abstract

You can use a multi-tenant architecture to save costs in using Azure cloud services. The multi-tenant architecture allows users to share cloud services. However, some things need to be considered when using multi-tenant services, one of which is database security between users. One solution to overcome this is to do a separate database approach. This study aims to improve data security in the multi-tenant method with a separate database approach on Microsoft Azure services. The method used in this research is a multi-tenant method with a database architecture approach used in a separate database, and the type of cloud service used is Paas (Platform as a Service) which uses Azure App Service and Azure SQL Database in carrying out the migration process. The result of this study is to migrate databases using a separate database approach from each service user to Azure services. This can improve data security between users so that data leakage does not occur because each user has its database. The multi-tenant separate database architectural design has the benefit of better data security because each tenant has its database so that data from each tenant is not mixed. become one. Meanwhile, in terms of cost, the Azure App service is more economical than the Azure Virtual Machine service.

Keywords: *Azure services, cloud security, data security, multi-tenant architecture, separate database*

1. PENDAHULUAN

Kata *cloud computing* dikenal juga dengan sebutan komputasi awan merupakan sebuah teknologi yang berkembang yang secara konsisten menghasilkan dampak pada dunia industri teknologi informasi serta akademisi, yang mana dalam melakukan seluruh tugas komputasi dilakukan melalui internet dengan menggunakan teknik *virtualisasi* dan tetap terisolasi dari infrastruktur perangkat keras dan perangkat lunak yang luas dan rumit[1]. *Cloud computing* memiliki layanan yang dapat mendukung pengembangan sistem tanpa harus

p-ISSN: 2775-4227
e-ISSN: 2775-4219

membeli server secara fisik, layanan ini disebut dengan istilah *Infrastructure as a Service* (IaaS). IaaS merupakan layanan *on-demand* dari komputasi abstrak infrastruktur atau platform virtualisasi yang menyediakan server, sistem operasi, penyimpanan, dan jaringan untuk mengembangkan, menghosting, dan menjalankan aplikasi.[1]

Cloud computing adalah sebuah model yang memungkinkan akses jaringan dimana-mana, nyaman, sesuai permintaan ke jaringan, bersama kumpulan sumber daya komputasi yang dapat dikonfigurasi (seperti jaringan, server, penyimpanan, aplikasi dan layanan), yang dapat dengan cepat disediakan dan dirilis dengan upaya manajemen minimal atau interaksi penyedia layanan.[2] *Cloud computing* memiliki karakteristik utama yang membuat layanan *cloud* lebih unggul jika dibandingkan dengan *on-premise* atau infrastruktur yang dibangun sendiri, karakteristik yang dimiliki oleh layanan *cloud computing* diantaranya yaitu layanan mandiri berdasarkan permintaan, akses jaringan yang luas, pengumpulan atau penyediaan sumber daya, elastisitas dan skalabilitas yang cepat, layanan terukur atau penetapan harga berbasis utilitas, kemandirian lokasi, efektivitas biaya, dan multi penyewa atau disebut juga dengan istilah *multi-tenant*. [1] Berdasarkan kelebihan yang dimiliki oleh *cloud computing*, maka banyak proses bisnis yang bermigrasi dari server *on-premise* mereka ke layanan *cloud computing* dengan tujuan untuk menghemat biaya.

Untuk semakin menghemat biaya dalam menggunakan layanan pada *cloud computing*, maka terbentuklah arsitektur *multi tenant* yang mana dapat memungkinkan aplikasi tunggal untuk bekerja lebih dari satu instan aplikasi telah direkomendasikan sebagai solusi untuk masalah tersebut.[3]

Ishrat Ahmad dalam penelitian *"Cloud Computing - A Comprehensive Definition"* menjelaskan bahwa layanan *cloud* memiliki beberapa fitur yang mana penyedia layanan *cloud* dapat menyediakan satu infrastruktur sebagai layanan untuk banyak pelanggan , fitur ini disebut sebagai *multi tenant*.[1] *Multi tenant* merupakan salah satu karakteristik dari layanan *cloud.*

Multi tenant adalah desain arsitektur yang memungkinkan pembagian infrastruktur untuk digunakan secara bersama oleh banyak vendor/*tenant*, sehingga efektif dan efisien dari segi biaya, perawatan, dan skalabilitasnya.[3]

Aplikasi *multi tenant* berbasis *cloud computing* memiliki banyak keuntungan karena menyediakan elastisitas dan skalabilitas sumber daya komputasi dan membebaskan pengguna dari banyak pekerjaan seperti konfigurasi, pengaturan, dan perawatan sumber daya teknologi informasi.[4]

Beberapa penelitian terdahulu yang berkaitan serta membahas mengenai metode *multi-tenant* pada layanan *cloud computing* diantaranya yang dilakukan oleh Erick dan Restyandito dalam penelitian *"Migrasi Aplikasi Multi Tenancy pada Layanan Komputasi Awan"* menjelaskan bagaimana metode *multi tenat* dapat digunakan untuk membangun sebuah layanan web berbasis ASP.NET dengan menggunakan perancangan sistem basis data terpisah (*separate database*) yang mana telah berhasil memigrasikan layanan web tanpa memperhatikan keamanan pada layanan tersebut.[5]

Hussain Aljahdali dan kawan-kawan dalam penelitian *"Multi-Tenancy in Cloud Computing"* menjelaskan bahwa *multi-tenant* merupakan hasil alami dari percobaan yang memanfaatkan keuntungan finansial dari komputasi awan dengan cara virtualisasi dan berbagi sumber daya. *Multi-tenant* mengacu pada berbagi sumber daya di layanan komputasi awan.[6]

Joan dan kawan-kawan dalam penelitian "*Layanan Infrastruktur Komputasi Multi Tenat dengan OpenStack di Lingkungan MaaS"* menyebutkan bahwa dalam lingkungan *Metal as a Service* (MaaS) untuk menyediakan layanan infrastruktur *multi tenant* berupa mesin-mesin virtual sebagai *Infrastructure as a Service* (IaaS). Dimana kapasitas sistem IaaS ini dapat dikonfigurasi berdasarkan kebutuhan sumber daya komputasi.[7]

Penelitian yang dilakukan oleh Mella Marliana dengan judul "*Keamanan dan Pencegahan Database Cloud Computing Untuk Pengguna Layanan"* menjelaskan terkait dengan *database* yang dipakai di komputasi awan yaitu *cloud database* yang mana memperhatikan terkait dengan ancaman dan pencegahan pada layanan *database* tersebut dengan mengelompokan menjadi *confidentiality, integrity* dan *availability*.[8]

Dalam menggunakan konsep *multi tenant*, terdapat dua arsitektur manajemen data yang dibedakan berdasarkan rentang, yaitu *shared data* (data berbagi) dan *isolated data*(data tertutup). Pada arsitektur manajemen *isolated data* terdapat pendekatan yang berguna untuk menjadi pertimbangan pada saat melakukan penyimpanan data, pendekatan tersebut adalah *separated database*. Sementara pada arsitektur manajemen *shared data*, terdapat dua pendekatan yang berfungsi untuk melakukan penyimpanan data yaitu *shared databases, separate schemas* dan *shared database, shared schema*.[9]

Pada *separated database* perangkat lunak yang dibangun akan digunakan secara bersama-sama, tetapi tiap *tenant* memiliki tempat penyimpanan data yang terisolasi satu sama lain dengan tujuan untuk melindungi privasi data antar *tenant*. Sementara untuk *shared database, separated schema*, semua *tenant* berada dalam satu lingkup basis data yang sama, dan untuk pemisah data dari tiap pengguna dilakukan dengan cara pengelompokan tabel-tabel dalam basis data pengguna ke dalam suatu kumpulan yang disebut *schema*.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

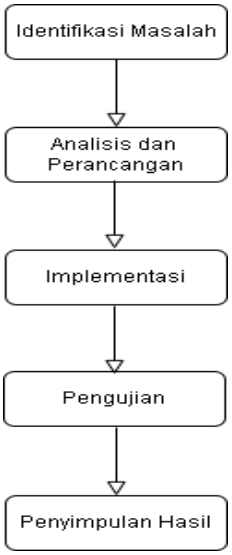
Untuk *shared database, shared schema* pendekatan yang dilakukan adalah menggunakan database yang sama dan tabel yang sama untuk menampung beberapa data dari pengguna, sehingga data-data tiap pengguna akan disimpan dalam satu basis data dan satu *schema*. Untuk membedakan data antar pengguna pada pendekatan ini , maka dilakukan penambahan kolom pada setiap tabel di basis data untuk menyimpan identitas setiap pengguna.

Berdasarkan uraian di atas, maka sistem keamanan data pada arsitektur layanan *multi tenant* perlu diperhatikan terkhusus dalam pendekatan metode *separate database*. Penelitian ini bertujuan untuk peningkatan keamanan data pada metode *multi tenant* dengan pendekatan *separate database* pada layanan Microsoft Azure dengan implementasi database yang digunakan adalah SQL Server.

2. METODE PENELITIAN

Penelitian dilakukan dengan tahapan seperti pada gambar 1, berikut ini merupakan penjelasan pada gambar 1. Proses diawali dengan melakukan identifikasi masalah, yaitu bagaimana cara mengimplementasikan metode *multi tenant* pada layanan database di Microsoft Azure dengan pendekatan *separate database*. Kedua adalah merencanakan dan menganalisa mengenai kebutuhan sistem dan proses simulasi dimana menganalisa kebutuhan seperti memilih jenis layanan *cloud* yang akan dipakai, memilih jenis dan tipe database yang akan digunakan di Azure, menganalisa terkait dengan biaya pada layanan-layanan di Azure, seperti biaya menggunakan Azure App, Azure SQL Database, serta menganalisa mengenai kebutuhan untuk membangun aplikasi menggunakan Visual Studio Community 2019. Ketiga implementasi, dilakukan dengan menerapkan metode *multi tenant* pada layanan *cloud* milik Microsoft Azure, membangun aplikasi yang akan di migrasi ke layanan Azure, serta membangun database untuk masing-masing *tenant*. Keempat adalah pengujian dari hasil implementasi yang telah dilakukan. Tahap terakhir adalah penarikan kesimpulan dari serangkaian implementasi dan pengujian yang telah dilakukan. Adapun alasan peneliti memilih layanan komputasi awan Microsoft Azure dikarenakan aplikasi basis data yang digunakan adalah basis data SQL Server, dimana selain dari vendor yang sama yaitu Microsoft, terdapat juga beberapa kemudahan untuk bekerja dengan menggunakan platform SQL Server dan layanan Microsoft Azure, selain itu alasan peneliti juga menggunakan Visual Studio Community 2019 untuk membangun aplikasi dikarenakan pada Visual Studio Community 2019 memiliki kemudahan saat melakukan migrasi ke layanan Azure.

Selain itu dilakukan juga studi kepustakaan yang mana dengan mereview beberapa jurnal penelitian terdahulu yang berkaitan dengan penelitian yang dilakukan, sehingga menghasilkan suatu referensi yang bermanfaat.



Gambar 1. Alur dalam metode penelitian

3. HASIL DAN PEMBAHASAN

Pada bagian ini dapat diuraikan mengenai hasil dari penelitian beserta pengujian yang telah dilakukan. Selain itu, disampaikan juga mengenai pembahasan dari penelitian maupun pengujian yang telah dilakukan

Ada beberapa cara yang dapat digunakan untuk melakukan proses migrasi dari server *on-premise* ke layanan *cloud* pada Microsoft Azure, diantaranya adalah dengan menggunakan layanan Azure VM (*Virtual*

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Machine), tipe layanan ini termasuk kedalam tipe IaaS (*Infrastructure as a Service*), selain menggunakan Azure VM ada juga layanan Azure App, tipe layanan ini termasuk kedalam tipe PaaS (*Platform as a Service)*.[10]

Pada penelitian ini, tipe migrasi yang digunakan adalah tipe PaaS dengan memanfaatkan layanan Azure App untuk memasang aplikasi web server. Adapun aplikasi web yang dibangun menggunakan ASP.NET core 5, sementara basis data yang peneliti gunakan pada server *on-premise* adalah SQL Server 2019, serta menggunakan arsitektur *separate database*.

Langkah pertama yang harus dilakukan adalah dengan membuat aplikasi web, pada penelitian ini aplikasi web yang dibangun menggunakan ASP.NET Core 5 Web app MVC(Model-View-Controller) dengan menggunakan Visual Studio Community 2019, kemudian lakukan perubahan pada tampilan View untuk bagian index dan layout, dimana tampilan ini akan menjadi tampilan awal yang akan ditampilkan pada saat melakukan migrasi nanti. Adapun pada tampilan ini peneliti membuat tampilan untuk data mahasiswa.

Langkah selanjutnya adalah membuat model data yang akan digunakan untuk menyimpan data di database nantinya, tetapi sebelum itu kita perlu menambahkan paket NuGet Microsoft Diagnostic Entity Framework Core dan Entity Framework Sql Server di Visual Studio, yang mana paket ini membantu mendeteksi dan mendiagnosa kesalahan dengan EF Core pada saat migrasi.

Adapun untuk contoh tabel basis data yang digunakan adalah tabel course, enrollment dan student, dimana setiap tabel memiliki entitasnya masing-masing. Tabel 1,2 dan 3 akan menjelaskan atribut-atributnya.

Tabel 1. Attribut pada entitas course

attribute	content	type
CourseID	ID Course	int
Title	Judul course	varchar
Credits	Beban SKS	int

Tabel 2. Attribut pada entitas enrollment

attribute	content	type
EnrollmentID	ID Enrollment	int
CourseID	ID Course	int
StudentID	ID Student	int
Grade	Nilai student	int

Tabel 3. Attribut pada entitas student

attribute	content	type
ID	ID Student	int
LastName	Nama akhir/ keluarga student	varchar
FirstMidName	Nama awal dan tengah student	varchar
EnrollmentDate	Waktu pada saat student daftar	DateTime

Selanjutnya kita membuat kelas pada folder Model di Visual Studio, jumlah kelas disesuaikan dengan jumlah entitas pada model data yang ingin dibuat.

Jika kelas sudah dibuat pada folder Model, selanjutnya adalah membuat folder baru yang berisi Data, dimana pada folder ini akan digunakan sebuah kelas yang digunakan untuk membuat database dengan pernyataan DbSet. Kelas pada folder data ini selanjutnya bisa dikonfigurasikan untuk *connection string* pada file app setting.json di Visual Studio.

Selanjutnya melakukan pengujian pada database yang sudah dibuat dengan membuat kelas baru pada folder Data yang berisi *initialize*, serta memasukkan data-data pada kelas tersebut. Data-data yang dimasukkan akan ditampilkan pada database saat migrasi nanti.

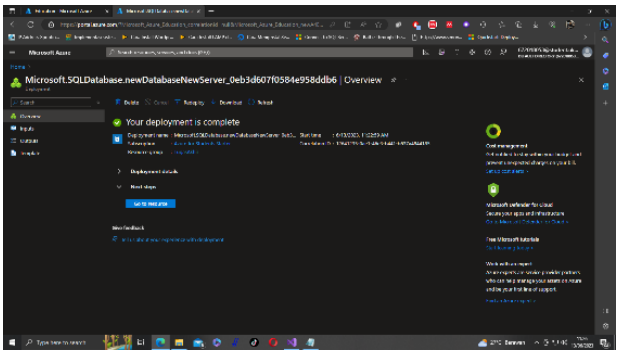
Selanjutnya adalah membuat controller, pada ASP.NET MVC kita dapat menambahkan controller dengan mudah, yaitu dengan menambahkan *MVC Controller with views, using Entity Framework* di aplikasi yang dibuat. Yang harus diperhatikan pada saat membuat controller ini adalah pengambilan kelas-kelasnya, dimana kita perlu menyesuaikan untuk kelas pada folder Model dan Data. Pada ASP.NET Core 5 ini kita dapat melakukan injeksi yang dapat mengurus serta meneruskan *instants* dari kelas yang dibuat ke *controller* secara otomatis, serta dapat menampilkan data yang sudah kita buat pada folder Data ke kelas Index.

Untuk menguji apakah database yang sudah dibuat sudah berhasil atau tidak dapat menggunakan menu *SQL Server Object Explorer* (SSOX) di Visual Studio. Jumlah aplikasi serta *instant* database dapat disesuaikan dengan jumlah *tenant* yang akan melakukan migrasi ke Azure. Jika jumlah *tenant* ada 3, maka buat 3 instant secara terpisah. Hal ini sesuai dengan arsitektur *separate database* pada metode *multi tenant.*

p-ISSN: 2775-4227
e-ISSN: 2775-4219

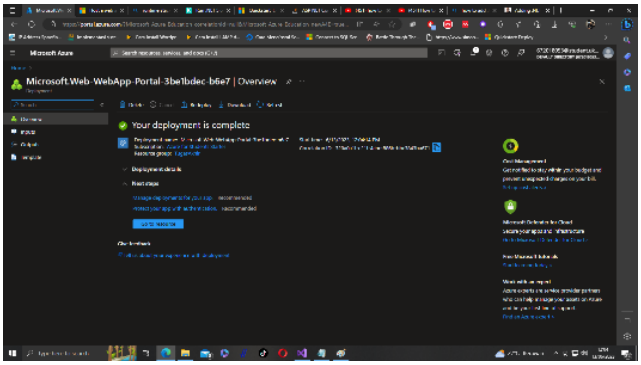
Selanjutnya adalah melakukan proses migrasi data dan aplikasi dari server lokal ke layanan Azure. Adapun untuk langkah-langkahnya adalah sebagai berikut :

Pertama kita membuat layanan Azure SQL Database pada portal Azure, untuk jumlah instant dapat disesuaikan dengan jumlah *tenant* yang akan melakukan migrasi, jadi masing-masing tenant memiliki SQL database tersendiri, hal ini bertujuan agar data tiap tenant tetap aman dan terhindar dari kebocoran data. Tambahkan username dan password untuk akses ke Azure SQL Database tersebut. Untuk hasil dari pembuatan layanan Azure SQL Database dapat dilihat pada gambar 2 berikut.



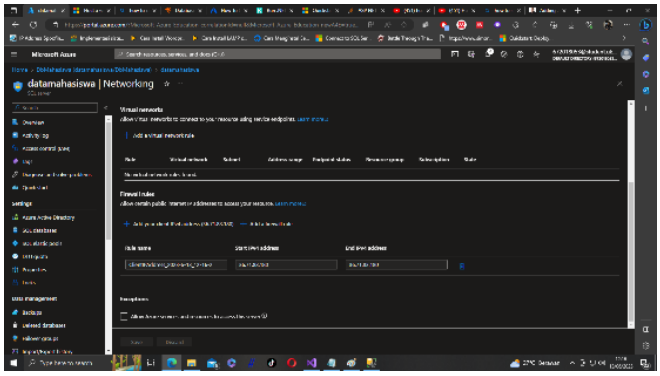
Gambar 2. Membuat layanan Azure SQL Database

Kemudian buat layanan Azure App Service pada Azure portal, yaitu dengan membuat Azure web app, serta tambahkan extension ASP.NET 5 di App Service yang sudah dibuat, dengan menggunakan layanan Azure App Service dapat menghemat biaya di Azure, karena tarif yang digunakan lebih hemat jika dibandingkan dengan membuat layanan Azure Virtual Machine, serta lebih mudah digunakan karena tidak perlu melakukan instalasi seperti pada Azure Virtual Machine. Untuk hasil dari pembuatan layanan Azure App Service terlihat seperti pada gambar 3 berikut.



Gambar 3. Membuat layanan Azure App Service

Kemudian buat IP Firewall dari SQL Database Azure yang sudah dibuat agar bisa diakses dari server lokal. Untuk hasil pembuatan IP Firewall untuk *client* pada Azure portal terlihat seperti gambar 4 berikut.



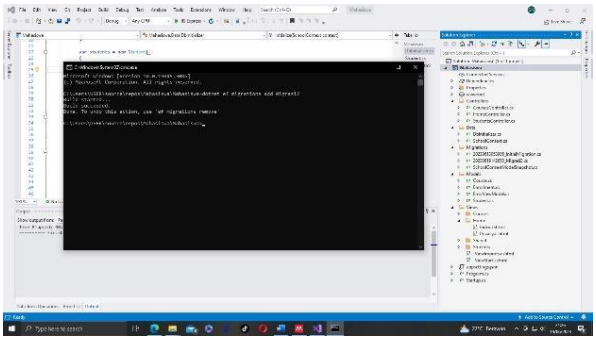
Gambar 4. Membuat IP Firewall pada Azure Portal

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Tahap selanjutnya adalah dengan mengganti *connection string* yang ada pada server local menjadi *connection* string yang ada pada Azure, serta mengganti username dan passwordnya sesuai dengan masing-masing SQL Database. Untuk hasil *connection string* dari Azure terlihat sebagai berikut : *Server=tcp:datamahasiswa.database.windows.net,1433;Initial Catalog=DbMahasiswa;Persist Security Info=False;User ID=samuel;Password={your_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate =False;Connection Timeout=30;*

Dengan adanya *connection string* yang diperoleh masing-masing *tenant*, maka tidak akan terjadi kesalahan dalam pengambilan data di dalam database, karena setiap *tenant* memiliki url yang berbeda-beda.

Kemudian lakukan proses migrasi untuk masing-masing *tenat* dari server lokal ke layanan Azure, adapun table perintah kode program yang digunakan untuk proses migrasi dari server lokal ke Azure terlihat pada table 4. Dimana proses migrasi diawali dengan klik kanan pada program yang sudah dibuat di Visual Studio, kemudian pilih menu *Open Folder in File Explorer*, maka nanti akan muncul jendela dari program tersebut, kemudian masukkan perintah *cmd* untuk membuka tampilan terminal, yang mana proses migrasi akan dilakukan melalui tampilan tersebut. Untuk lebih jelasnya dapat diperhatikan pada gambar berikut :

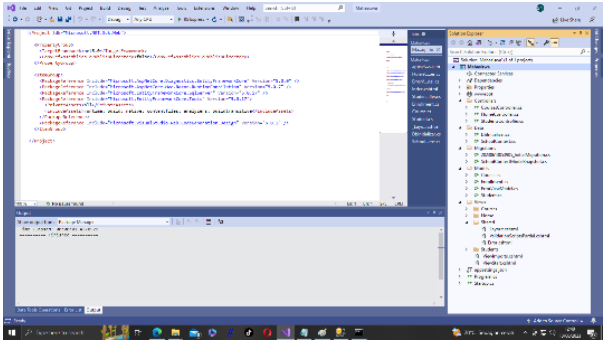


Gambar 5. Proses migrasi dari server lokal ke Azure

Tabel 4. Kode program untuk melakukan migrasi

kode program	keterangan
dotnet ef migrations add InitialCreate	Melakukan migrasi untuk pertama kali
dotnet ef database update	Untuk update dari server lokal ke Azure setelah migrasi

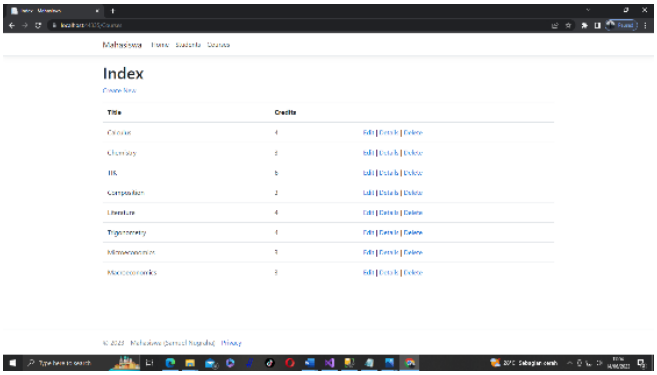
Kemudian lakukan proses migrasi untuk masing-masing *tenant*, hal ini termasuk kedalam pendekatan *separate database* dimana setiap *tenant* memiliki database yang terpisah antar pengguna layanan. Jika ada perubahan setelah melakukan proses migrasi maka masukkan perintah database update pada terminal *cmd* seperti pada tabel 4. Dengan melakukan migrasi database dari masing-masing *tenant*, hal ini dapat meningkatkan keamanan database pada layanan Azure, sehingga tidak terjadi kebocoran data antar *tenant* di Azure, karena masing-masing *tenant* memiliki databasenya sendiri. Hal ini yang menjadi kelebihan dari pendekatan arsitektur *separate database*, dimana selain lebih aman dari kebocoran data di Azure, juga lebih mudah saat melakukan migrasi. Meski sumber daya pada Azure digunakan oleh banyak pengguna atau disebut juga dengan *multi-tenant* tetapi karena dilakukan keamanan database dengan menggunakan pendekatan *separate database* maka data antar *tenant* tetap aman. Jika proses migrasi berhasil maka terdapat folder *Migration* pada Visual Studio seperti pada gambar 6.



Gambar 6. Folder Migrasi pada Visual Studio Code

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Untuk melakukan publish aplikasi dari server lokal ke Azure App dapat dilakukan dengan publish melalui akun Azure atau mendownload file *publish profile* pada Azure portal. Untuk hasil dari aplikasi yang dibuat pada server lokal ditunjukkan pada gambar 6 berikut.



Gambar 7. Output aplikasi pada localhost

Perlu diingat bahwa untuk dapat melakukan publish pada layanan Azure perlu menggunakan subscriptions seperti Pas-as-you-go.

4. DISKUSI

Terdapat beberapa penelitian terdahulu seperti yang dilakukan oleh Erick dan Restyandito dalam penelitian *"Migrasi Aplikasi Multi Tenancy pada Layanan Komputasi Awan"*[5], Joan dan kawan-kawan dalam penelitian "*Layanan Infrastruktur Komputasi Multi Tenat dengan OpenStack di Lingkungan MaaS"*[7], serta yang dilakukan oleh Hussain Aljahdali dan kawan-kawan dalam penelitian *"Multi-Tenancy in Cloud Computing"* [6] dan yang dilakukan oleh oleh Mella Marliana dengan judul "*Keamanan dan Pencegahan Database Cloud Computing Untuk Pengguna Layanan"* [8]. Namun pada penelitian tersebut menggunakan arsitektur *shared database, separated schema* yang mana hal ini dapat menyebabkan masalah di database karena database masing-masing *tenant* tersimpan pada satu database saja. Serta untuk proses migrasi yang dilakukan pada penelitian tersebut menggunakan layanan Azure VM yang mana layanan ini kurang sederhana karena harus melakukan proses instalasi, kemudian harus melakukan konfigurasi untuk layanan web server IIS serta biaya yang digunakan untuk satu VM Azure tergolong mahal. Keterbaharuan pada penelitian ini adalah dengan menggunakan layanan Azure App Service yang mana selain lebih mudah untuk dikonfigurasi biaya yang digunakan juga lebih murah.

Selain itu dalam penelitian yang dilakukan oleh Matheus [11], menggunakan layanan *cloud* yaitu *Salesforce* untuk melakukan proses migrasi, ada juga yang melakukan proses virtualisasi untuk melakukan migrasi ke layanan *cloud* [12] , ada juga yang melakukan migrasi dengan metode copy-pasted [13] ,serta menggunakan layanan IaaS [14] ,sementara penelitian ini menggunakan layanan *cloud* yaitu Azure serta menggunakan tipe PaaS [15], alasan memilih Azure adalah karena proses pembuatan aplikasi menggunakan Visual Studio dan database yang digunakan adalah SQL, selain dari vendor yang sama, terdapat juga beberapa *tools* yang memudahkan saat melakukan proses migrasi.

Persamaan antara penelitian ini dan penelitian-penelitian terdahulu yaitu sama sama menerapkan metode *multi tenant* dan melakukan migrasi layanan. Namun terdapat perbedaan dengan penelitian terdahulu, dimana penelitian ini menggunakan migrasi dari server lokal ke Azure dengan menggunakan layanan Azure App, sehingga tidak perlu melakukan konfigurasi untuk web server IIS.Selain itu Azure juga mudah dan dapat diakses dari banyak *platform* [16].Kemudian arsitektur pendekatan yang digunakan adalah *separated database* [17], sehingga data dari masing-masing tenant lebih aman karena data tidak terdapat dalam satu database saja, dan studi kasus yang digunakan adalah pembuatan database mahasiswa.

5. KESIMPULAN

Terdapat beberapa kesimpulan dari penelitian ini yaitu layanan komputasi awan dengan tipe PaaS (*Platform as a Service)* yaitu Azure App adalah tipe layanan yang lebih mudah dan hemat dalam melakukan proses migrasi dari server lokal menuju Azure. Kemudian desain arsitektur *multi tenant separated database* memiliki keamanan yang lebih baik karena tiap *tenant* memiliki database tersendiri sehingga data dari tiap *tenant* tidak bercampur menjadi satu. Sementara dari segi biaya layanan Azure App Service lebih hemat jika dibandingkan dengan pembuatan *Virtual Machine* (VM).

p-ISSN: 2775-4227
e-ISSN: 2775-4219

DAFTAR PUSTAKA

[1] I. Ahmad, H. Bakht, And U. Mohan, "Cloud Computing-A Comprehensive Definiton Cloud Computing-A Comprehensive Definition," 2017. [Online]. Available: Https://Www.Researchgate.Net/Publication/314072571

[2] P. M. Mell And T. Grance, "The Nist Definition Of Cloud Computing," Gaithersburg, Md, 2011. Doi: 10.6028/Nist.Sp.800-145.

[3] H. Aljahdali, A. Albatli, P. Garraghan, P. Townend, L. Lau, And J. Xu, "Multi-Tenancy In Cloud Computing," In *Proceedings - Ieee 8th International Symposium On Service Oriented System Engineering, Sose 2014*, Ieee Computer Society, 2014, Pp. 344–351. Doi: 10.1109/Sose.2014.50.

[4] J. Fiaidhi, I. Bojanova, J. Zhang, And L.-J. Zhang, "Guest Editors' Introduction," 2012. [Online]. Available: Www.Appirio.Com/Ecosystem/Appirio.Swf

[5] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, And Q. Li, "Comparison Of Several Cloud Computing Platforms," In *2nd International Symposium On Information Science And Engineering, Isise 2009*, Ieee Computer Society, 2009, Pp. 23–27. Doi: 10.1109/Isise.2009.94.

[6] E. Kurniawan Restyandito Abstrak, K. Kunci, K. Awan, And A. Multitenancy, "Migrasi Aplikasi Multitenancy Pada Layanan Komputasi Awan."

[7] J. R. Panggabean, A. B. Prasetijo, And E. D. Widianto, "Layanan Infrastruktur Komputasi Multitenant Dengan Openstack Di Lingkungan Maas," *Jurnal Teknologi Dan Sistem Komputer*, Vol. 5, No. 4, P. 142, Oct. 2017, Doi: 10.14710/Jtsiskom.5.4.2017.142-146.

[8] M. Marliana, "Keamanan Dan Pencegahan Database Cloud Computing Untuk Pengguna Layanan," Vol. 3, No. 2, 2019.

[9] "150031-Id-Implementasi-Modul-Modul-Enterprise-Reso".

[10] Y. Fauziah, "Tinjauan Keamanan Sistem Pada Teknologi Cloud Computing," 2014.

[11] M. S. Rumetna, "Title Case," *Jurnal Teknologi Informasi Dan Ilmu Komputer*, Vol. 5, No. 3, P. 305, Aug. 2018, Doi: 10.25126/Jtiik.201853595.

[12] W. S. Prabowo, M. H. Muslim, And S. B. Iryanto, "Pusat Data Privat Virtual Pemerintah Berbasis Komputasi Awan (Studi Empiris Pada Lembaga Ilmu Pengetahuan Indonesia) Pusat Data Privat Virtual Pemerintah Berbasis Komputasi Awan (Studi Empiris Pada Lembaga Ilmu Pengetahuan Indonesia) Government Virtual Private Data Center Based On Cloud Computing (Empirical Study On Indonesian Institute Of Sciences-Lipi)."

[13] M. Gilang, W. Utama, R. Latuconsina, And M. F. Ruriawan, "Live Migration Pada Cloud Computing Dengan Metode Post-Copy Live Migration In Cloud Computing Using Post-Copy Method."

[14] P. Aptika Dan Ikp, B. Litbang Sdm, And K. Jl Medan Merdeka Barat No, "Implementasi Cloud Computing Di Beberapa Instansi Pemerintahan Cloud Computing Implementation In Several Government Institutions Faiq Wildana."

[15] "Cloud Computing-An Overview."

[16] S. R. Gundu, C. A. Panem, And A. Thimmapuram, "The Dynamic Computational Model And The New Era Of Cloud Computation Using Microsoft Azure," *Sn Comput Sci*, Vol. 1, No. 5, Sep. 2020, Doi: 10.1007/S42979-020-00276-Y.

[17] B. Gao *Et Al.*, "A Non-Intrusive Multi-Tenant Database Software For Large Scale Saas Application," In *Proceedings - 2011 8th Ieee International Conference On E-Business Engineering, Icebe 2011*, 2011, Pp. 324–328. Doi: 10.1109/Icebe.2011.23.