SQL Server Modernization on Azure

Microsoft training

Contents

Introduction

Welcome to the SQL Server Migration and Modernization Course in Azure. This course was created by **Pablo Ariel Di Loreto**, and will allow you to learn the basics of SQL Server in the Microsoft Azure cloud, including IaaS and PaaS services.

Keep in mind that this course requires some knowledge of technology and general concepts of software development, which are not covered in it.

The Contents of this course are optimized to prepare you in the best possible way to learn in a practical way. Likewise, in addition to having access to this course, you will be participating in the Microsoft Workshop with the same name for 2 days of 3 hours each.

Thank you for your participation!

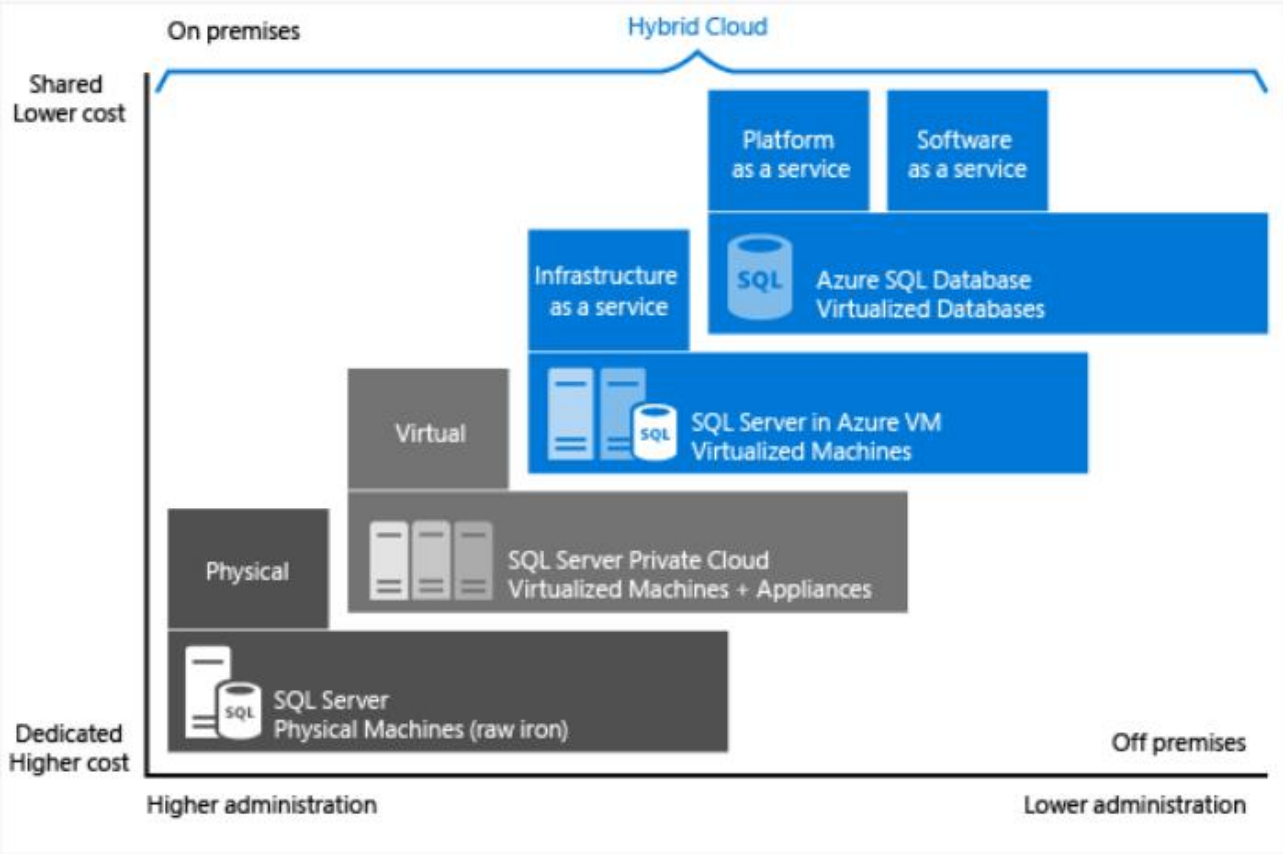
About Azure SQL

Azure SQL is a family of intelligent, secure, and managed products that use the Azure cloud SQL Server database engine. The options currently present are:

- Azure SQL Database – Support modern cloud applications in an intelligent, managed database service that includes serverless compute.
- Azure SQL Managed Instance – Modernize your existing SQL Server applications at scale with an intelligent fully managed instance as a service, which has near 100% feature parity with the SQL Server database engine. the best option for most cloud migrations.
- SQL Server on Azure Virtual Machines: Lift-and-shift your SQL Server workloads with ease and retain 100% SQL Server compatibility and OS-level access.

Azure SQL is built on the familiar SQL Server engine, so you can easily migrate applications and continue to use the tools, languages, and resources you're familiar with. Your skills and experience move to the cloud, so you can get more done with what you already have.

As seen in the diagram, each service offer can be characterized by the level of administration that is given over the infrastructure and the degree of cost-effectiveness.



In Azure, you can have SQL Server workloads run on a hosted service (PaaS) or on a hosted infrastructure (IaaS). In PaaS, you have multiple product options and service levels within each option. The key question to ask when deciding between PaaS or IaaS is whether you want to manage your database, patch, and perform backups, or delegate these operations to Azure.

Exercise 1: SQL Virtual Machines

Introduction to the exercise

In this section we will make the first experience together. This experience requires you to have a Microsoft Azure subscription set up and available and consumes any credits you have on it. Please review this topic to avoid charges on subscriptions where you don't want them to exist.

In this experience we will:

- Create a SQL Server on Virtual Machine with Windows.
- Learn how to check images available in Azure for VMs with integrated SQL Server.
- Access administration tools that we will use throughout the course.
- Know what is the IaaS agent for virtual machines with SQL Server.

About SQL Server Virtual Machines

SQL Server on Azure Virtual Machines allows us to use full versions of SQL Server in the cloud without having to manage all the local hardware. SQL Server on Azure Virtual Machines is an IaaS option that also simplifies license costs when you pay as you go through a consistent experience that we'll cover in this experience.

Let's explore some points about Licensing, Availability and Management.

SQL Server Licensing on Azure Virtual Machines

SQL Server software on an Azure VM must be licensed. There are automatic mechanisms to detect that a computer is running SQL Server, so one of the following models must be chosen.

Pay Per Use License

There are virtual machine images that implement SQL Server on a pay-as-you-go model. The scope is for both Enterprise, Standard, Web, Developer and Express editions (depending on the version of SQL Server). This will cause the cost of the license to be charged to the monthly price of the virtual machine (or virtual machines).

BYOL license

You can also bring your own license (BYOL). In this scenario, you only pay for the virtual machine without any additional SQL Server license charges in the monthly price. Bringing your own license can save money over time in ongoing production workloads.

To use your own license, you can convert an existing pay-as-you-go SQL Server virtual machine, or deploy an image with {BYOL} as a prefix.

Managing SQL Server on Azure Virtual Machines

Updates

SQL Server on Azure Virtual Machines provides facilities for scheduling SQL Server and Windows updates. This is possible through a special configuration panel for SQL Server on VMs. However, do not forget that being IaaS you are responsible for its maintenance.

Backups

SQL Server on Azure Virtual Machines can take advantage of Automated Backup, which regularly backs up the database to an Azure storage account, specifically in block blobs. However, do not forget that being IaaS you are responsible for its maintenance.

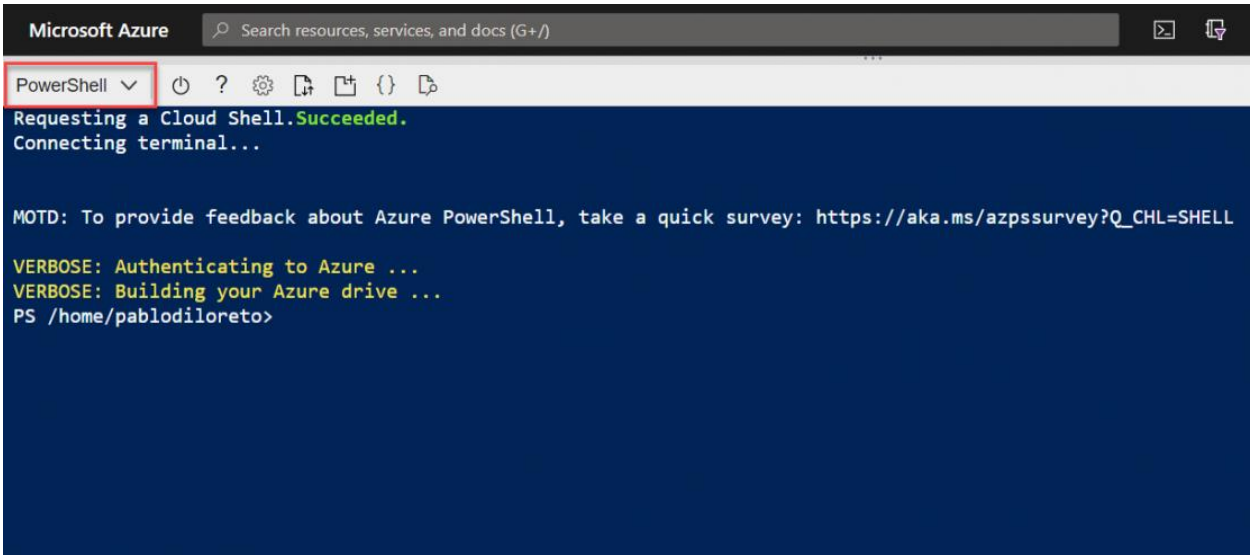
Query Available Images of Microsoft SQL Server on Azure

We are going to explore by command line what are the offers of Microsoft SQL Server in Azure (that is, in VMs). This can be done by pre-specifying an Azure location. Doing it from the command line is a simple and fast way to get the results, without exploring extensive documentation or the graphical interface.

Enter the URL https://shell.azure.com/ and log in with your Azure user:

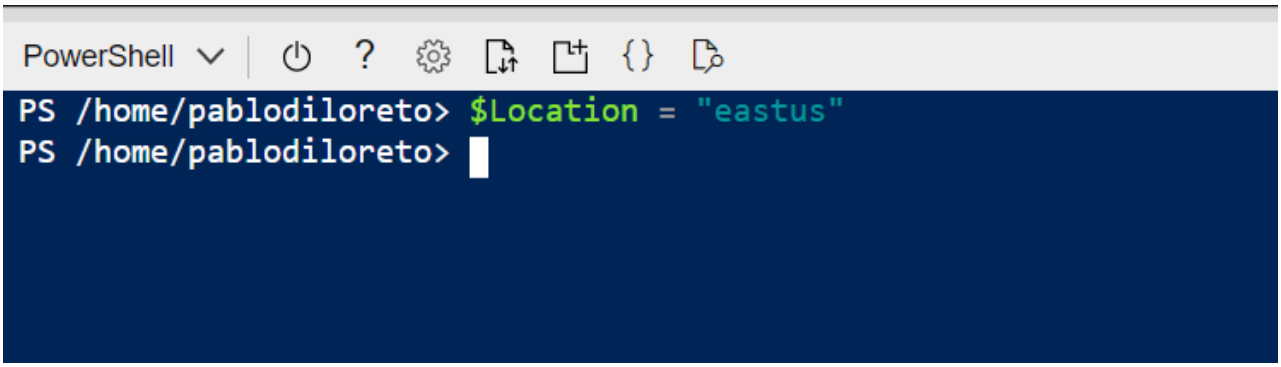
If you have never started Cloud Shell before, you may be prompted to create a storage account. Follow the steps to generate it.

Once you've done that, choose "PowerShell" as your scripting option:



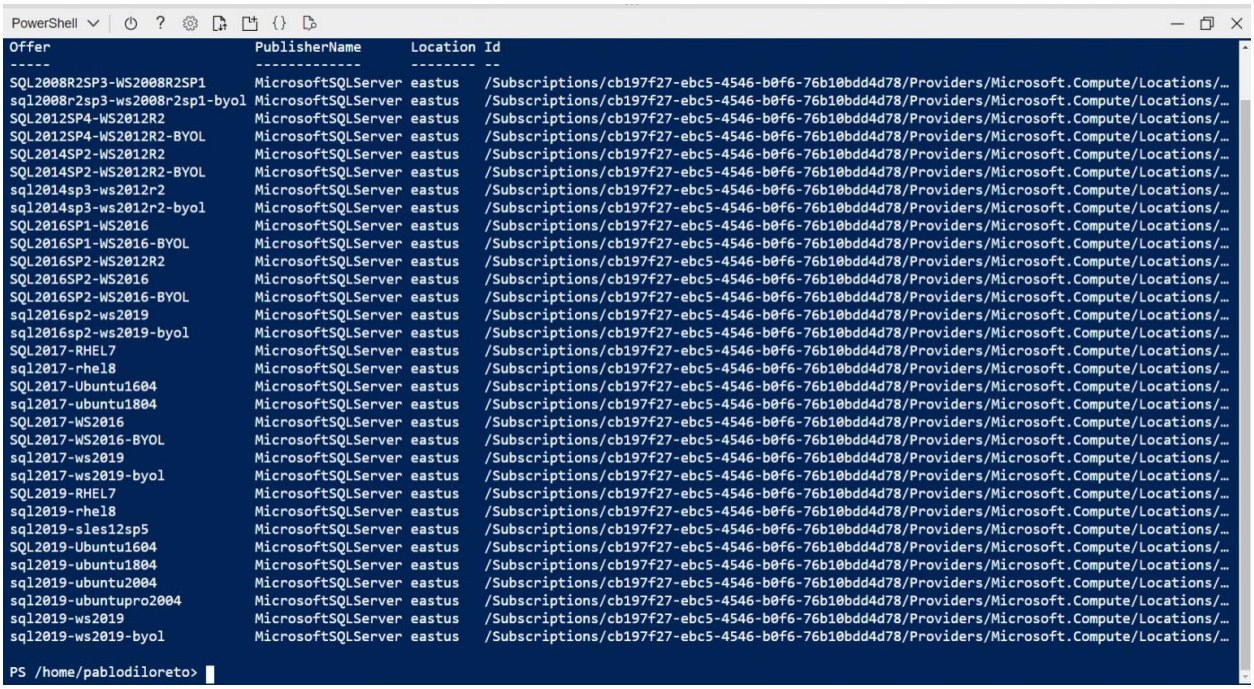
To find out about SQL Server offerings on Azure for “IaaS”, and as we said before, we must specify a location. Let's generate a variable called $Location and store a location there: eastus:

$Location = "eastus"



You will run the following command to find out the offers, reusing the "Location" variable and specifying the Publisher in MicrosoftSQLServer:

Get-AzVMImageOffer -Location $Location -Publisher 'MicrosoftSQLServer'



You are welcome to build queries using the $Location variable with other locations. The list of current locations can be obtained with the following PowerShell command:

Get-AzLocation | select displayname,location

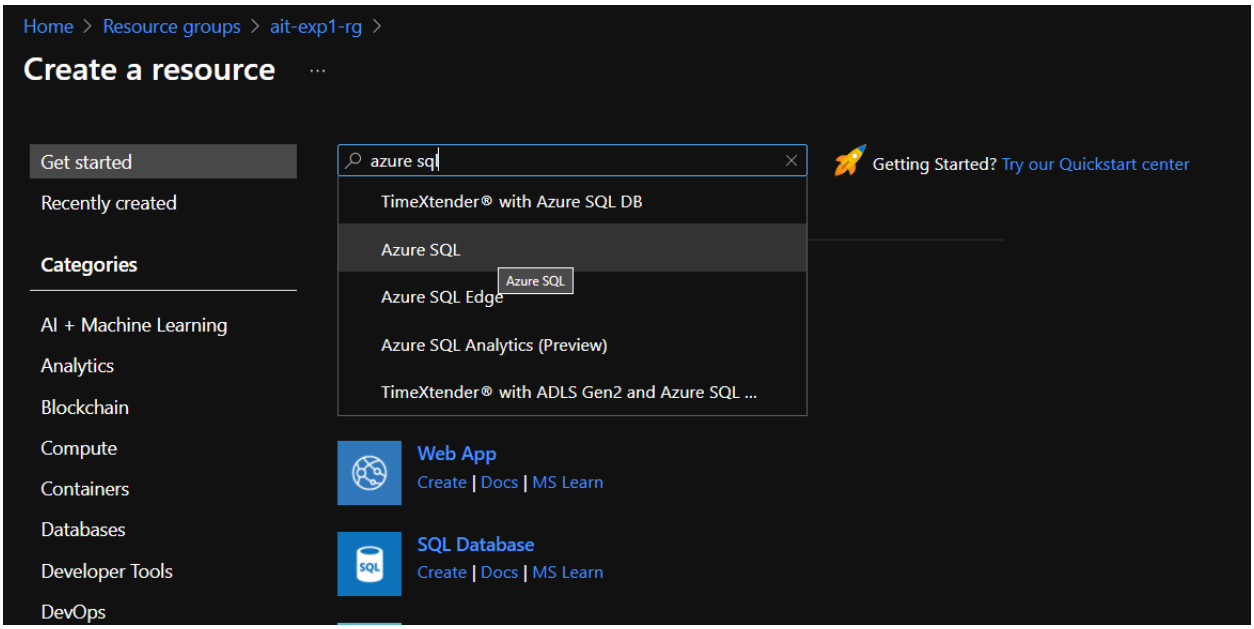
Creating a Windows VM with SQL Server

Let's create our first VM in Azure with Integrated SQL. In this case we are going to use a virtual computer with Windows operating system. Remember that you also have Linux options.

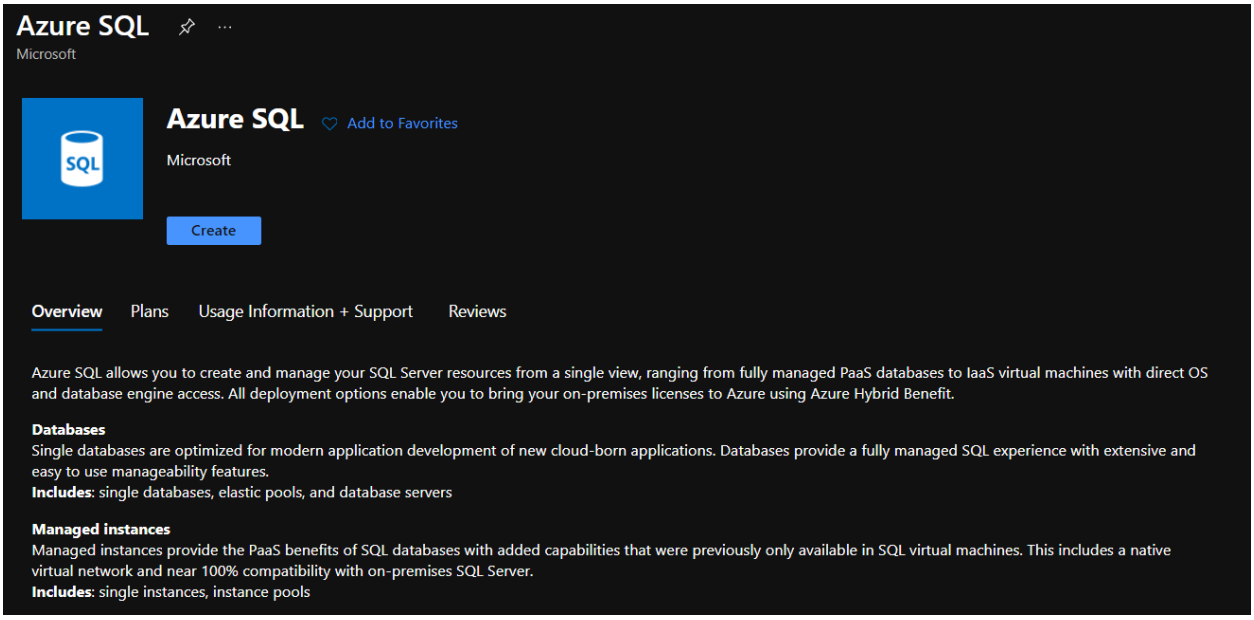
Selecting the IaaS Deployment Method

From the Azure portal we are going to work on the same Resource Group that we worked on before, whose suggested name is “sql101a-exp1-rg”. Once there you will click on new resource.

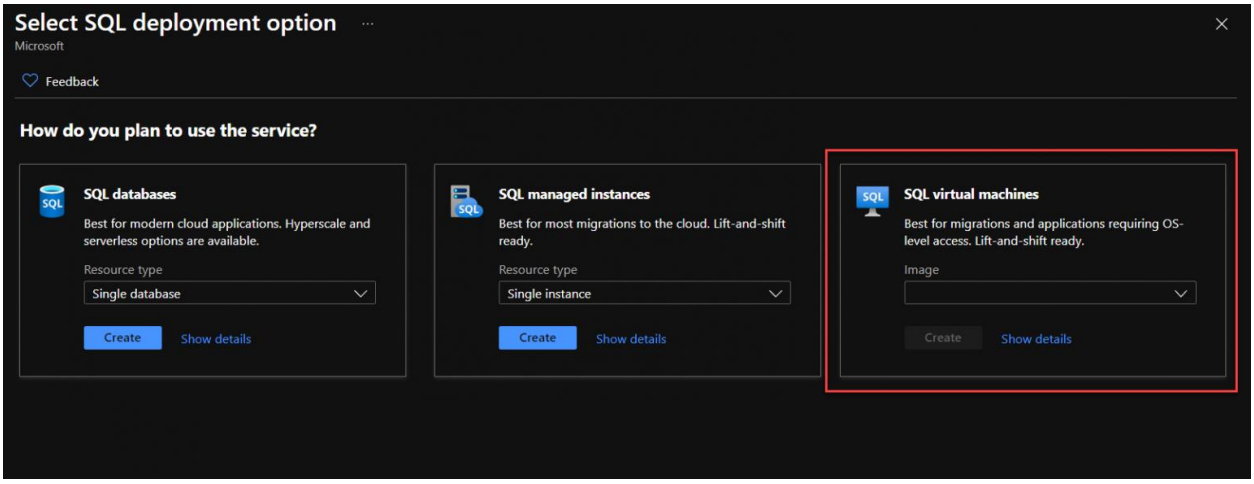
In the new resource window you will search for “azure sql” as seen on the screen:



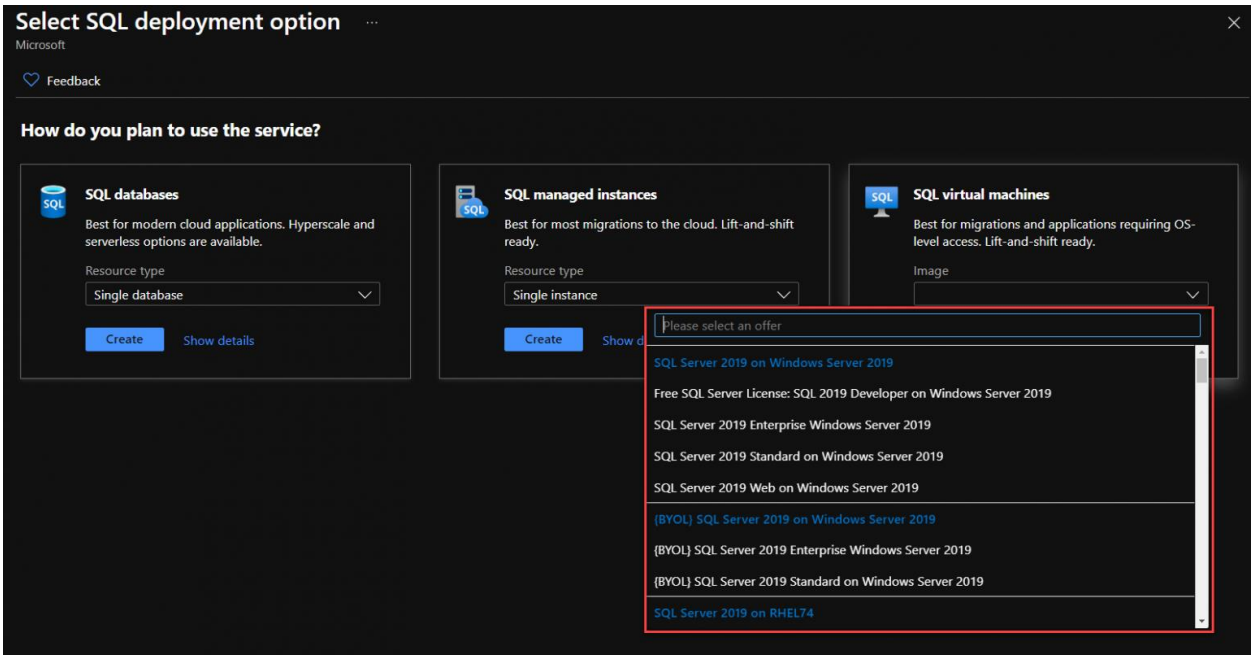
The result that will appear will be the one published by Microsoft, which is called “Azure SQL. We will click on Create:



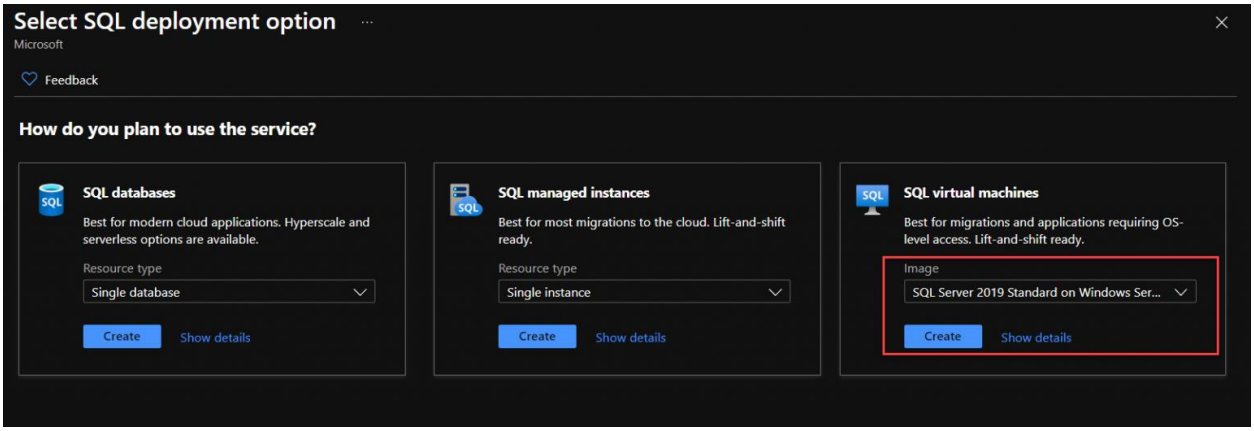
A set of options will appear that is the same as the one we have seen in the theoretical introduction. We can choose to create an Azure SQL as a Database, as a Managed Instance and as a Virtual Machine. We will focus on Virtual Machine:



If we display the image option, we can quickly see all the images offered:



We are going to choose the option “SQL Server 2019 Standard ok Windows Server 2019”:

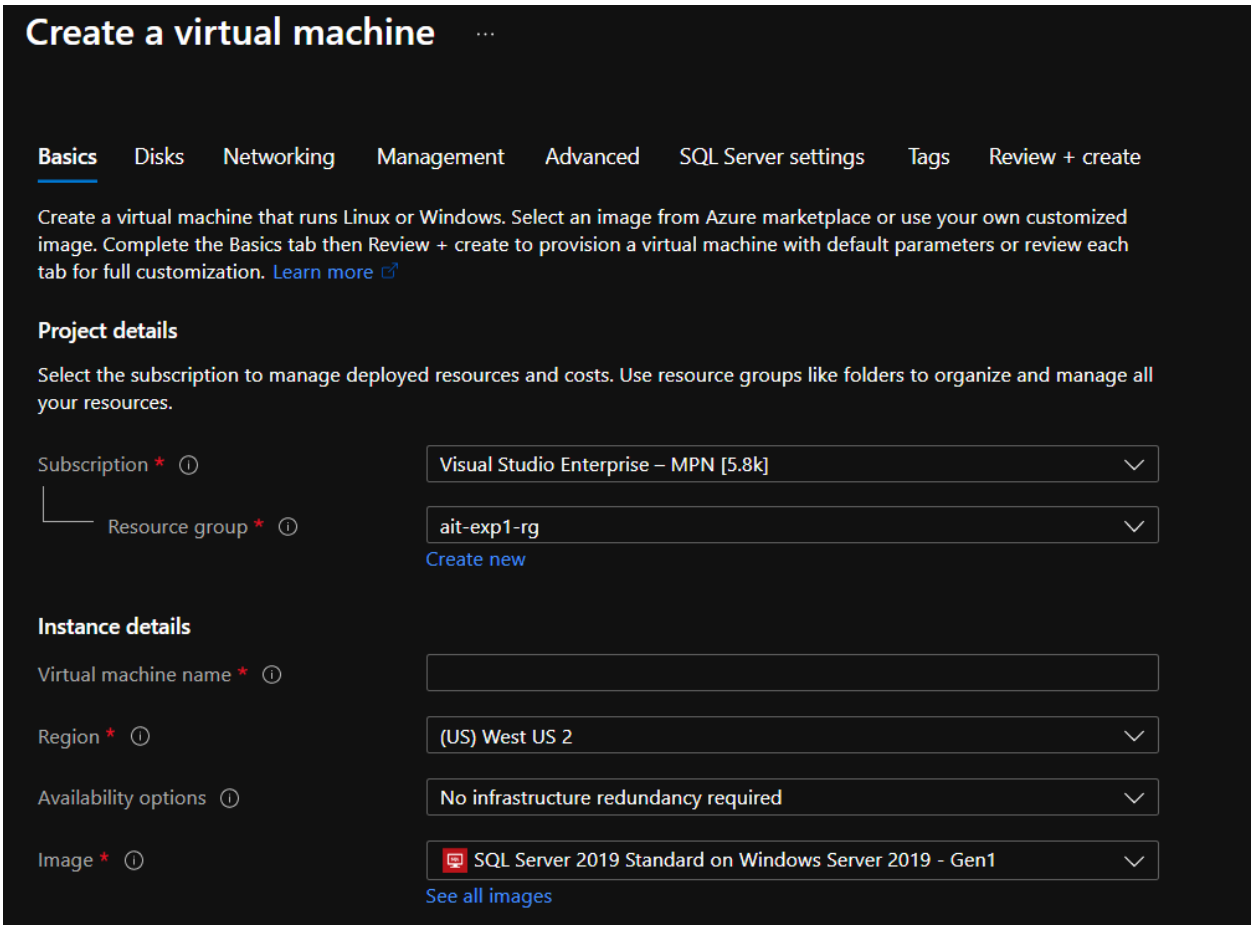


Creation of the VM

The wizard that we will find next is nothing more than the creation of a virtual machine, although with a specific image with integrated SQL. Beyond the similarity, we will discover that we will have a tab of the wizard dedicated to SQL Server settings.

Basic Options

The basic options of the virtual machine are those that we must enter YES OR YES to advance in the wizard. This flap looks like the following image:



We are going to create the machine with the following configuration:

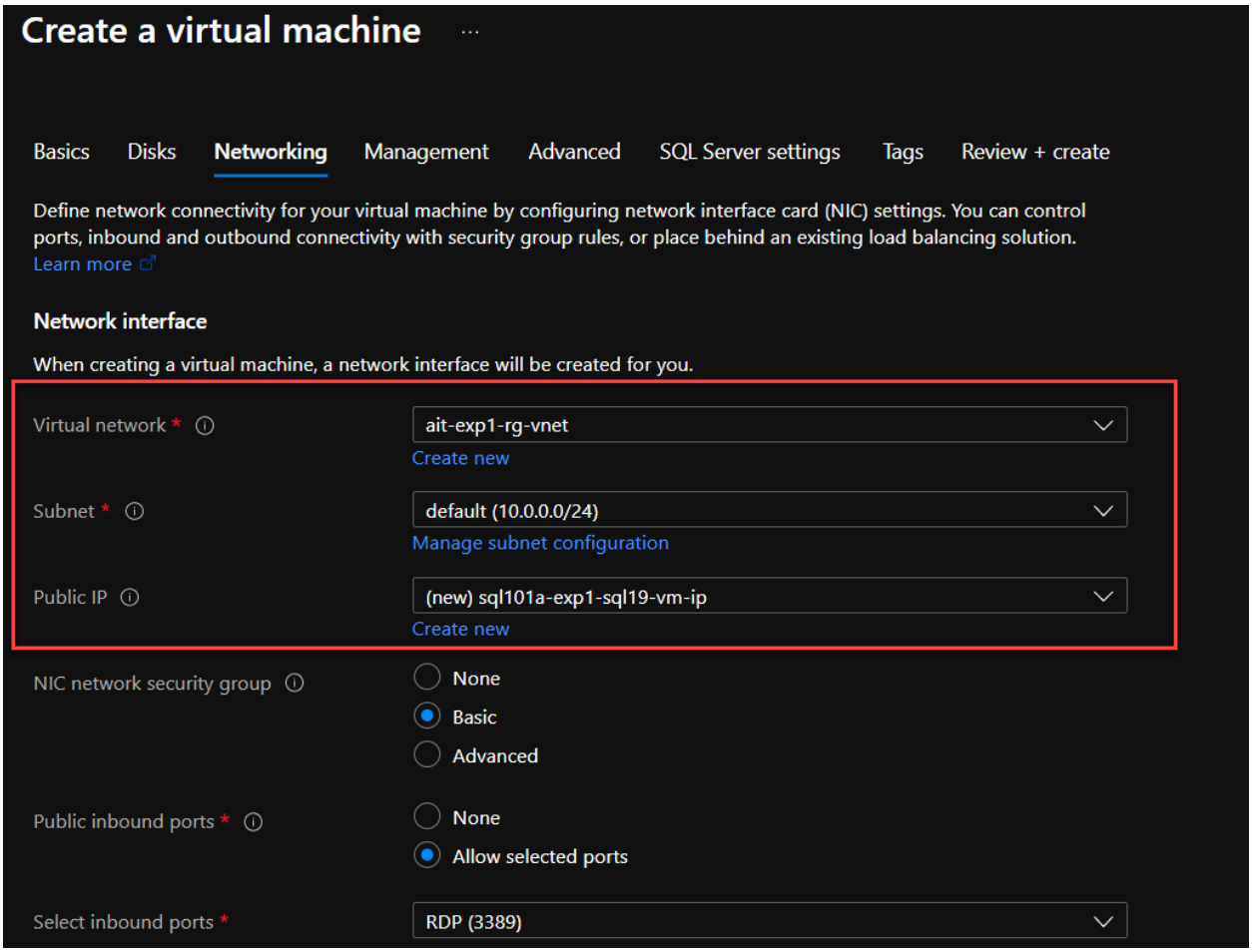
- Resource Group: sql101a-exp1-rg
- Operating System: SQL Server 2019 on Windows Server 2019
- VM Name: sql101a-exp1-sql19-vm
- VM Size: Standard_B2ms
- Suggested Region: East US
- VM username and password: generate a personalized one
- RDP access: enabled
- Disks: 1 disk only (operating system)
- Network configuration: leave the suggested ones
- Boot diagnostics: disabled

- Auto shutdown: set at a convenient time

We will leave the disk options at default.

Network Options

It is important that when we are configuring the network options, we leave the default options making sure that the virtual network is /16 and the subnet created is /24.



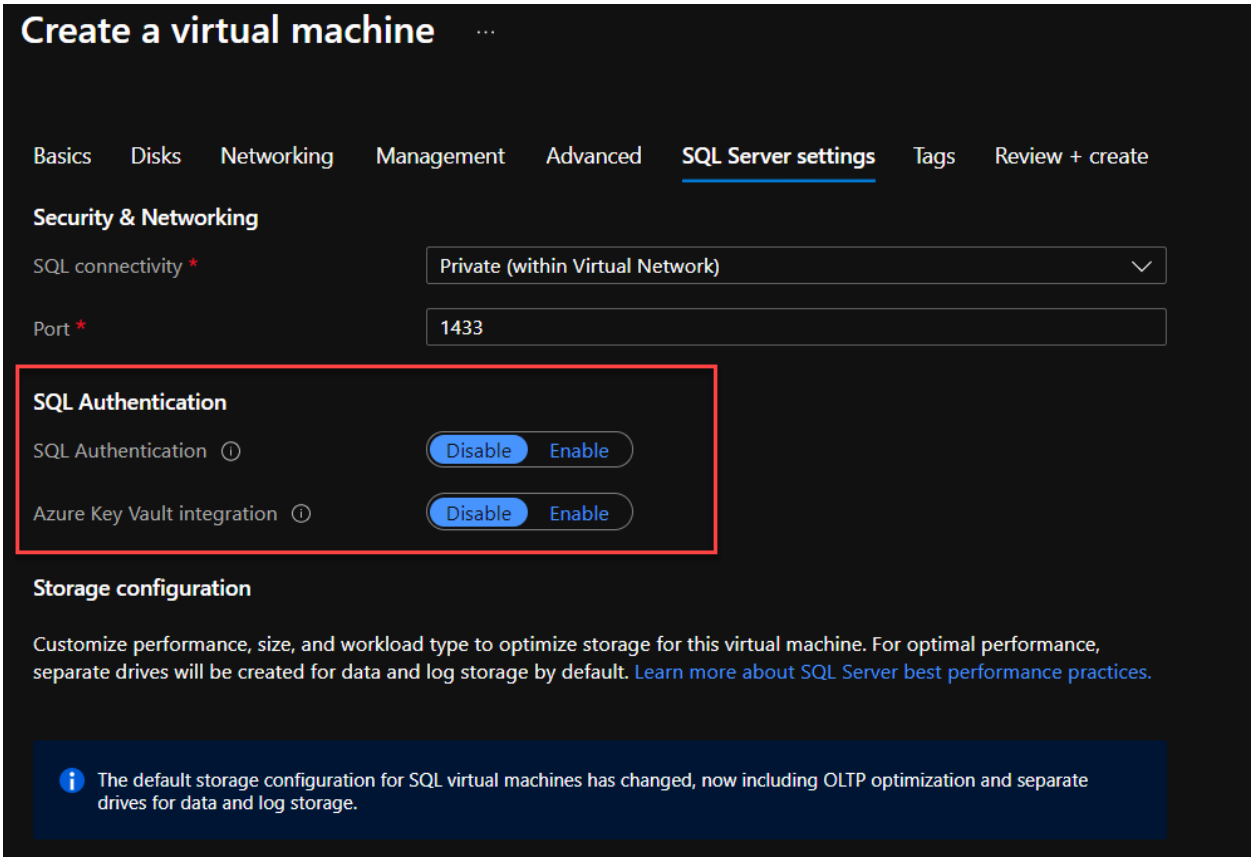
We will not modify the Management and Advanced options. We will move on to the SQL Server settings option.

SQL Server Options

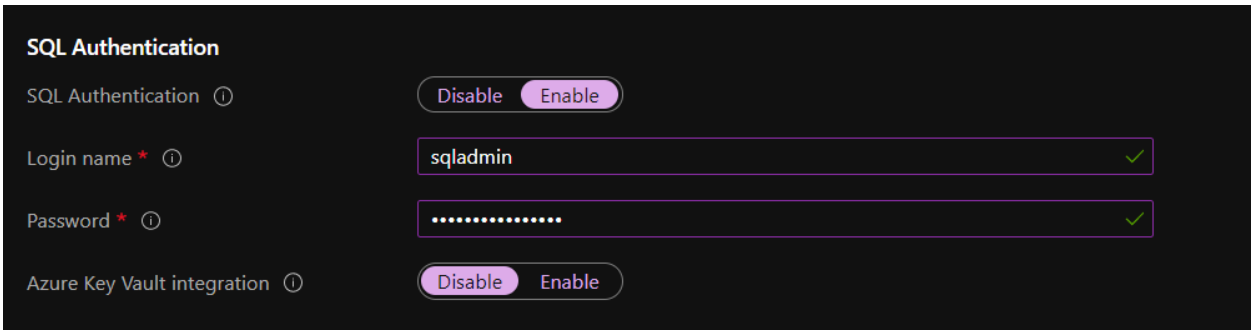
This tab is a novelty, since we are talking about virtual machines for SQL Server. From here we can configure several aspects that we do not have in a normal virtual machine.

Authentication

To begin we can configure the authentication part:

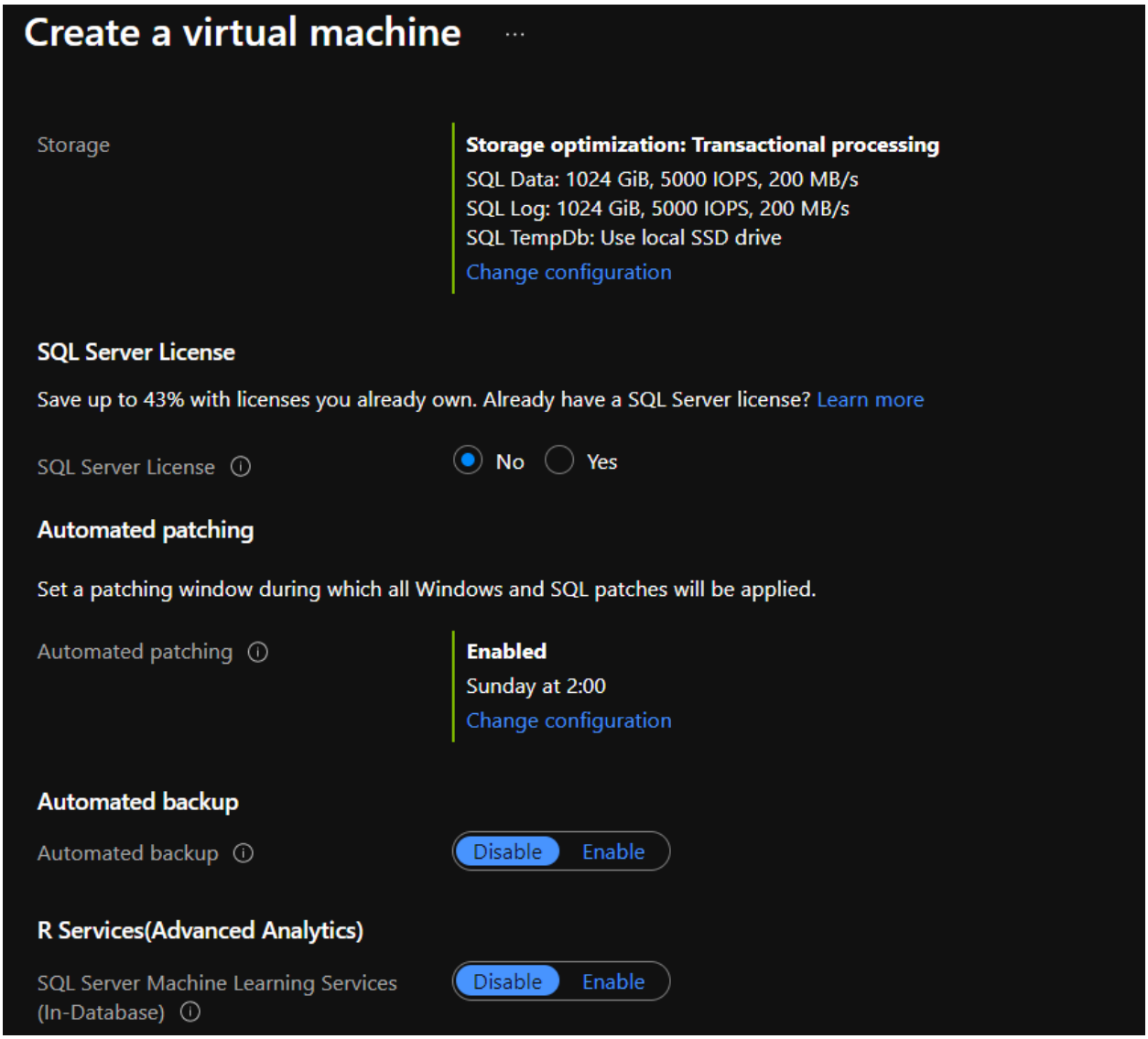


If we enable SQL authentication, we can generate a local authentication user on the server. Let's generate one with the username sqladmin and a password of your own choosing:

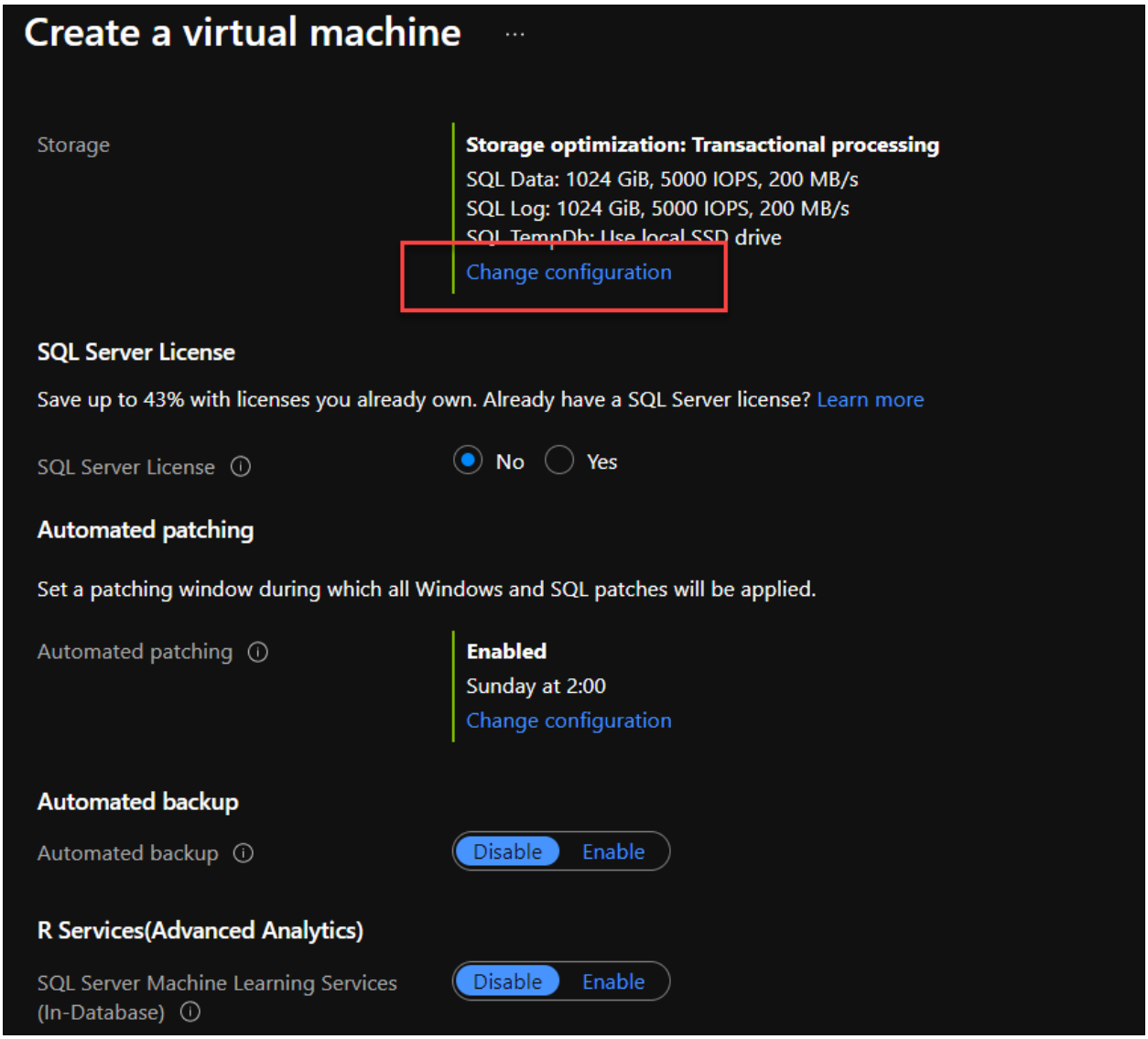


Storage Settings

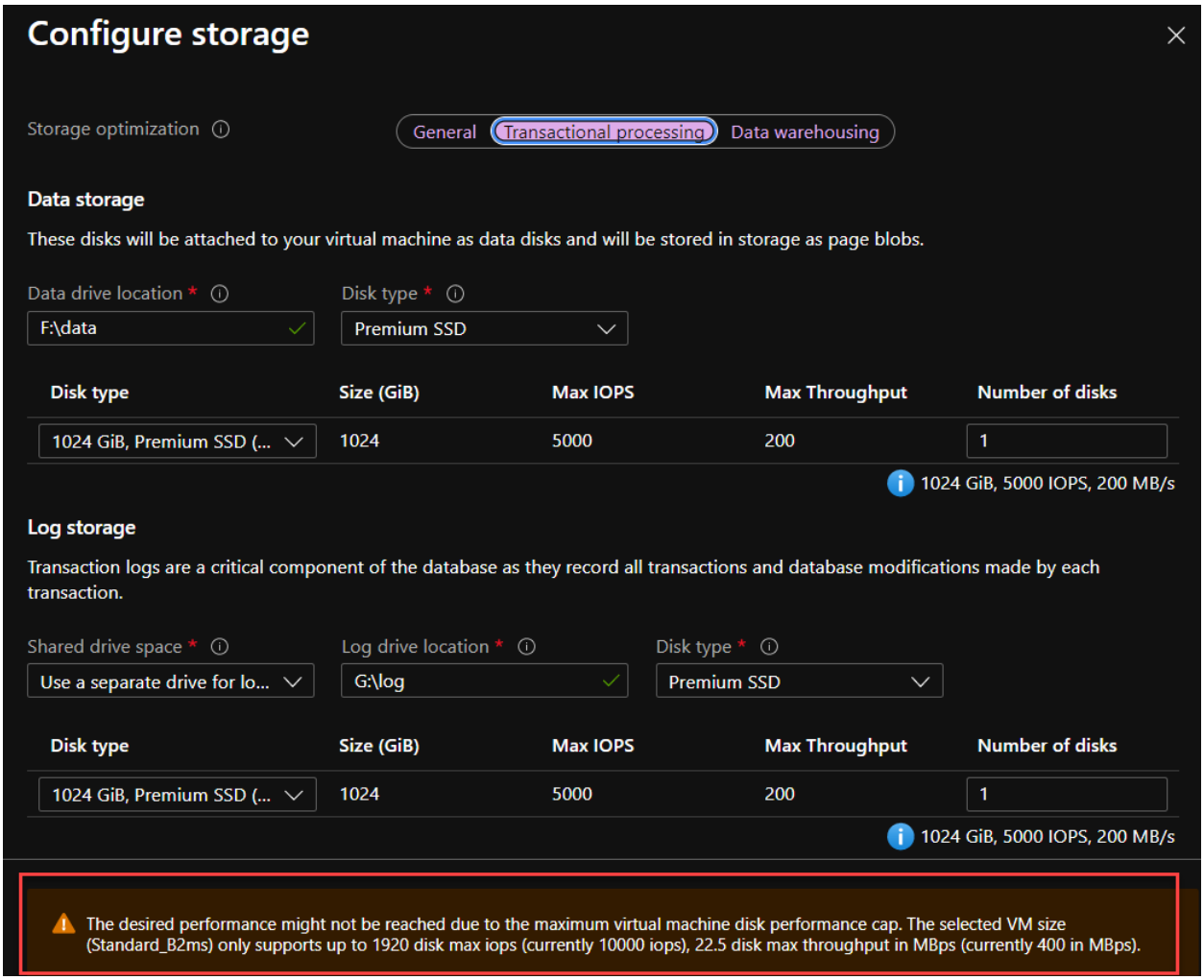
On the other hand, we can find a complete storage configuration. For this reason we have previously skipped the disk configuration in the VM creation wizard. This option allows us to focus on the storage for SQL Server:



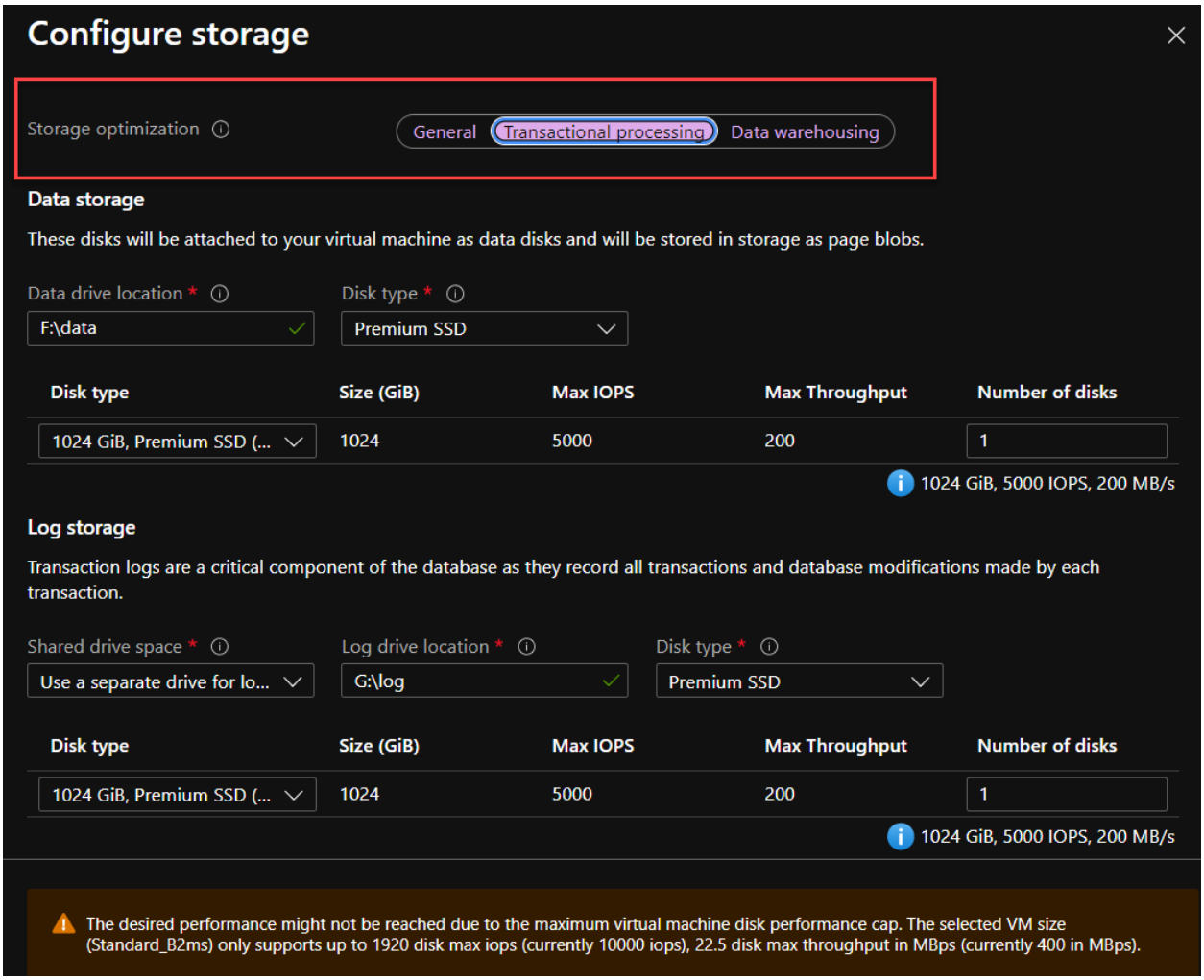
We are going to select “Change configuration” to customize the disk storage:



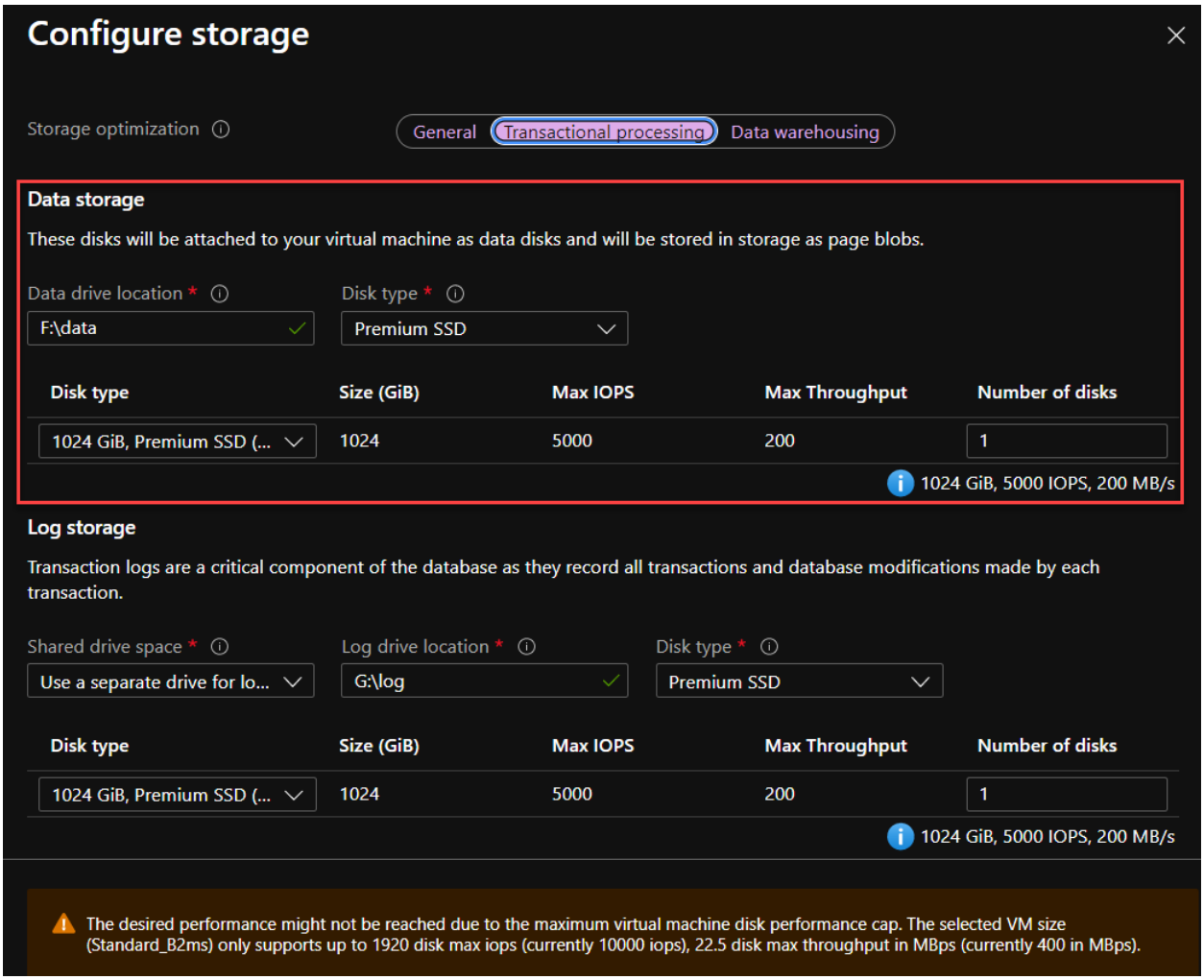
We will be able to observe that, if you chose the size that we recommend, an alert will appear since the desired performance for the selection of "Storage optimization" is not according to the size of VM that we have. Let's dismiss this alert and continue with the wizard:



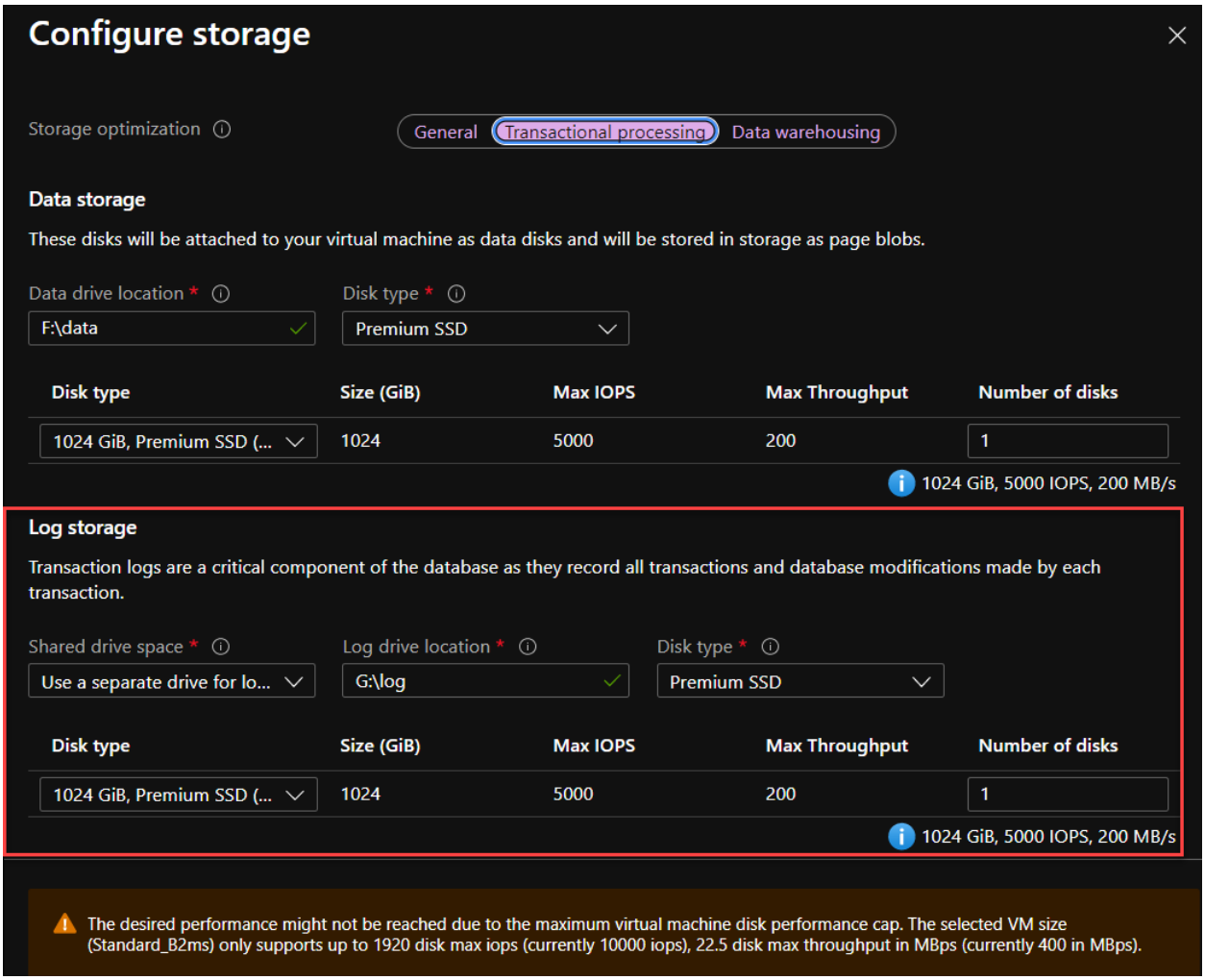
Note that we have options to choose from three storage optimization options: for general use, for transactional use, and for data warehouse. Let's choose the transactional use:



Once we choose transactional use, we will be able to customize the disk configuration for SQL Server Data. We propose you to configure the disk you want according to the options of location of the unit (Data drive location) and type of disk (Disk type):



Below we will have the option to customize the storage of logs. By default, SQL Server on VMs logically separates configuration data and configuration logs onto two separate disks. As in the previous step, we invite you to customize these options:



Keep in mind that the Data and Logs configuration must be sized, according to the use that we will give to the SQL Server and the amount of data that we will have to host, as well as the periodicity of the backups and their strategy.

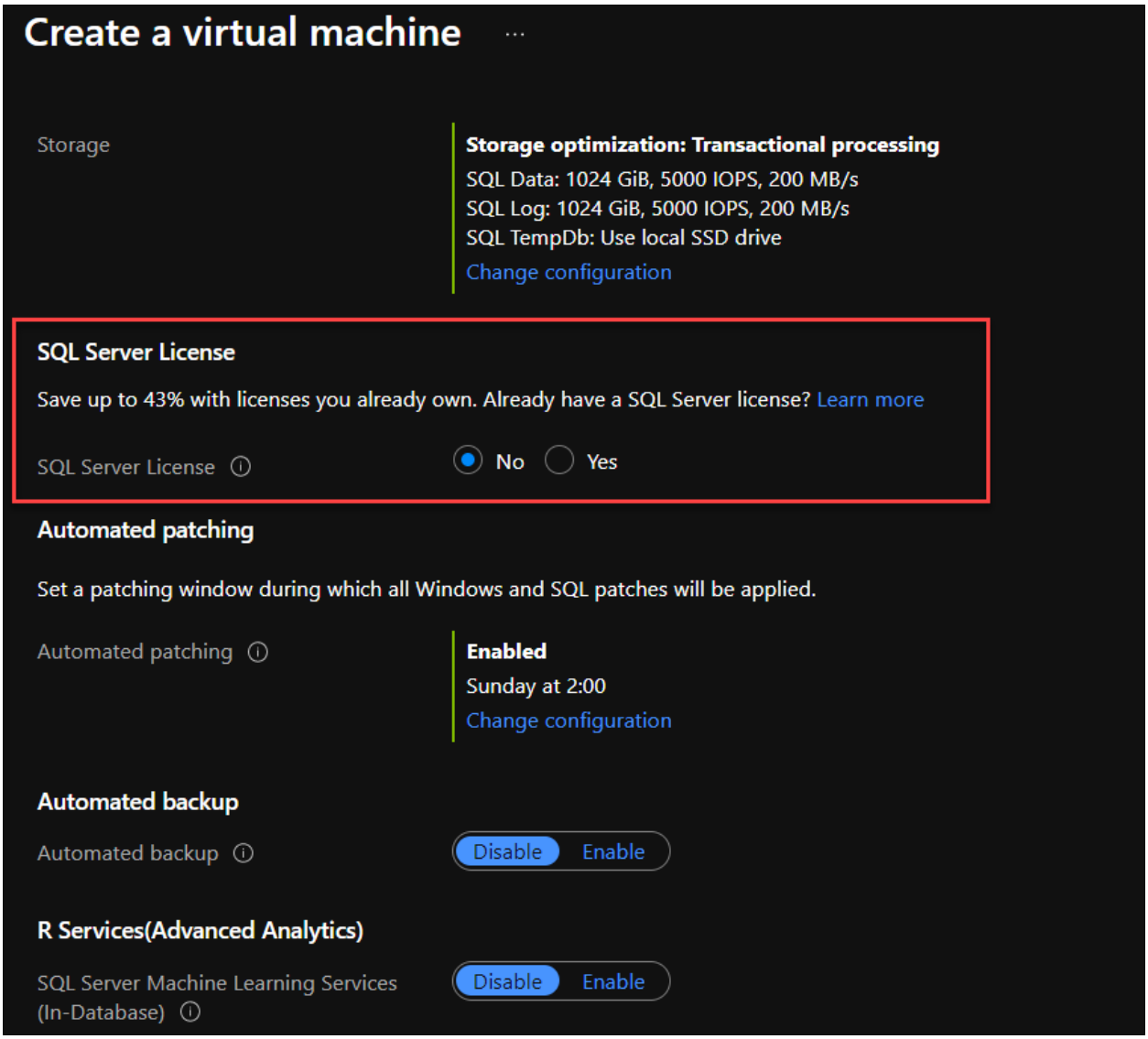
SQL Server License Configuration

We will accept this configuration and move on to configure the Licensing aspects. At this time we can:

- Choose pay per use according to the selected image (in our case SQL Server Standard).

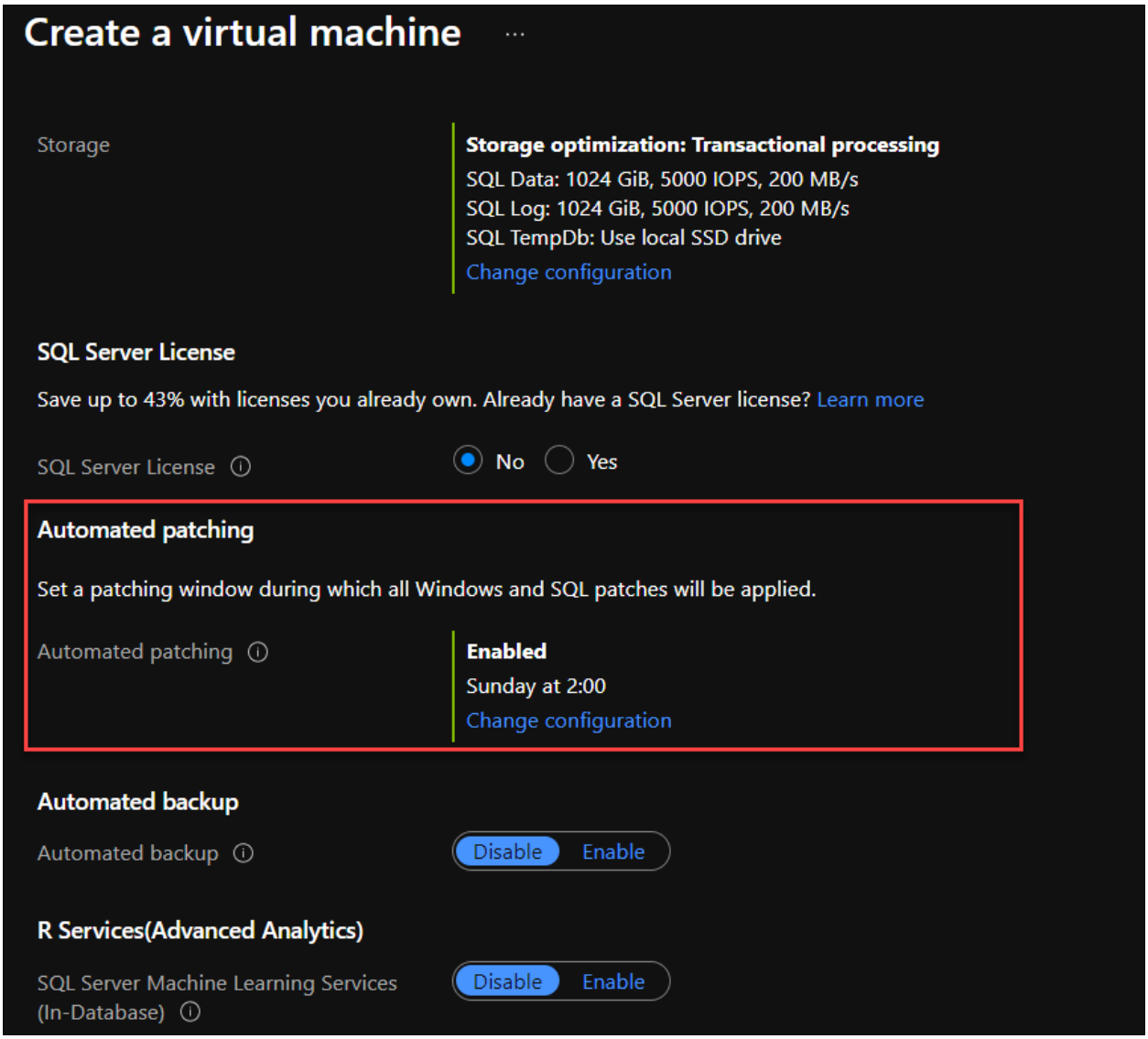
- Choose a hybrid benefit option, which is the case that we already have a SQL Server license that can be applied to an Azure VM.

If you have questions about your license, you should consult a Microsoft or partner licensing specialist. In our case we will choose the "No" option (that is, we will advance by payment for use):

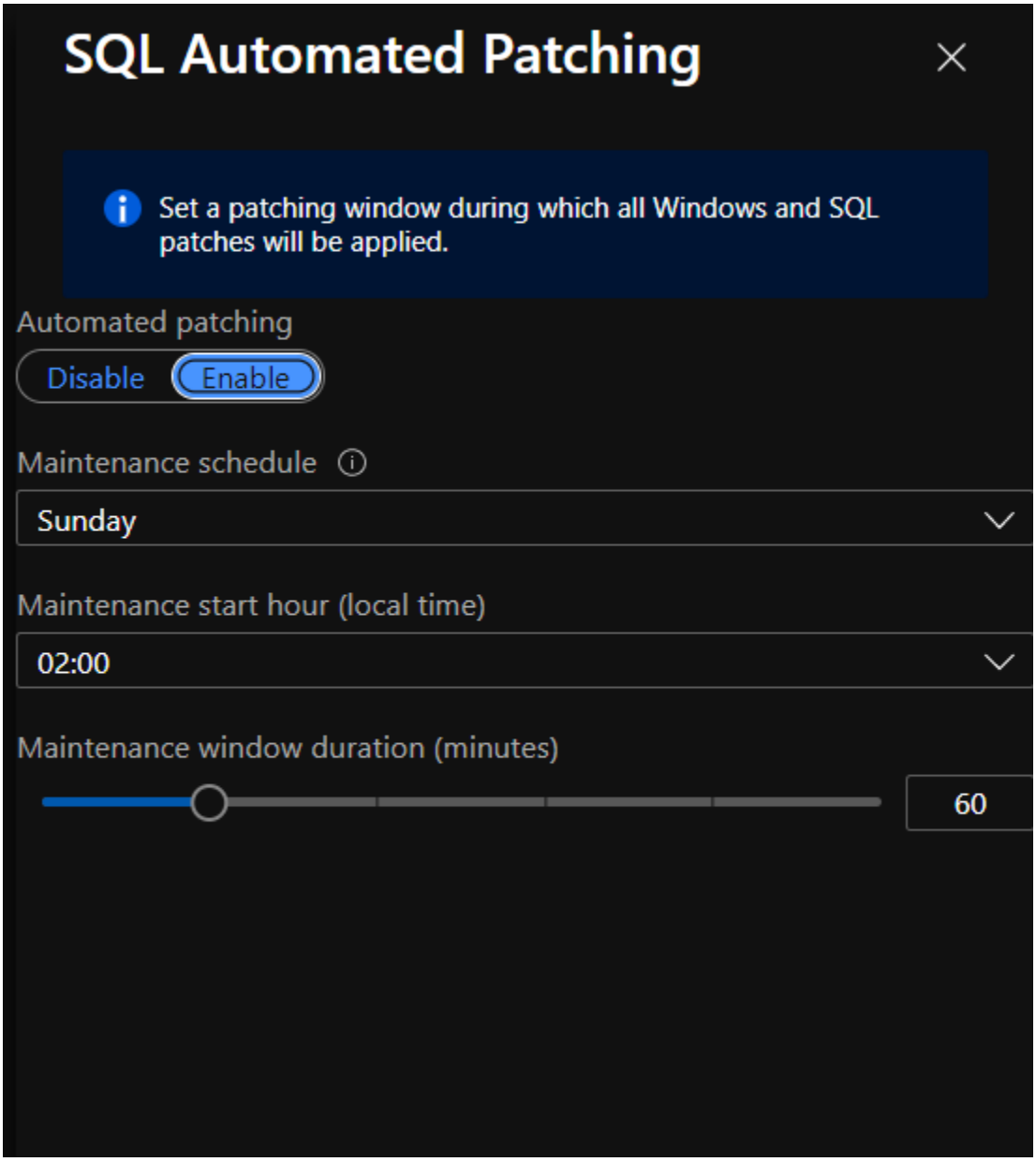


Update Settings

SQL Server in virtual machines allows us to have facilities in relation to the installation of updates. The wizard allows us to configure automatic patching, which applies updates to SQL Server at a pre-established day and time. This does not remove our responsibility for updates: we are responsible for any problems in the VM, since we are in the IaaS delivery method. Let's choose Change configuration to explore the options we have:



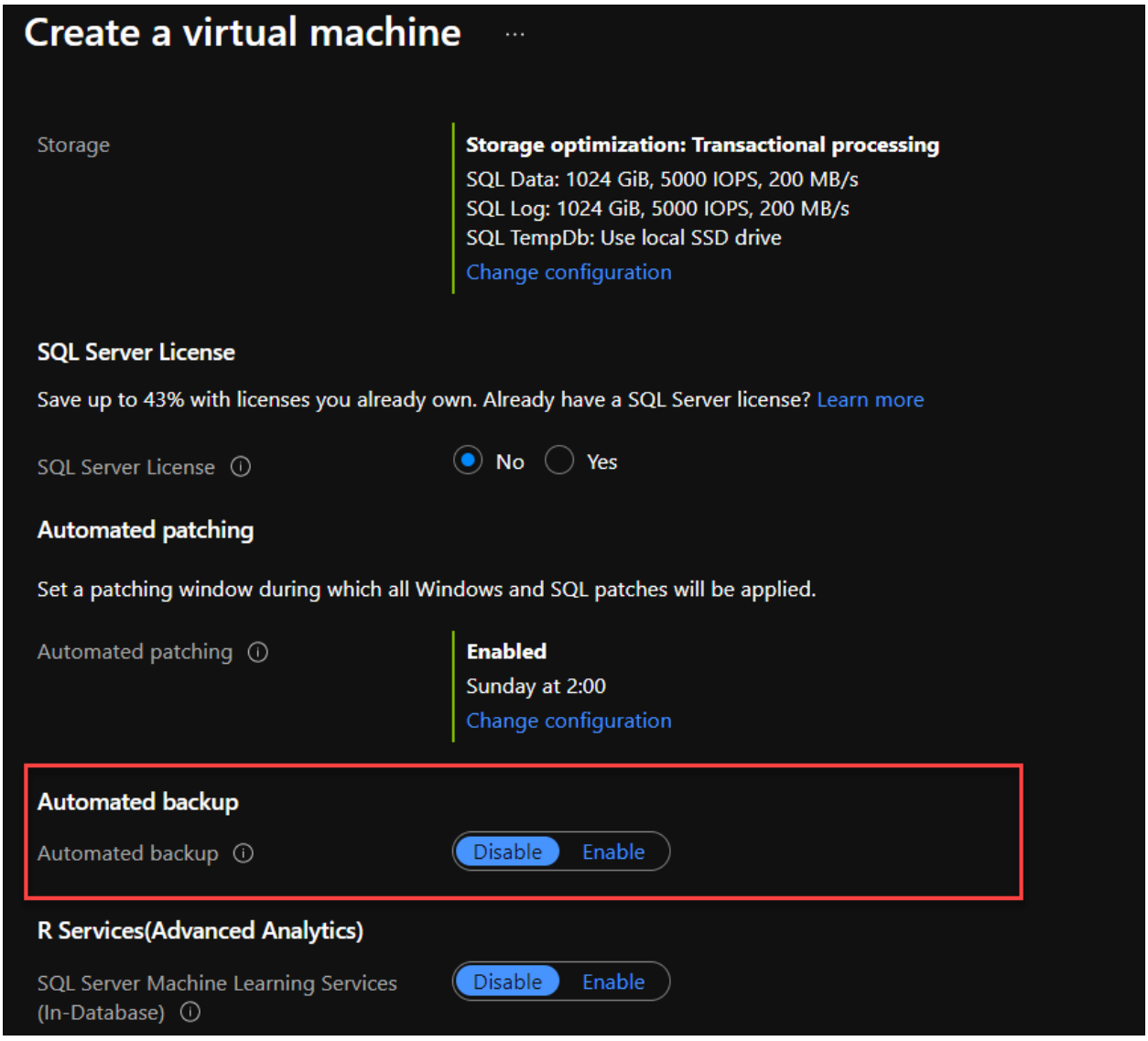
From here we can enable or disable automatic patching, and if we choose, we can select the day and time preference with a maintenance window. The maintenance window allows a space for patches to be applied.



Backup Configuration

Another aspect with which we have facilities when we talk about SQL Server VM is backup copies. We can choose to have automatic backups made in the VM to protect our databases. However, this aspect cannot be customized from here, only enabled and disabled.

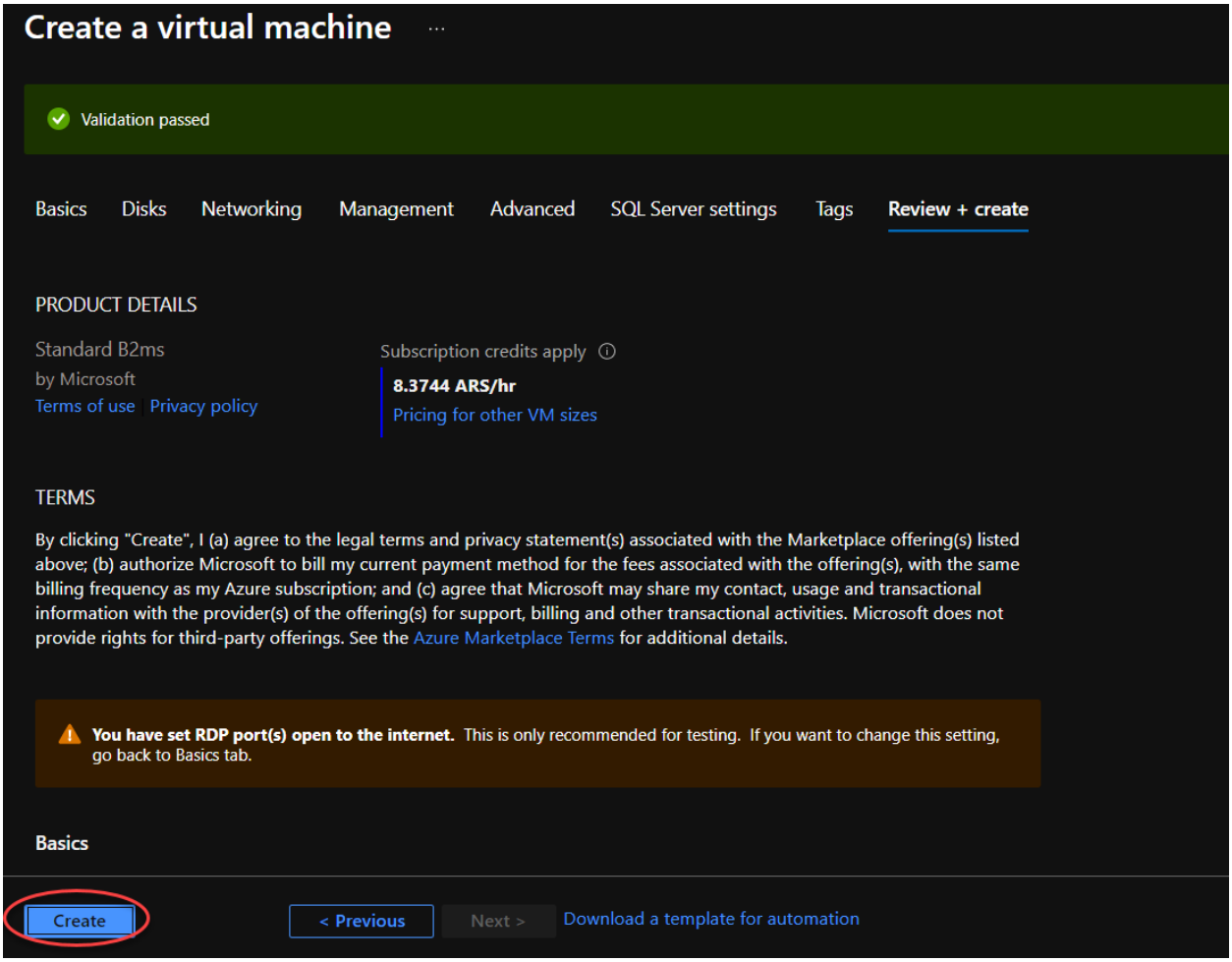
Remember that we are in IaaS delivery method: this means that the configuration we select does not remove our responsibility for backup and recovery, it is only an enabler.



At the moment we will not configure the advanced analytics services.

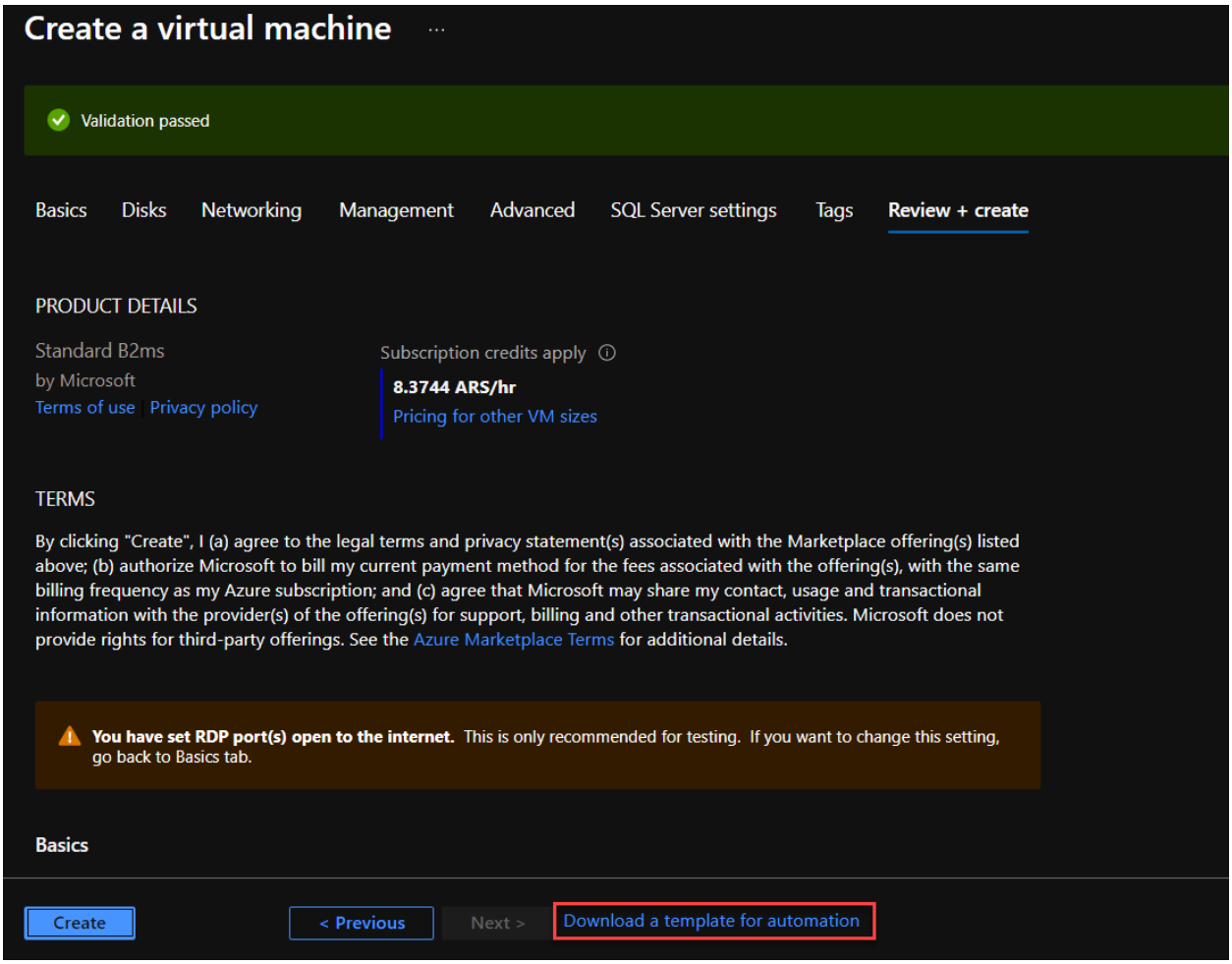
Validation and Creation of VM

The last step is the validation of the selected options. If all goes well, we can generate the VM:



Build by Infrastructure as Code

If we want to automate the tasks performed by the graphical interface, we can choose to export the template for future use:



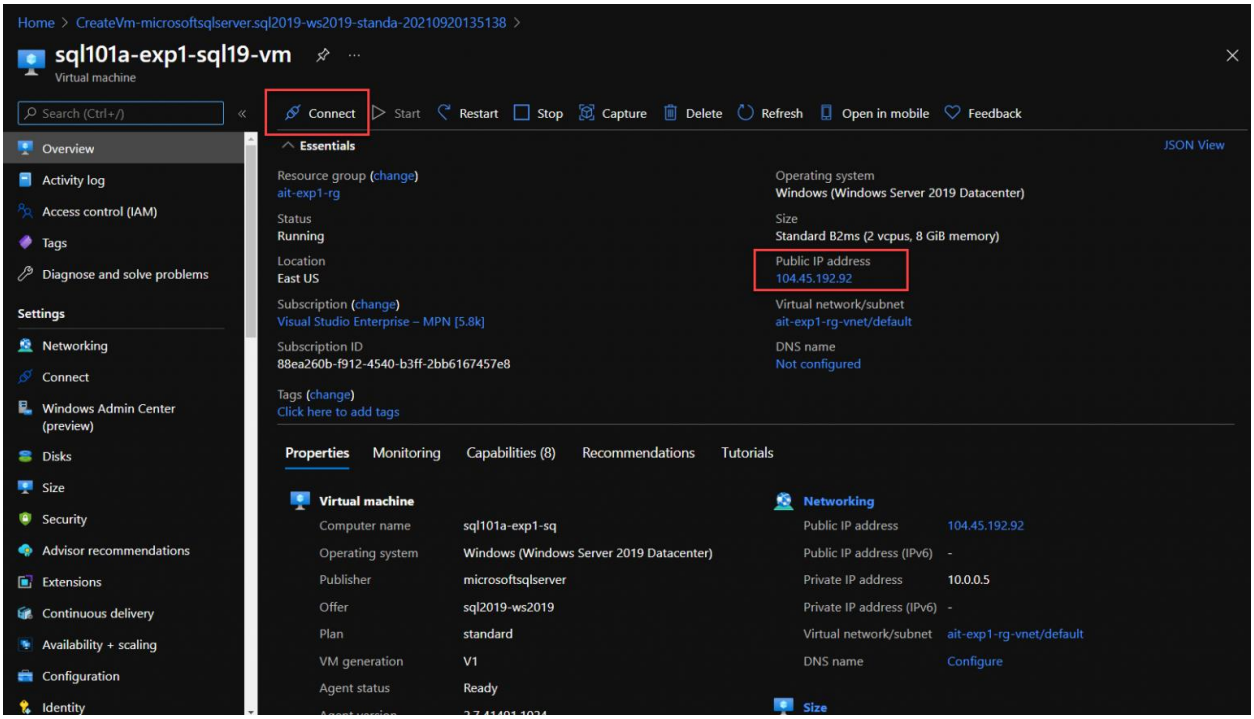
When the VM finishes creating, we can enter. We suggest you move forward.

Login to our Windows VM with SQL Server

Once the VM has been generated, we can log in via RDP. In the case of a VM with Linux, we can use SSH.

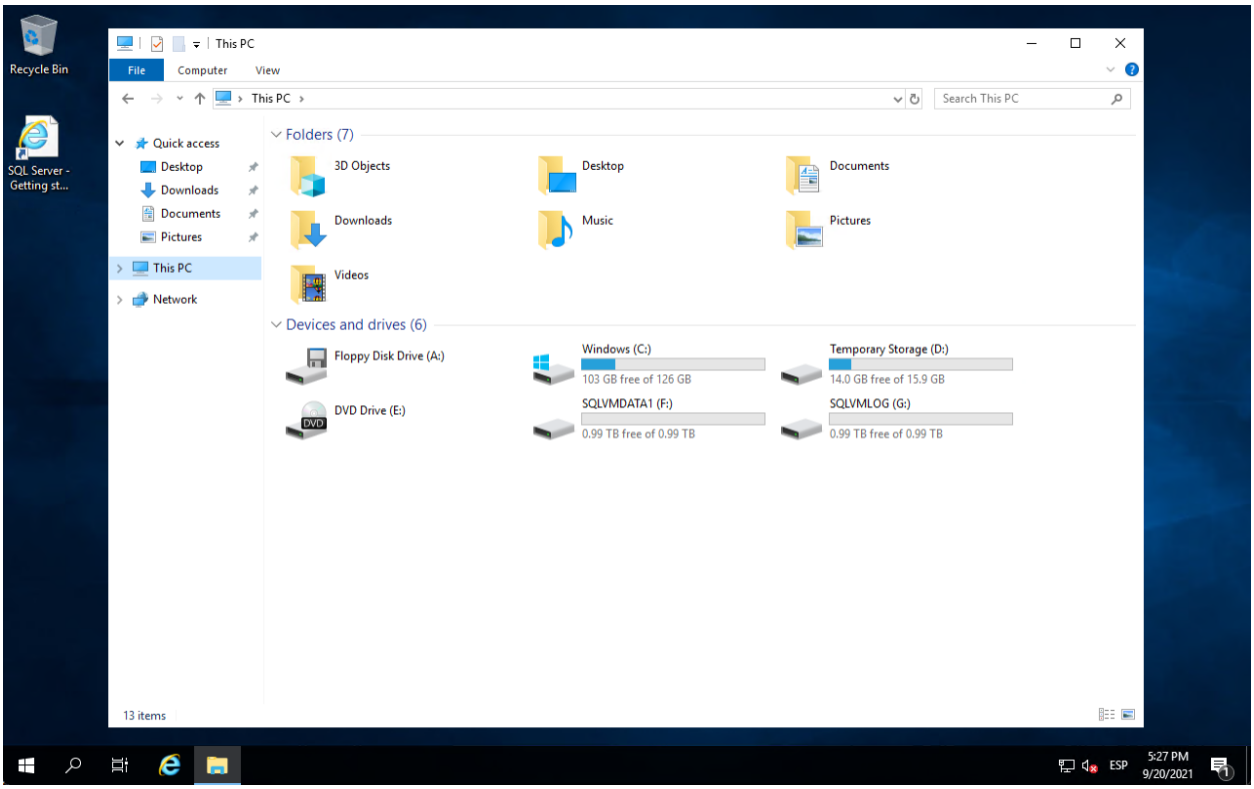
Connect to the VM via RDP

Since the goal of the course and this experience is to create a simple SQL Server VM, we have done basic configurations on our computer. To enter via RDP we will have to use the Connect button or the Public IP address from an RDP client:

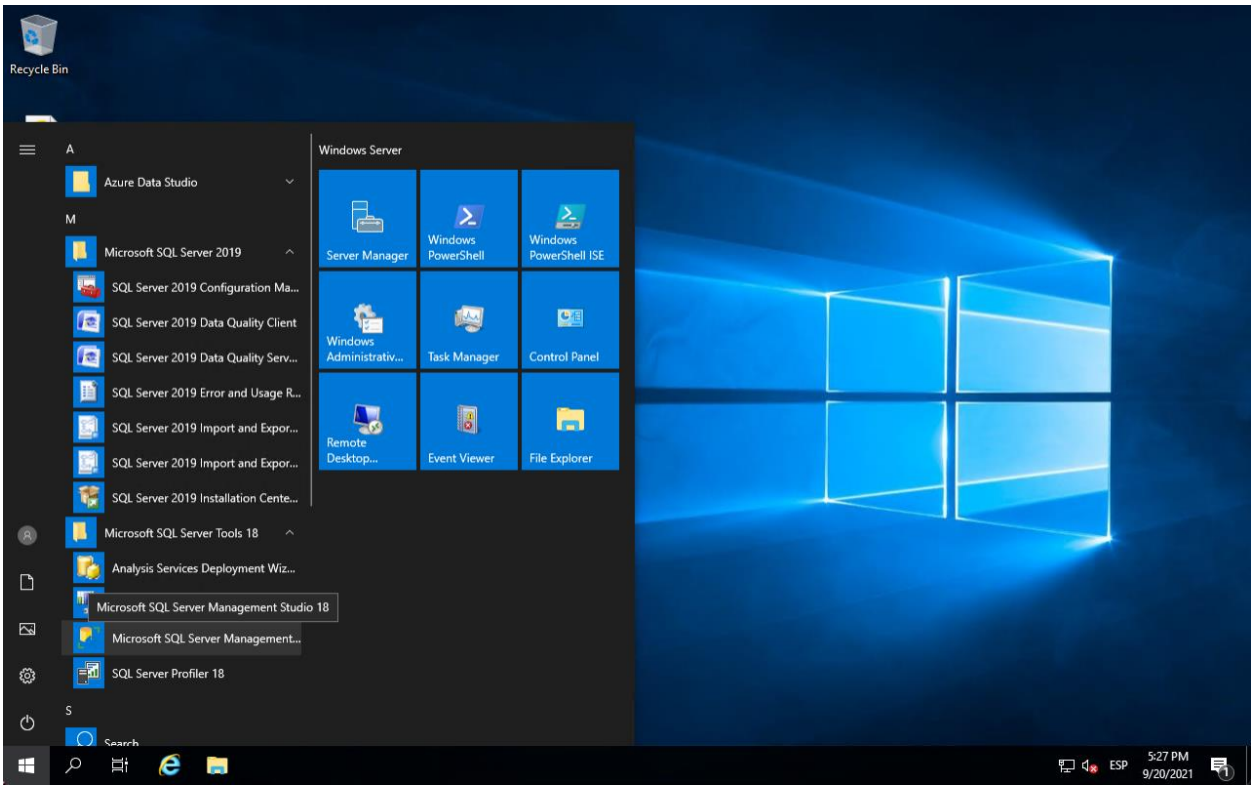


SQL Server VM Validations

We can see that the virtual disk configuration in the VM is as chosen in the configuration:

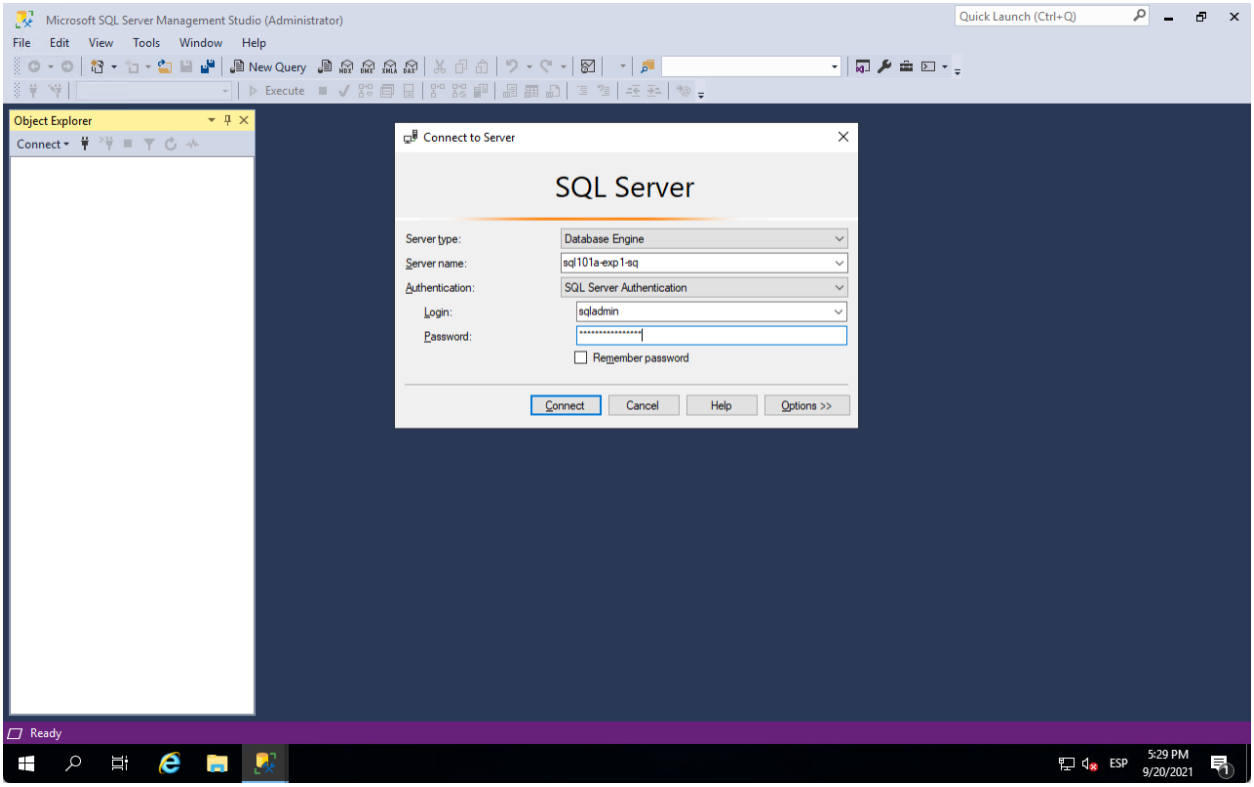


On the other hand, we can validate that management tools such as SQL Management Studio have been installed:

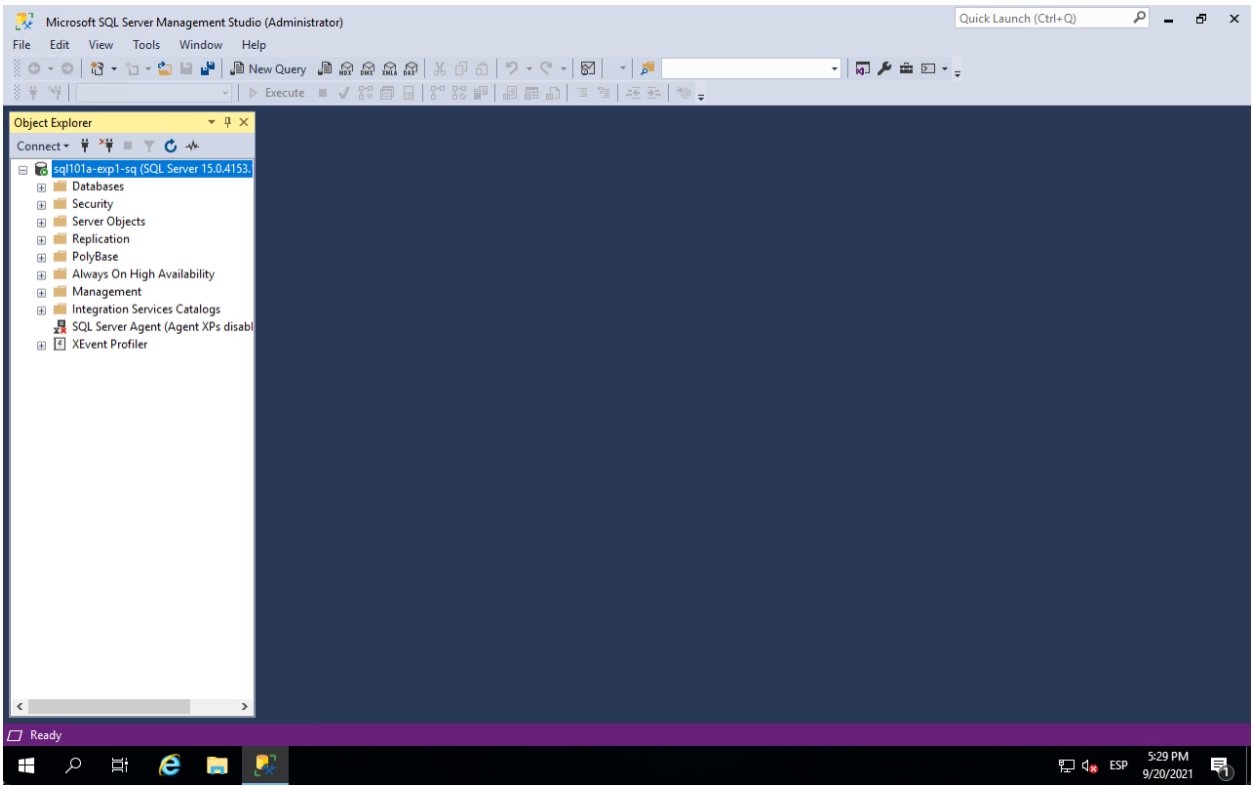


Connection to local SQL Management Studio

If we enter the SQL Management Studio we can connect with our local engine. For this we will use the credentials created during the creation of the VM, in the SQL Server settings tab:



If all goes well, we have been able to connect to the local SQL Server:



You will also be able to use Azure Data Studio to connect. Check with your instructor.

Exercise 2: Azure SQL Database (part 1)

Introduction to the exercise

In this section we will do the second experience together. This experience requires you to have a Microsoft Azure subscription set up and available and consumes any credits you have on it. Please review this topic to avoid charges on subscriptions where you don't want them to exist.

In this experience we will:

- Learn more about Azure SQL Database.
- Create a SQL Database Server.
- Configure Networks and Firewall of the SQL Database Server.
- Connect to SQL Database Server from administrative tools (such as SQL Management Studio and Azure Data Studio).
- Create single database type SQL Databases, DTU-based as well as vCPU-based.

This experience is going to be intense! Let's keep going!

About Azure SQL Database

Since 2008, when Windows Azure was announced, SQL on Azure was one of its five main components. From the beginning, SQL has been a big part of Azure. SQL Azure was built to provide a cloud-hosted version of SQL Server. Later, Windows Azure was renamed to Microsoft Azure, SQL Azure was renamed to Azure SQL, and both have increased services ever since, very noticeably.

SQL Database is an Azure SQL PaaS deployment option that abstracts both the operating system and the instance of SQL Server from users.

We are going to explore some points about Deployment Models, Purchasing and Licensing Models.

Deployment Models in Azure SQL Database

In Azure SQL Database there are two resource types or deployment models: single database and elastic pool. Let's briefly review each one:

- A single database represents a fully managed isolated database. You can use this option if you have modern cloud applications and microservices that need a single, trusted source of data. A single database is similar to a standalone database in the SQL Server database engine.
- The elastic pool is a collection of single databases with a shared set of resources, such as CPU or memory. Single databases can be moved in and out of an elastic pool.

In this section we will concentrate on single databases.

Azure SQL Database purchasing models

Azure SQL Database allow you to easily purchase a fully managed platform-as-a-service (PaaS) database engine that meets your performance and cost needs. Depending on the deployment model you've chosen for Azure SQL Database, you can select the purchasing model that best suits your needs:

- Virtual core based purchasing model (recommended). This purchasing model allows you to choose between a provisioned compute tier and a serverless compute tier. With the provisioned compute tier, you choose the exact amount of compute resources that are always provisioned for the workload. With the serverless compute tier, you must specify autoscaling of compute resources using a range of configurable computes. With this compute level, you can also automatically pause and resume the database based on

workload activity. The unit price of a vCore per unit of time is lower at the provisioned compute tier than at the serverless compute tier.

- Purchasing model based on database transaction unit (DTU). This purchasing model provides bundled and balanced storage and compute packages for typical workloads.

Azure SQL Database Hybrid Benefit

At the provisioned compute level of the vCore-based purchasing model, you can exchange your existing licenses for discounted rates on Azure SQL Database and Azure SQL Managed Instance through the Azure Hybrid Benefit. This Azure benefit allows you to save up to 30% or even more on SQL Database and SQL Managed Instance when you use SQL Server licenses with Software Assurance.

Creating a SQL Database Server

The first step to get started with Azure SQL Database is to create a SQL Database Server.

What is a SQL Database Server?

A server is a logical construct that acts as the central administrative point for a collection of databases. At the server level, you can:

- Manage logins, firewall rules,
- audit rules,
- Threat detection policies and
- Automatic failover groups.

The server must exist before you can create a database, and all databases managed by a single server are created in the same region as that server. The SQL Database Server is FREE: there is no cost or cost to create it. The databases that we generate dependent on it, on the other hand, will have a cost.

A server is different than an instance of SQL Server, which you may be familiar with in the world of locales. In particular, there is no warranty regarding the location of the databases or the data storage database with respect to the server that manages them.

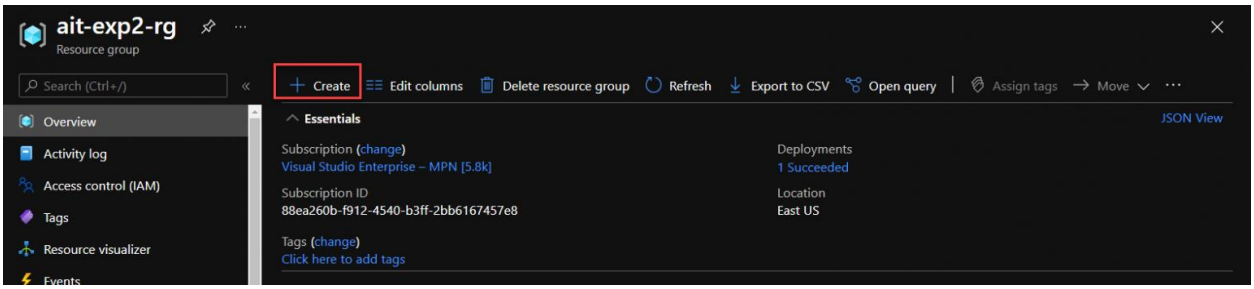
In this experience and the ones that follow, you will learn many things about a SQL Database Server that will surely surprise you.

Resource Group Creation

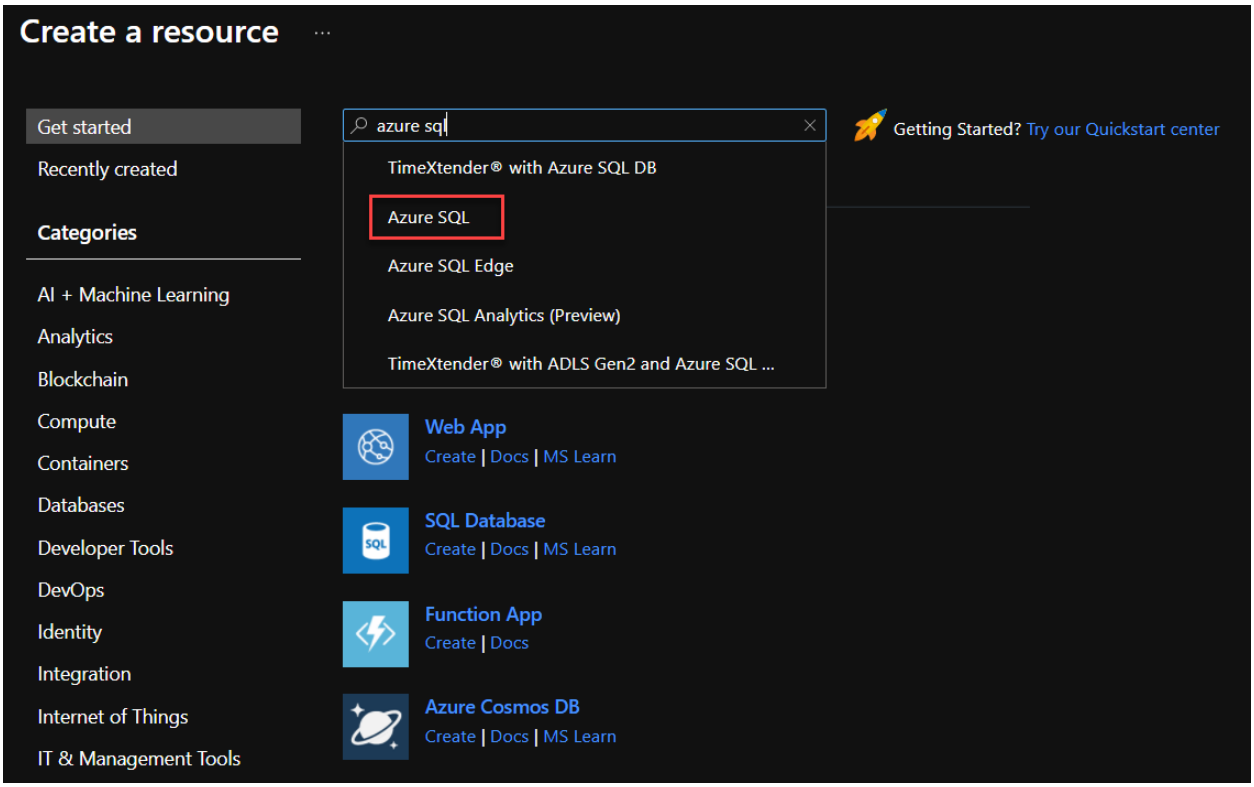
For this experiment we are going to work on the group of resources that you will have to create with the name "**sql101a-exp2-rg**".

Creating a SQL Database Server

We can create a SQL Database Server directly from the Azure Portal. We'll move to the suggested resource group for this experience and create a new resource:

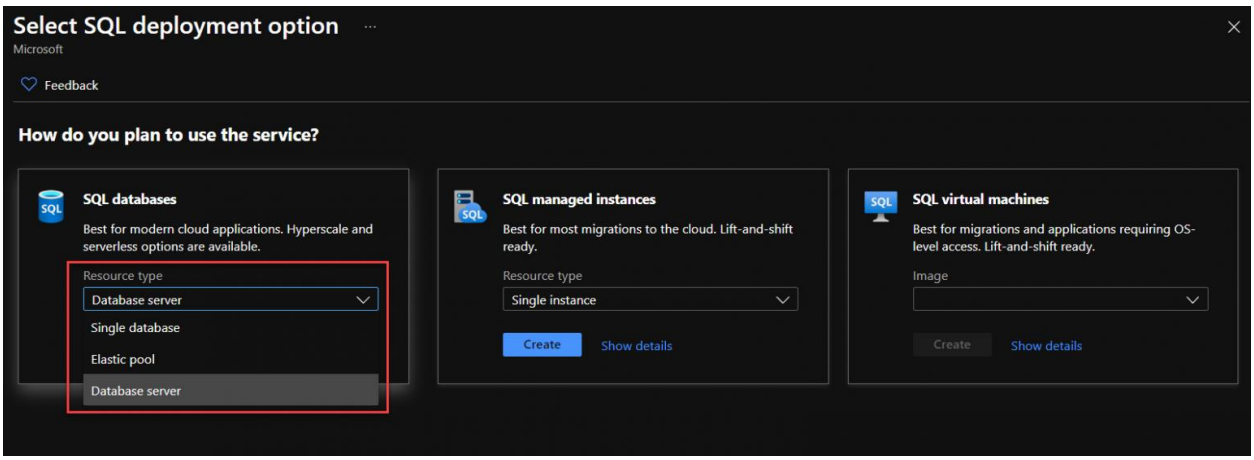


There we will choose “Azure SQL” in the search bar:



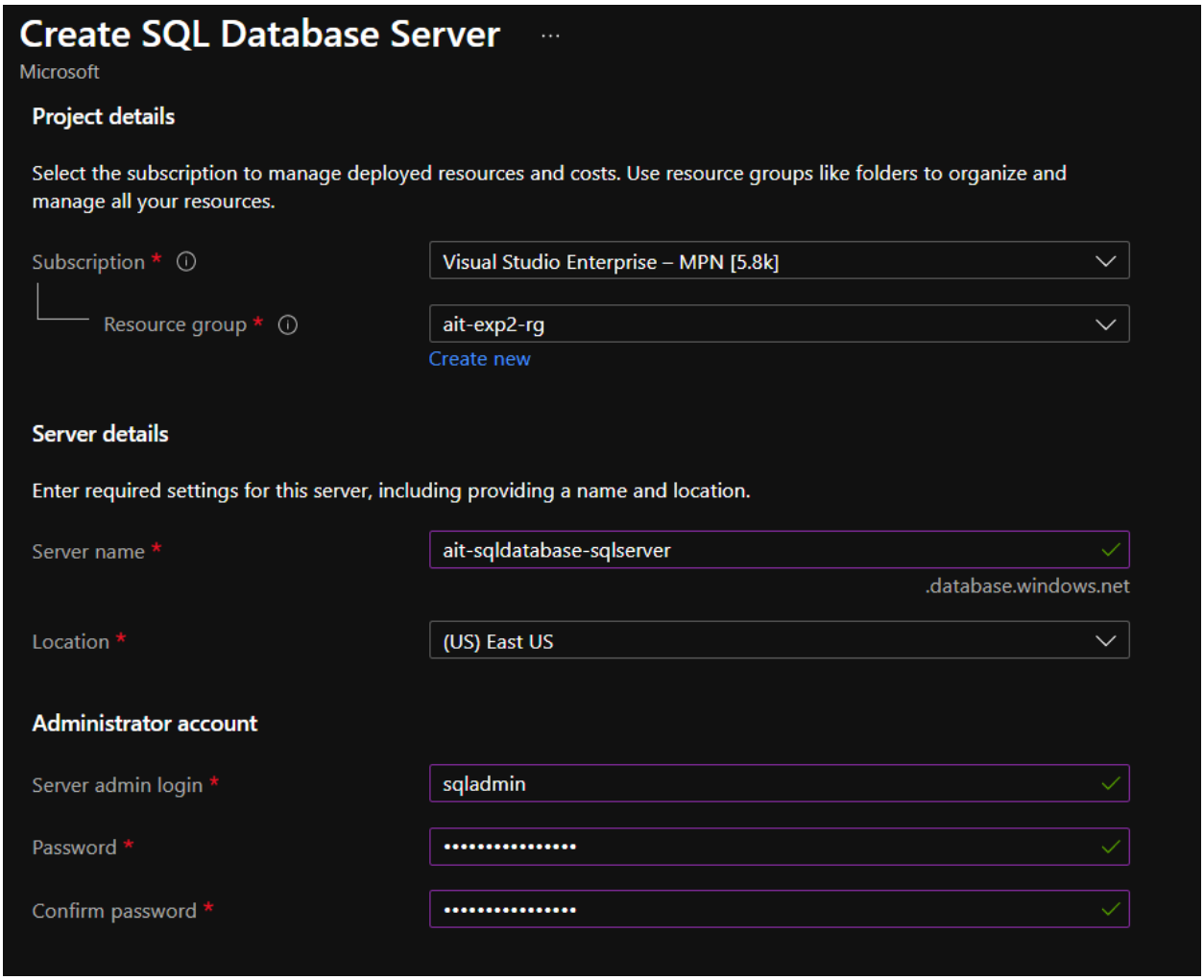
We will enter the screen with the Azure SQL deployment options. This screen should be familiar to us, since it is the same one that we have seen in the theoretical introduction.

We are going to select SQL Databases and, in the resource type, “Database server”:

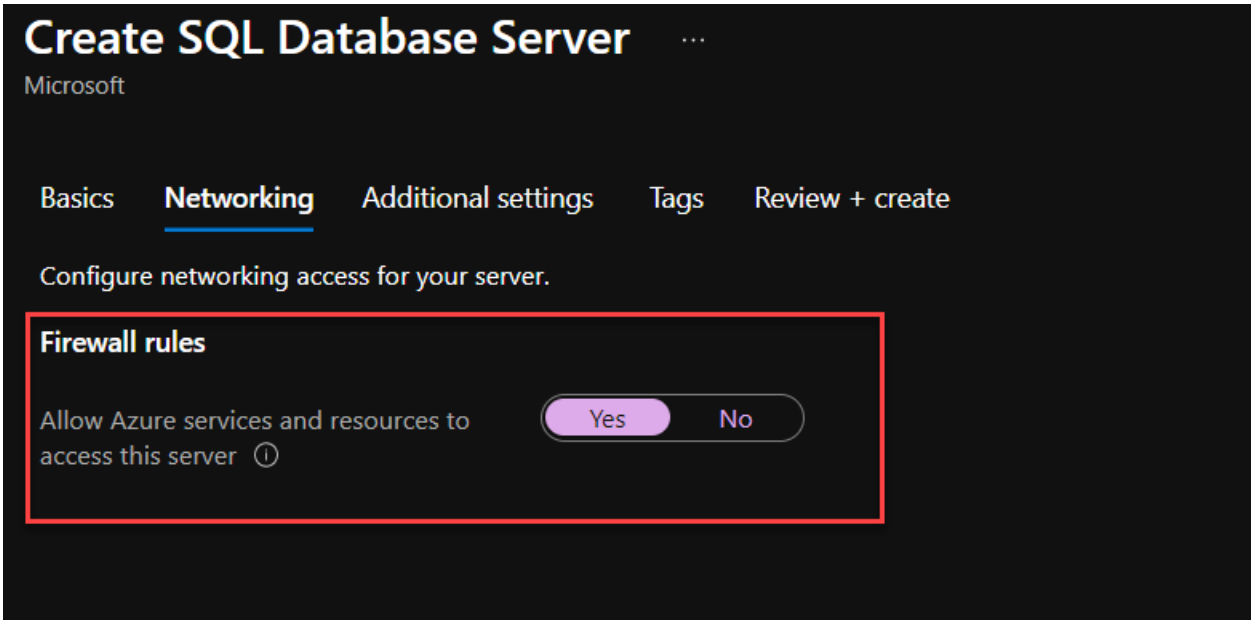


Now it's time to start entering data to create our SQL Database Server. We suggest you follow the following guidelines:

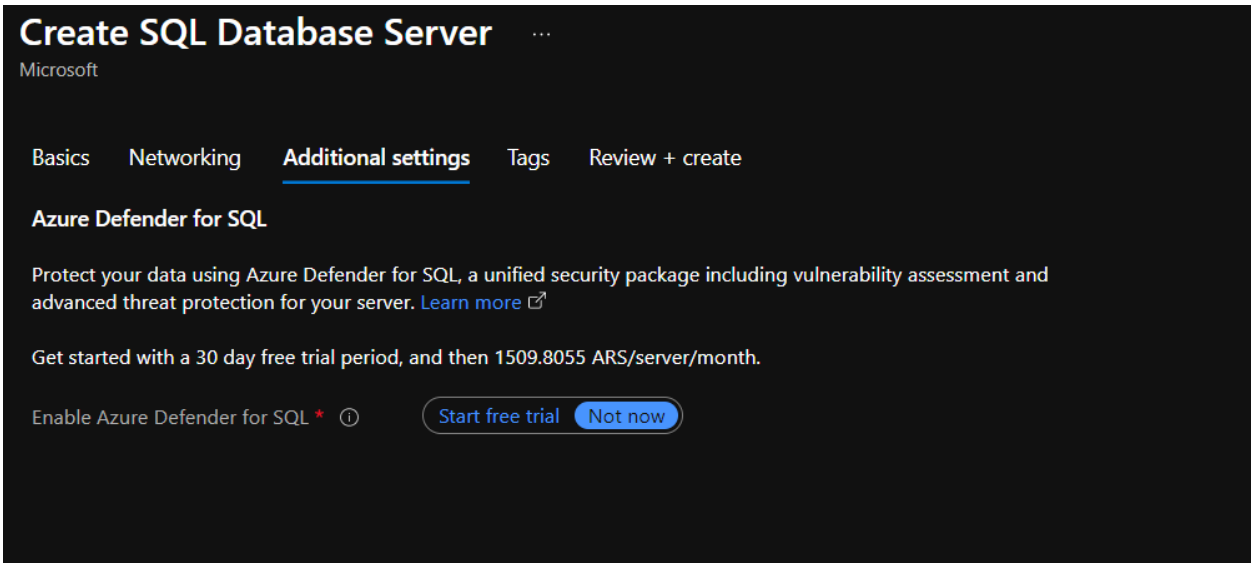
- Server name: ait-sqldatabase-<string>-sqlserver, where <yourstring> must be replaced with your own unique text and numbers (since this server name must be unique across all of Azure).
- Location: eastus
- Server admin login and password: the one you choose, and as a suggestion enter "sqladmin" in the login.



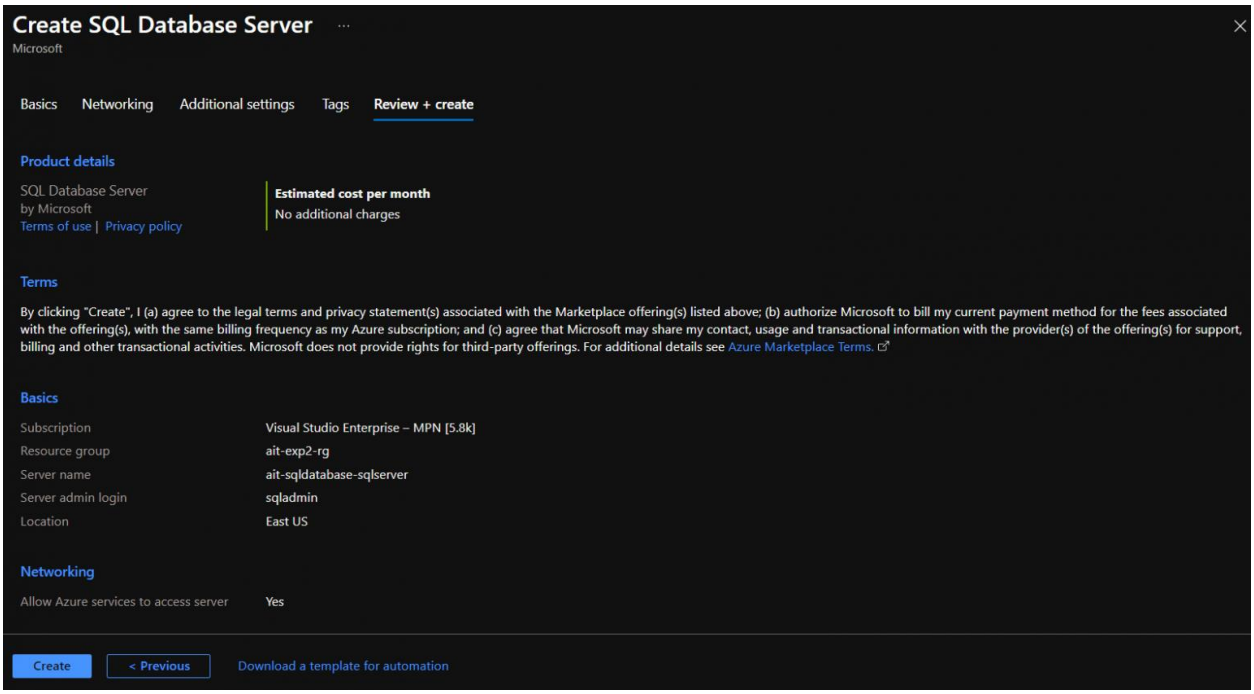
After completing the previous data, we will go on to configure the networking part. We will enable Azure services (VMs, Websites, etc) to connect to this SQL Database Server. This, later in the experience, we will modify:



In Additional settings we will have the possibility to configure Azure Defender. This time we will not:



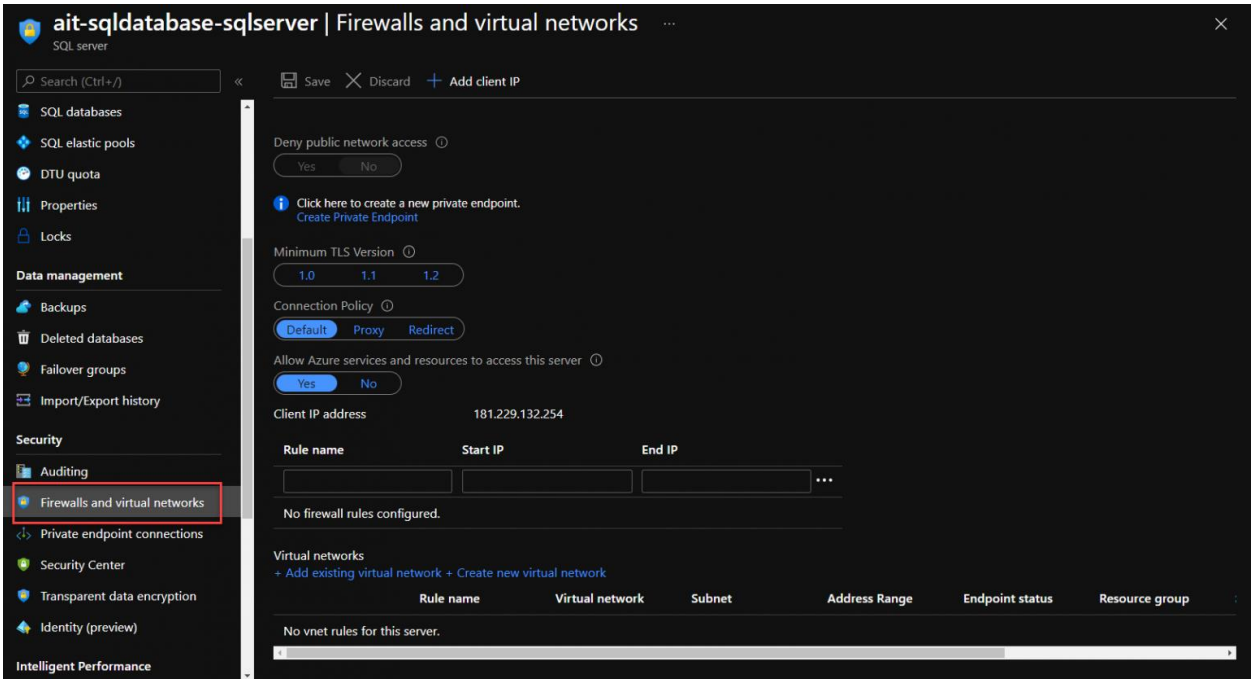
Finally, we are going to review the selected options and, when the validations have passed, we will create the SQL Database Server:



Ready! When you have finished generating it you will move on to the next lesson.

Network and Firewall Configuration

In order to connect to the SQL Database Server from management tools or from our apps, we must correctly configure the Networks and Firewall. This option is found within our server, as indicated on the screen:



In this lesson we will not carry out any configuration, we will only go through the options so that you know what each of them is for.

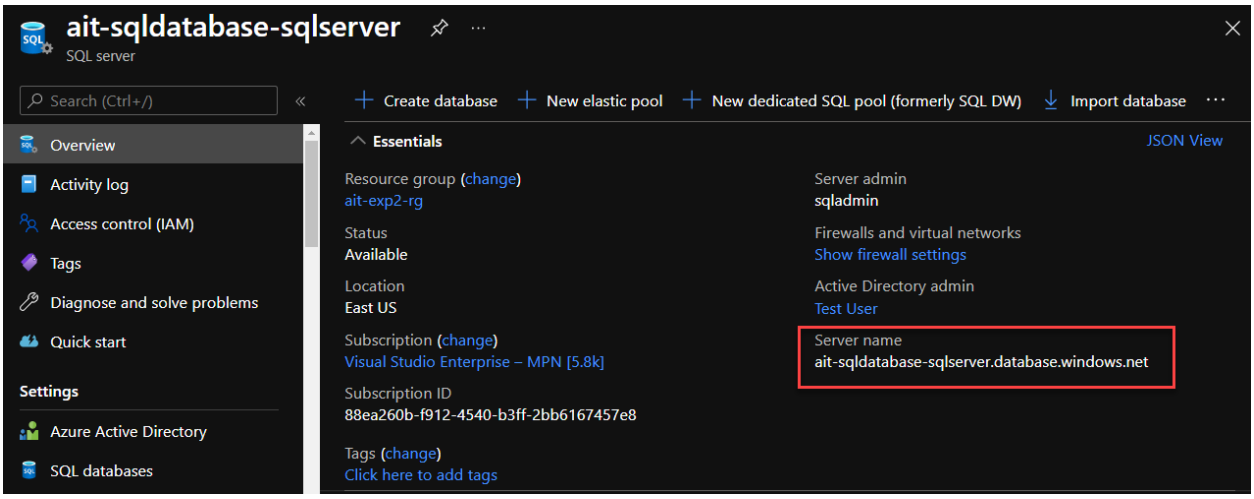
Connection from Administrative Tools

In this lesson we are going to validate that we can connect using SQL Management Studio and from Azure Data Studio from our Azure virtual machine. However, we are going to validate that we cannot do it from our desktop computer.

We will not generate databases yet, an action that we will leave for later in this experience.

Get the connection URL to the SQL Database Server

We are going to enter our created SQL Database Server resource, and in the Overview screen we will extract (copy) the Server name with the full connection URL:

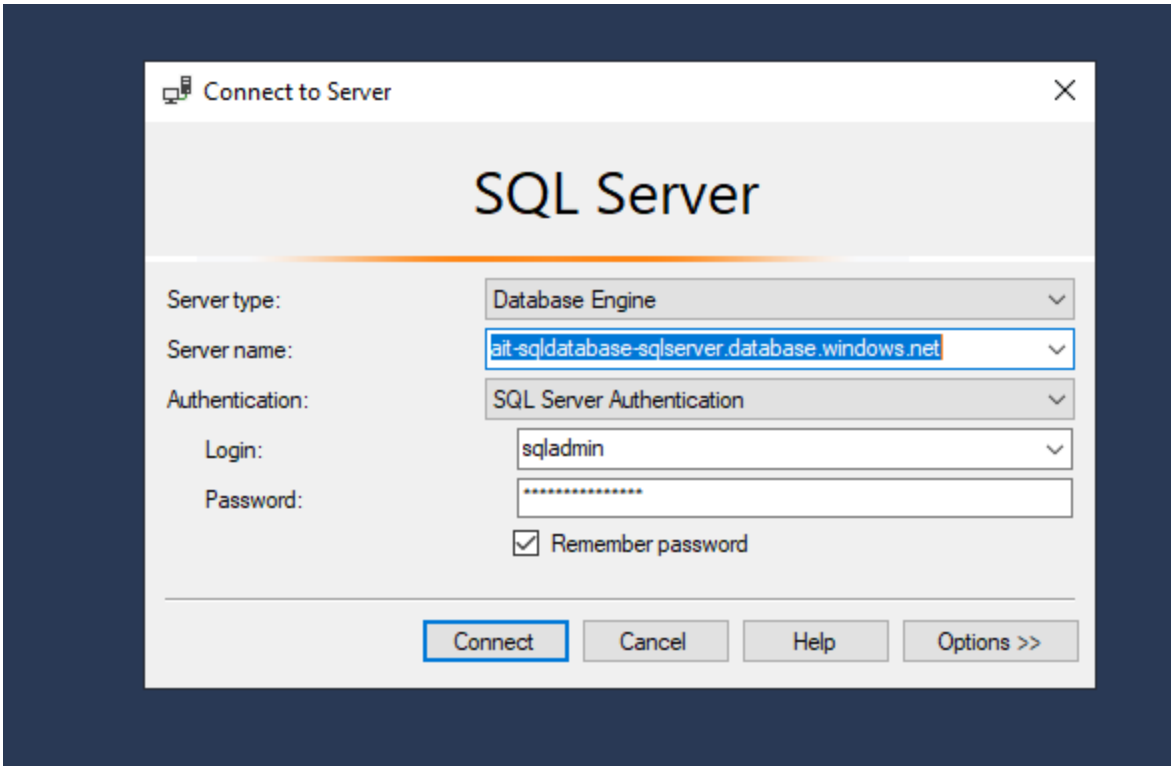


Connect from Azure using SQL Management Studio

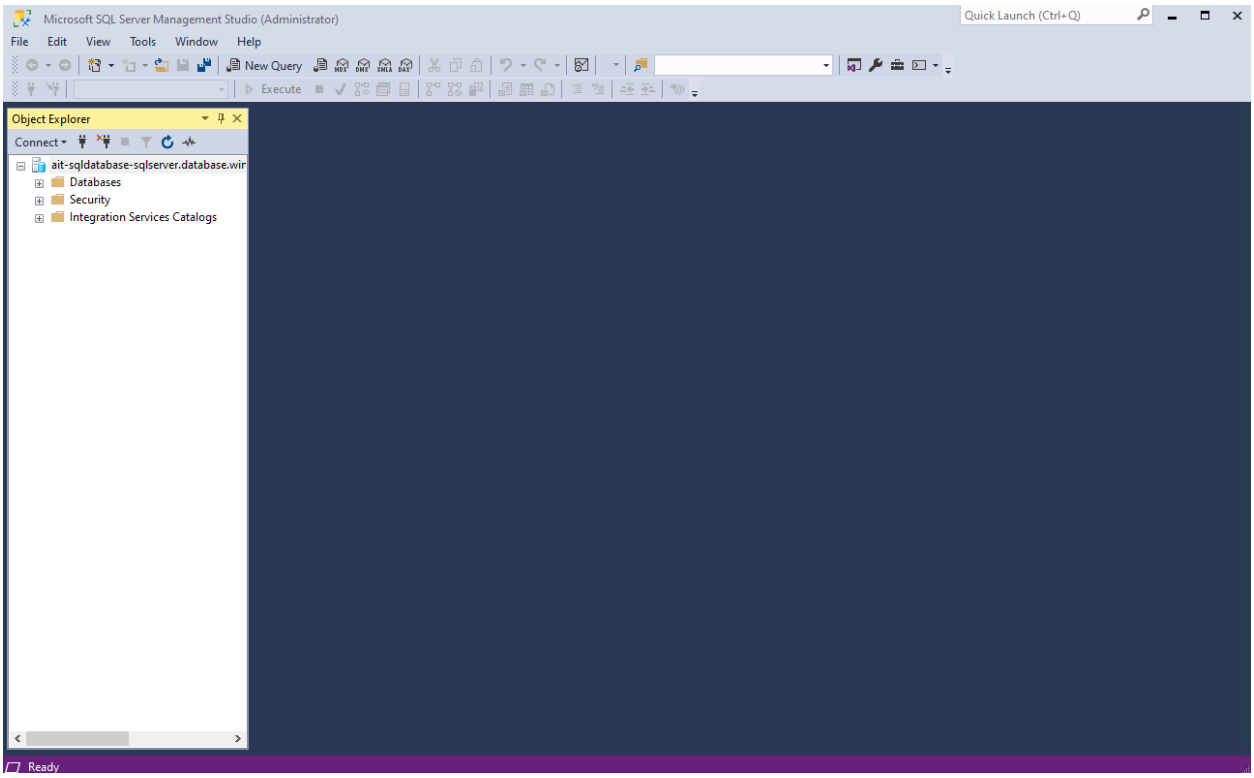
Since we have configured all the corresponding network and firewall aspects, enabling connections from Azure to be established with the server we just generated, we are going to go to our virtual team (created in the first experience) and open SQL Management Studio .

We will enter:

- In Server name the full URL of the SQL Database Server,
- Under Authentication “SQL Server Authentication”.
- Login and Password that we have assigned to our SQL Database Server.



We will be able to verify that our SQL Management Studio connected without problems to the SQL Database Server:



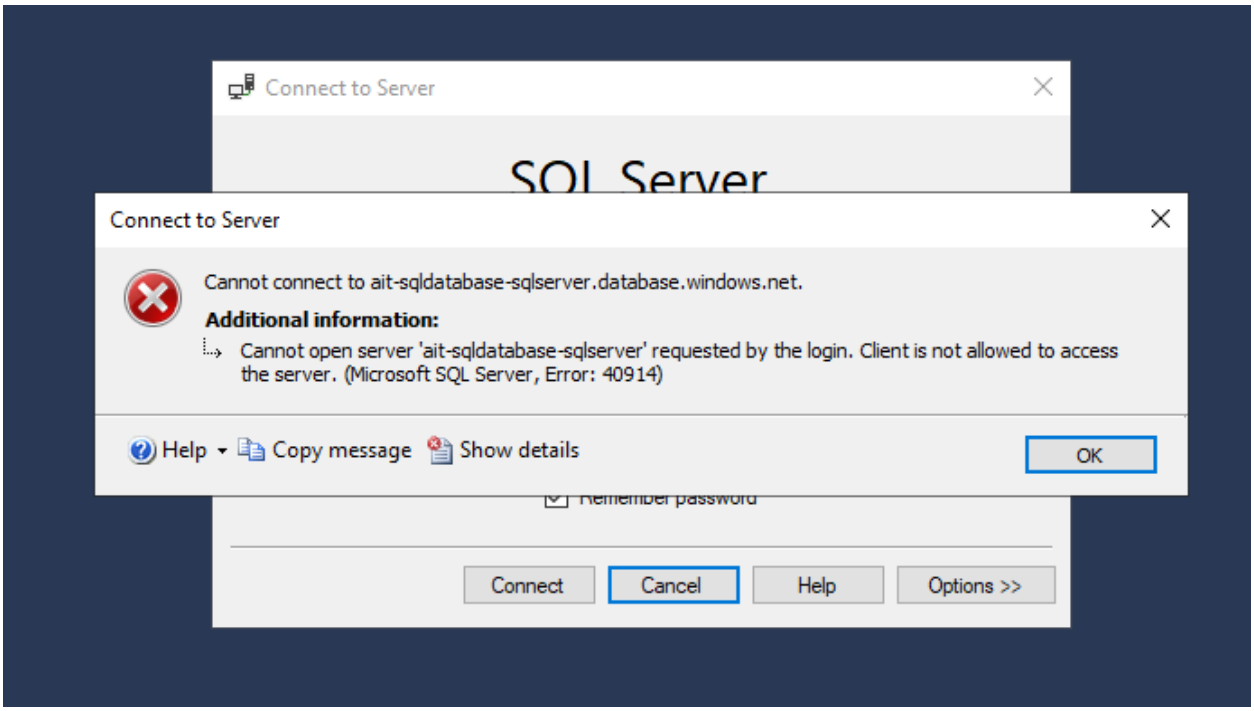
We will be able to verify that our SQL Management Studio connected without problems to the SQL Database Server:

Now... and what happens if I try the same connection from my house? Let's explore it.

Connect from another location using SQL Management Studio

To do this test we are going to need to have SQL Management Studio installed in our house, or at least in some computer :-). To download it, go to Experience 1 and use the same download URL.

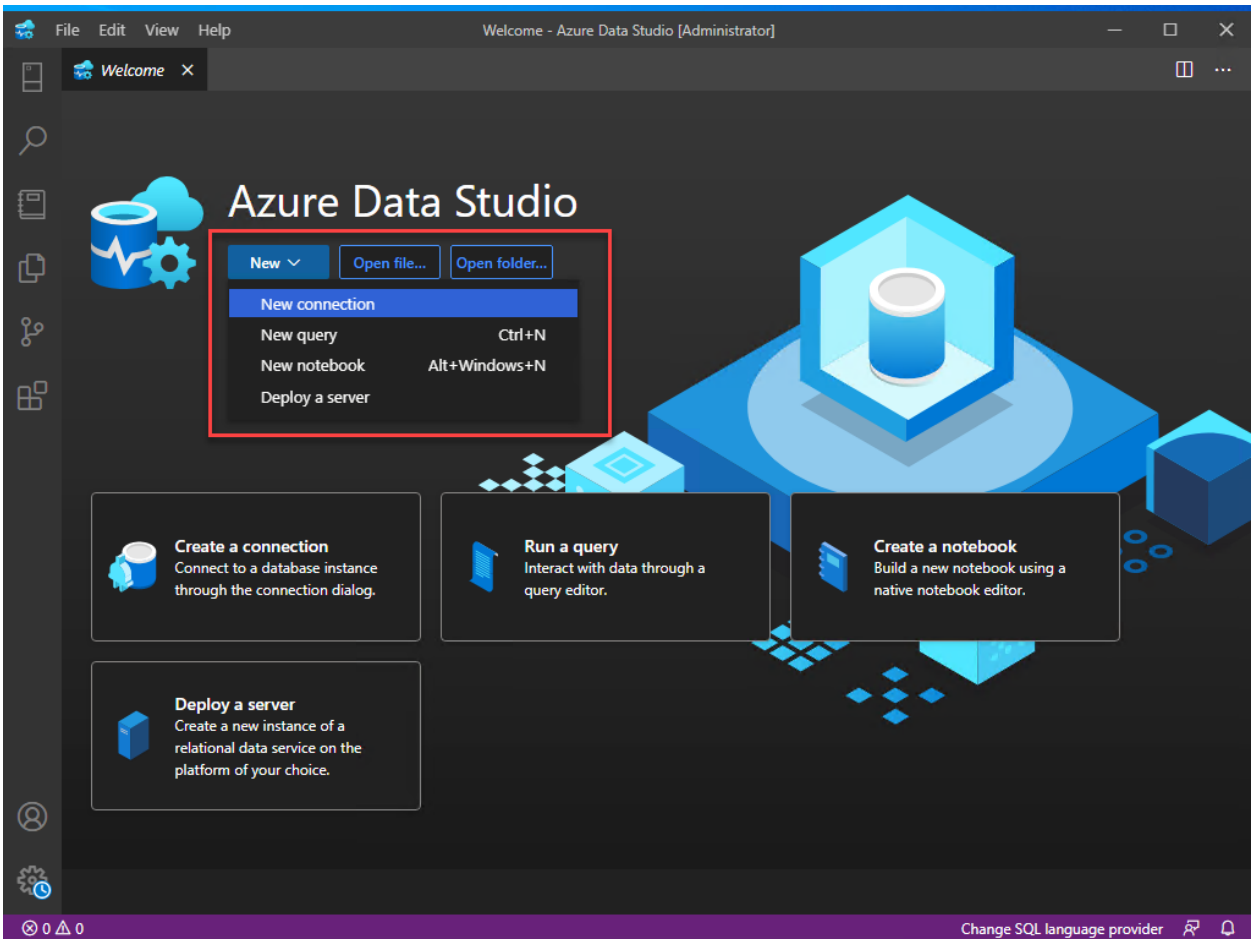
Entering the connection data, it will not be possible for us to connect since it will be denied with the error "Client is not allowed to access the server":



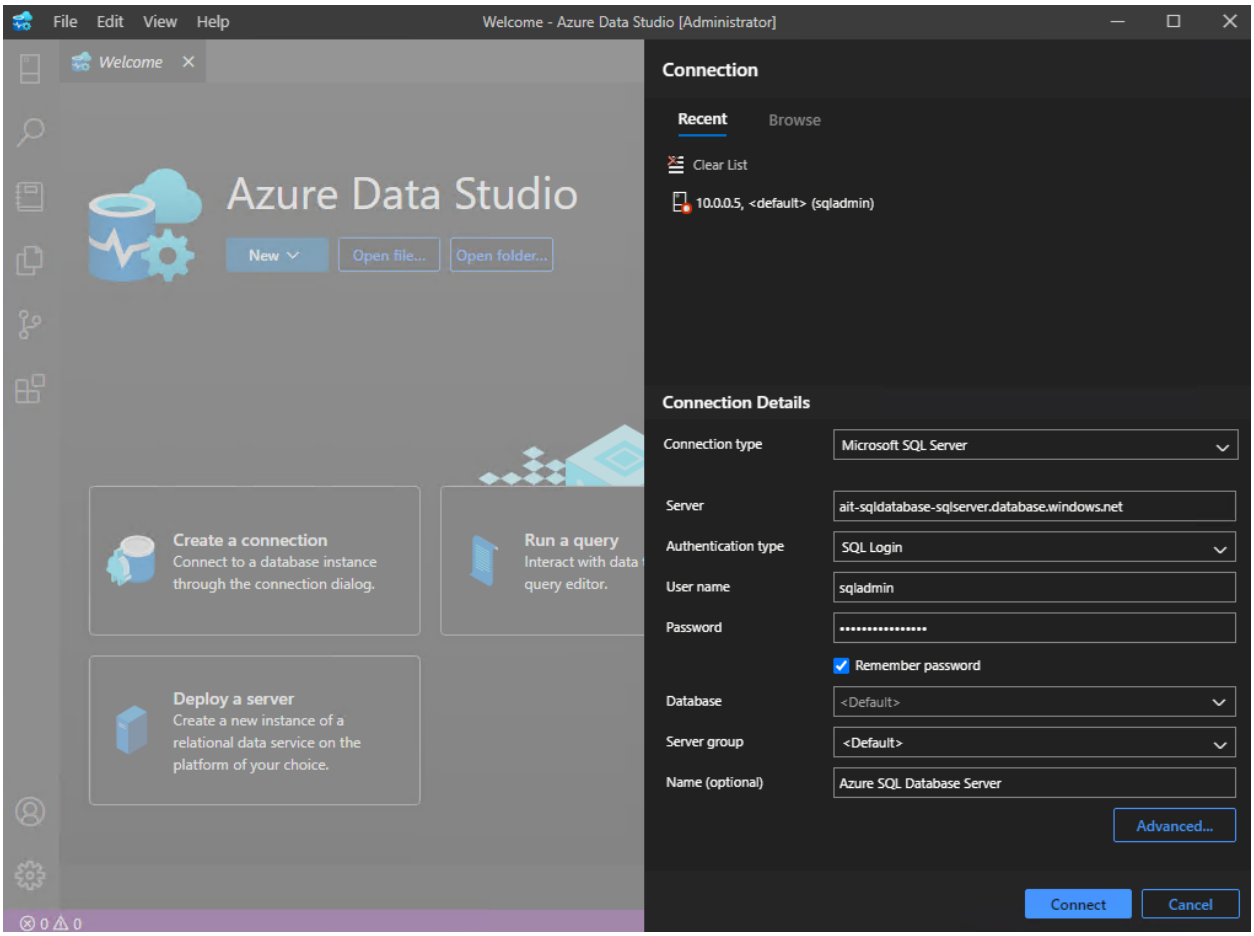
How do we go about solving it? Here's a hint: add your public (home) IP address in the SQL Database Server Firewall and try again.

Connect from Azure using Azure Data Studio

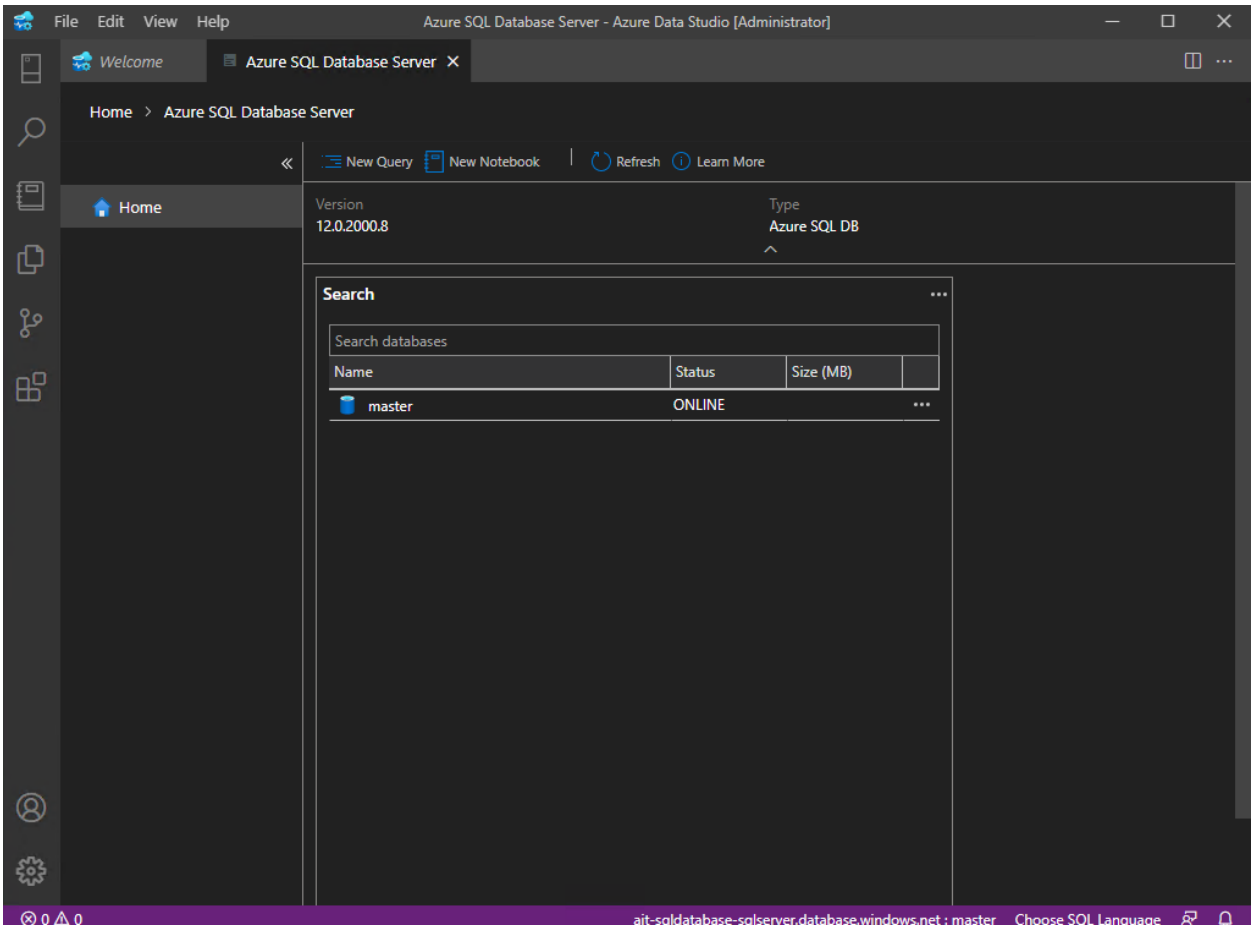
In the same way that we have tried connecting to SQL Management Studio from the Azure VM that we created in experience 1, we are going to connect to Azure Data Studio:



We will enter the server data and credentials:



Ready! We are already connected, too, with this tool:



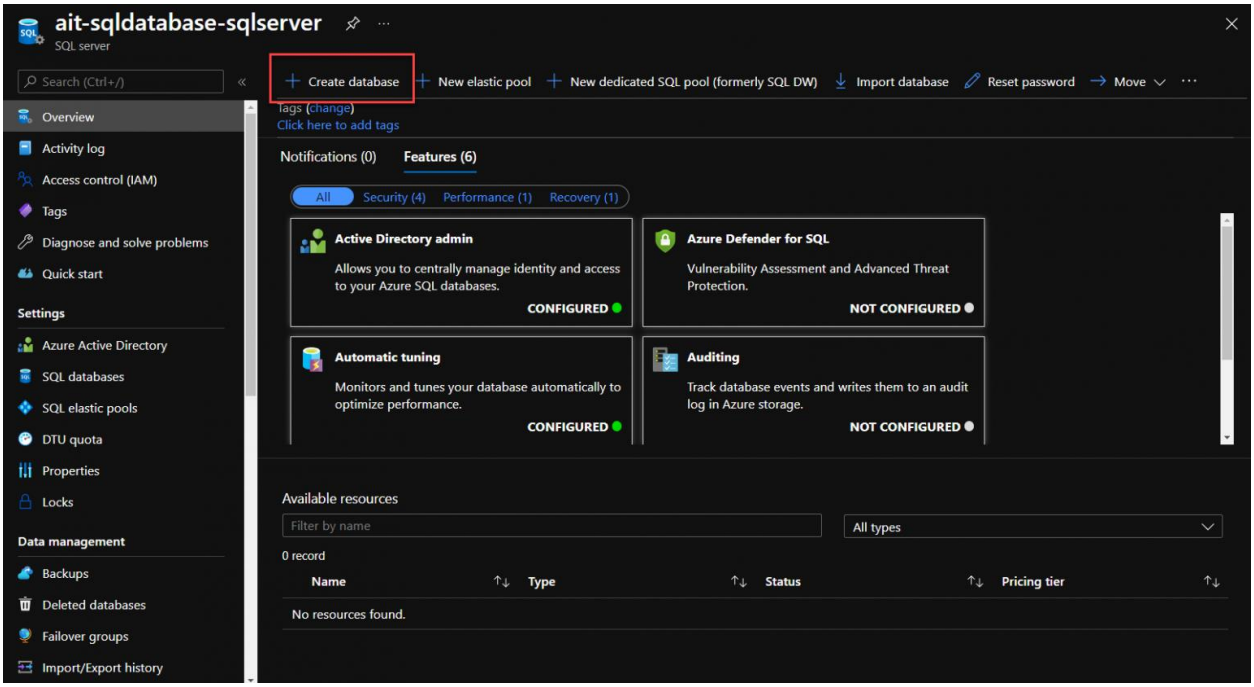
Let's move on to the next lesson!

Single Database Creation (DTU)

We are going to create our first unique database using the DTU-based purchasing model.

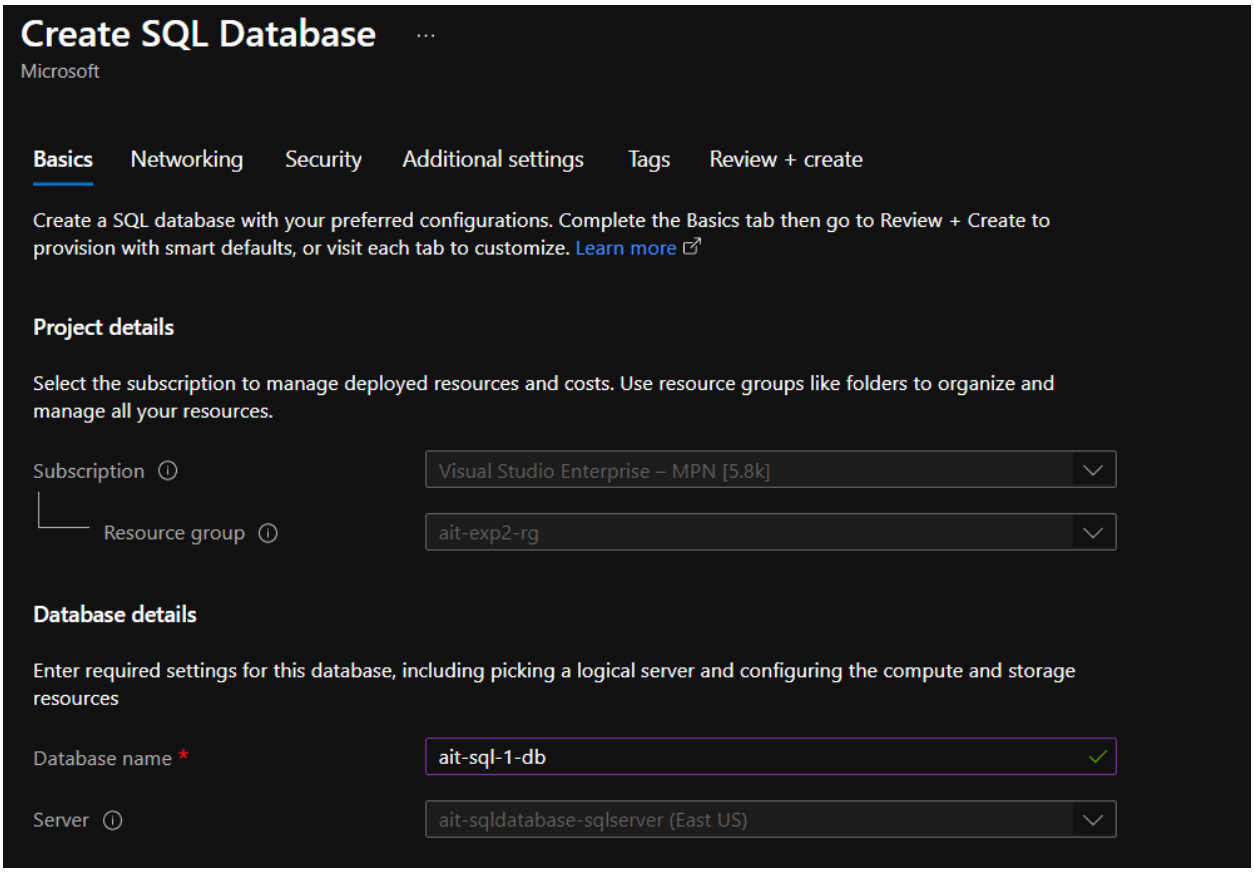
Unique database creation based on DTUs

From our Azure SQL Database Server, on the Overview screen, we are going to click on Create database:

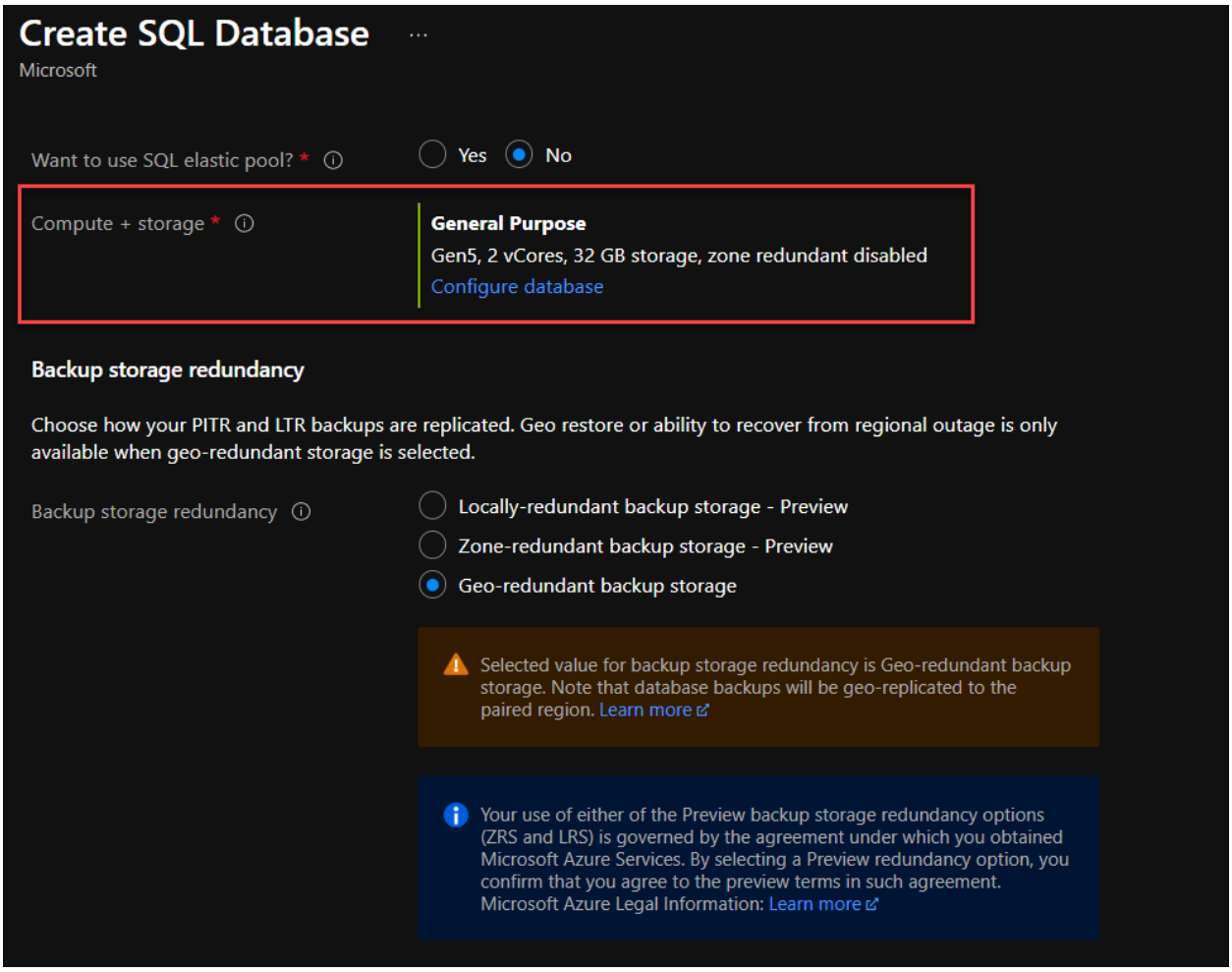


This will start the SQL Database creation wizard. In the same way, we could have generated a new resource in our Resource Group and searched for “Azure SQL”, and then chosen the “Single database” item in the “SQL database” option.

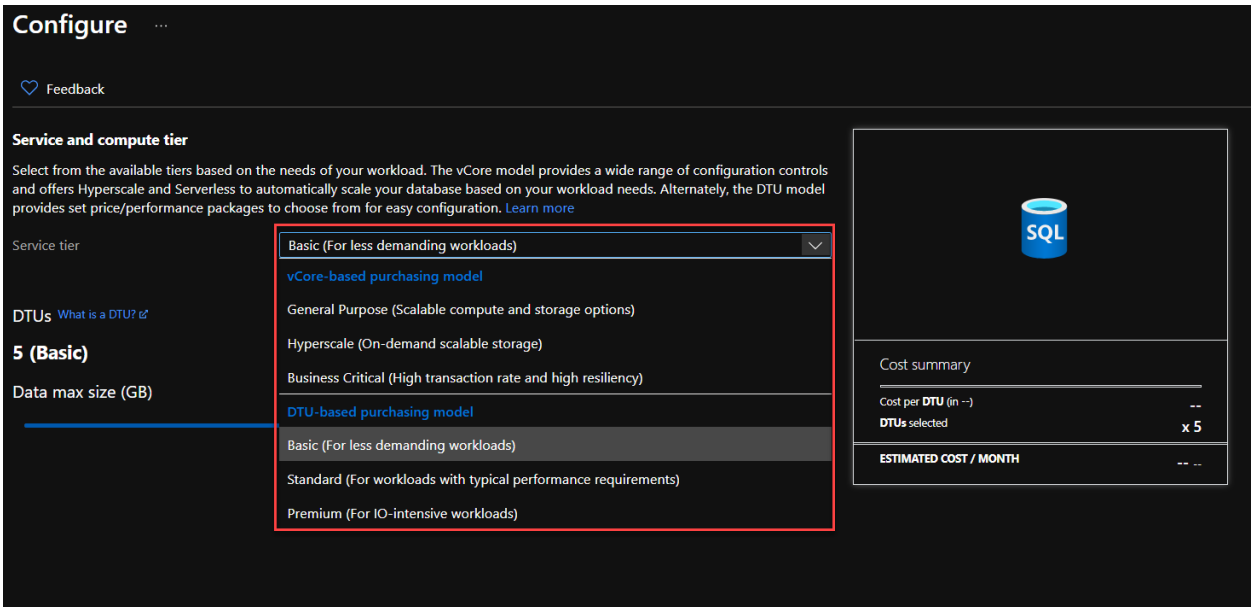
Note that we only need to fill in the name of the database. For our exercise we will enter ait-sql-1-db:



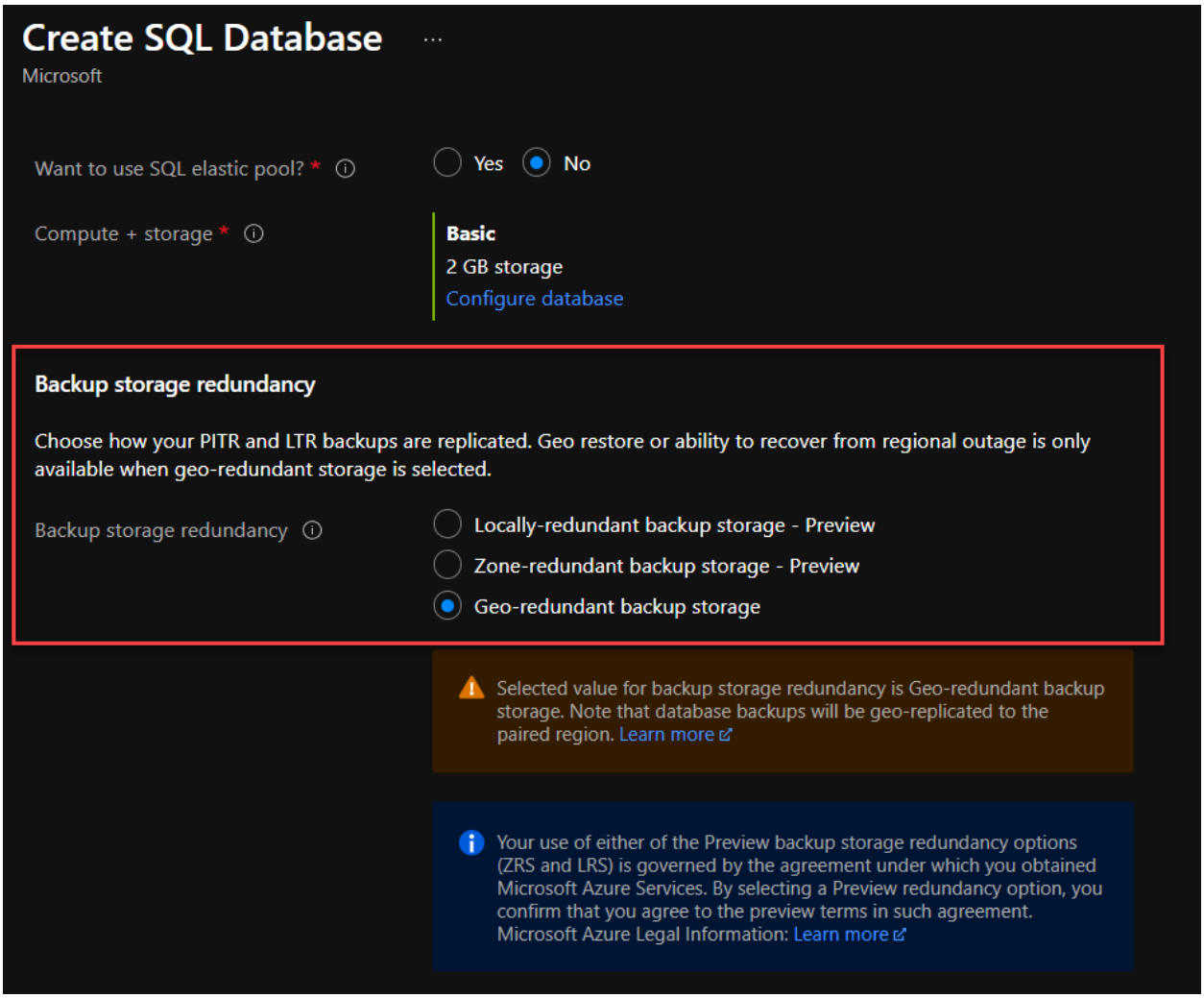
In the “Compute + storage” section we are going to choose the purchase model and the service level, by clicking on “Configure database”:



On the screen that is displayed we can see in "Service tier" the options based on vCore and DTU. Let's select the DTU-based option “Basic”:

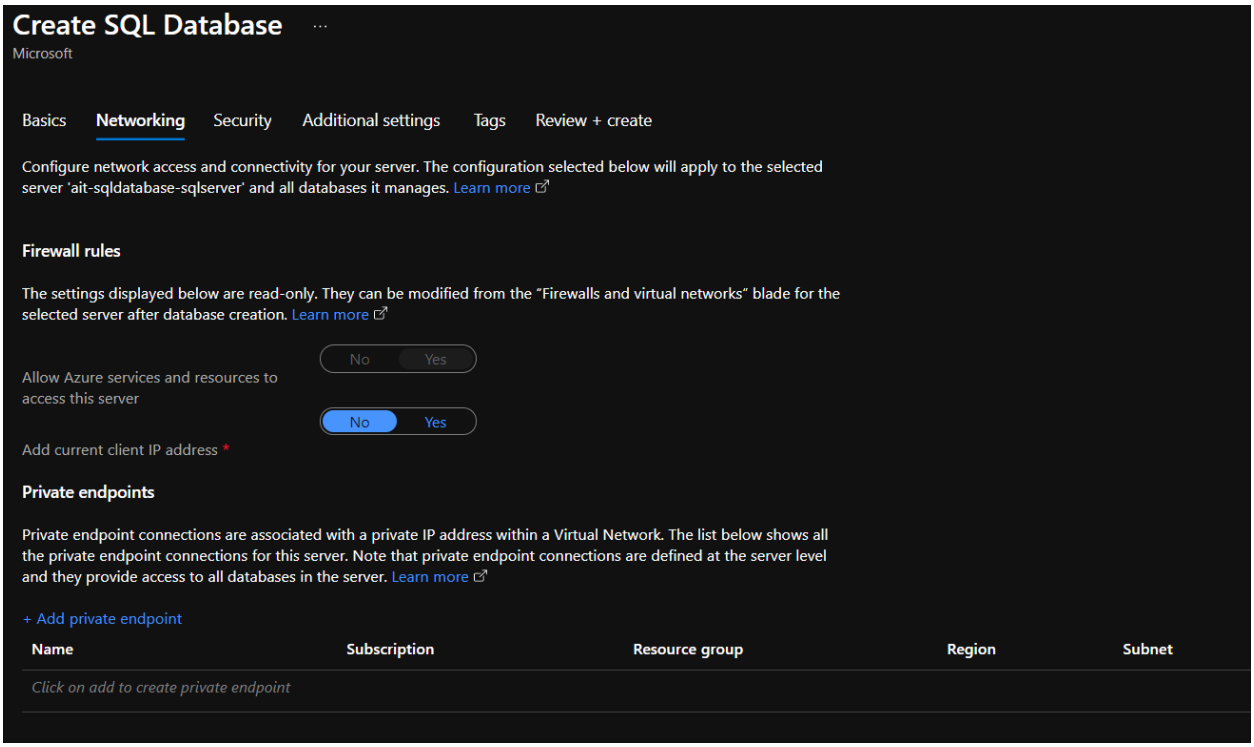


We can also configure aspects of backup copies and their redundancy, or leave the default value:

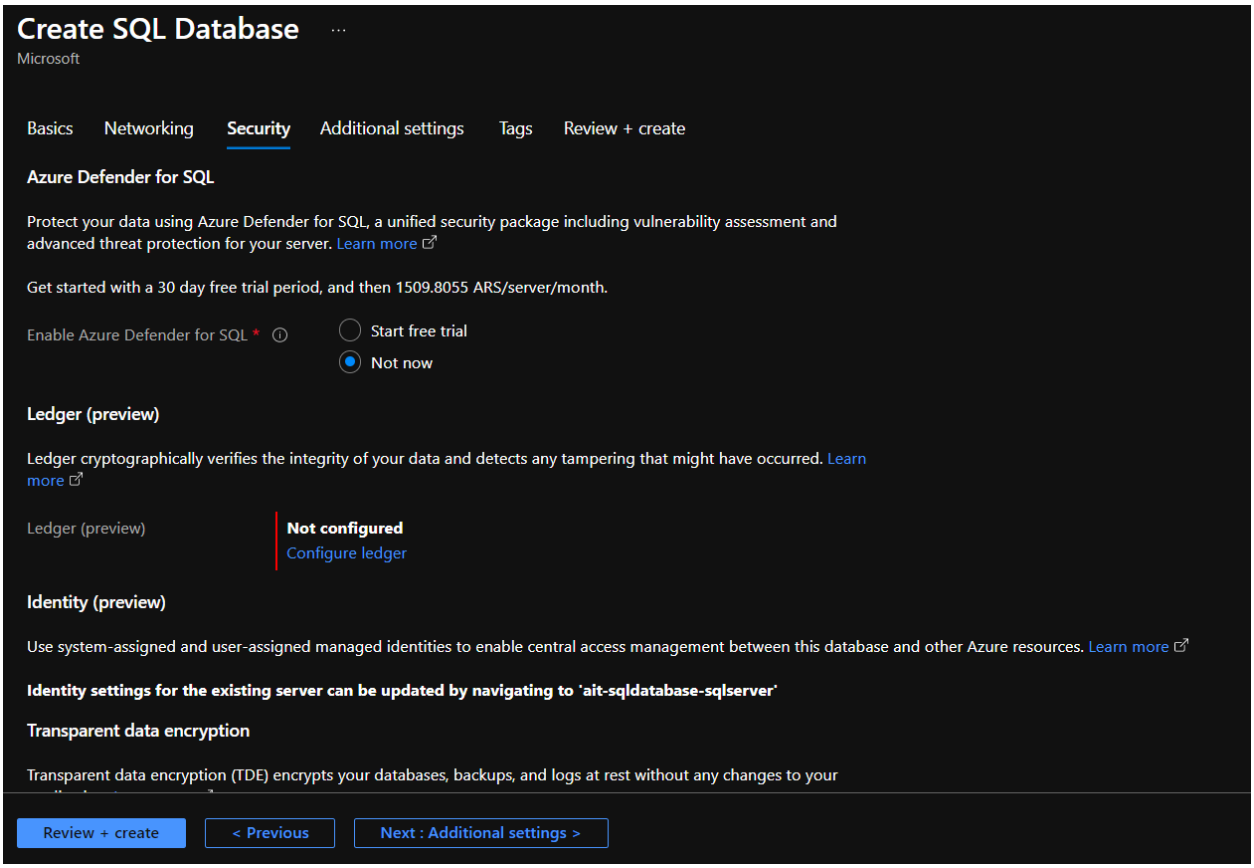


In the next step of the wizard, we will proceed to configure the network aspects. This configuration, in reality, corresponds to the SQL Database Server, although in the creation of the database it is presented to customize. Note that we cannot modify the Azure services option, but what we can do is enable (or not) a client IP (from which we are connecting) so that with any administration tool we can connect to the server and to the future database.

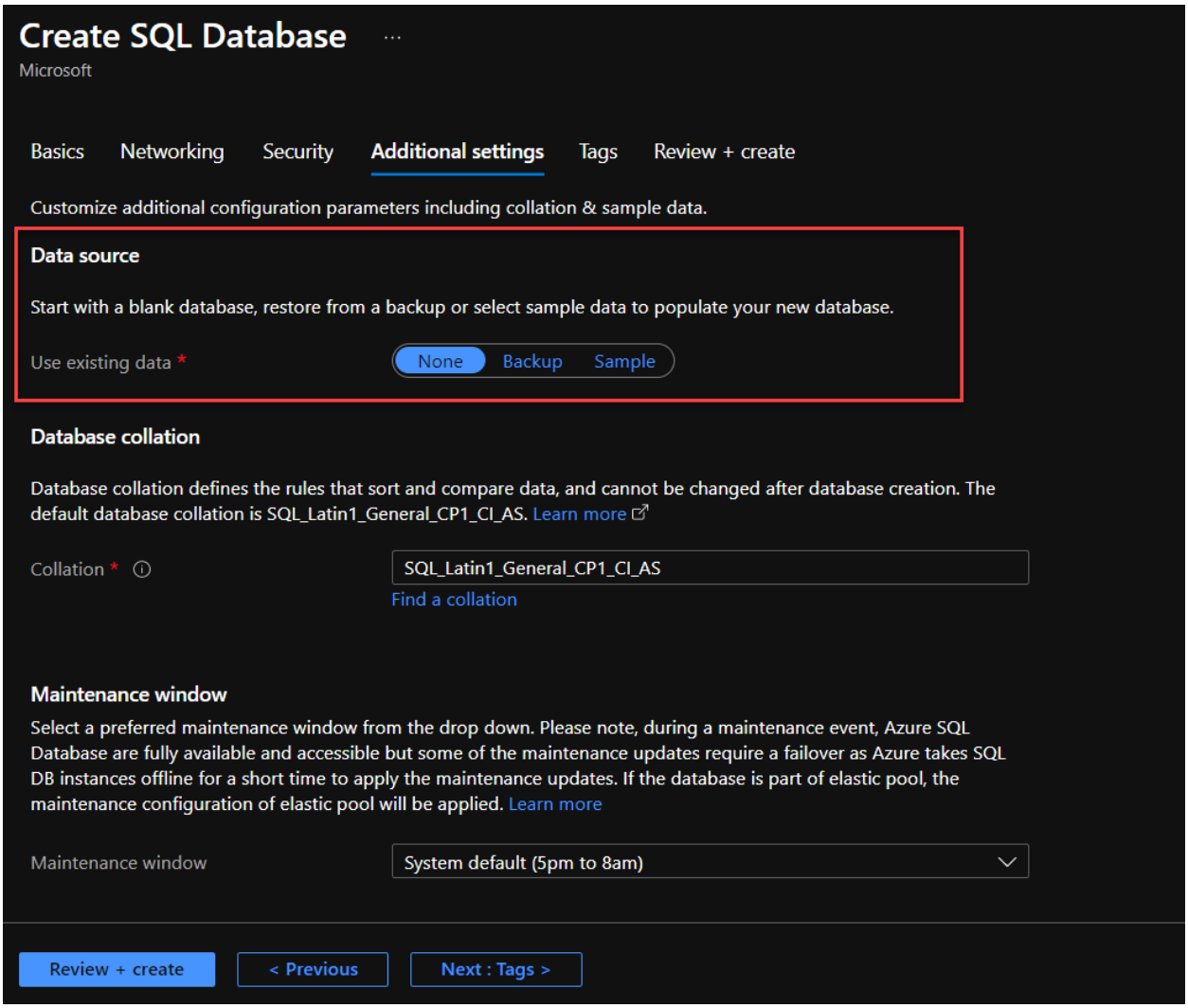
In our case we are going to set the option to “No”:



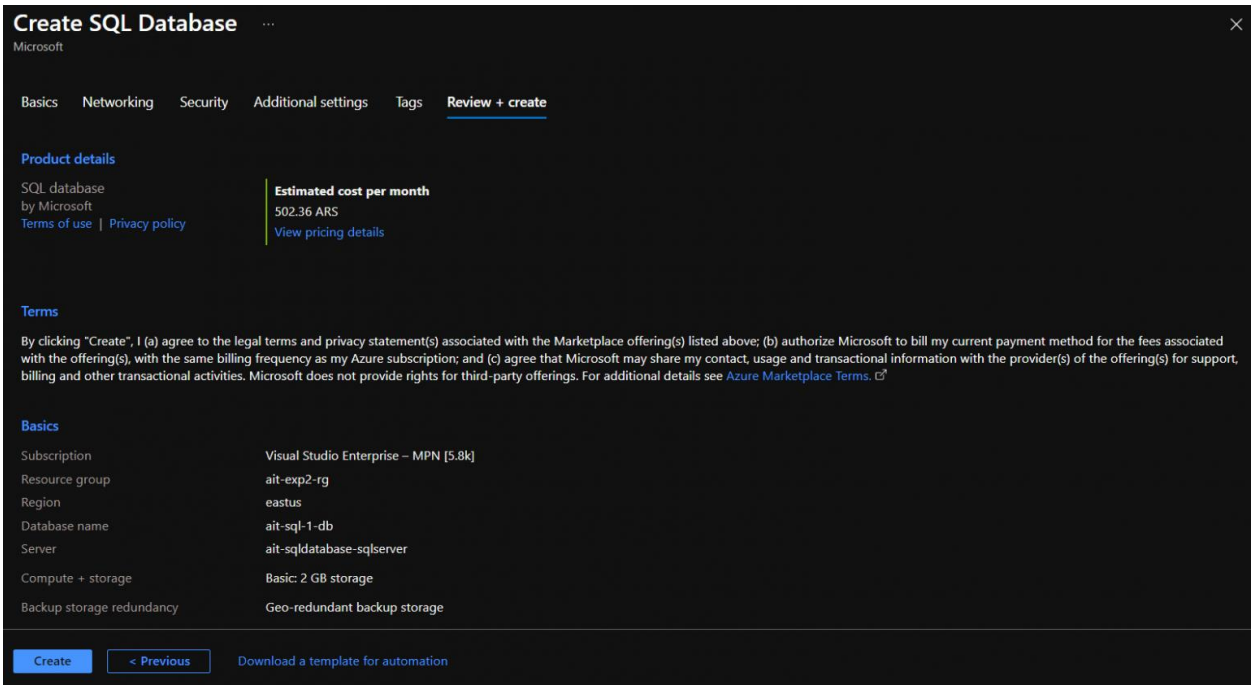
In the next step of the wizard, we will proceed to configure security aspects. For the purposes of this experience, we are not going to enable "Azure Defender for SQL" (at least for now) or configure Ledger:



In the "Additional settings" section we will find the possibility of configuring that the created database is blank, or starting with an example or backup restoration. We'll start with the blank database:

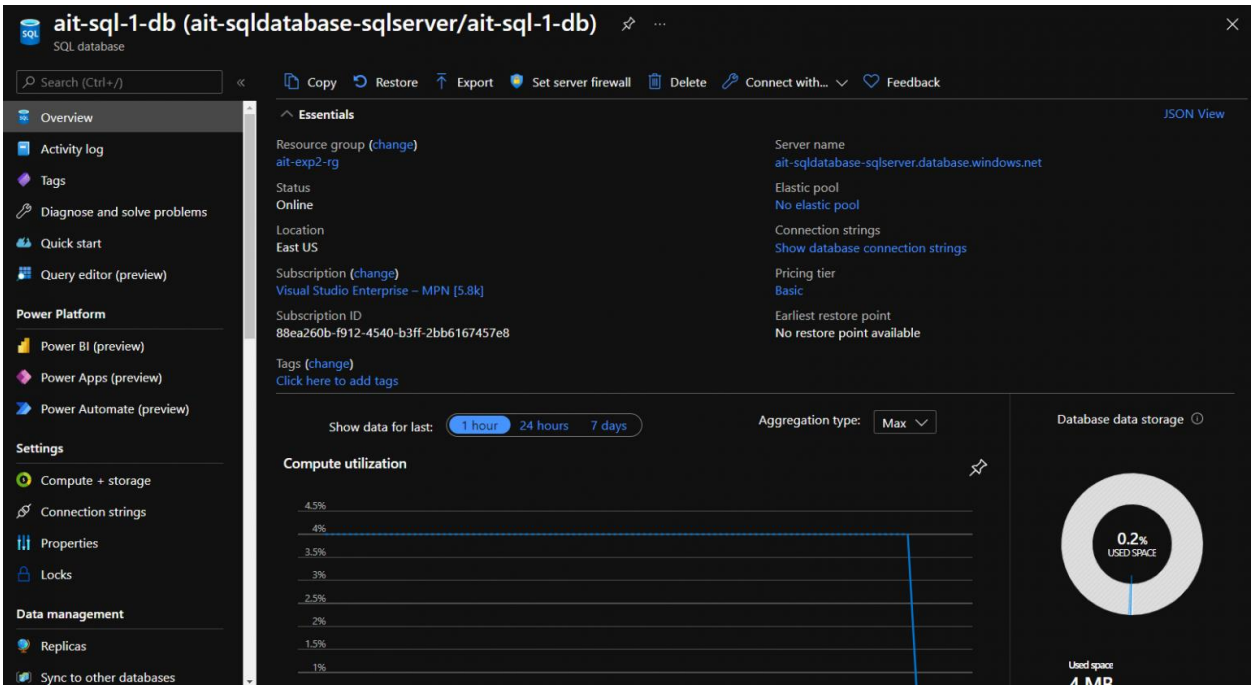


We validate the selected options in the summary part, and when we are ok we click on "Create":

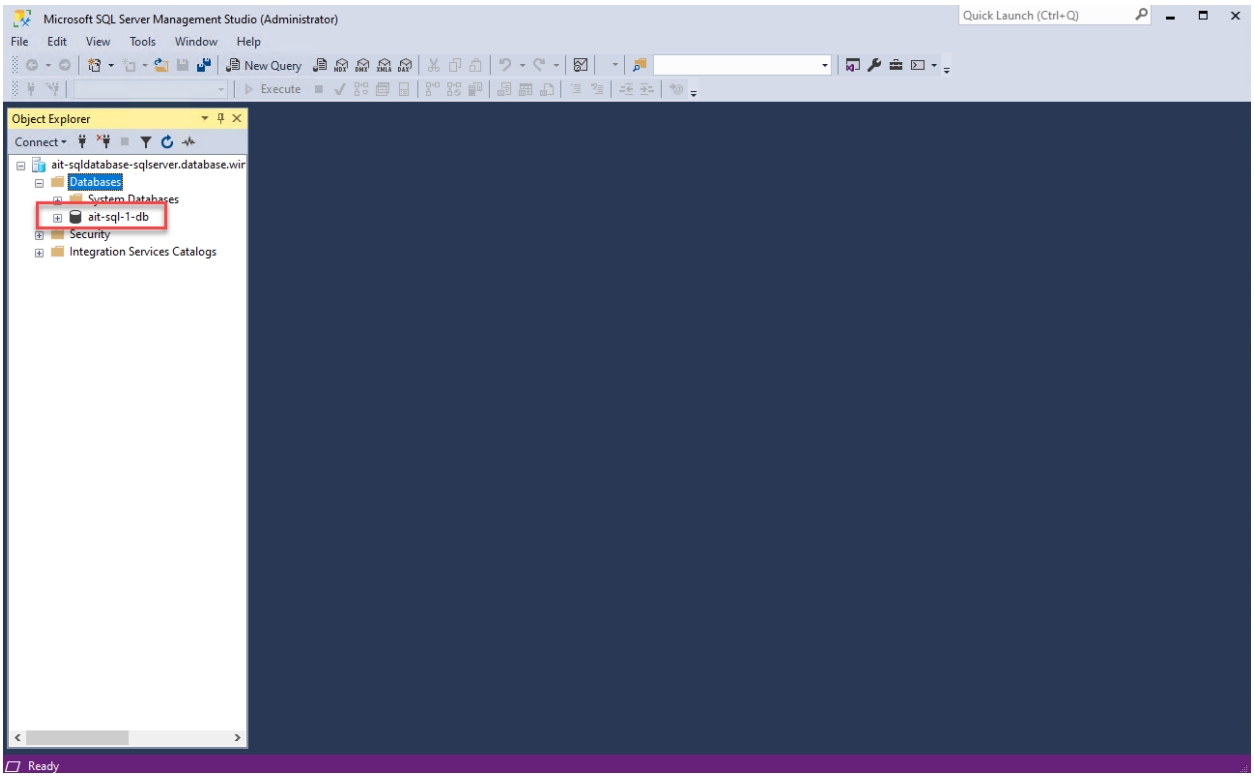


Database and connection validation

When the database creation task is finished, we can enter the particular database. We will find an Overview screen where we can see some specific metrics:



We can see the database from any of the administration tools that we have installed in our virtual machine:



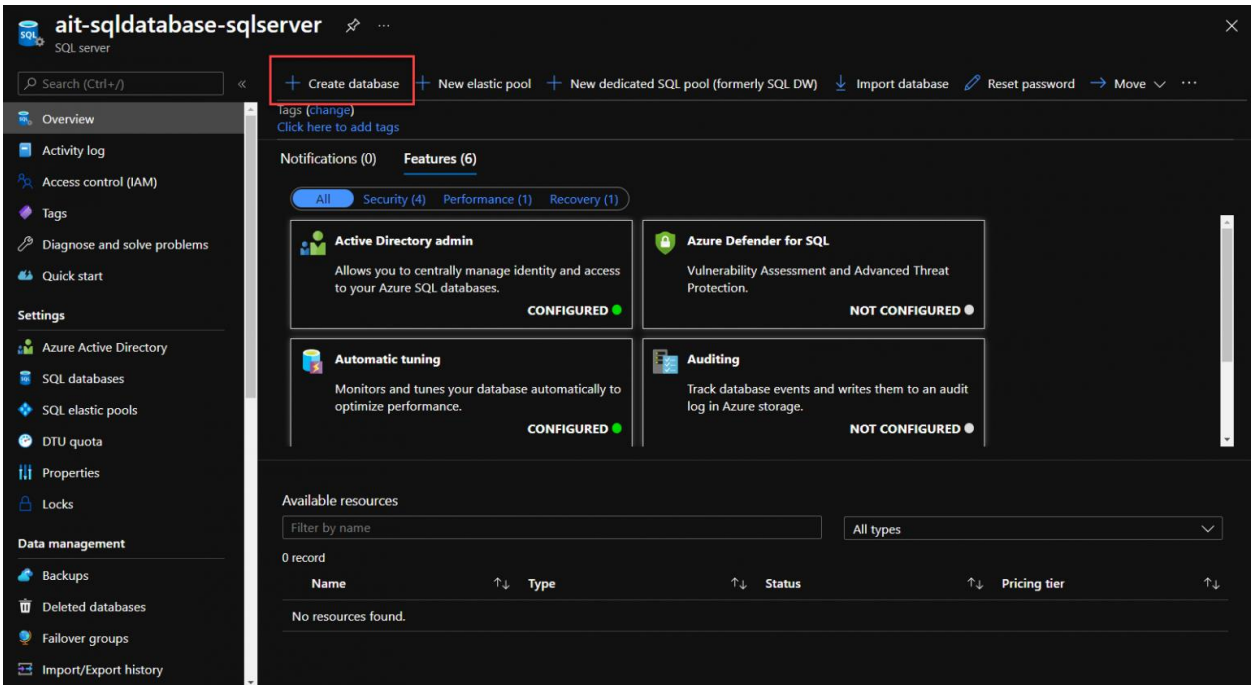
Ready! We have generated our first single database based on DTU.

Single Database Creation (vCore)

We are going to create our first unique database using the vCore-based purchasing model. The wizard is the same one we went through in the previous step, although the selected purchase model changes.

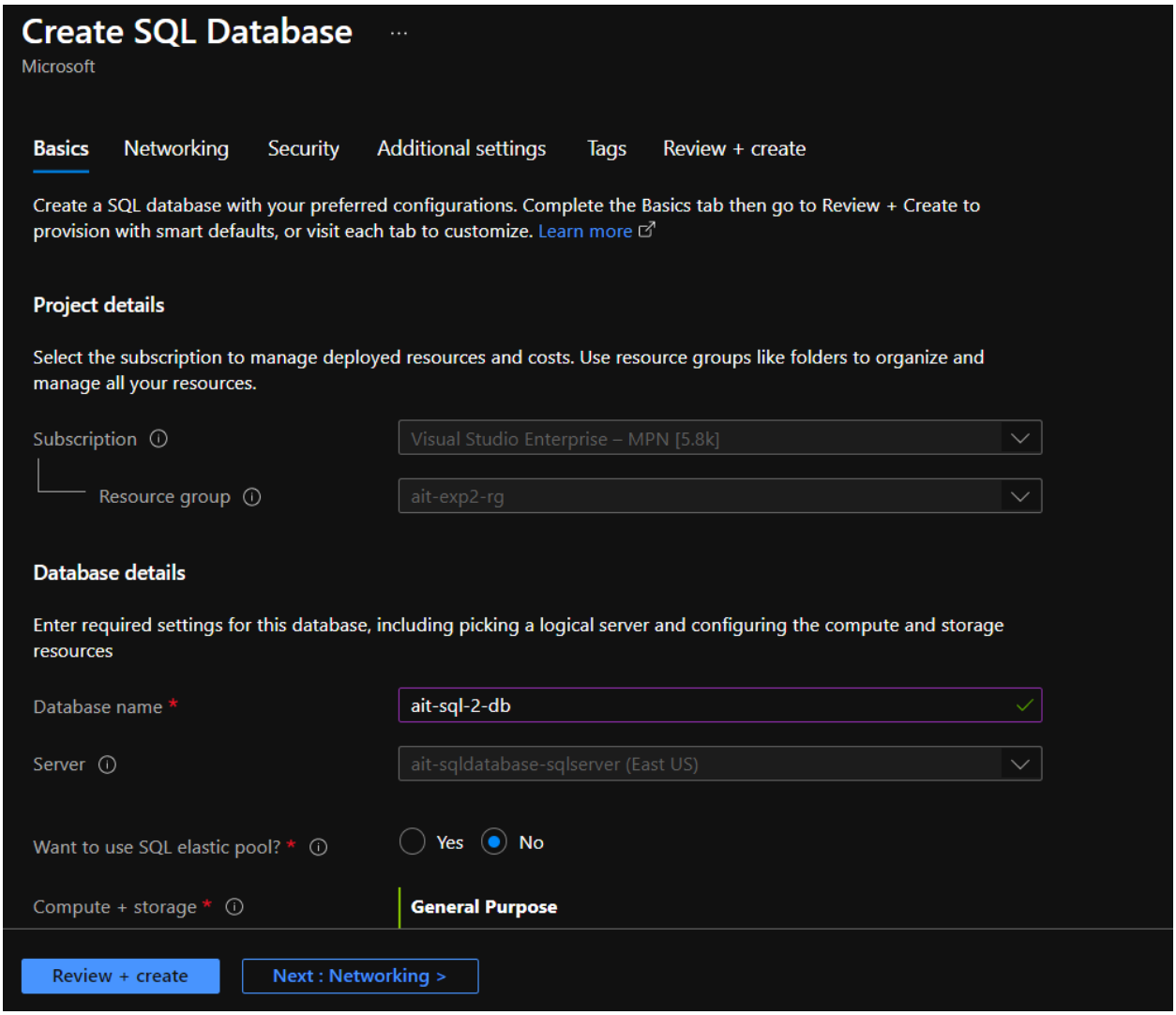
Creation of a single database based on vCore

From our Azure SQL Database Server, on the Overview screen, we are going to click on Create database:

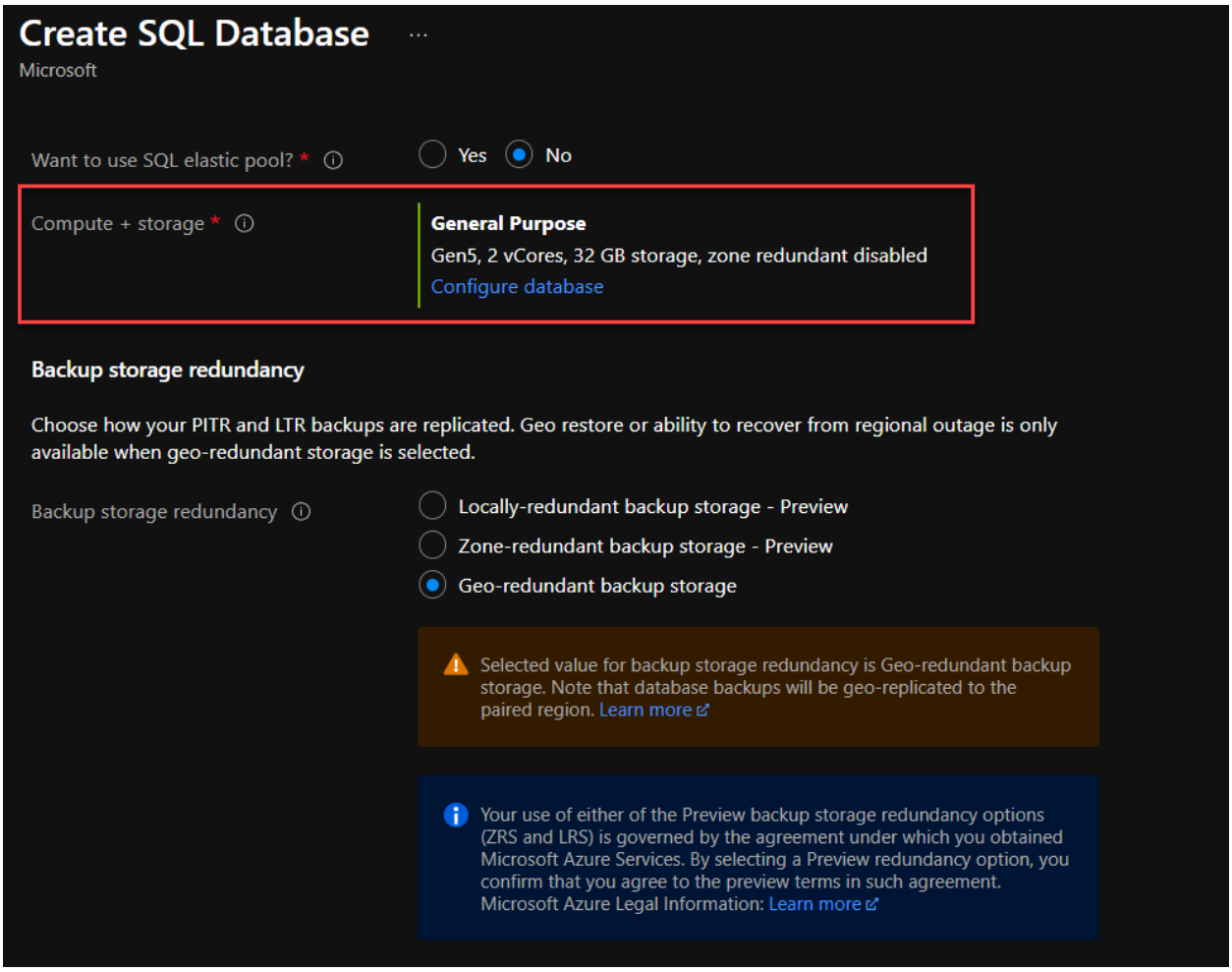


This will start the SQL Database creation wizard. In the same way, we could have generated a new resource in our Resource Group and searched for “Azure SQL”, and then chosen the “Single database” item in the “SQL database” option.

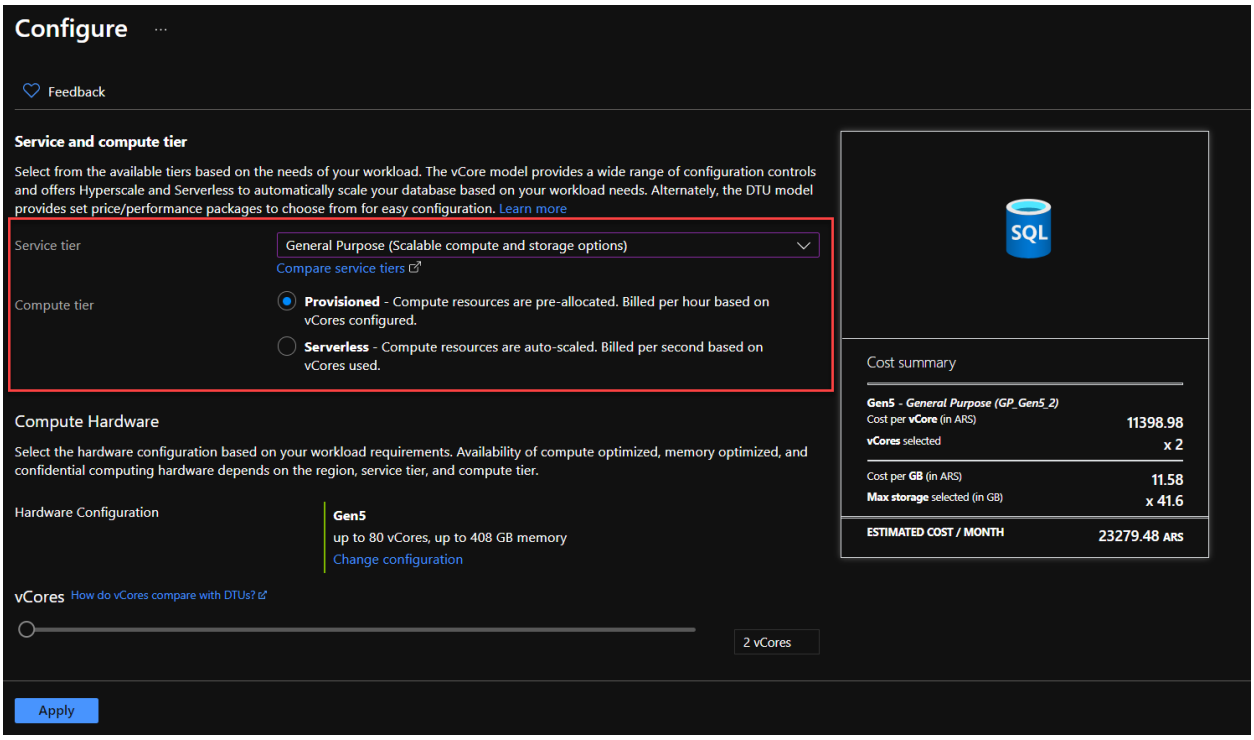
Note that we only need to fill in the name of the database. For our exercise we will enter ait-sql-2-db:



In the “Compute + storage” section we are going to choose the purchase model and the service level, by clicking on “Configure database”:



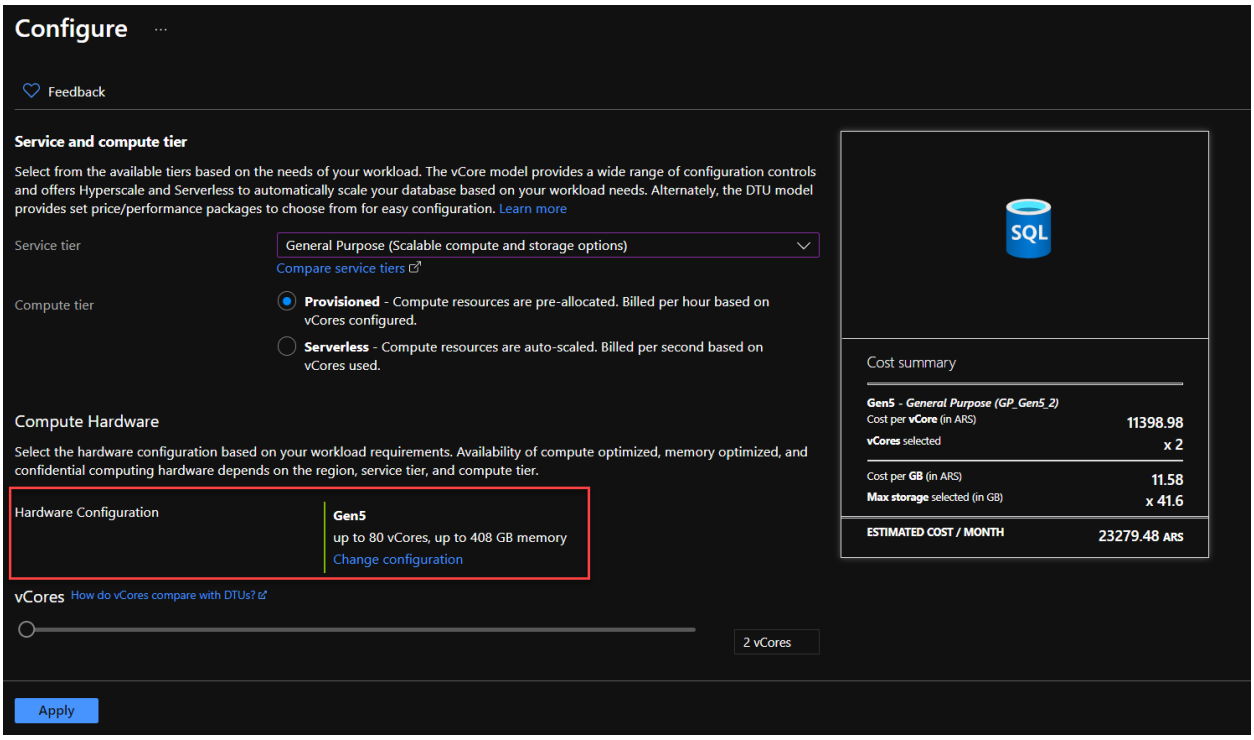
On the screen that is displayed we can see in "Service tier" the options based on vCore and DTU. We are going to select the option based on vCore “General Purpose”:



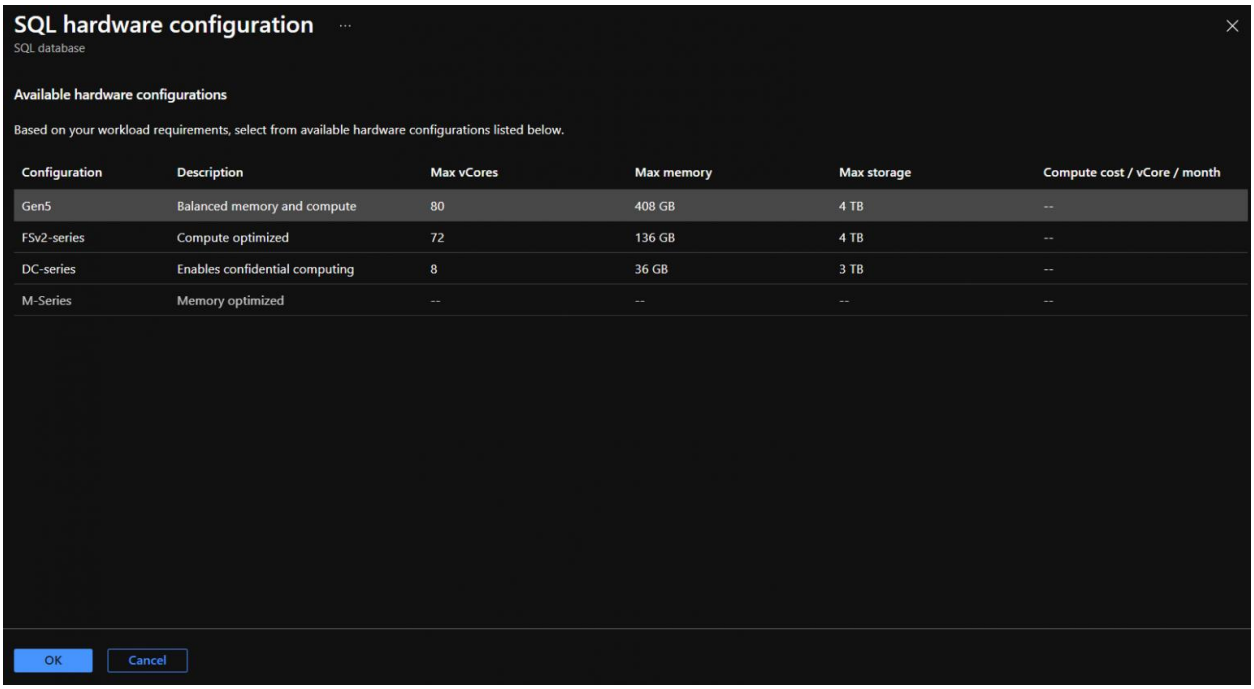
Note, in the image above, that we have the Compute tier available with two basic options:

- Provisioned – This means that the compute will be pre-allocated and paid for hourly based on the number of vCores selected.
- Serverless: Unlike the previous model, compute resources are not pre-allocated and only scale on demand. You will be billed per second of utilization based on the vCores ultimately used.

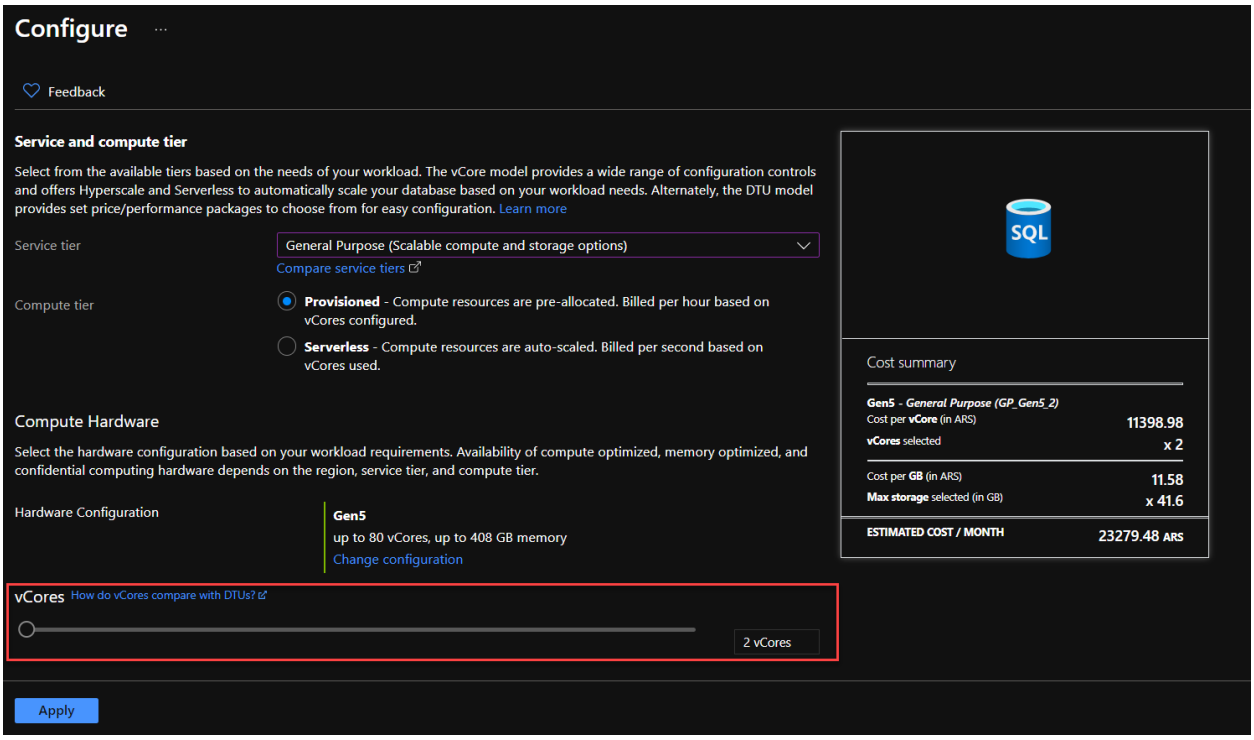
We will work on the “Provisioned” Compute tier, and this will allow us to configure the hardware as we see below. Let's explore the "Hardware Configuration" options:



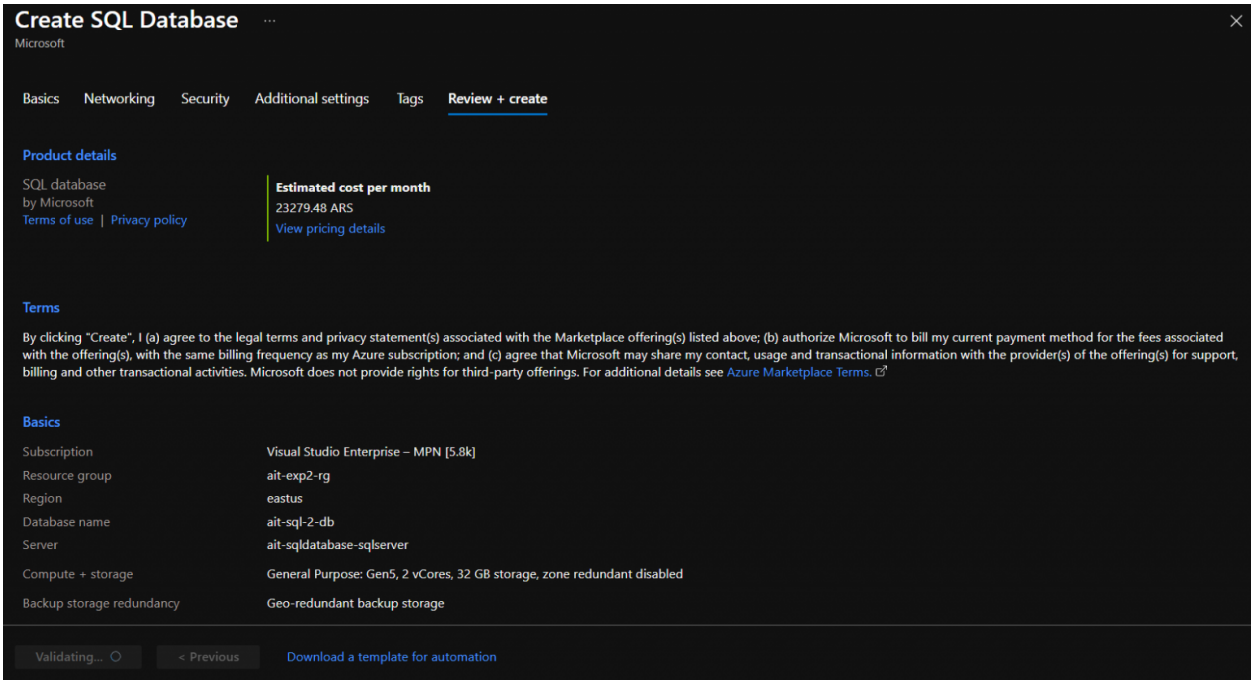
We have limited selection options, with a possibility of maximum use of Cores and Memory, as well as storage for our data. We will select the “Gen5” option for the purposes of this experience:



Once we select the Hardware, we can configure the vCores that we want to include in our provisioning. For the purposes of this experience we will configure 2 vCores (which is the minimum we can assign):

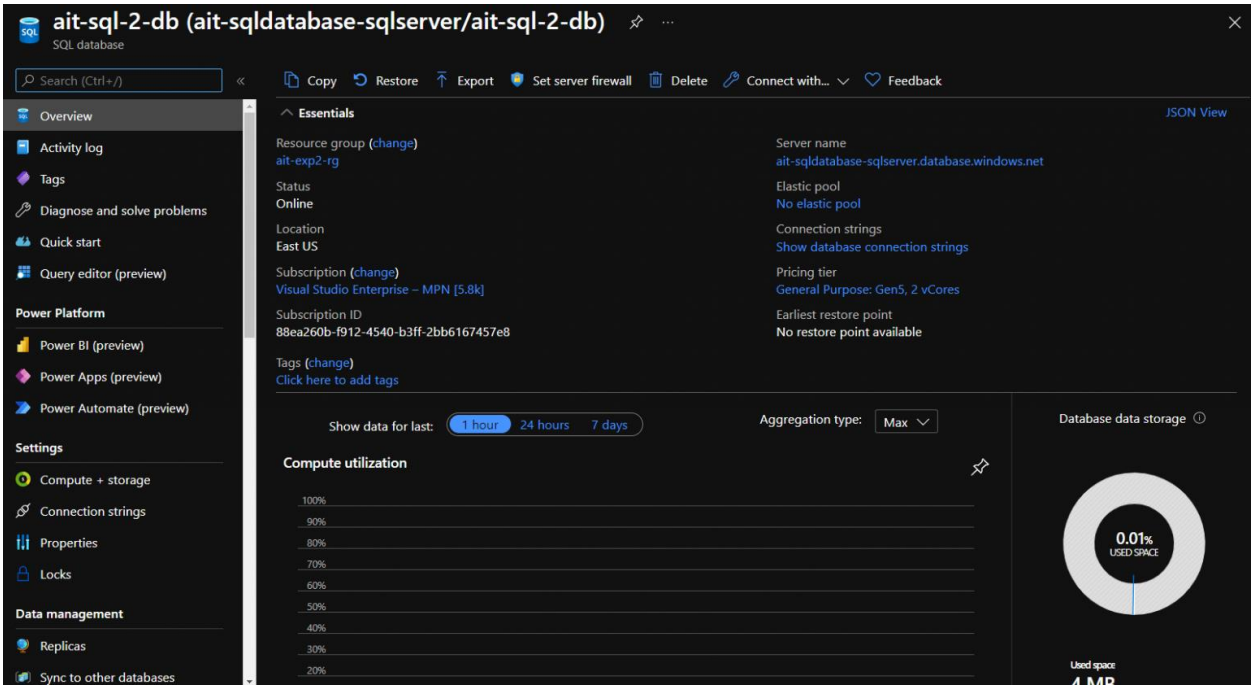


Let's move on to testing and creating our unique database based on vCores:

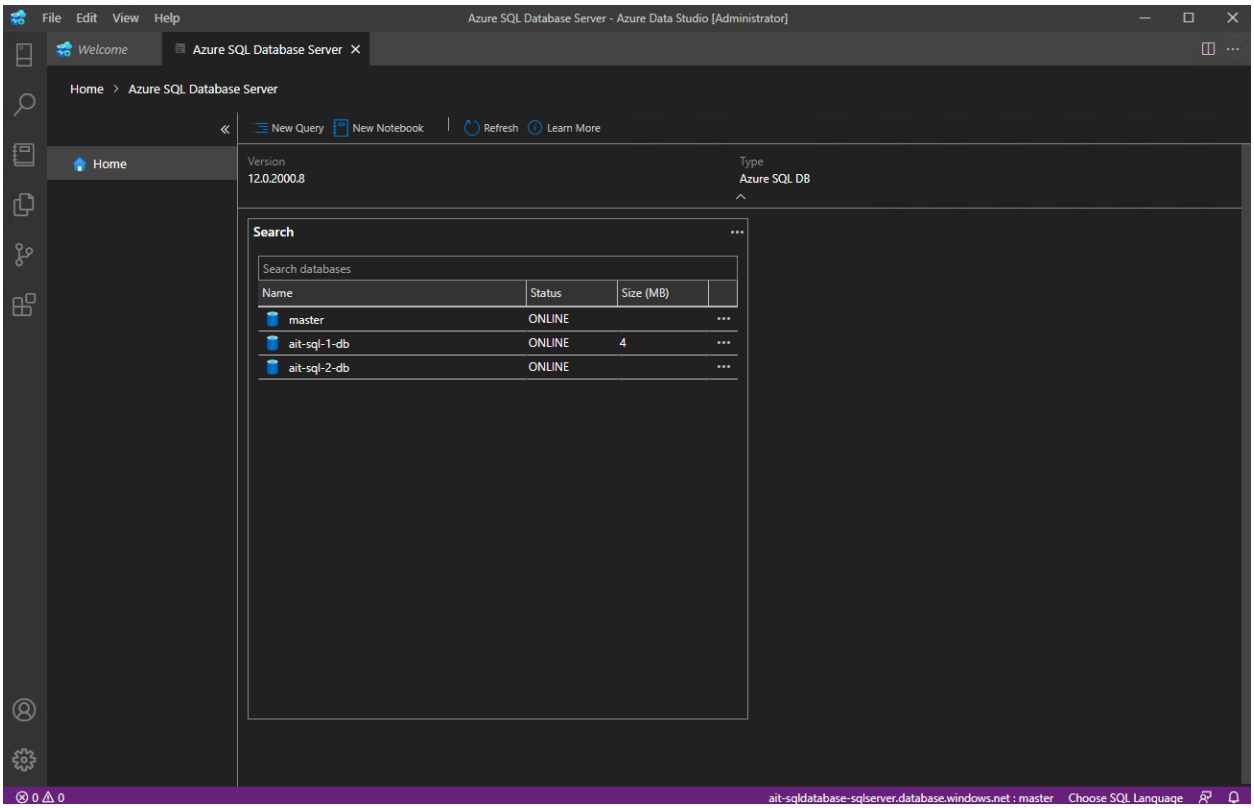


Database and connection validation

When the database creation task is finished, we can enter the particular database. We will find an Overview screen where we can see some specific metrics:



We can see the database from any of the administration tools that we have installed in our virtual machine:



In this case, we will see two databases: the one generated in the previous exercise and the one generated in this one, from Azure Data Studio.

Exercise 3: Azure SQL Database (part 2)

Introduction to the exercise

In this section we will do the third experience together. This experience requires you to have a Microsoft Azure subscription set up and available and consumes any credits you have on it. Please review this topic to avoid charges on subscriptions where you don't want them to exist.

In this experience we will:

- Learn more about Azure SQL Database.
- Create an Azure SQL Database elastic pool.
- Move SQL Databases from single DB to elastic pool, both for DTU and vCore.
- Configure authentication with Azure AD admin.
- Integrate with virtual networks to increase security.

This experience is going to be intense! Let's keep going!

About elastic groups

What is an elastic pool in Azure SQL Database?

As we have reviewed in the theoretical introductions, Azure SQL Database elastic pools are a simple and cost-effective solution for managing and scaling multiple databases that have different and unpredictable usage demands.

In an elastic pool, the databases reside on a single server and share a set number of resources at a set price. Elastic pools in Azure SQL Database allow SaaS developers to optimize price performance for a pool of databases within a prescribed budget while providing performance elasticity for each database.

Elastic groups in real life

SaaS developers build applications at the upper data tiers of the scale that consist of multiple databases. A common application pattern is to provision a unique database for each client. However, each customer often has variable and unpredictable usage patterns, and it is difficult to predict the resource requirements of each individual database user. Traditionally, there were two options:

- Overprovisioning resources based on peak usage and overpaying, or
- Under-provision to save costs at the expense of performance and customer satisfaction during peak hours.

Elastic pools solve this problem and ensure that databases get the performance resources they need, when they need them. They provide a simple resource allocation mechanism within a predictable budget.

Elastic pools for DTU and vCore

Elastic pools allow a developer to purchase resources for a shared pool across multiple databases to cope with unpredictable periods of usage by individual databases. You can configure the resources for the pool based on the DTU-based purchasing model or the vCore-based purchasing model. The resource requirement for a pool is determined by the aggregate usage of its databases. The amount of resources available to the pool is determined by the developer's budget.

The developer simply adds databases to the pool, optionally sets the minimum and maximum resources for the databases (min or max DTUs or min and max vCores, depending on the resource model choice), and then sets the resources of the group according to your budget. A developer can use pools to efficiently grow their service from a Lean Startup method to a mature business on an ever-increasing scale.

Elastic Pool Costs

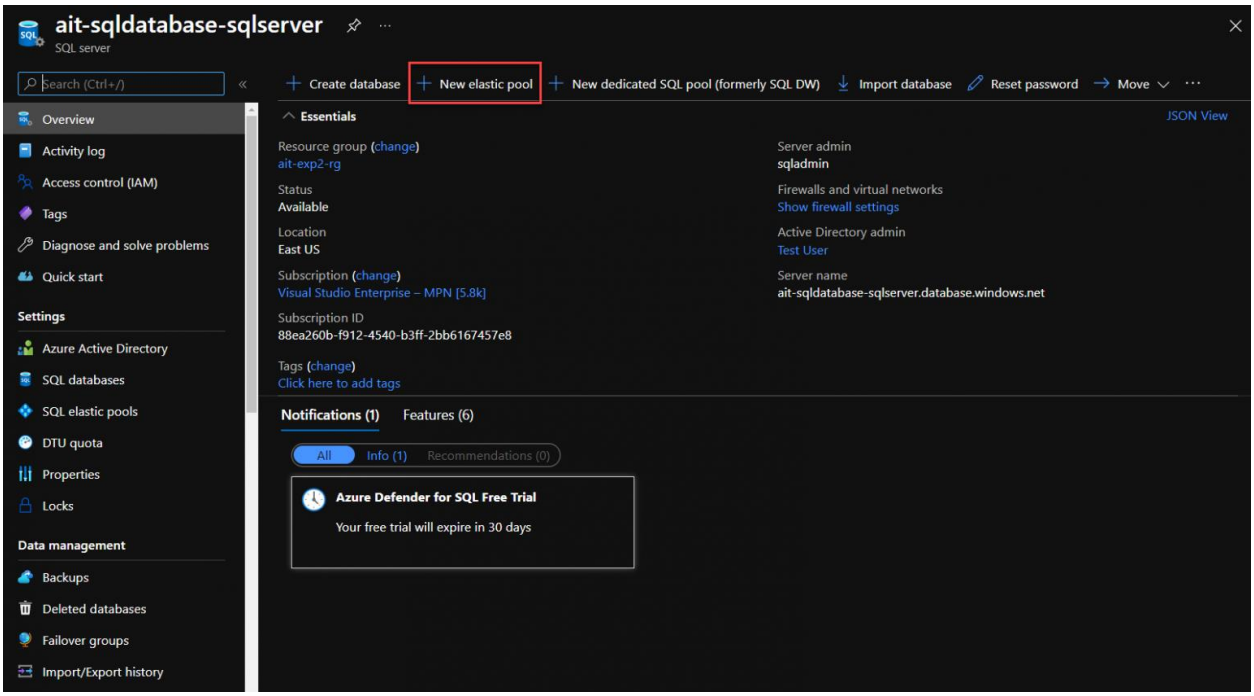
There are no database charges for elastic pools. You are charged for every hour that a pool exists in the highest eDTU or vCores, regardless of usage or whether the pool was active for less than one hour.

Elastic pool (DTU) creation and database conversion

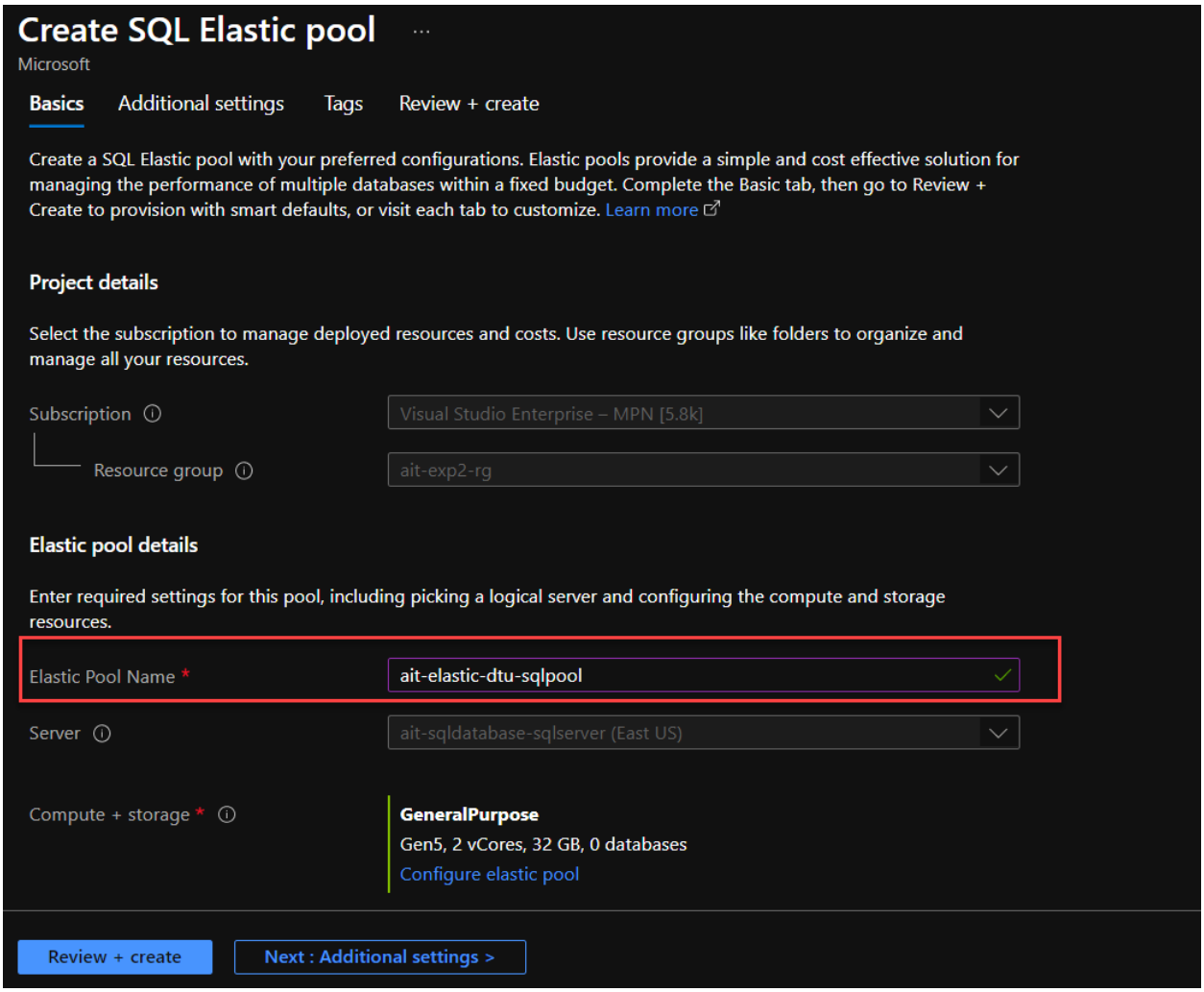
We are going to create an elastic pool based on DTU, and then we are going to move the first database named “ait-sql-1-db” and created earlier to this pool.

Creating an elastic pool based on DTUs

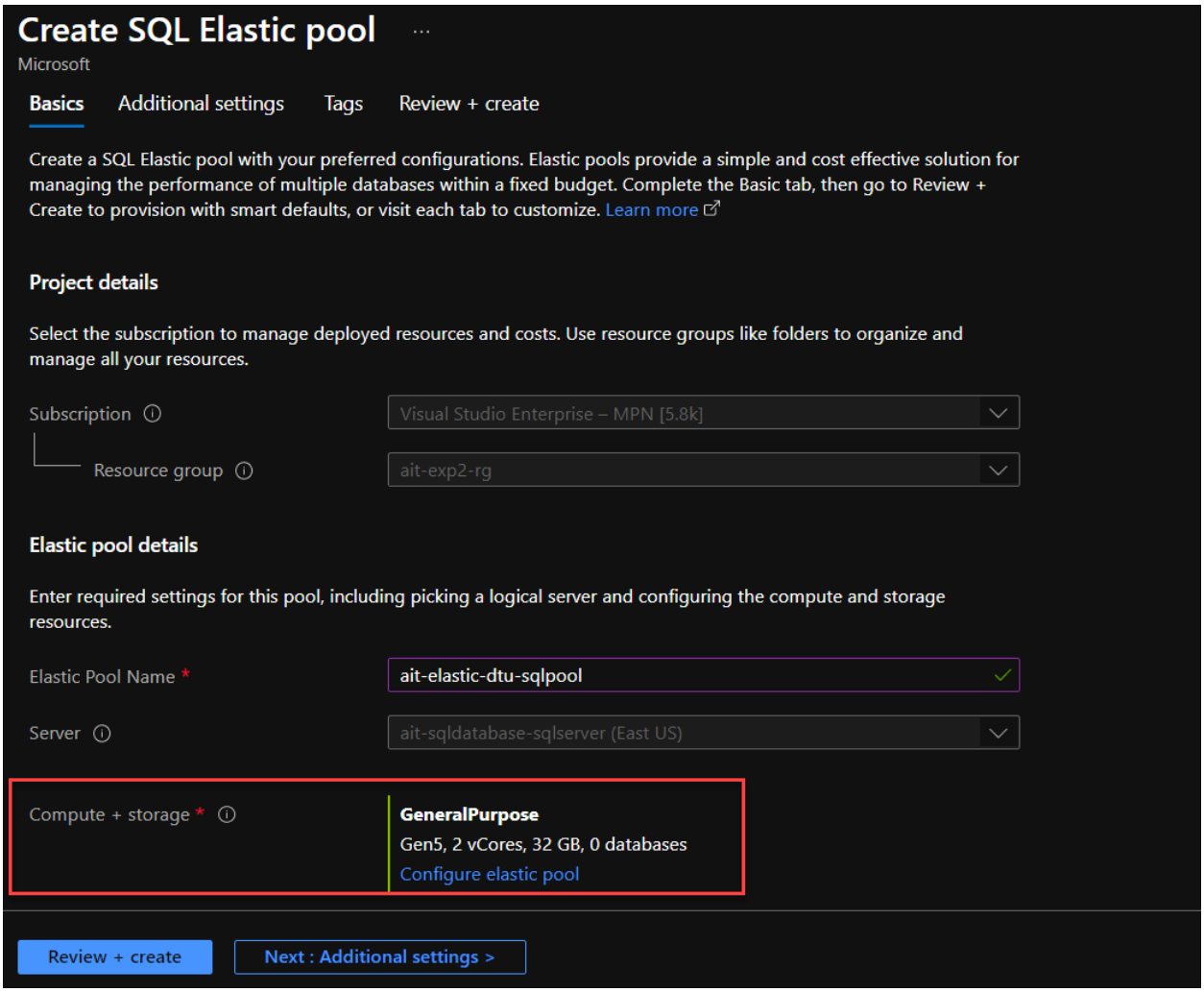
We are going to enter the SQL Database Server that we have generated in experience 2 and, in the Overview screen, we are going to click on “New elastic pool”:



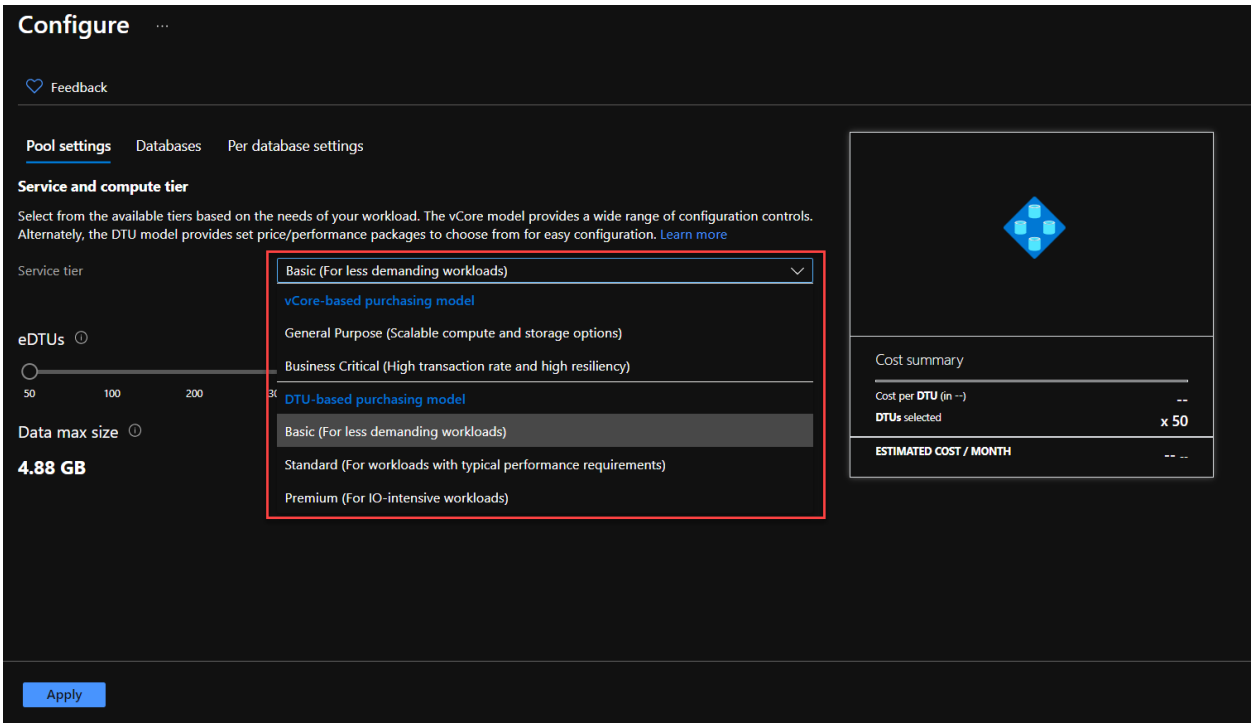
The wizard for creating a new SQL Elastic Pool will start here. Note that the parent of the elastic pool is always the SQL Database Server. We will enter a name for the elastic pool, and a suggested form is ait-elastic-dtu-sqlpool:



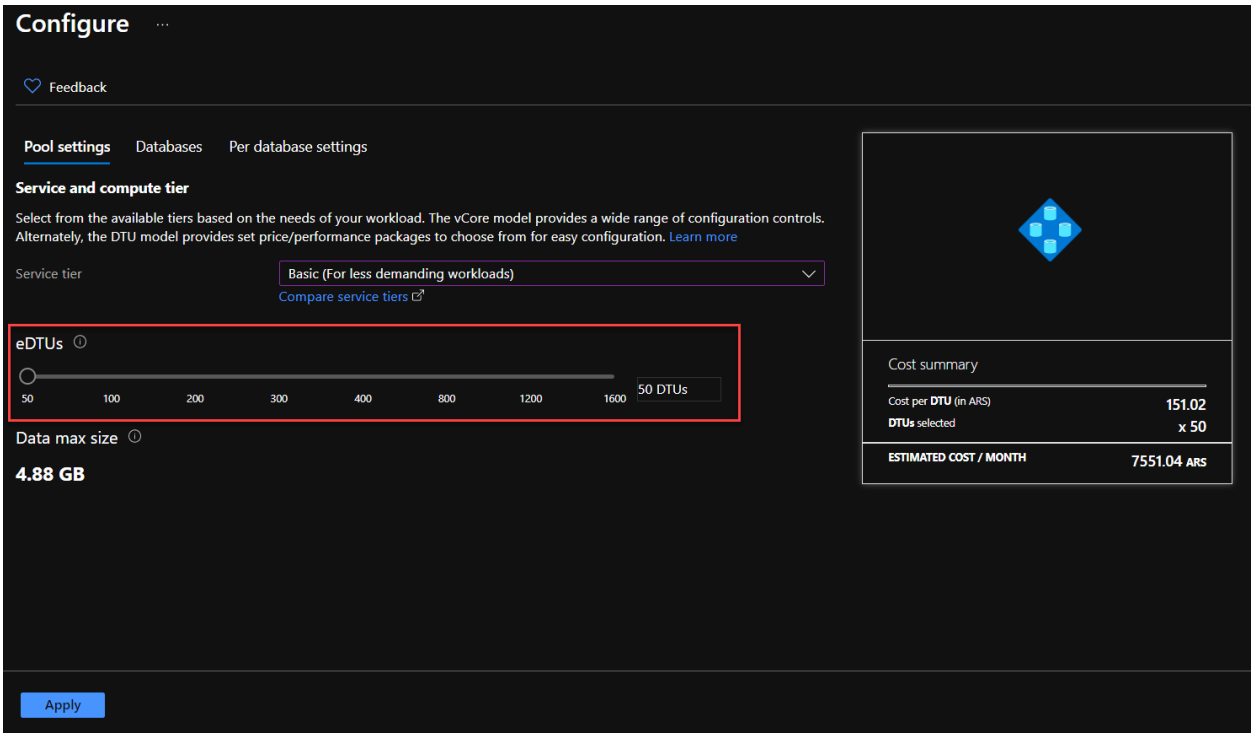
Having entered the name, it remains to configure the computing and storage part. We will click on “Configure elastic pool”:



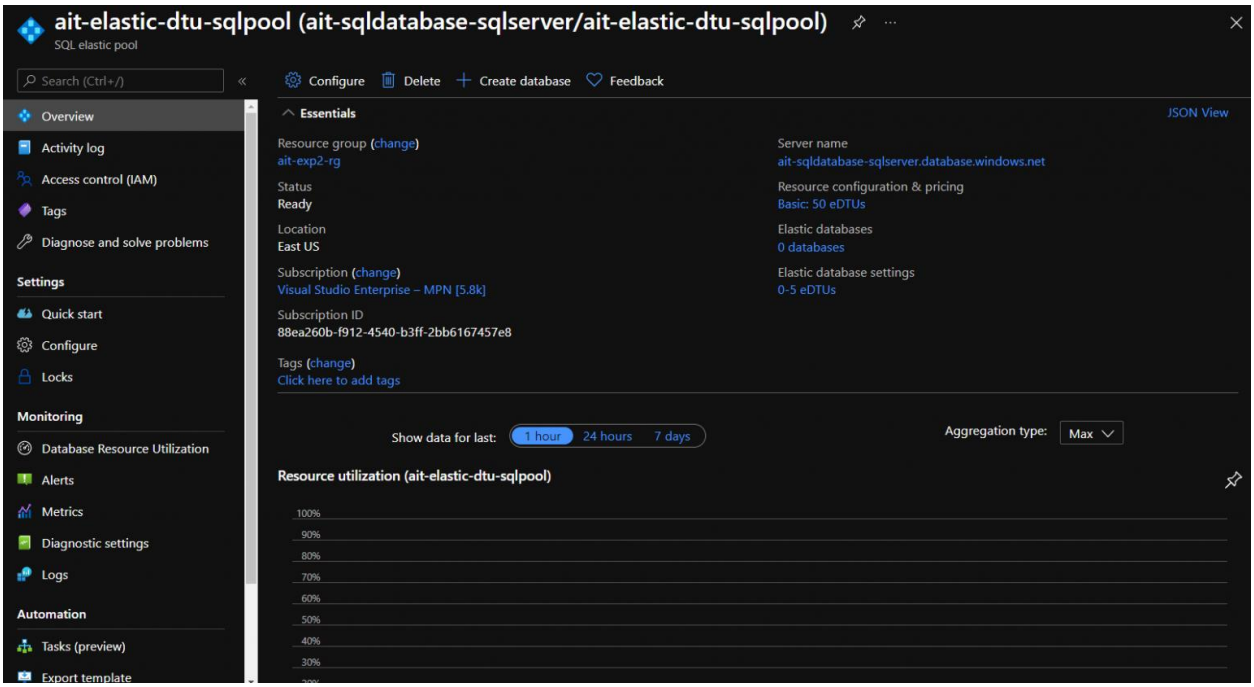
We will notice that we can select between the already known purchase models: vCore and DTU. Let's choose "Basic" from DTU:



Unlike a single database, here the DTUs are called "eDTUs" (Elastic DTUs) and the minimum purchase is 50. Also, the maximum amount of storage in the Basic plan is 4.88 GB:



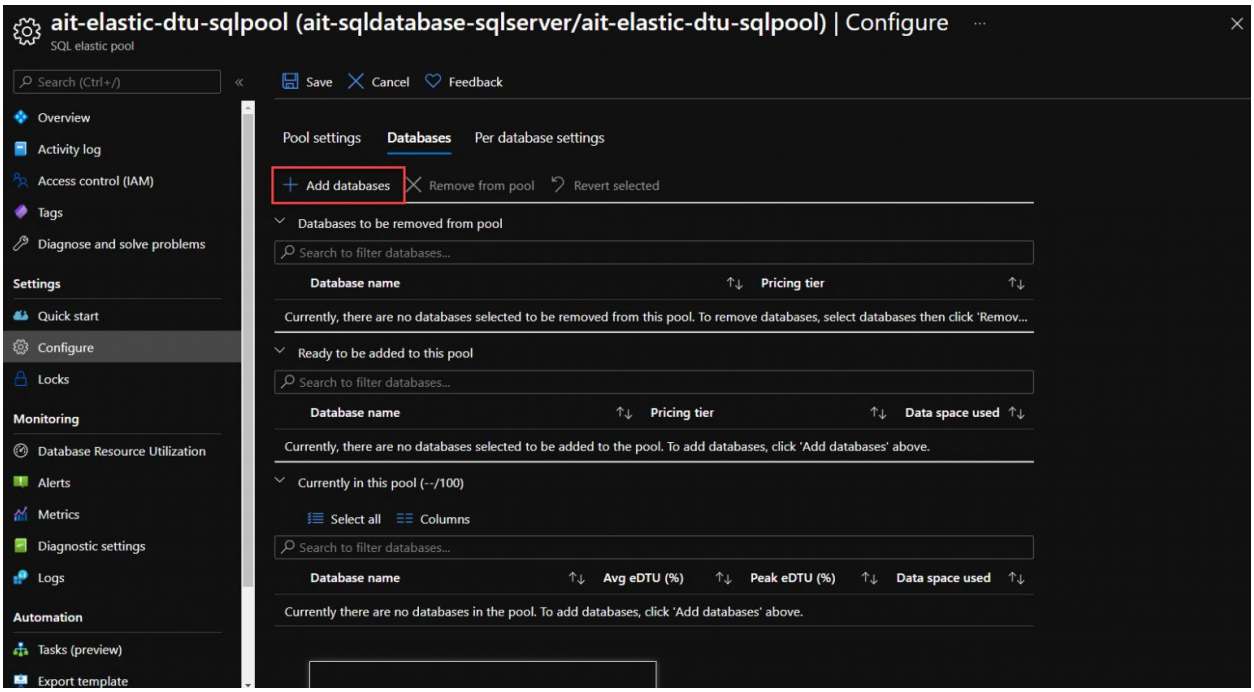
Let's generate the Elastic Pool. Once the generation is finished, we can enter the initial page of the resource that looks like the following:



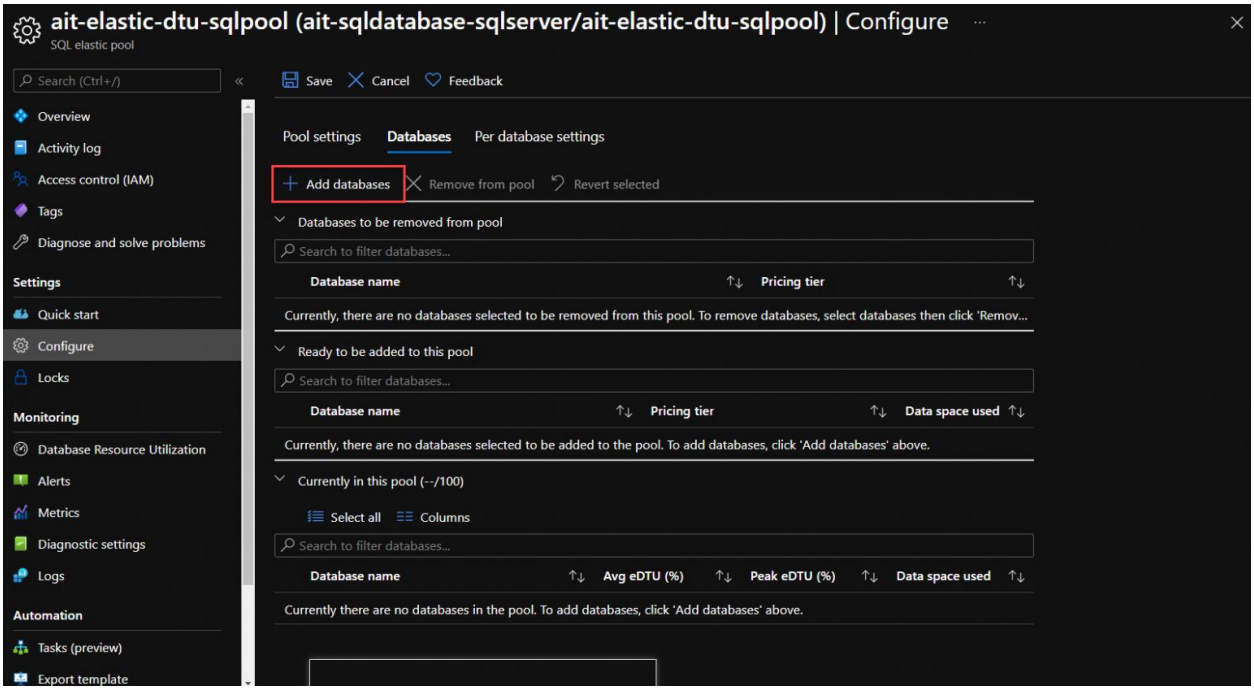
Now let's see how we can include databases in this pool!

Conversion from single database to elastic pool

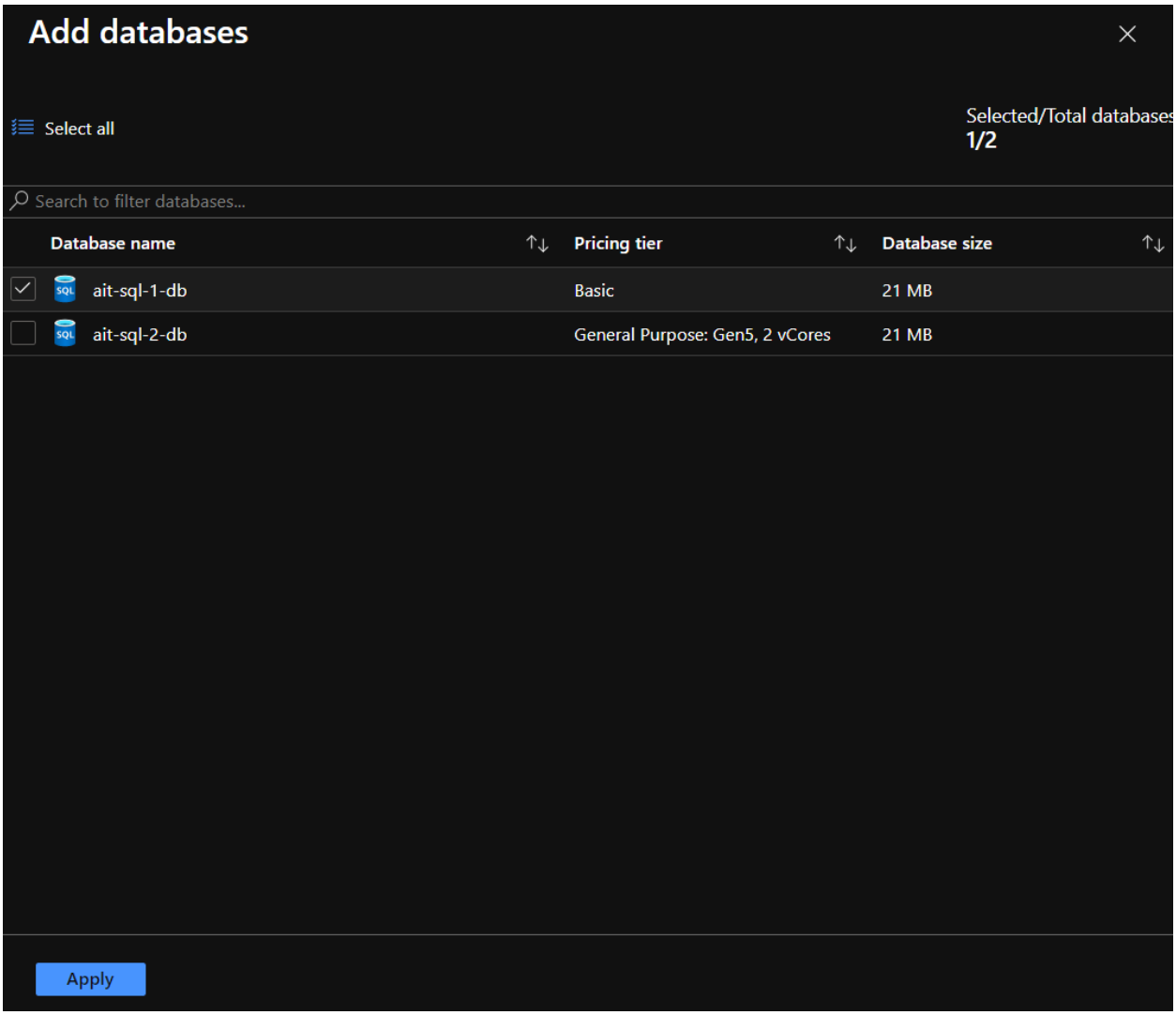
If we enter our SQL elastic pool, we are going to look for the "Configure" part to enter the following screen. From here we can include databases that ALREADY EXIST (we are not creating them now):



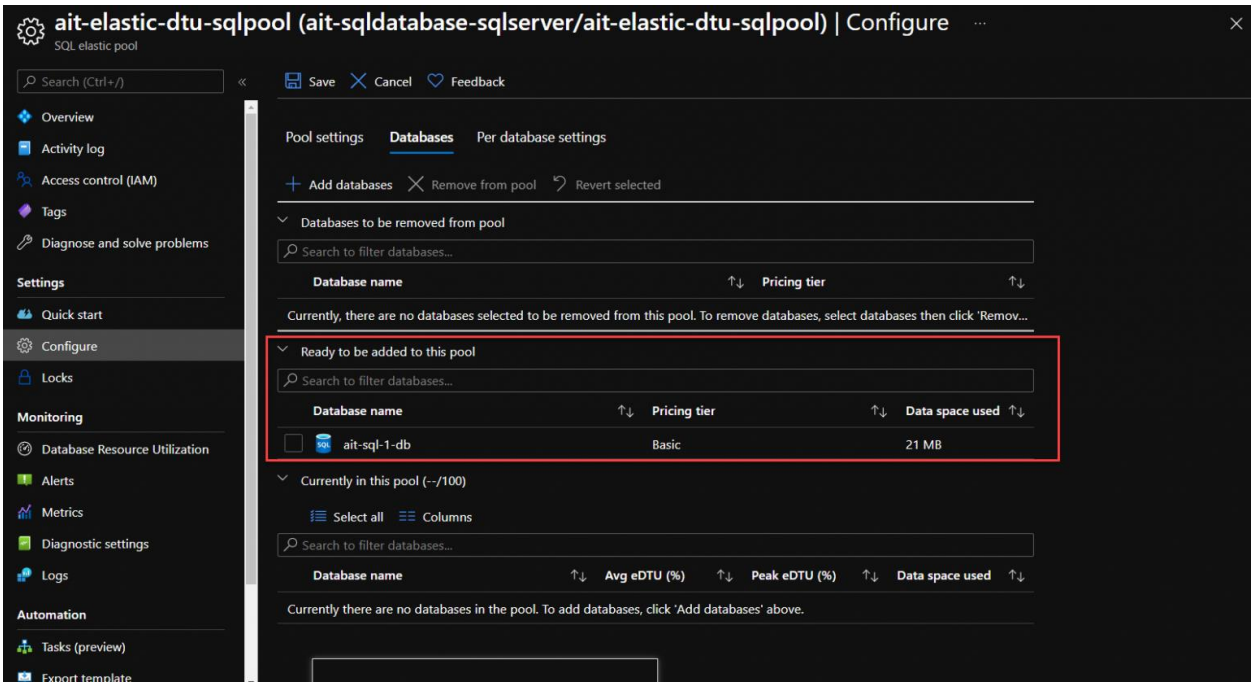
We will click on “Add databases”:



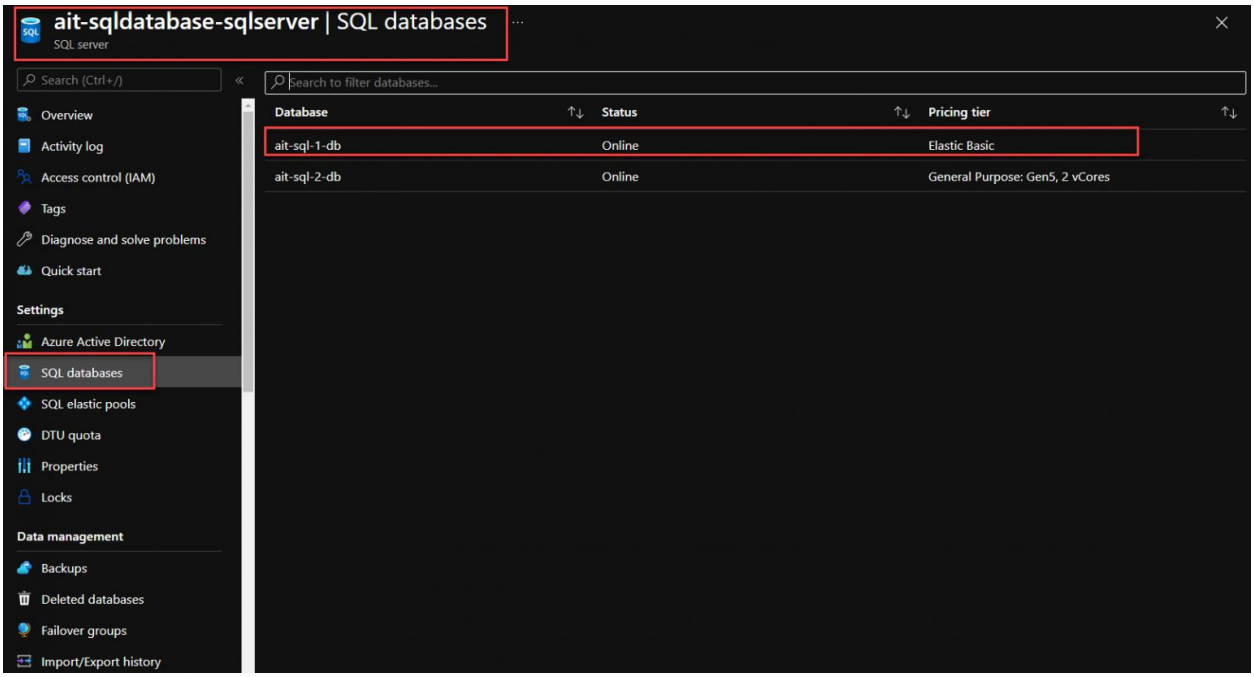
The wizard will open the available databases to add to the Pool. We will select the first one: ait-sql-1-db.



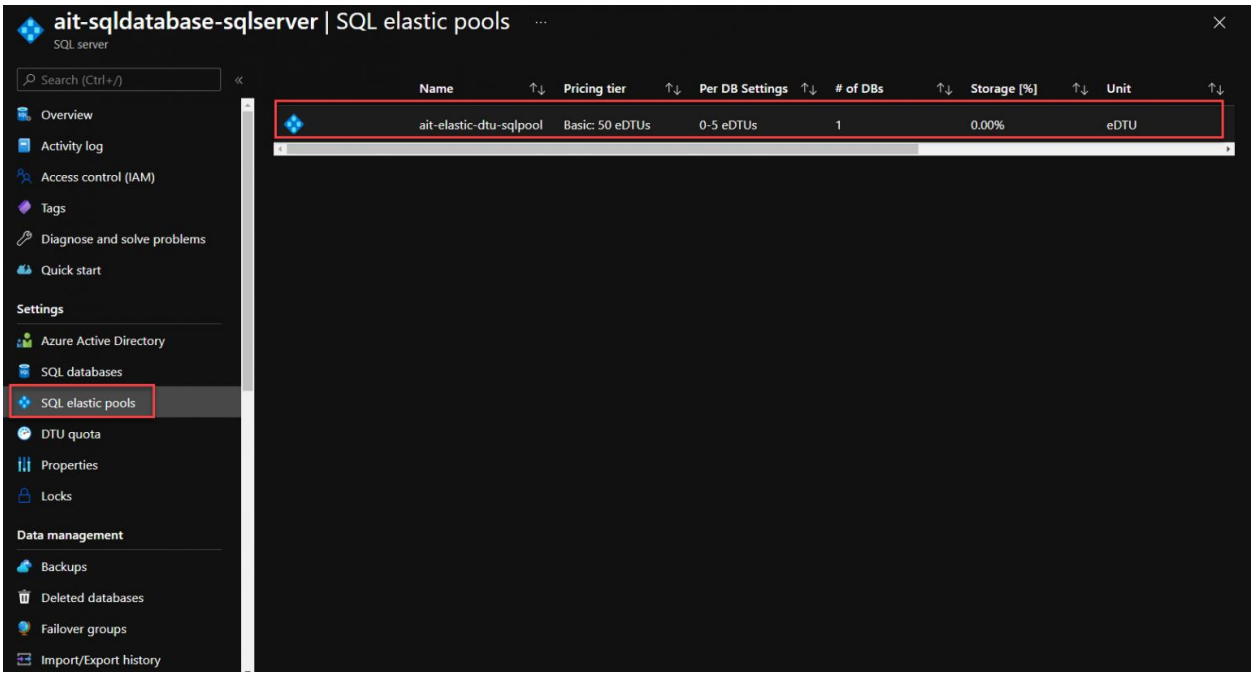
When we click ok, the database will appear in “Ready to be added to this pool”:



Once we confirm the action, and after a few minutes, we can verify that if we enter the SQL Database Server logical server and go to the "SQL databases" option, the database already has the "Pricing tier" of Elastic Basic :



At the same time, if we enter SQL elastic pools within our logical server, we will verify that our pool is already operational:



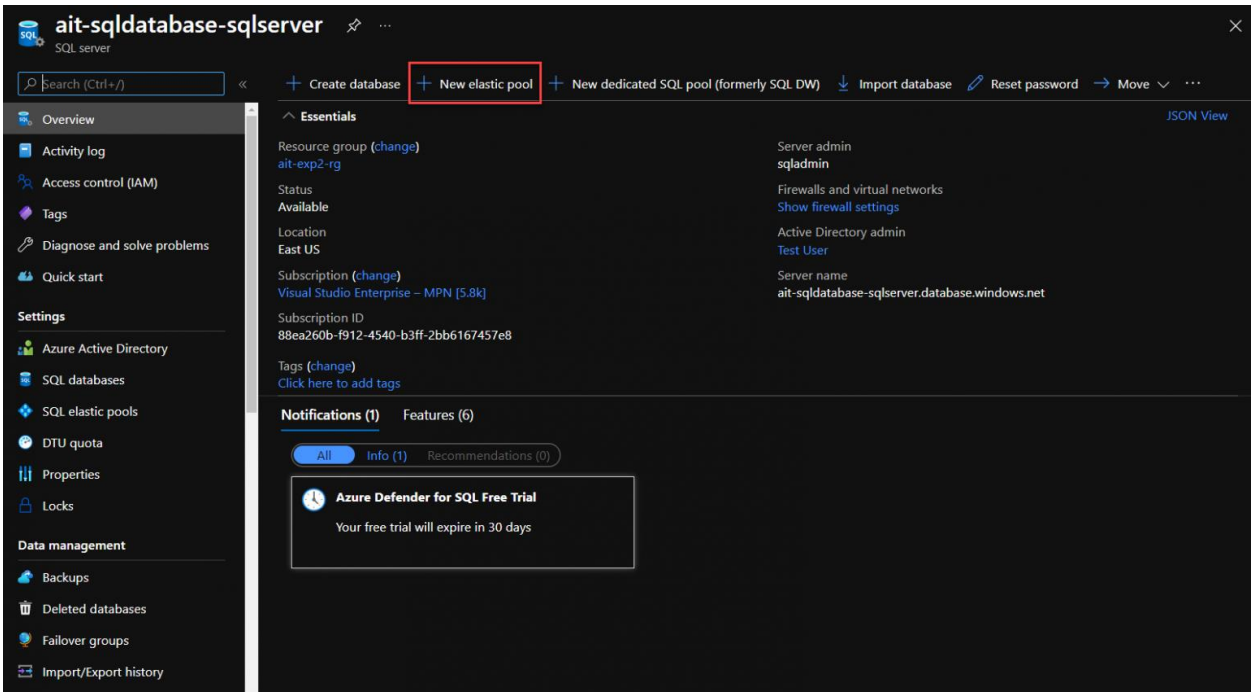
You can try connecting from SQL Management Studio, or from Azure Data Studio: you will see that your base remains the same.

Elastic pool creation (vCore) and database conversion

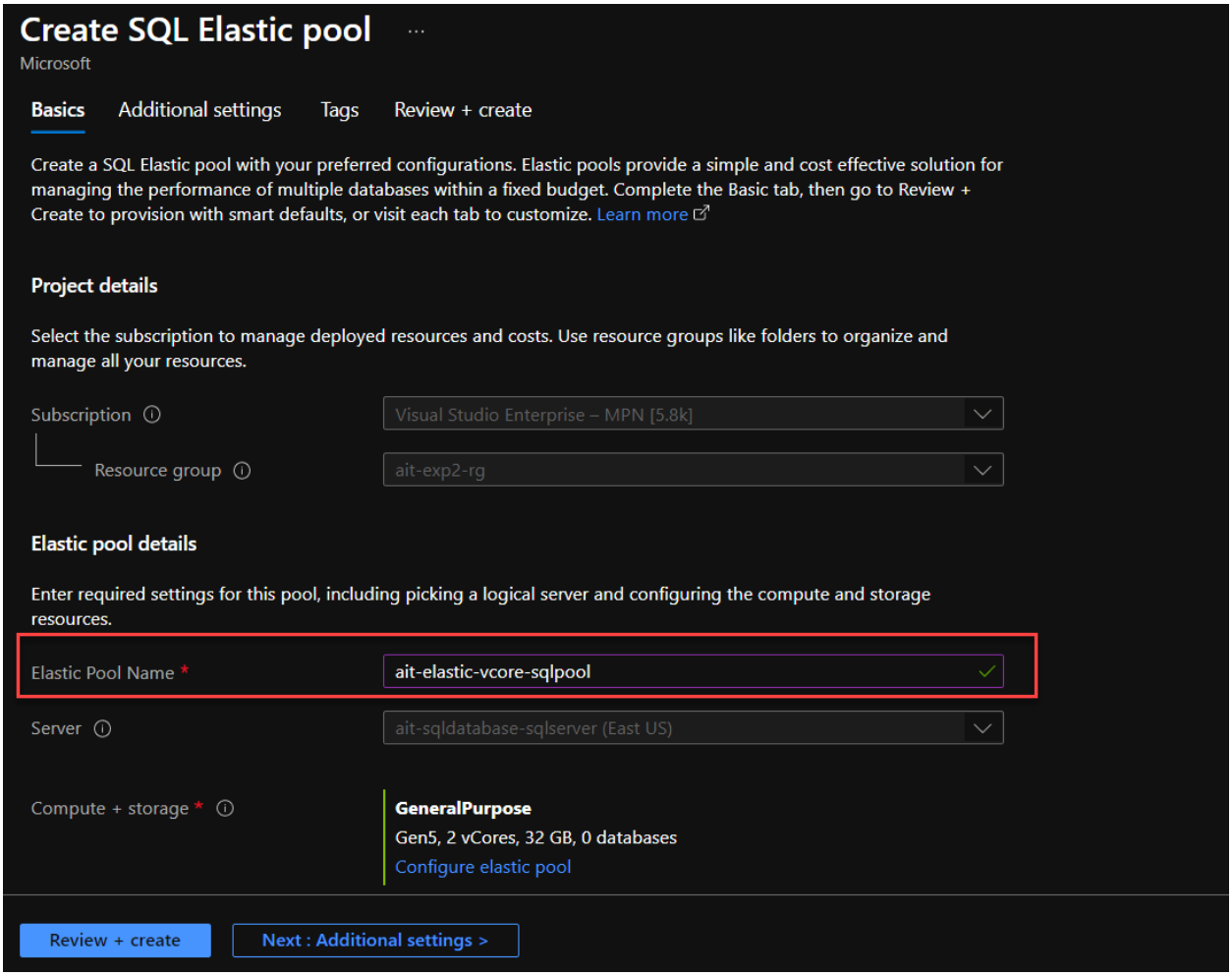
We are going to create an elastic pool based on vCore, and then we are going to move the first database named “ait-sql-2-db” and created earlier to this pool.

Elastic pool creation and single database conversion

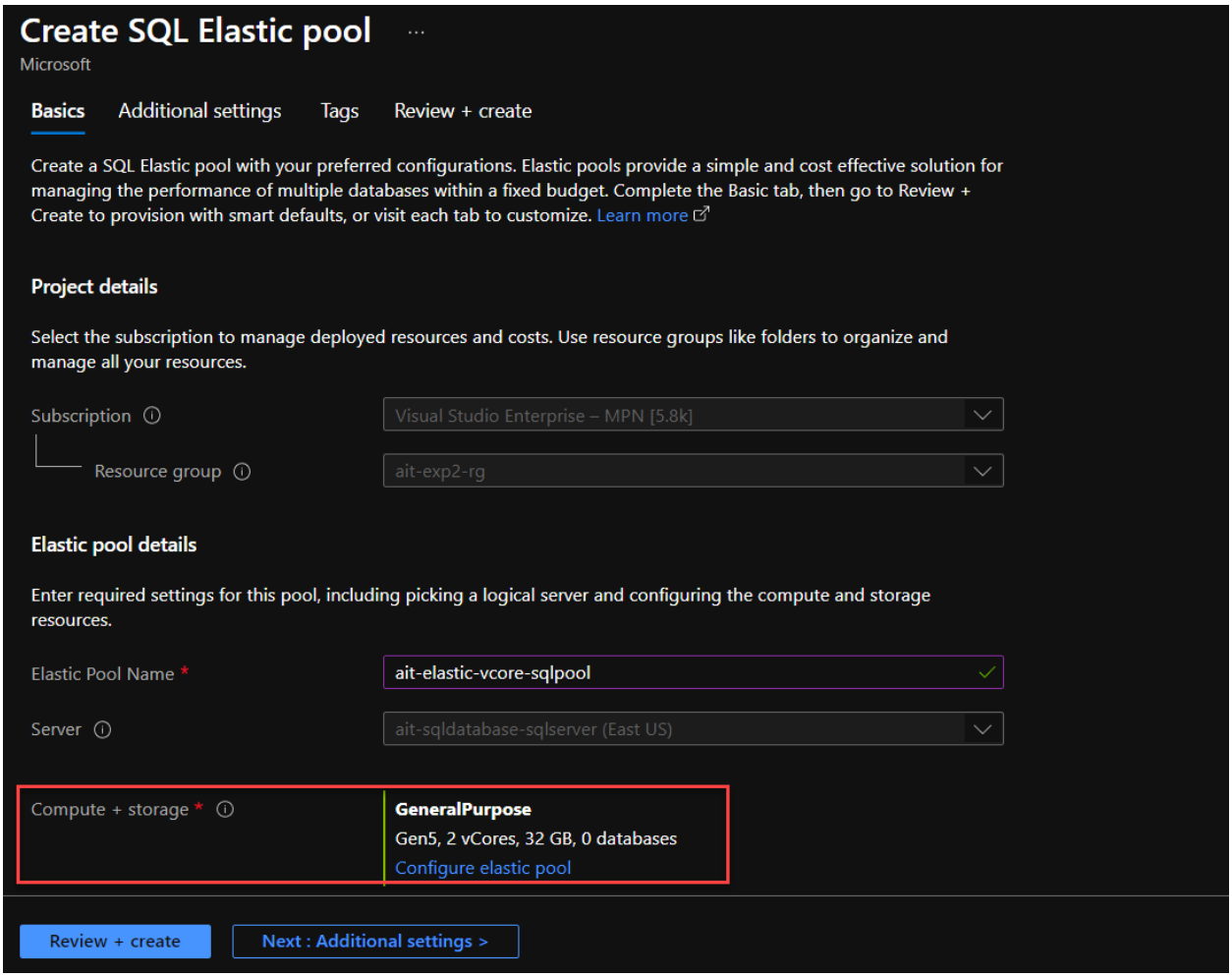
We are going to enter the SQL Database Server that we have generated in experience 2 and, in the Overview screen, we are going to click on “New elastic pool”:



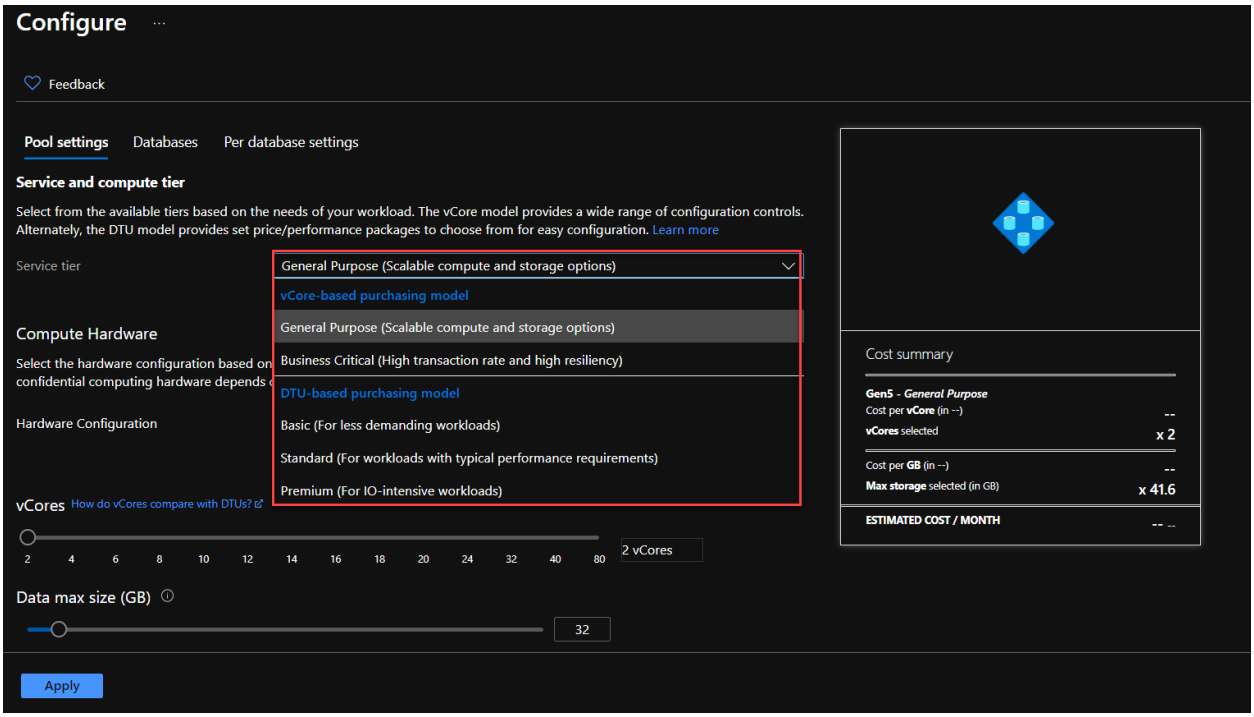
The wizard for creating a new SQL Elastic Pool will start here. Note that the parent of the elastic pool is always the SQL Database Server. We will enter a name for the elastic pool, and a suggested form is ait-elastic-vcore-sqlpool:



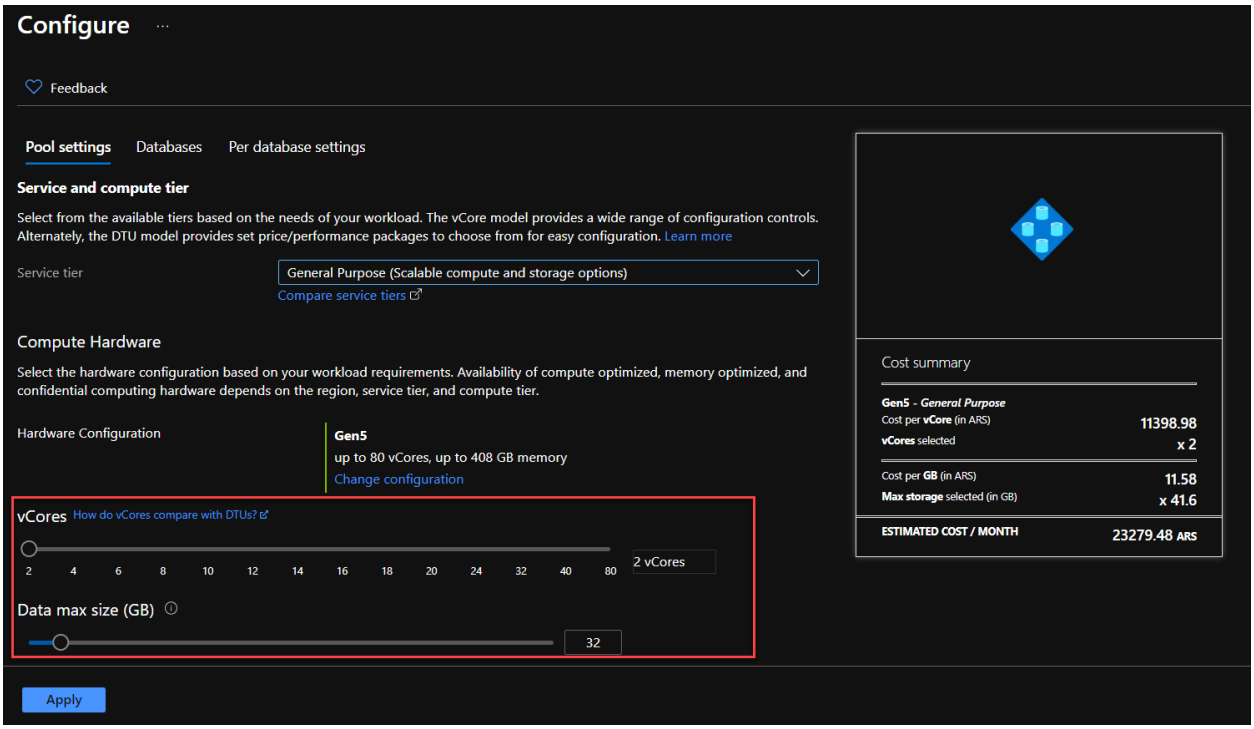
Having entered the name, it remains to configure the computing and storage part. We will click on “Configure elastic pool”:



We will notice that we can select between the already known purchase models: vCore and DTU. Let's choose "General Purpose" of vCore:



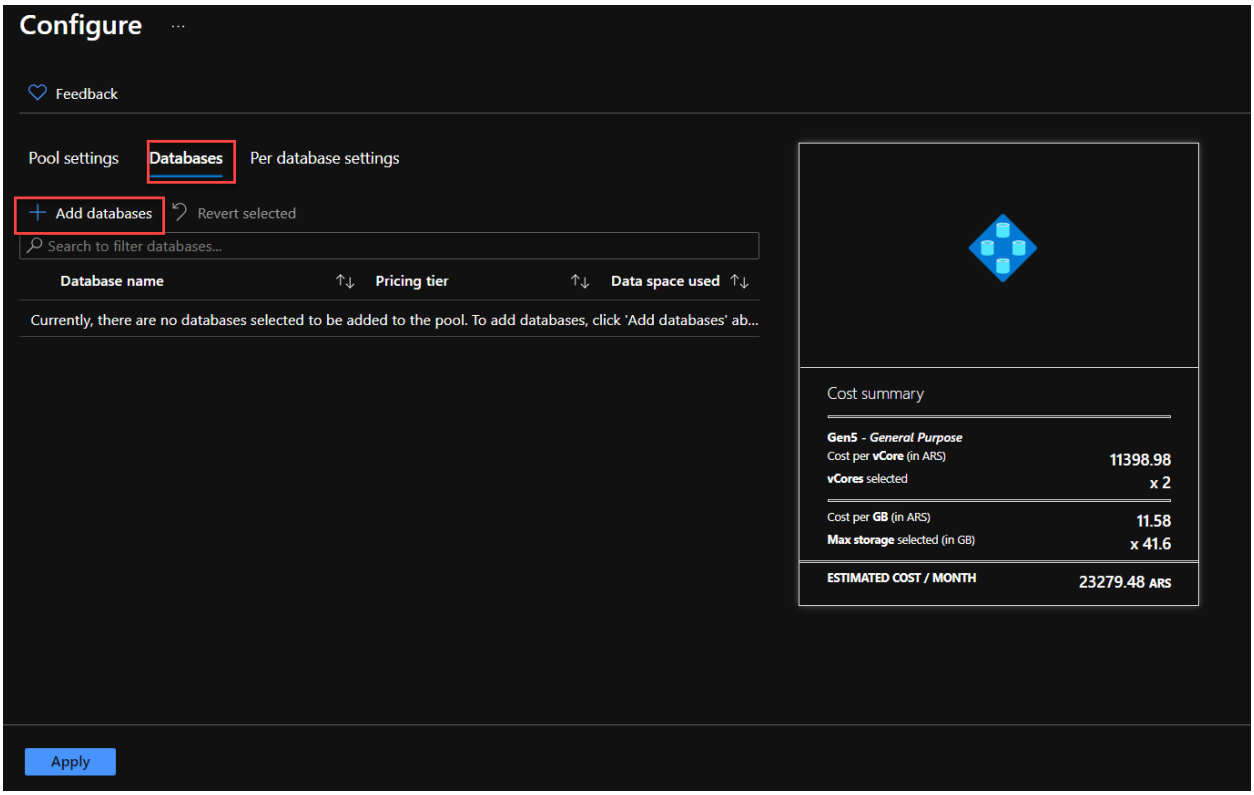
Unlike the previous case, where we created the elastic pool based on DTU, here we can choose vCores (for the experience we will select 2) and the maximum amount of pre-provisioned data (storage) (for the experience we will select 32 GB):



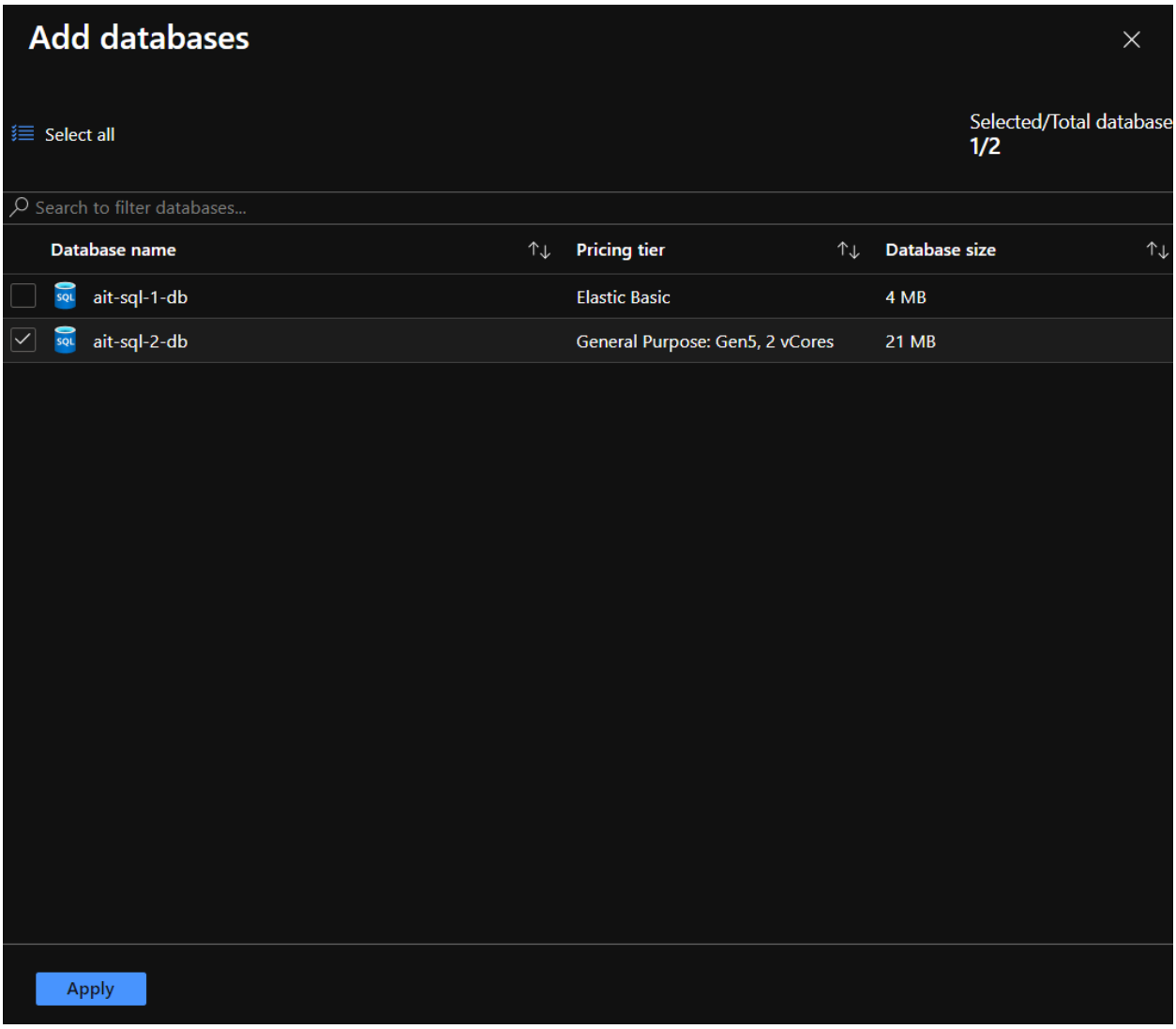
One-time database conversion during Pool creation

Unlike the previous lesson, this time we won't wait to convert databases and include them inside the instance. We'll explore a way to do this during Pool creation.

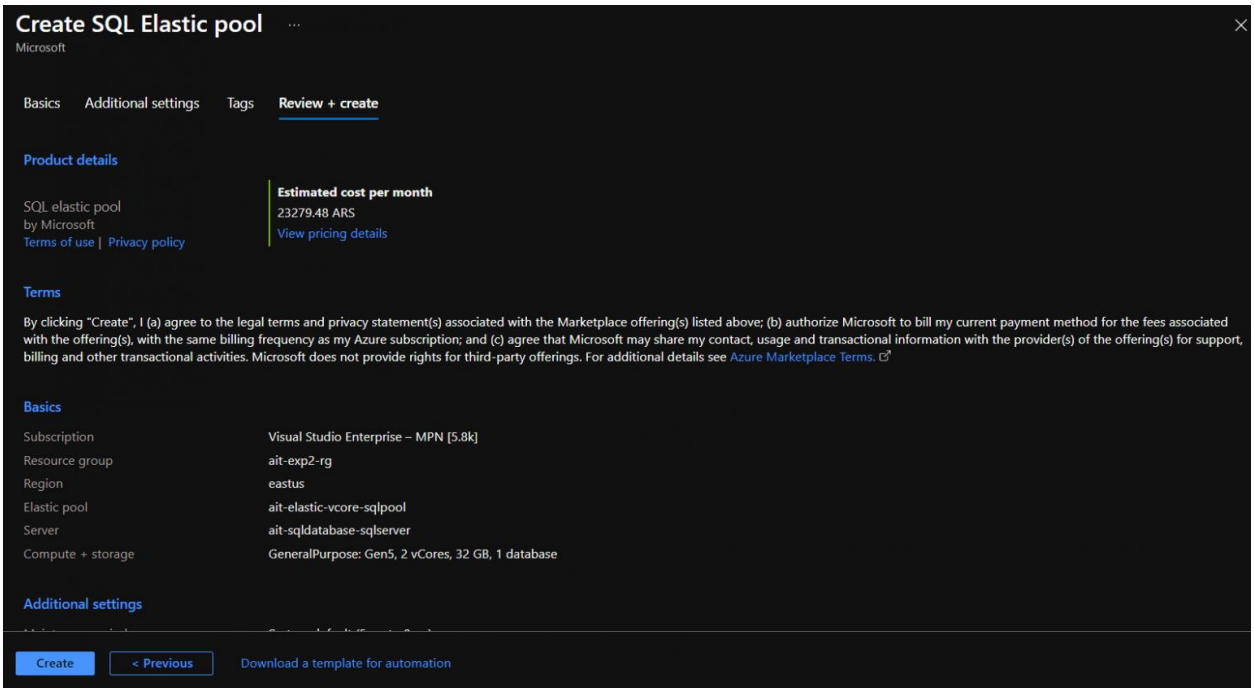
On the same Computing and Storage configuration screen, we will have a tab called “Databases”:



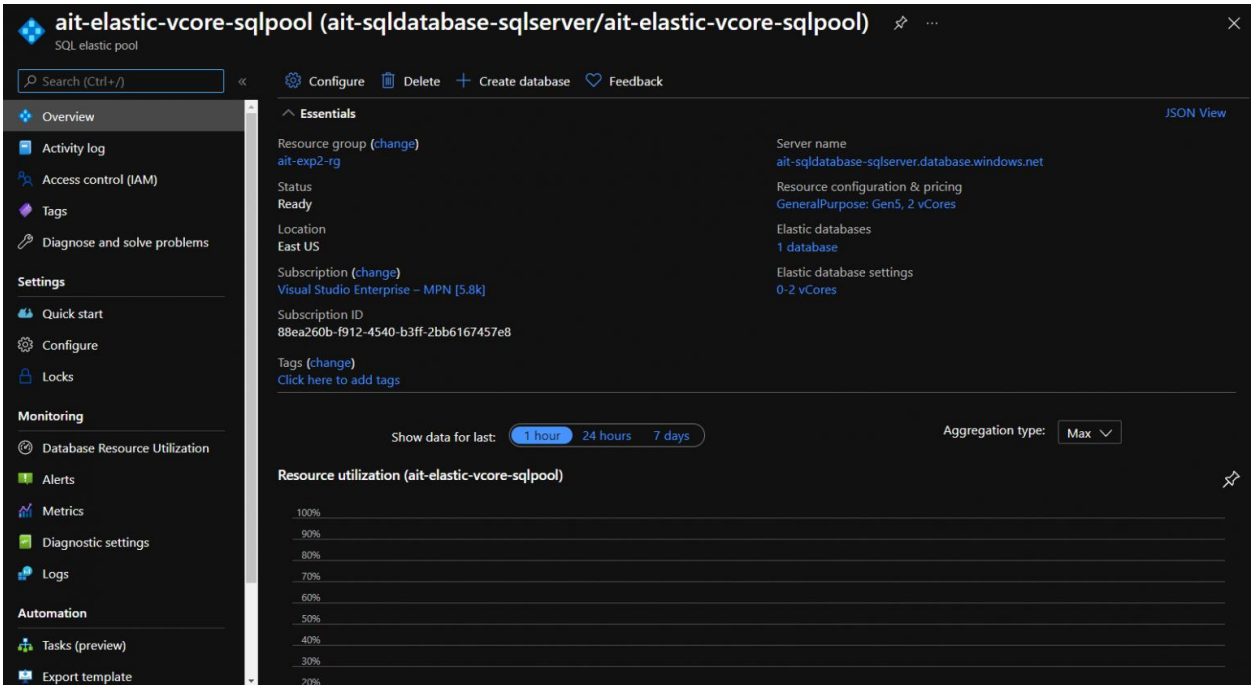
There we will select the "ait-sql-2-db" database to convert it and add it to the pool on creation:



In this way we can review the selected options and confirm the creation of the pool, including the inclusion of the second database within:



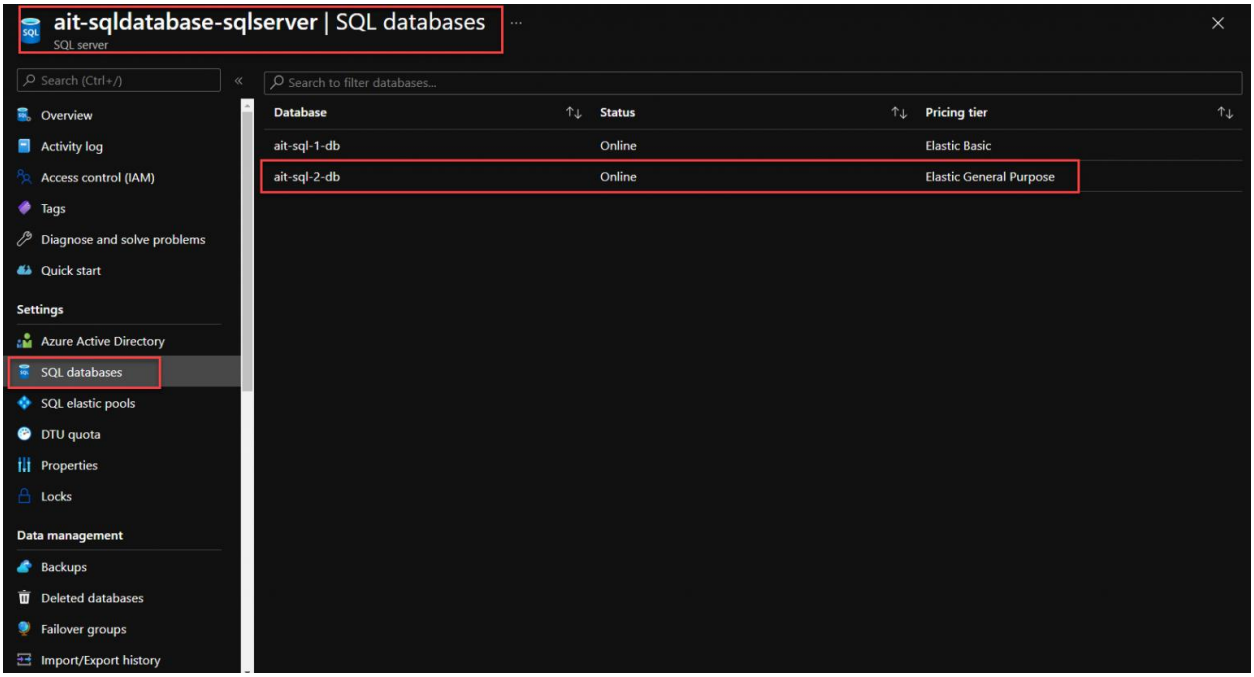
When the creation is finished, we will be able to view our new pool as a new resource:



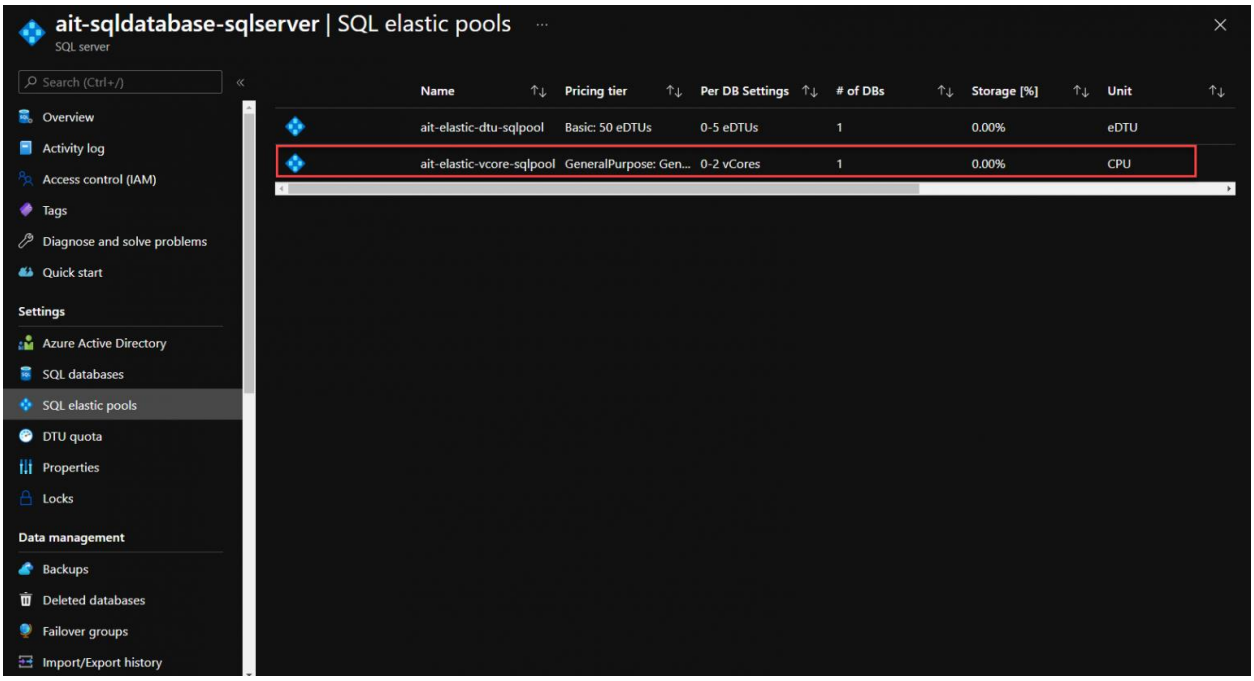
Let's validate the elastic group!

Elastic pool validation

If we enter our SQL Database Server, and go to the SQL databases part, we will verify that the second database is already part of the pricing tier of a pool, based on vCore:



In the same way, we can validate that in the SQL elastic pools section, within our SQL Database Server, we will find the second active pool:



You can try connecting from SQL Management Studio, or from Azure Data Studio: you will see that your base remains the same.

Replicas in Azure SQL Database

Before finishing this experience number 3 (intense! Right?), we are going to see one last topic: the possibility of carrying out a database replication to take advantage of availability between regions. And we will discover how easy it is to do it with Azure SQL Database.

What are replicas in Azure SQL Database?

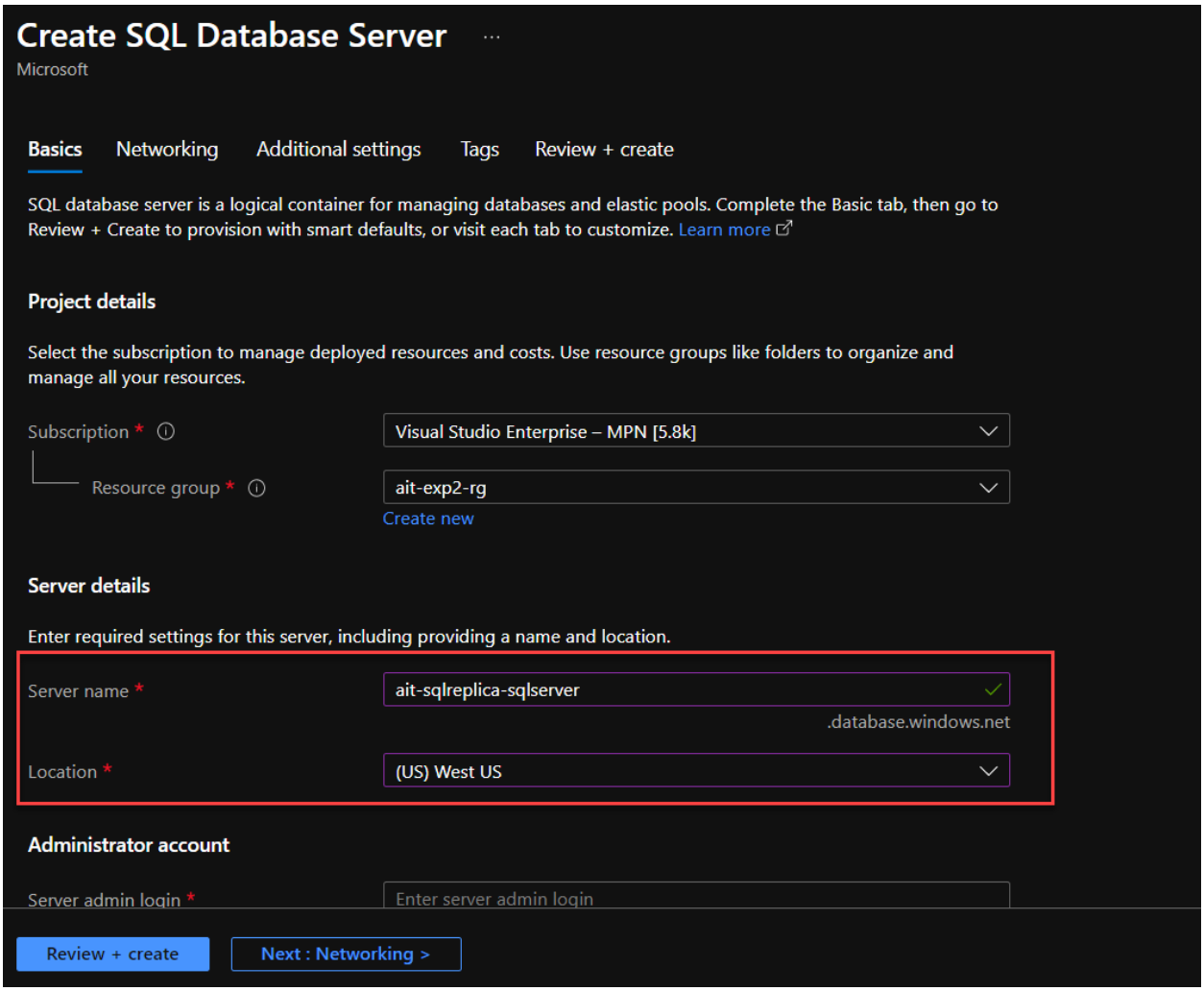
Azure SQL Database replicas allow us to have our data replicated in another region (or the same one, if we wish) with a read-only copy of our database.

Create an additional SQL Database Server (other region)

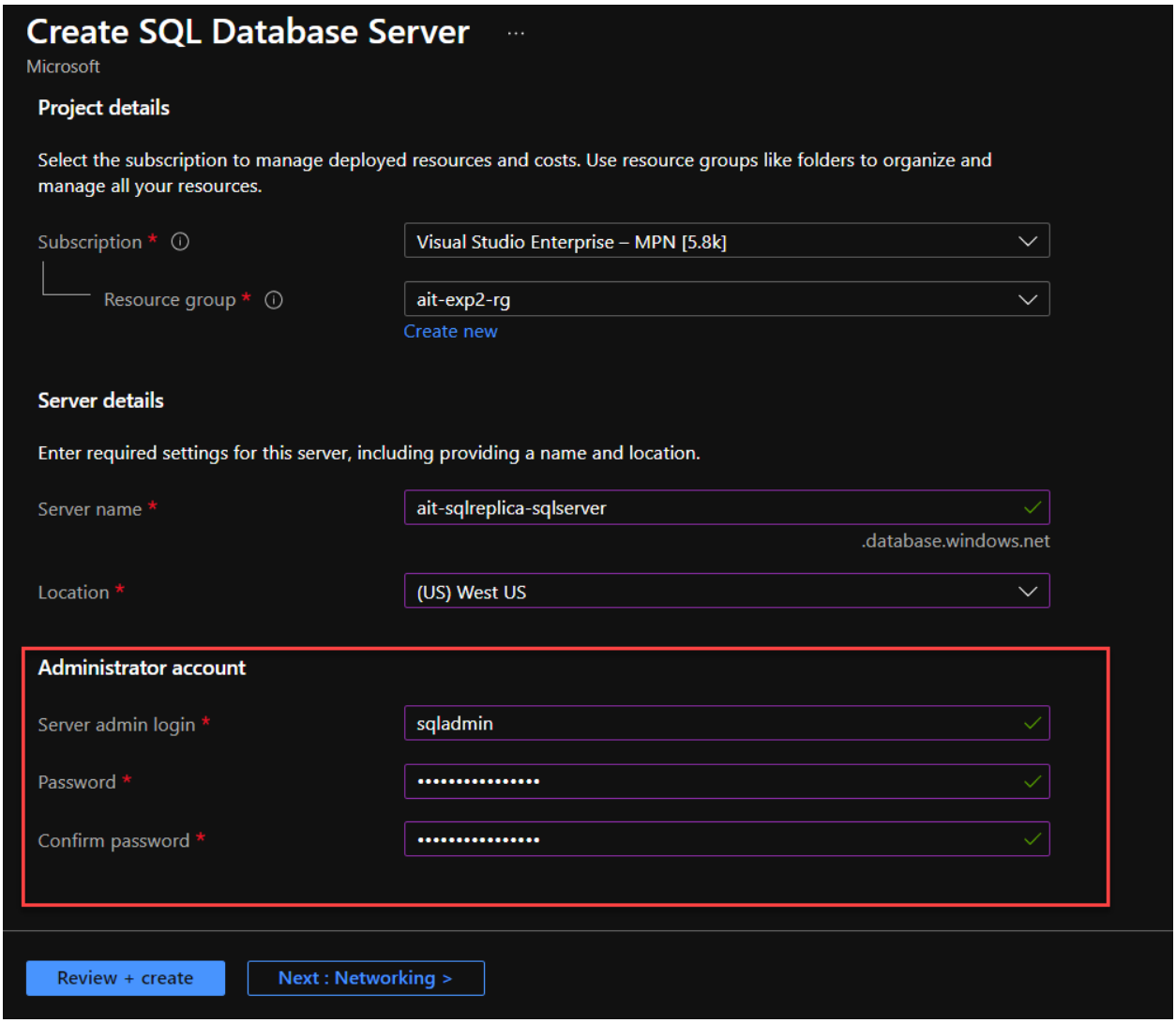
The first thing we need to do is create a SQL Database Server in a new region, for example WestUS. Remember that the region of the SQL Database Server determines the location of its databases.

We will create a new SQL Database Server with the following data:

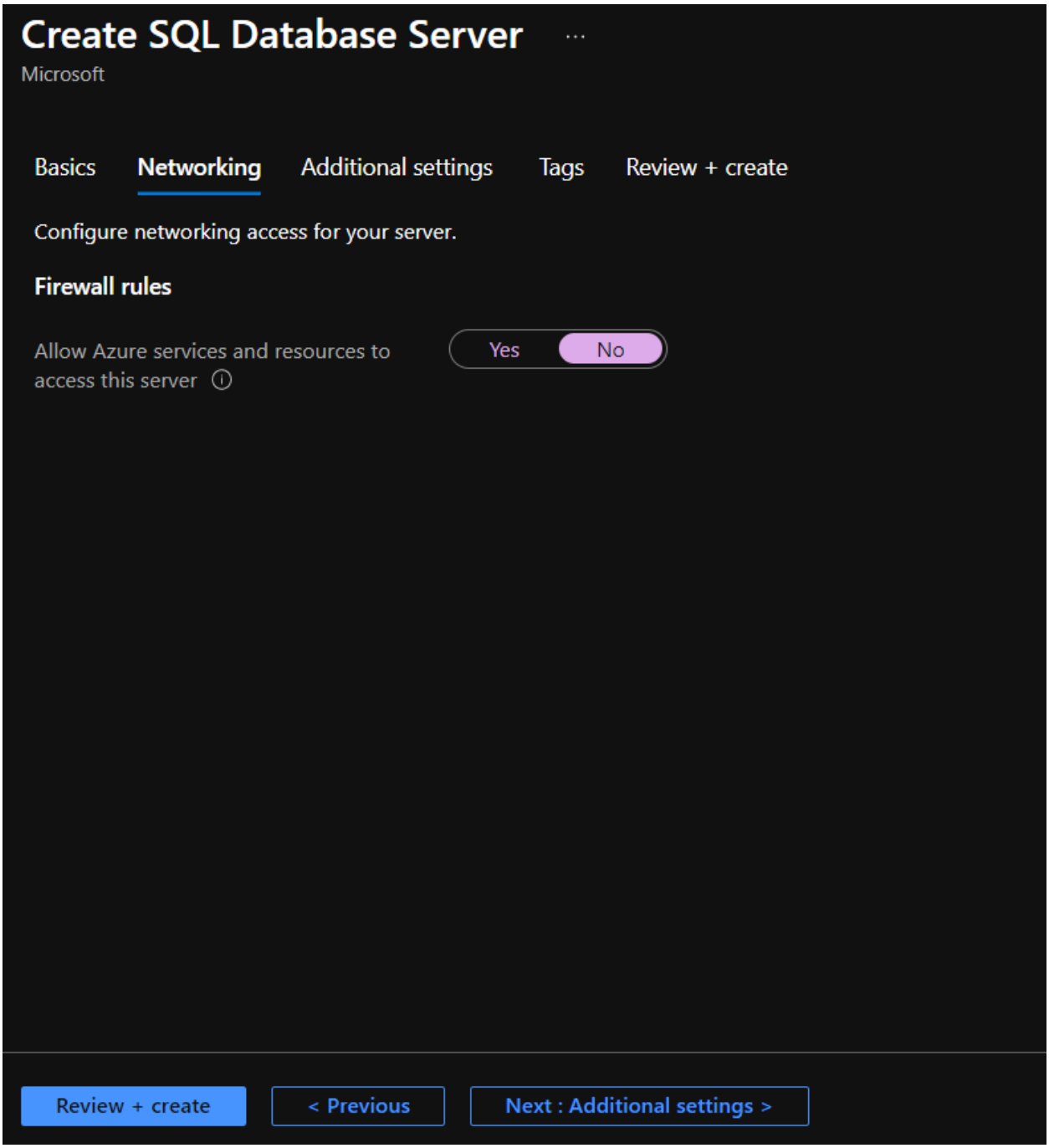
- Server name: ait-sqlreplica-<string>-sqlserver, where <yourstring> must be replaced with your own unique text and numbers (since this server name must be unique across all of Azure).
- Location: Westus



Server admin login and password: the one you choose, and as a suggestion enter “sqladmin” in the login.



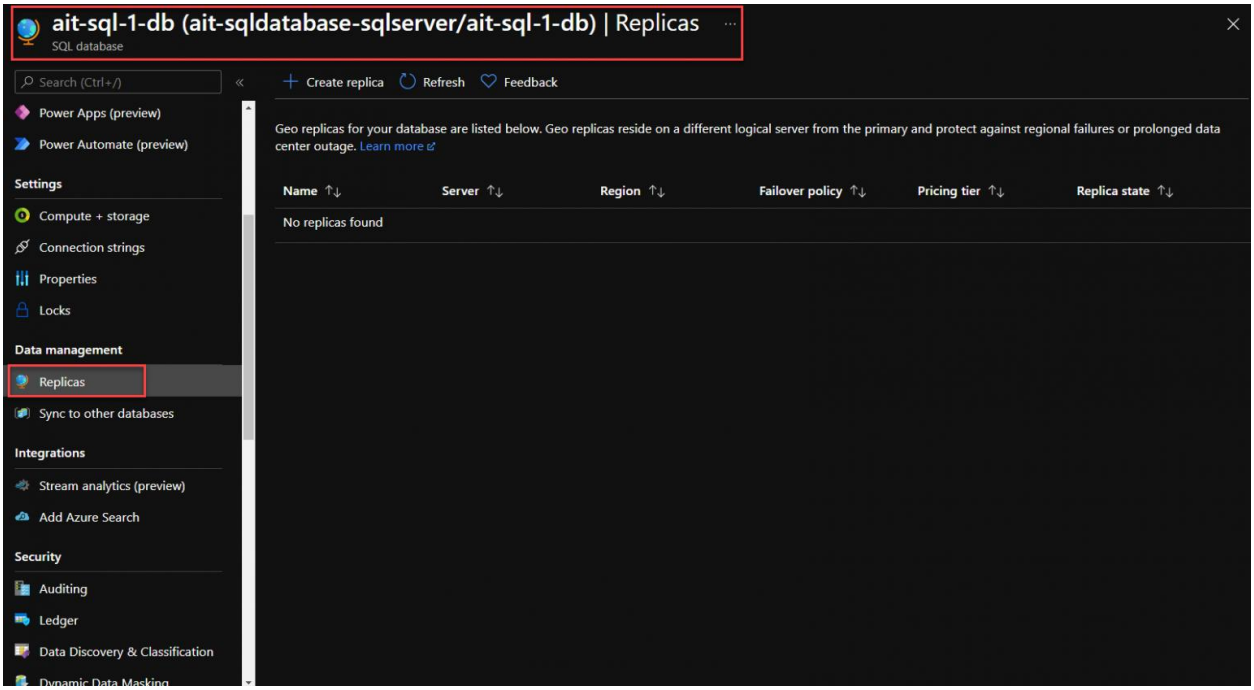
In relation to the Networking options, we will leave WITHOUT ACCESS to Azure services. In this way we can validate the independence of the service in relation to the replica itself, not the access:



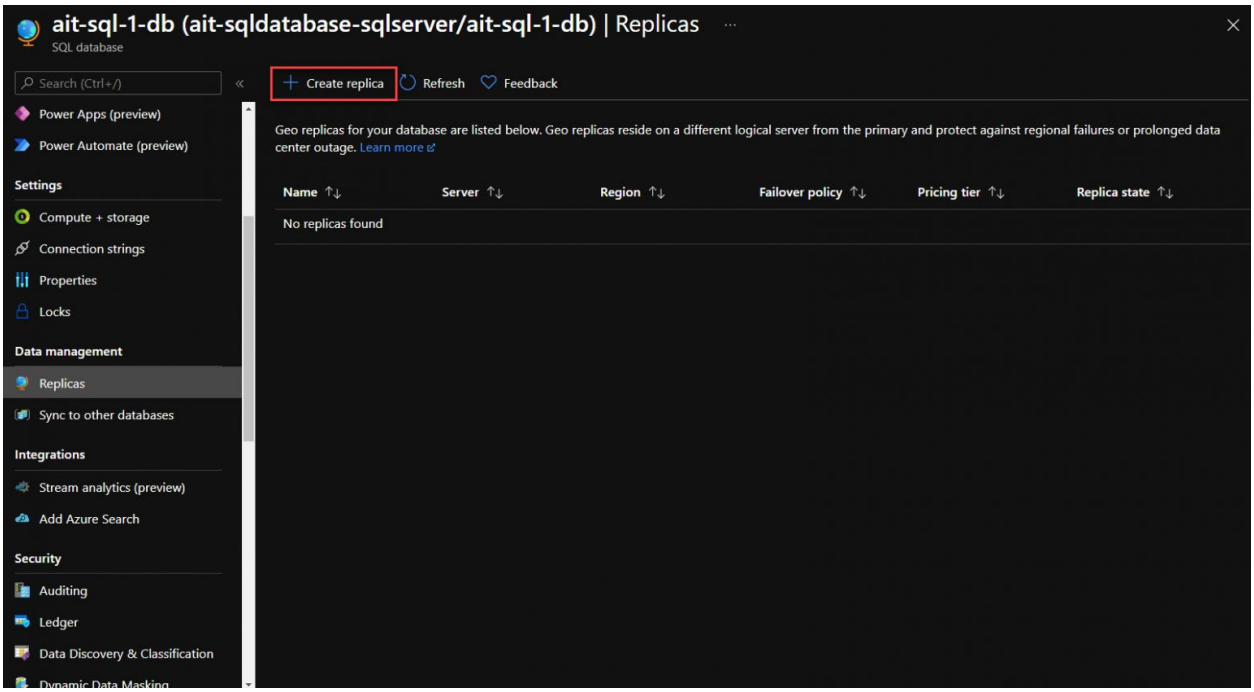
With the creation of our new SQL Database Server complete, we continue to move forward.

Create cross-region database replica

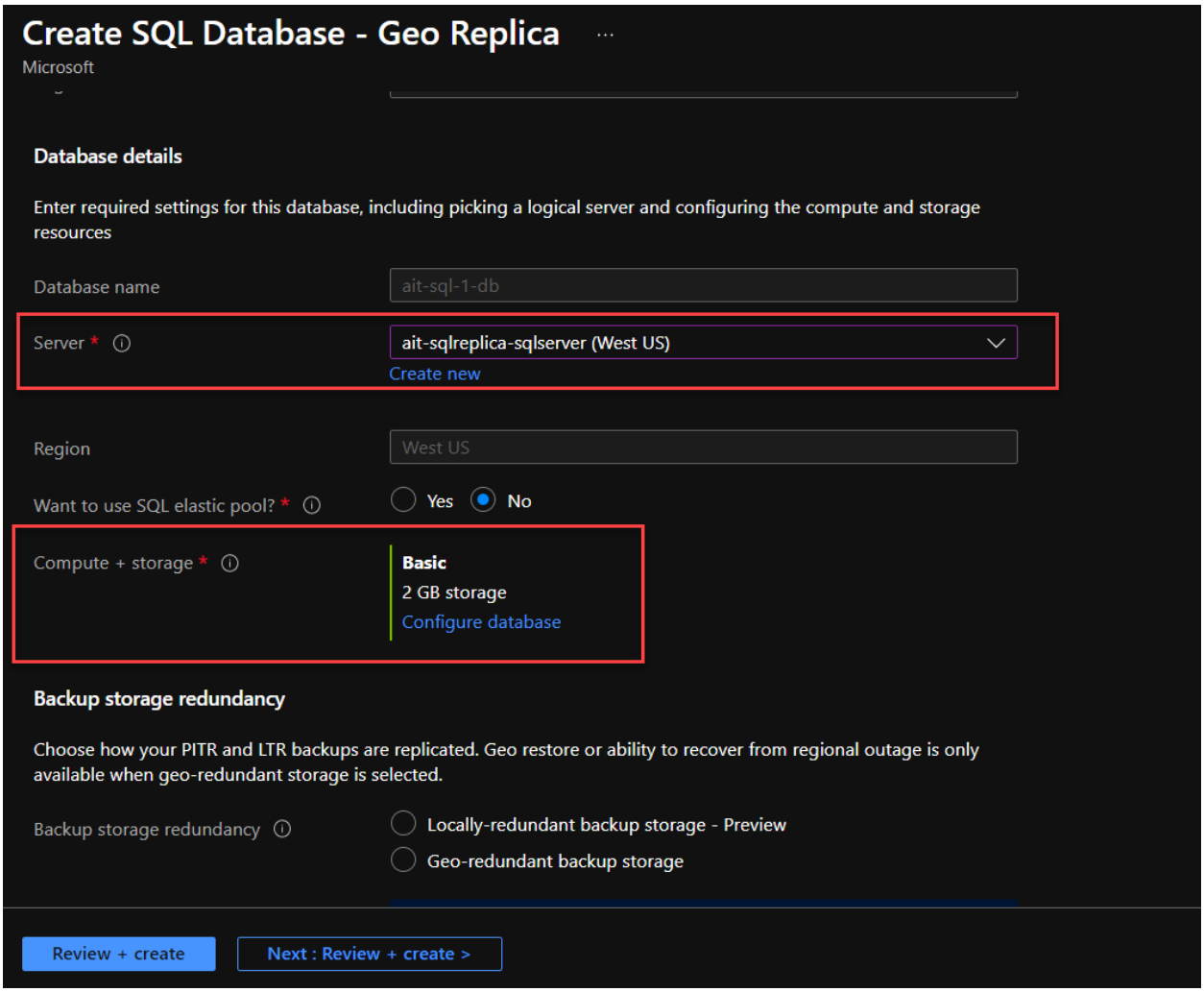
Within the database that we want to replicate (in this case the database is ait-sql-1-db) we go to the Data management section "Replicas":



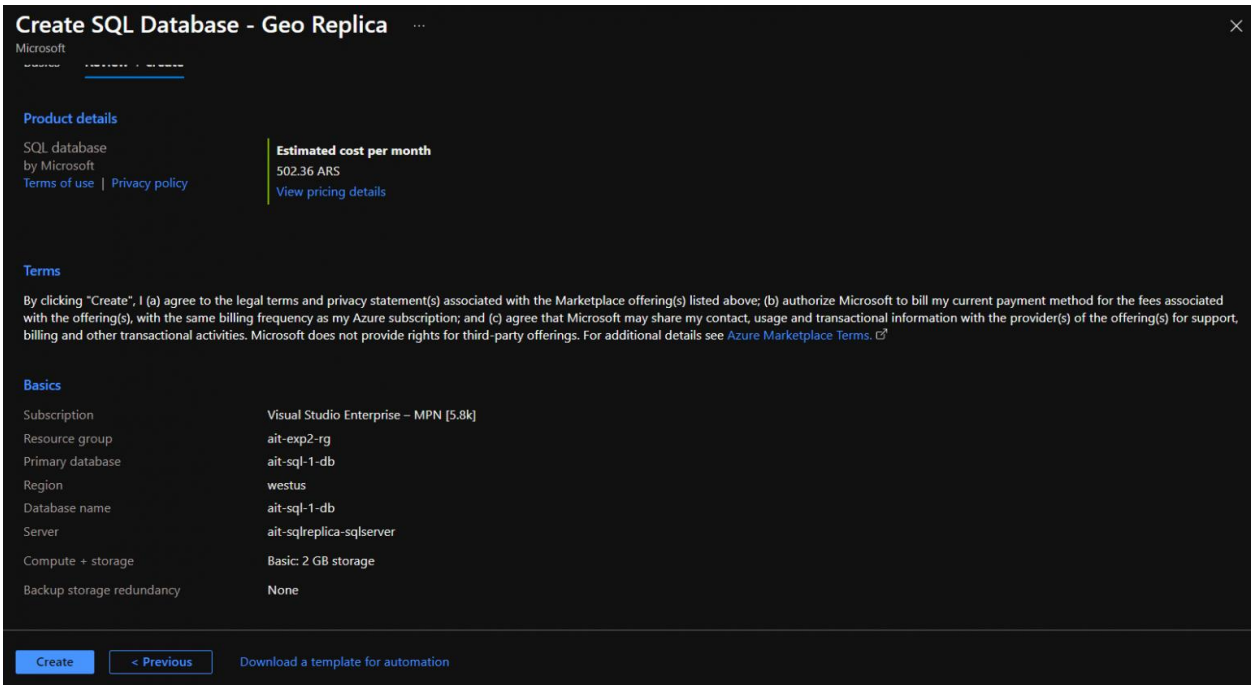
There we select the "Create replica" button:



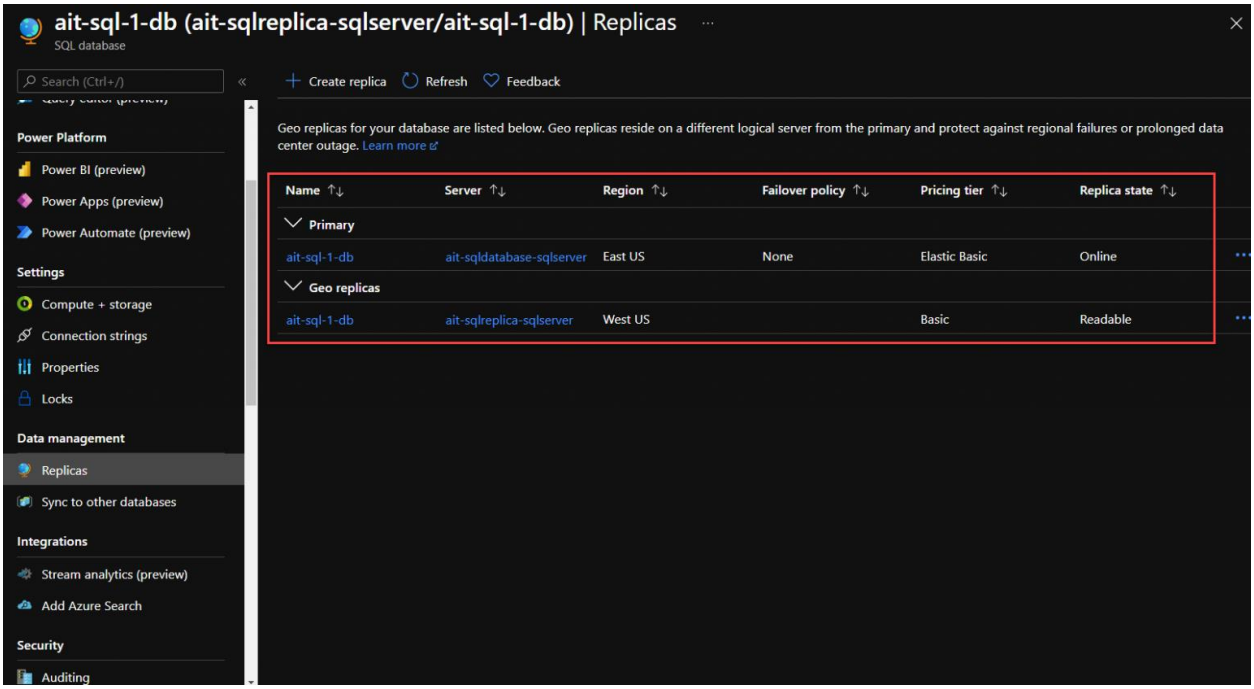
In the wizard, we will select the logical server that we want to use. In this case it is the recently created: ait-sqlreplica-sqlserver (West US). On the other hand, we must choose the “Compute + storage” options and preferences, and we can even configure aspects of an SQL elastic pool:



After having selected these options (we suggest that you choose them according to your criteria), we advance in the creation:



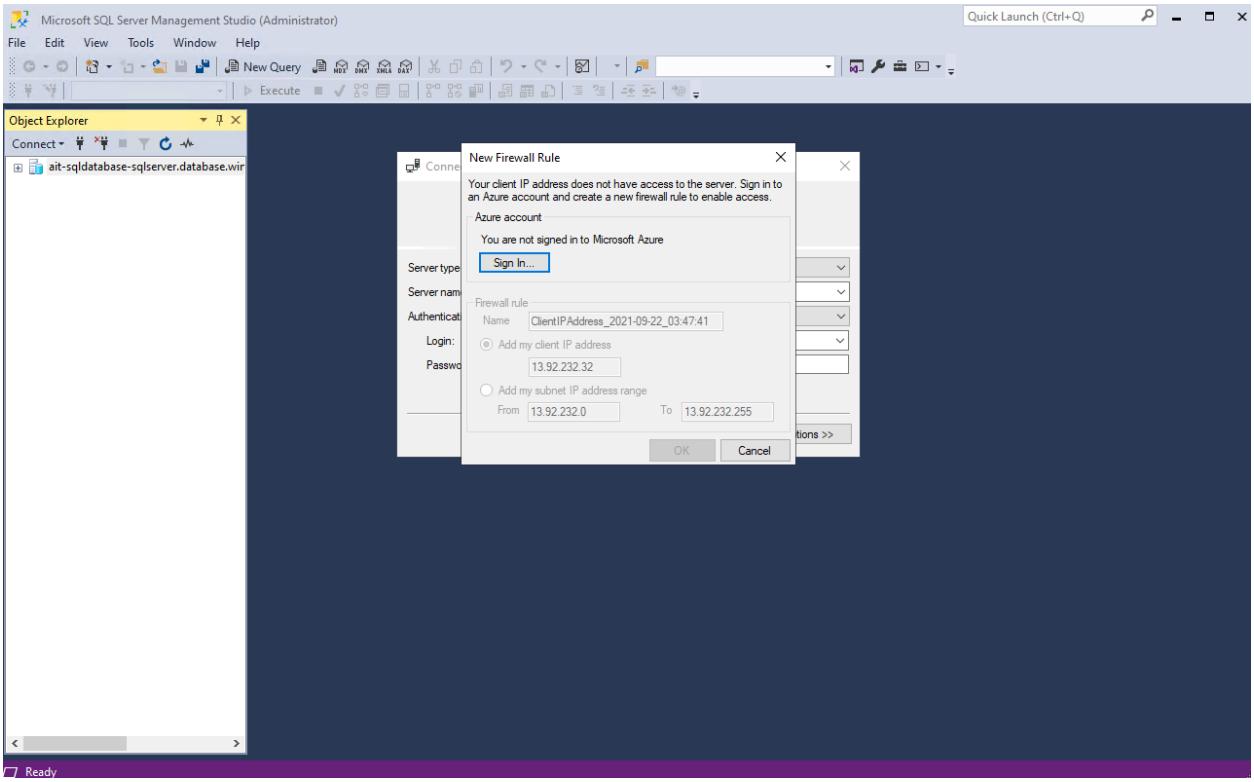
Once the replica has been generated, we are going to display them in the same section of Replicas within the ait-sql-1-db database:



Ready! Now we are going to test connecting and using the replica.

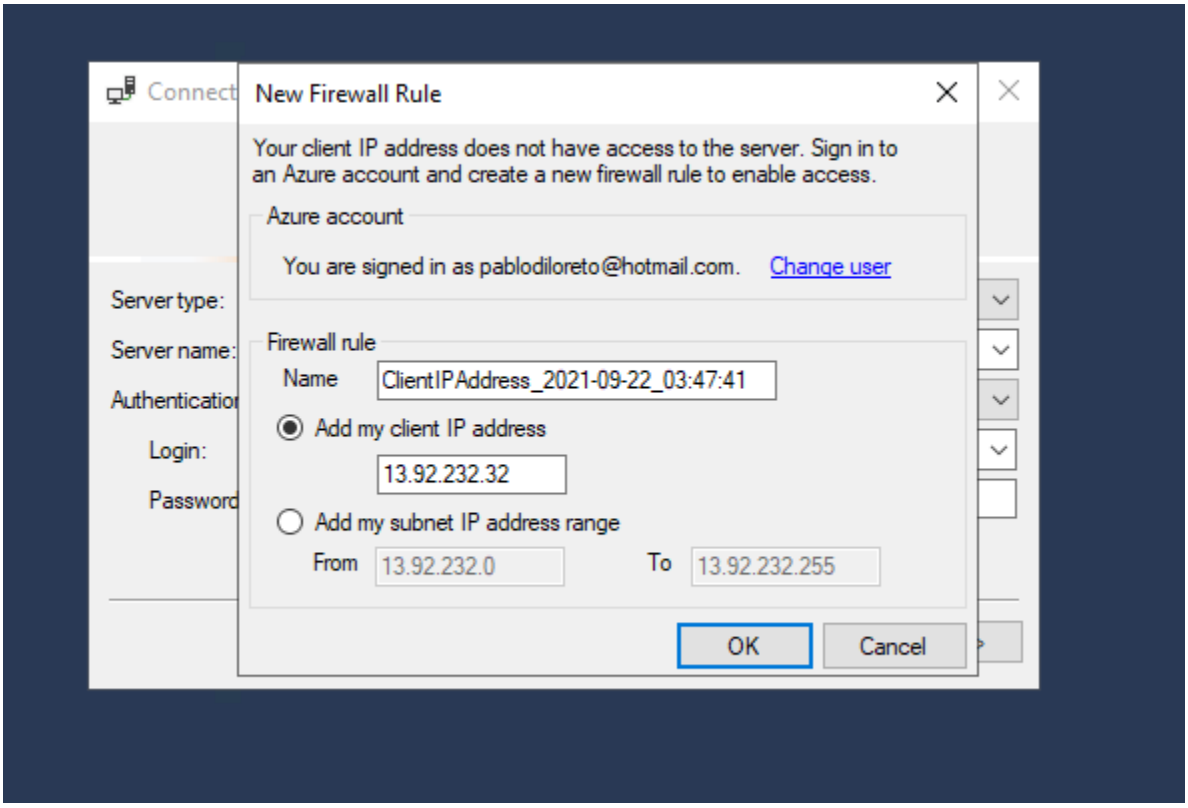
Connecting to the replica SQL Database Server

From the SQL Management Studio of our Azure team we try to enter using the replica server data, which we obtain from the Overview window (within the newly created logical server). We will find the following: we cannot connect.

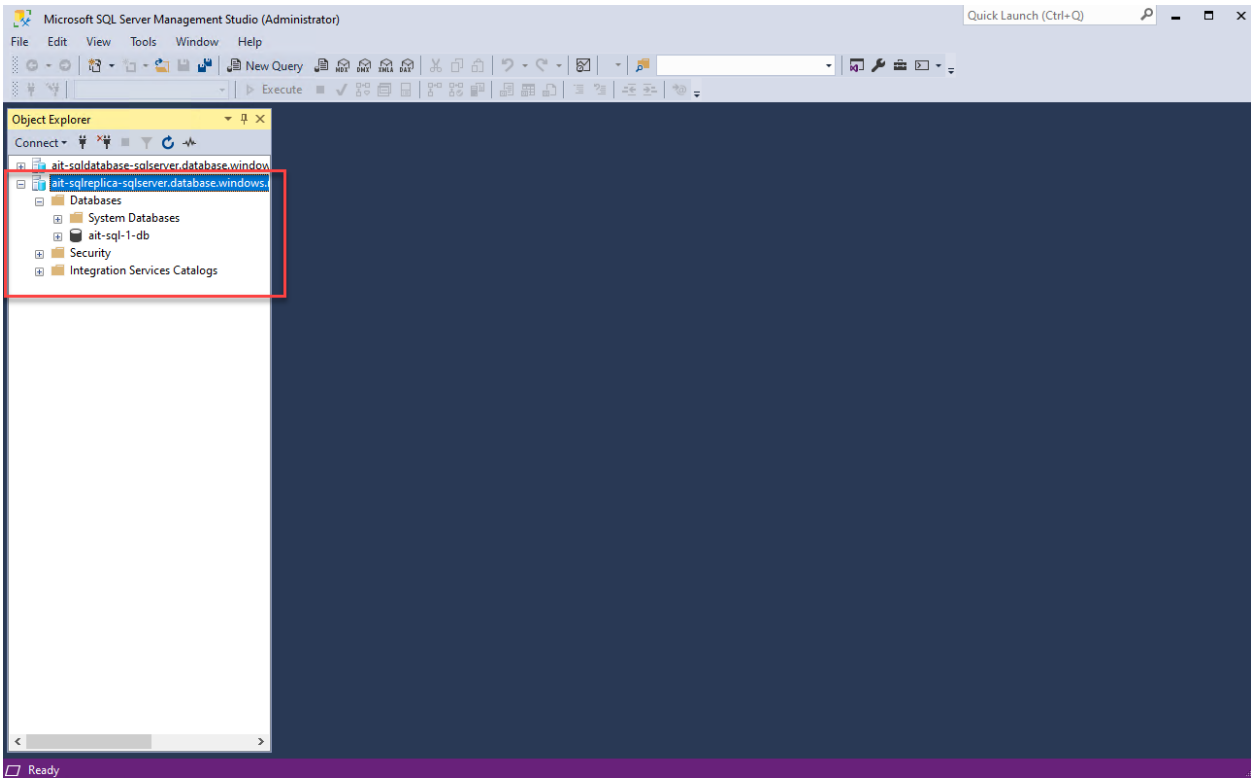


Why did this happen? If we remember, at the time of creating the new logical server we chose that said server not allow connections from Azure services. By not having Firewall exceptions either, we will not be able to establish a connection with it.

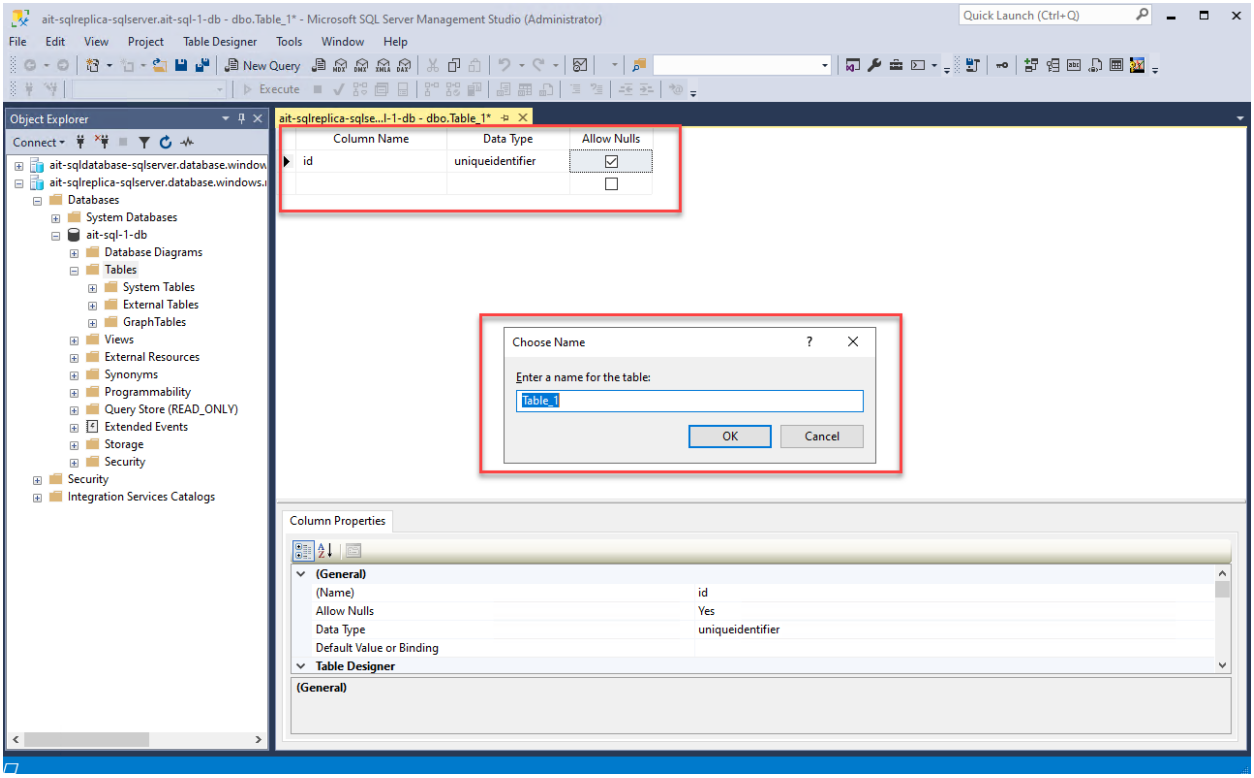
If we want, from SQL Management Studio itself, we can log in with our Azure user (as long as they have permission to modify the SQL Database Server resource) and we can add the rule with its IP exception from here:



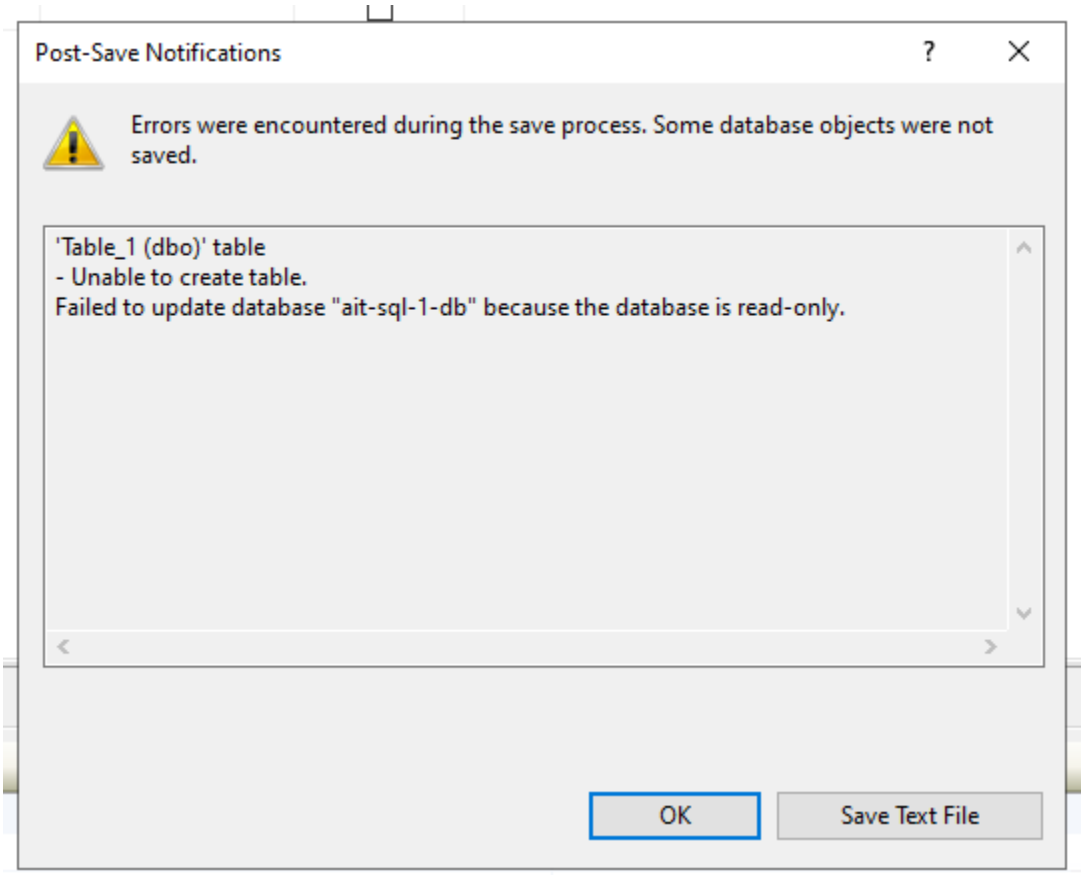
After having generated the exception, we can connect to the replica logical server:



If we try to generate a table on the replica server, we won't be able to:

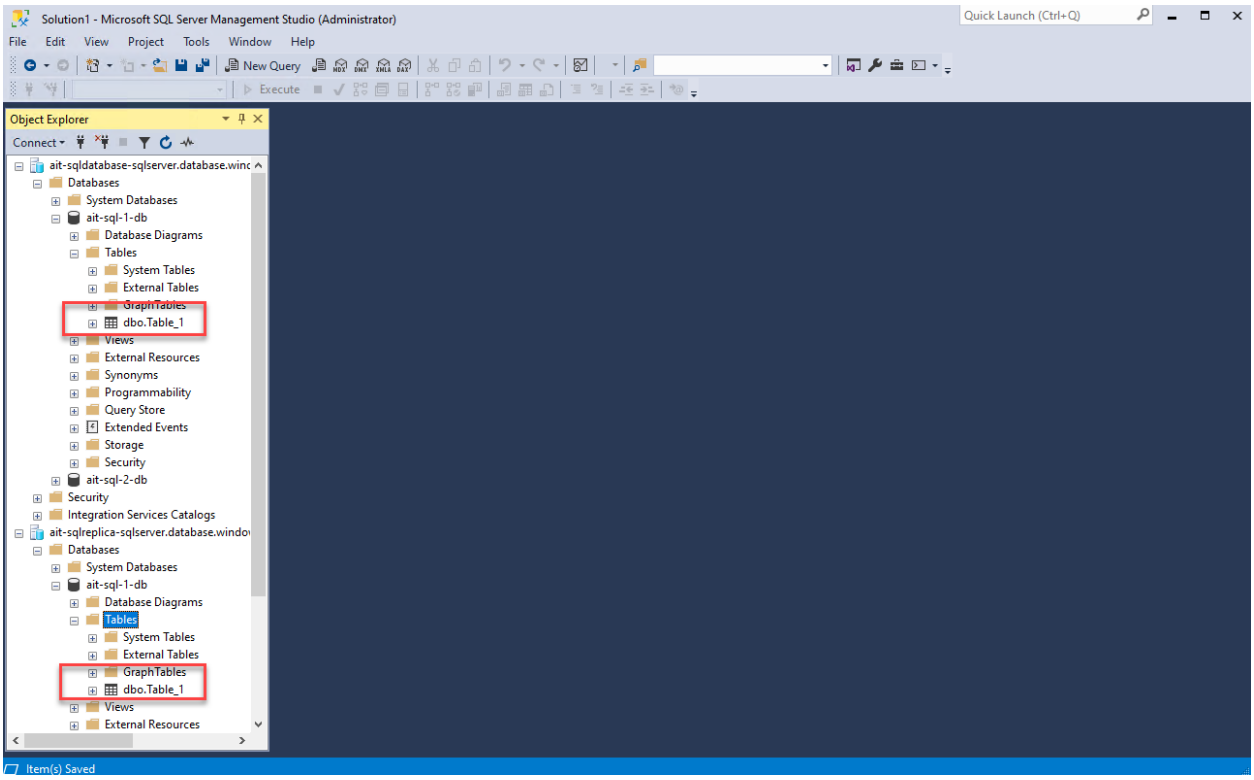


This is because that server is read-only:



Data replication validation

We are going to do the same thing we tried to do before, but now on the main server, ait-sql-1-db database. We will notice that we can create the table, and that in a few seconds it is already replicated on the West US server:



Finished experience! Now all that remains is to validate some knowledge :-).

Exercise 4: SQL Managed Instance

Introduction to the exercise

In this section we will do the fourth experience together. This experience requires you to have a Microsoft Azure subscription set up and available and consumes any credits you have on it. Please review this topic to avoid charges on subscriptions where you don't want them to exist.

In this experience we will:

- Learn more about Azure SQL Managed Instance.
- Create a SQL Managed Instance (takes approx 6 hours).
- Create databases in our SQL Managed Instance.
- This experience is going to be intense! Let's keep going!

About Azure SQL Managed Instance

Azure SQL Managed Instance is an intelligent and scalable cloud database service that combines the best compatibility with the SQL Server database engine and all the benefits of a fully managed and always-on platform-as-a-service.

SQL Managed Instance is nearly 100% compatible with the latest SQL Server (Enterprise Edition) database engine, providing a native virtual network (VNet) implementation to address common security issues, and a favorable business model for existing SQL Server customers.

Purchasing Models

The purchase model available in SQL Managed Instance is vCore based. This model gives you flexibility, control, transparency, and the ease of moving your workload needs from on-premises to the cloud. This model allows you to change compute, memory, and storage resources based on the needs of your workload. The vCore model also allows you to enjoy up to 55% savings with the Azure Hybrid Benefit for SQL Server.

Service levels

SQL Managed Instance is available in two levels of service:

- General Purpose – Designed for applications with common performance and I/O latency requirements.
- Business Critical – Designed for applications with low I/O latency requirements and minimal underlying workload impact from maintenance operations.

Both service levels guarantee 99.99% availability and allow you to select storage size and throughput independently.

Create SQL Managed Instance

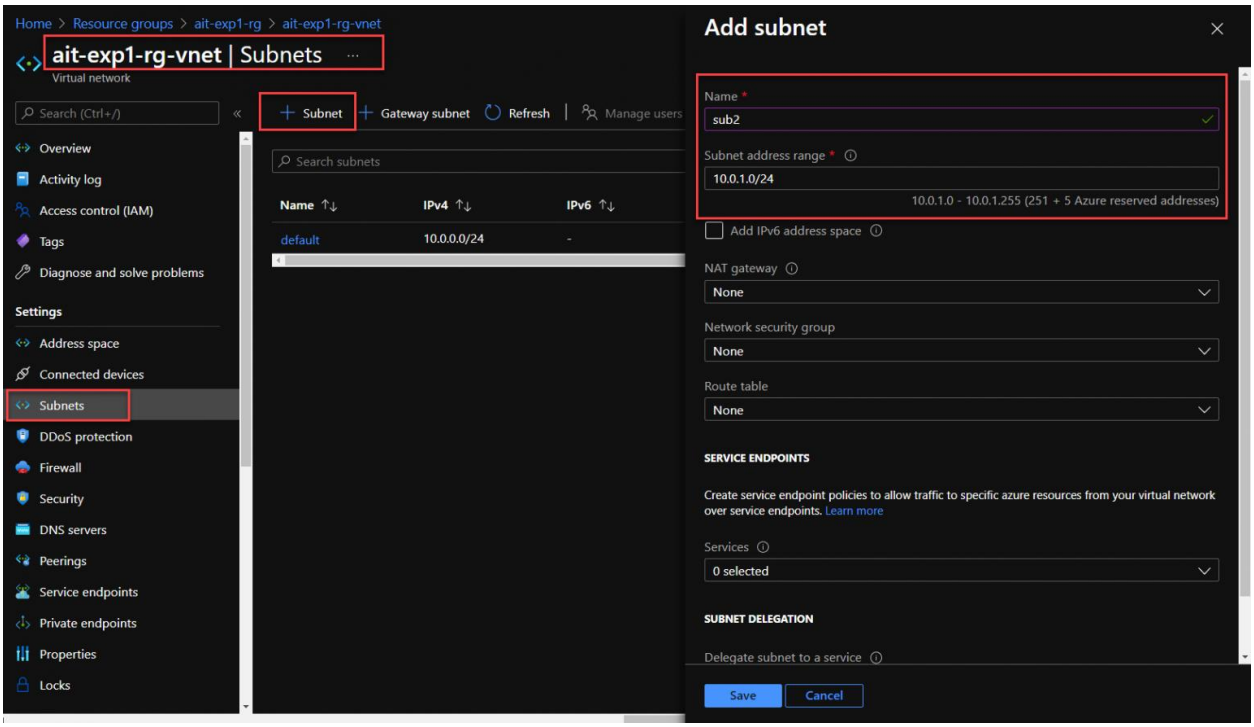
Let's create a SQL Managed Instance. It is important that you take into account that the creation of this type of resource can take up to 6 hours from the start. In addition, this type of resource has a very high economic value, so when you start this practice, the idea is that you complete it in the next few days (and if you are in a free trial account, do not let more than 2 or 3 days for this). If you are not sure about finishing it yet, it is preferable that you delay its creation.

Also, in TRIAL accounts you may find yourself with some limitations in the number of vCores. If this happens to you, ask your instructor for assistance.

Subnet Creation

The first thing we are going to do is create an additional subnet in the existing virtual network created in experience 1. We must do this because SQL Managed Instance requires a special subnet for exclusive use.

We are going to go to our network called (if you respected the proposed nomenclature) “ait-exp1-rg-vnet” and we will click on Subnets:

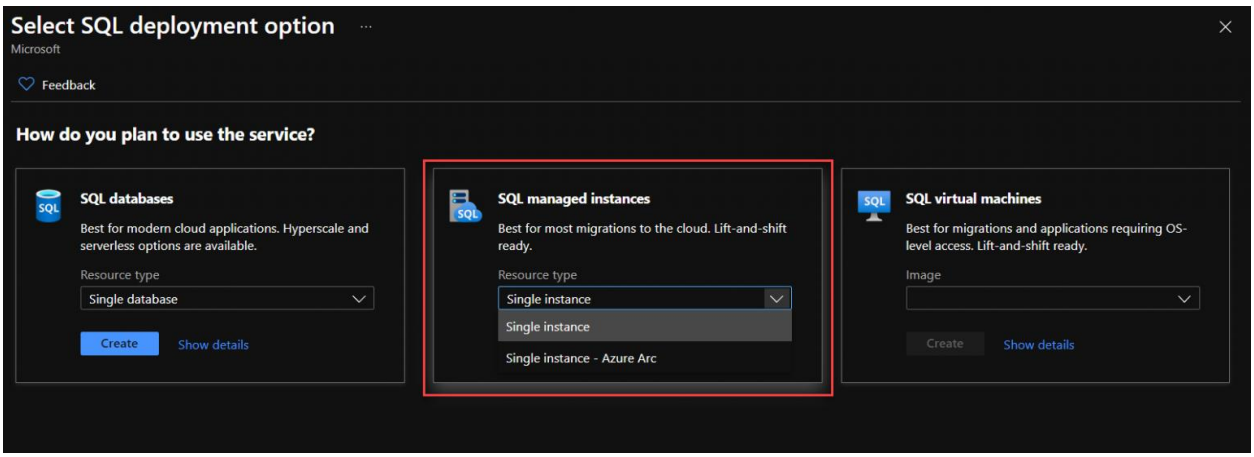


There we will click on “+ Subnet” to generate a new subnet. The data for this new subnet (if you followed our directions for network scope) is 10.0.1.0/24. What name you can give the new subnet is up to you, and we suggest sub2.

Once you have completed the creation of the new subnet, you can continue.

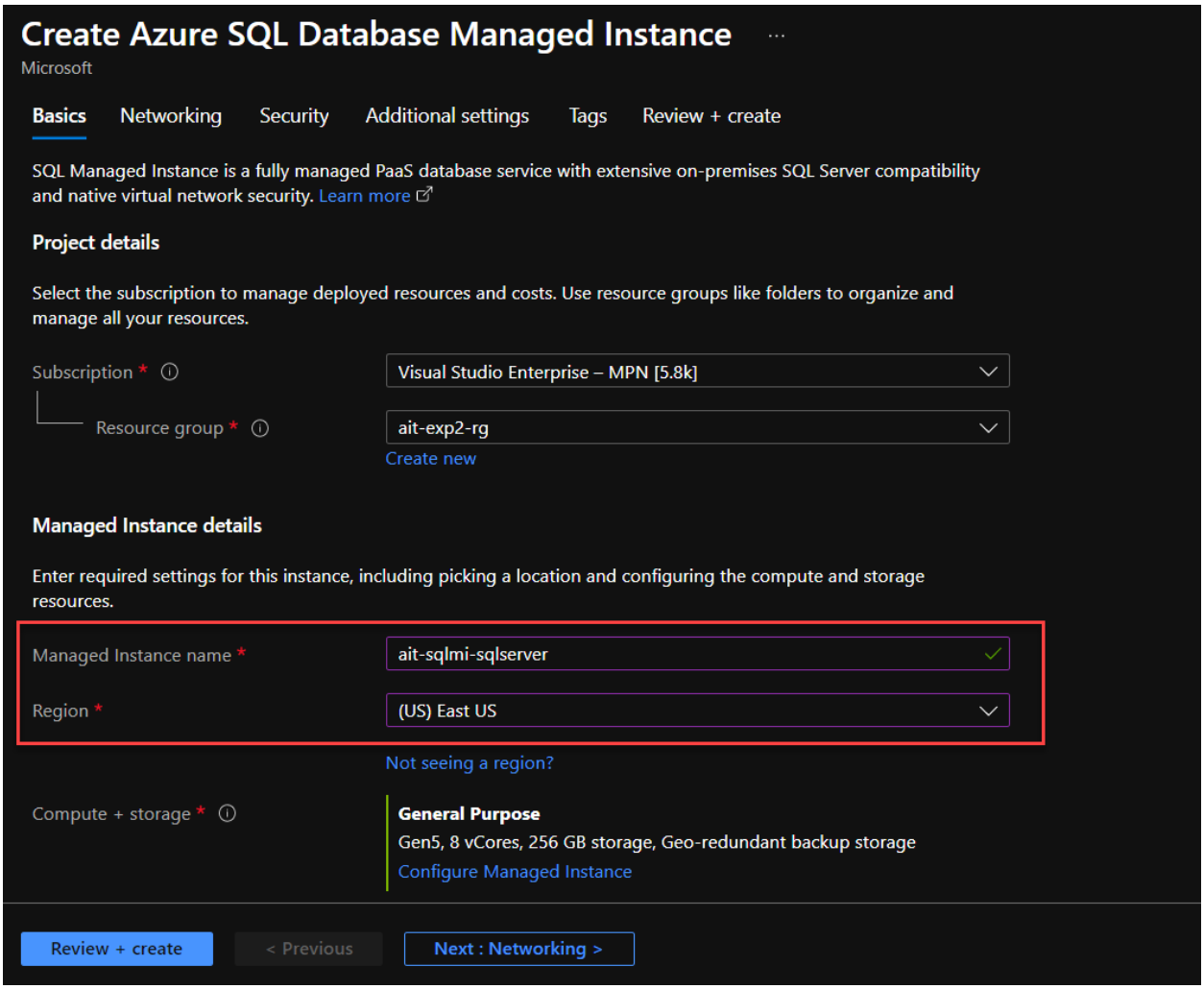
Create SQL Managed Instance

Let's create a new resource group called "ait-exp4-rg#. There we will create a new resource of type “Azure SQL” (by now you should know how to get to the following window):

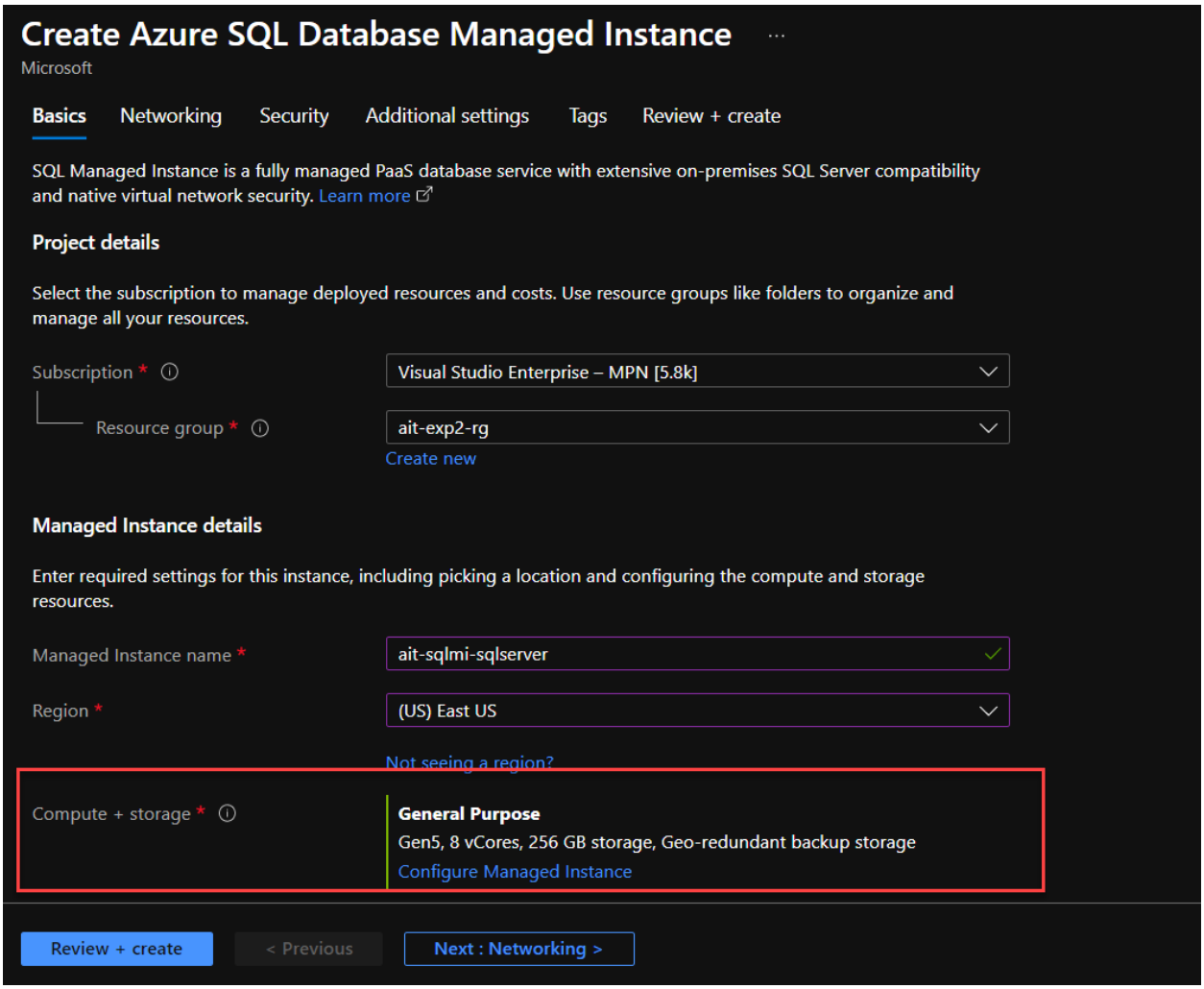


There we are going to select Single instance of “SQL Managed Instances” and we will complete the basic data such as:

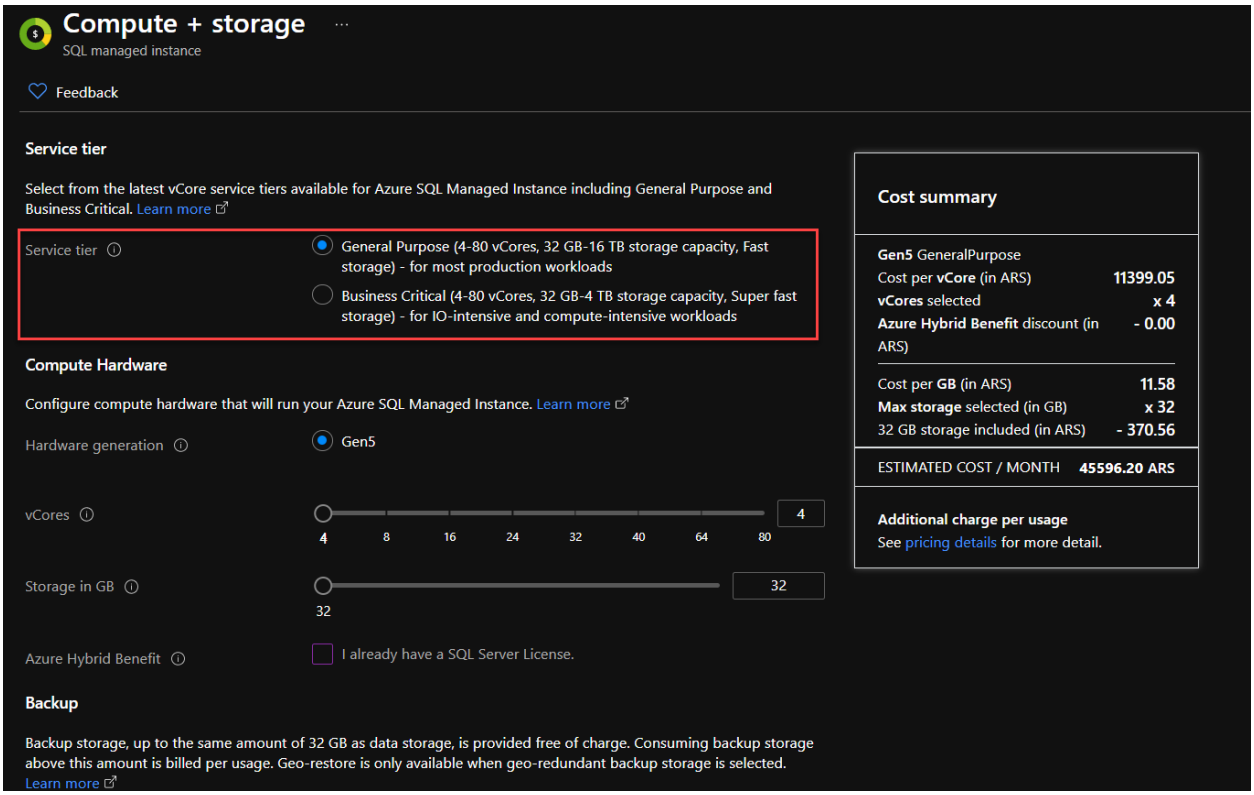
- Name: ait-sqlmi-<string>-sqlserver (where <string> are letters and numbers proposed by you).
- Region: eastus



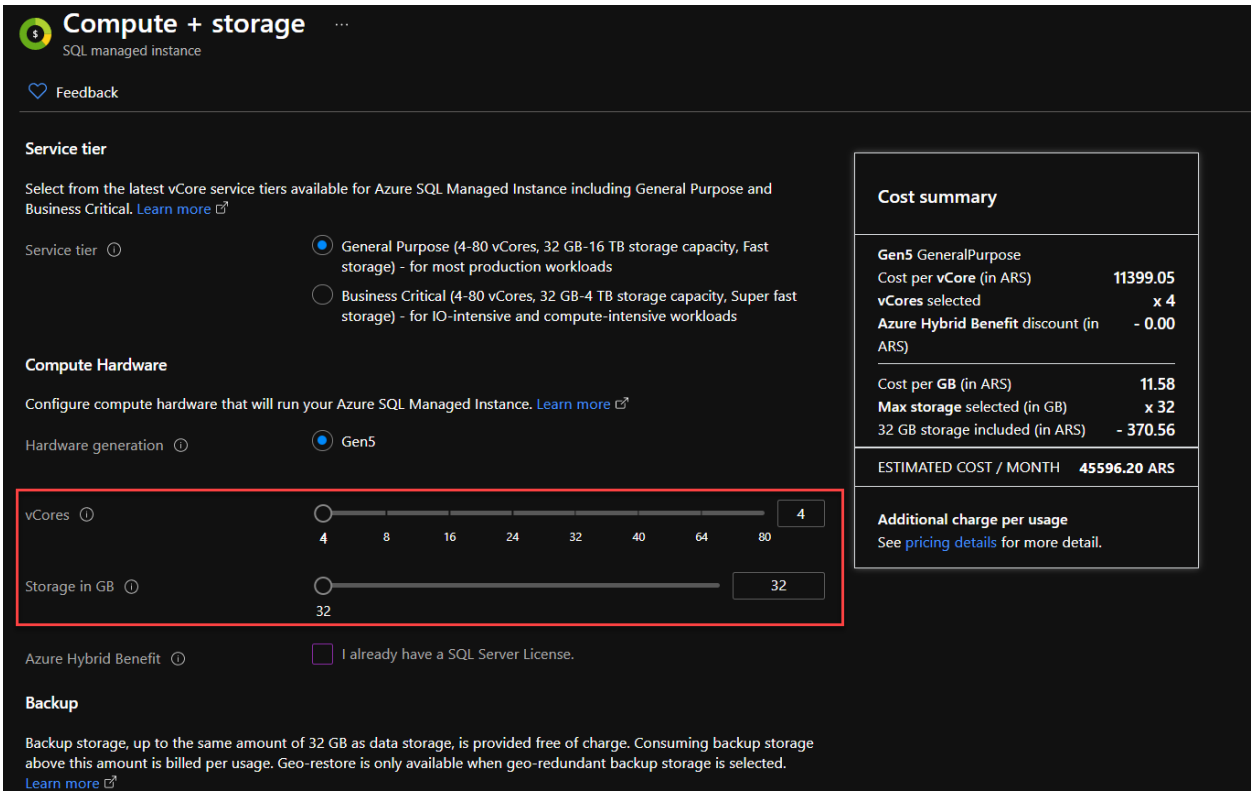
Then we need to configure the Compute + storage aspects. To do this we will click on "Configure Managed Instance":



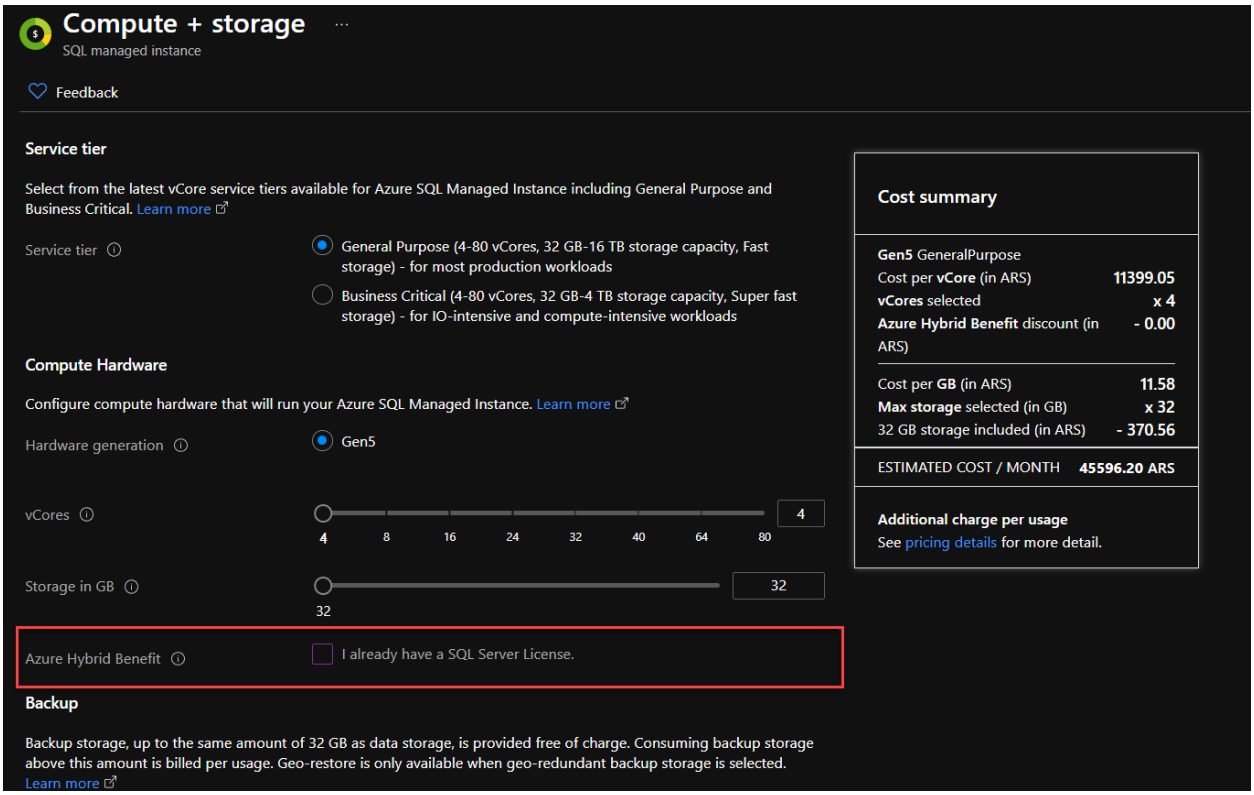
There we will notice that we can select between two levels of service: General Purpose and Business Critical. For this experience we will stay with “General Purpose”:



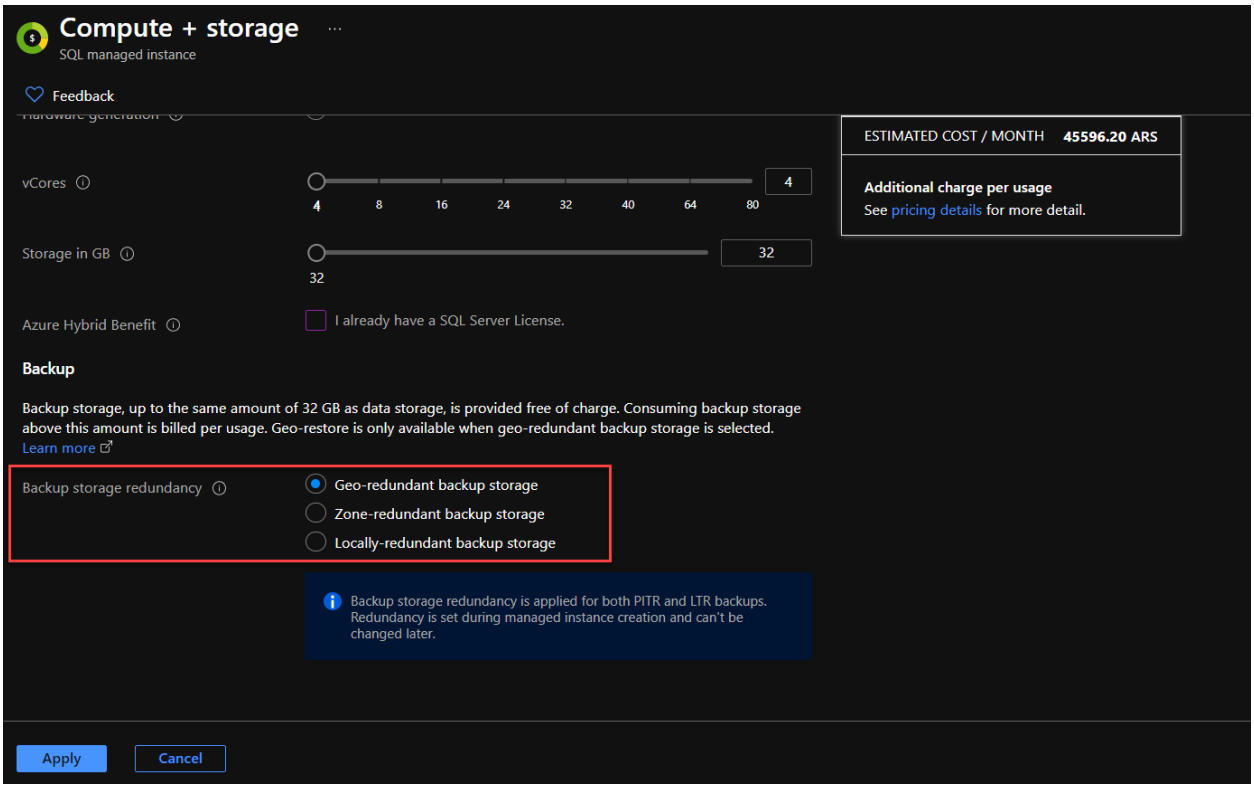
Note that we can configure the aspects of vCores and Storage in GB. For our experience we will select 4 vCores and 32 GB:



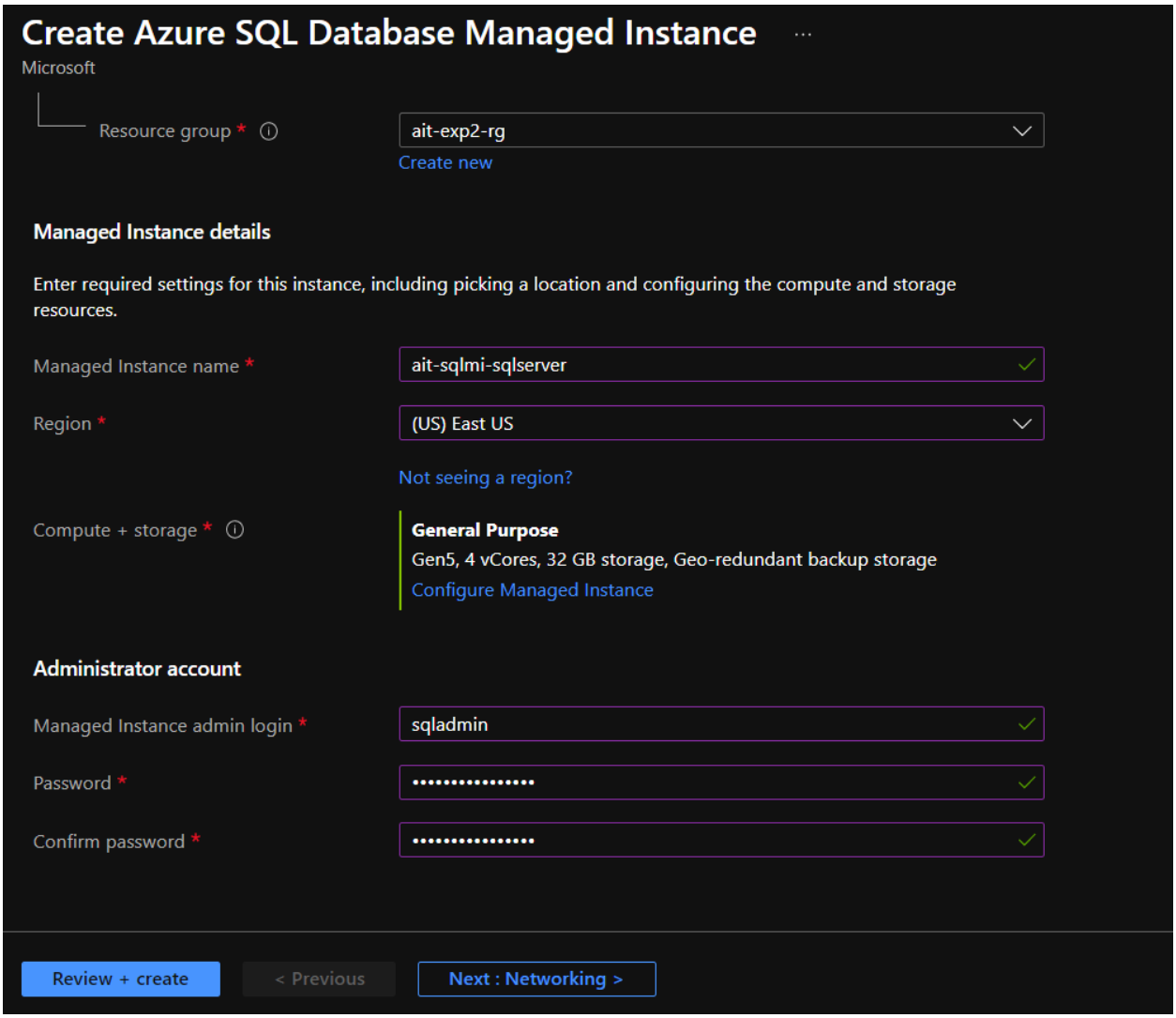
It is important to note that we have the possibility of applying to hybrid benefits, something that reduces the final cost of the service since we are reusing licenses that we already have purchased (if applicable):



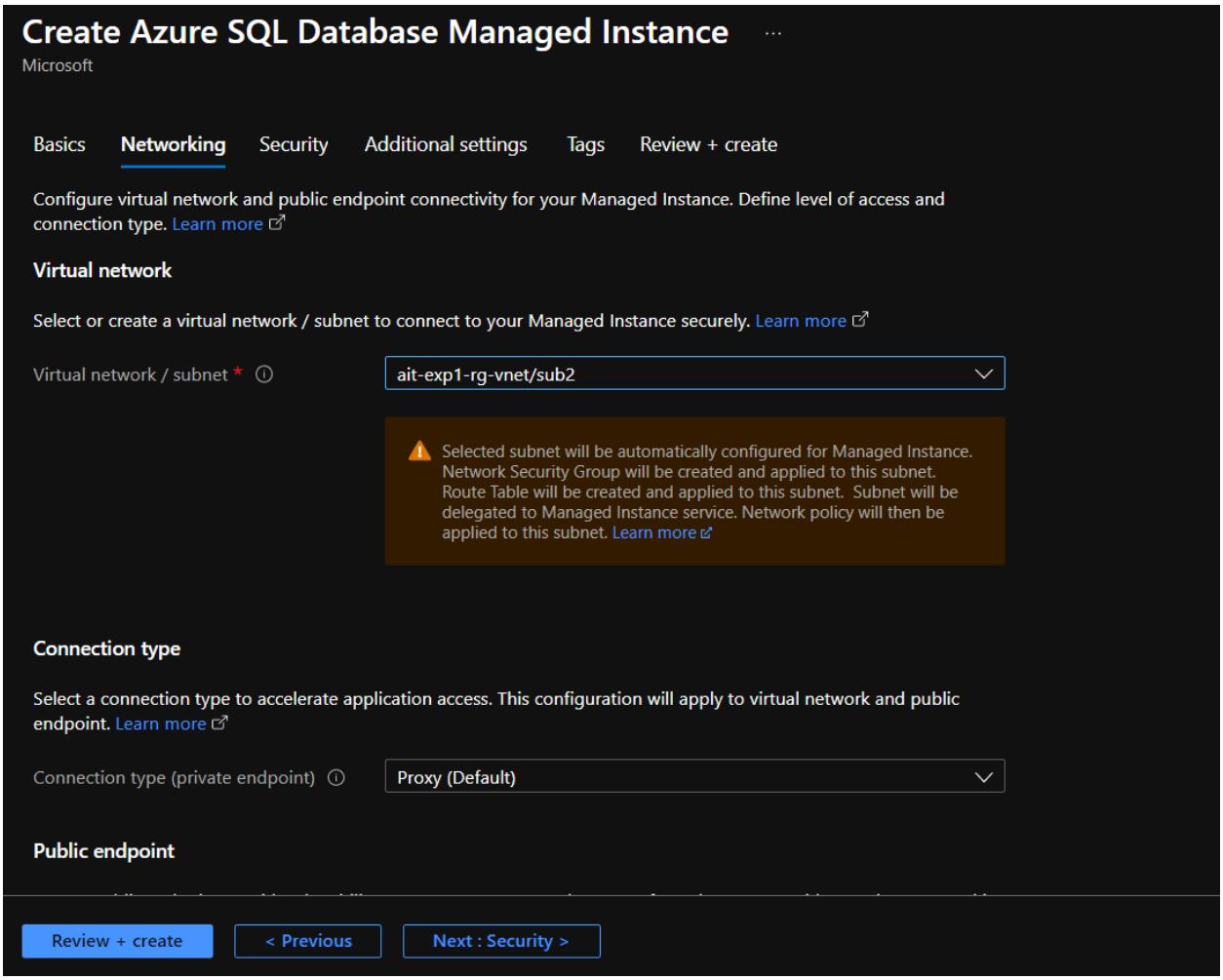
Finally, we can configure backup aspects and its redundancy in terms of storage:



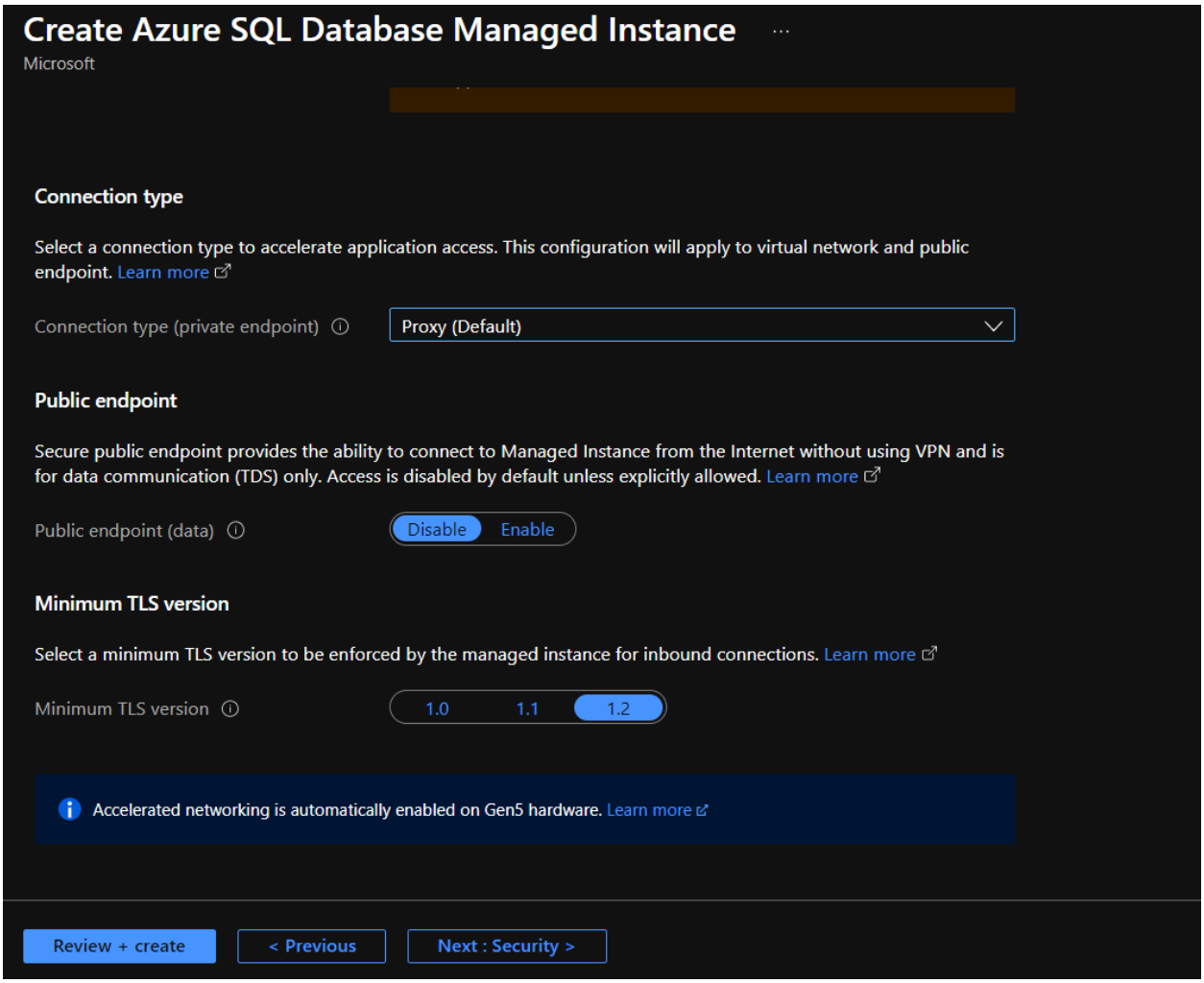
Once all these aspects are confirmed, we must enter a username (admin login) and a password twice. This user will be the one that allows us to enter our SQL Managed Instance:



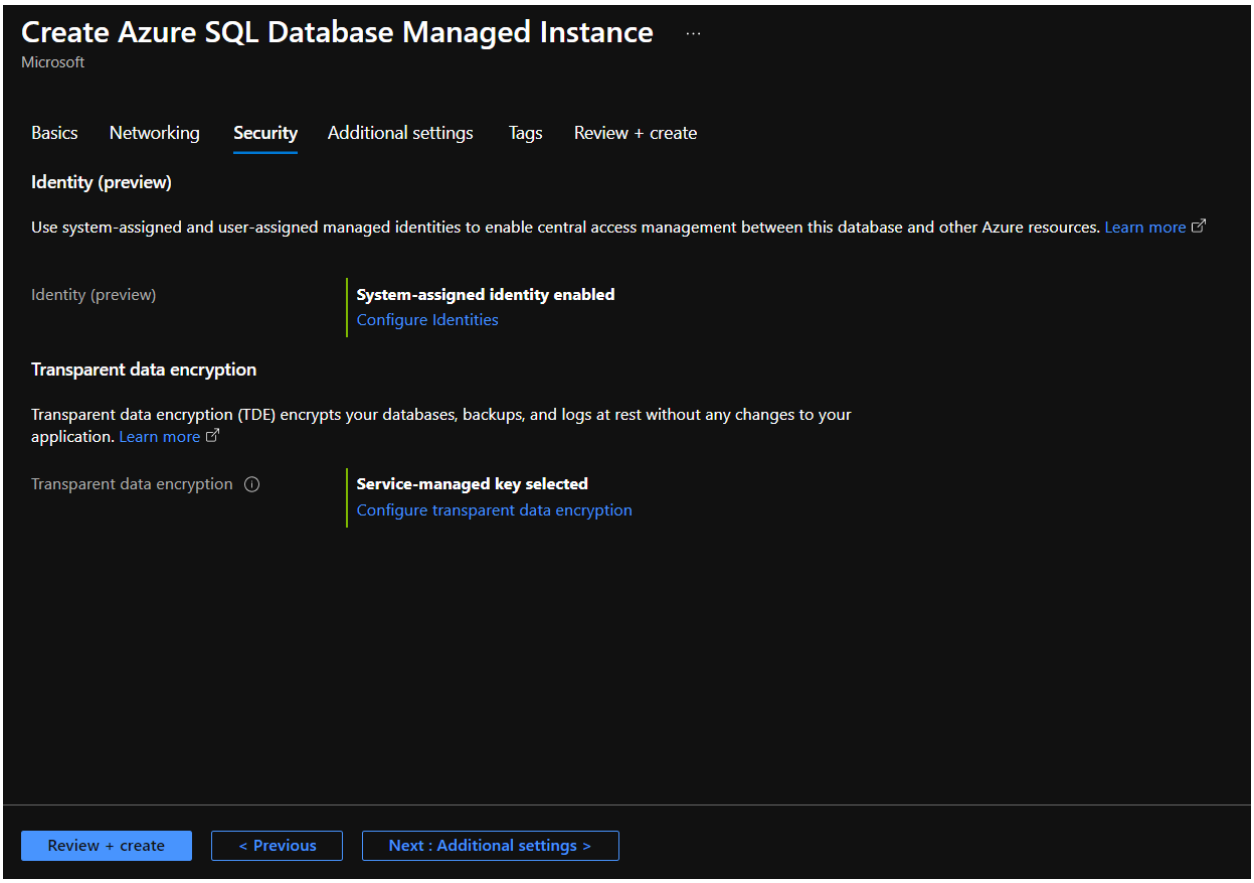
In the next tab we will have aspects to configure related to Networks. It is important to mention that by default our Managed Instance does not have access from the Internet. Here we will select the subnet that we just generated in previous steps:



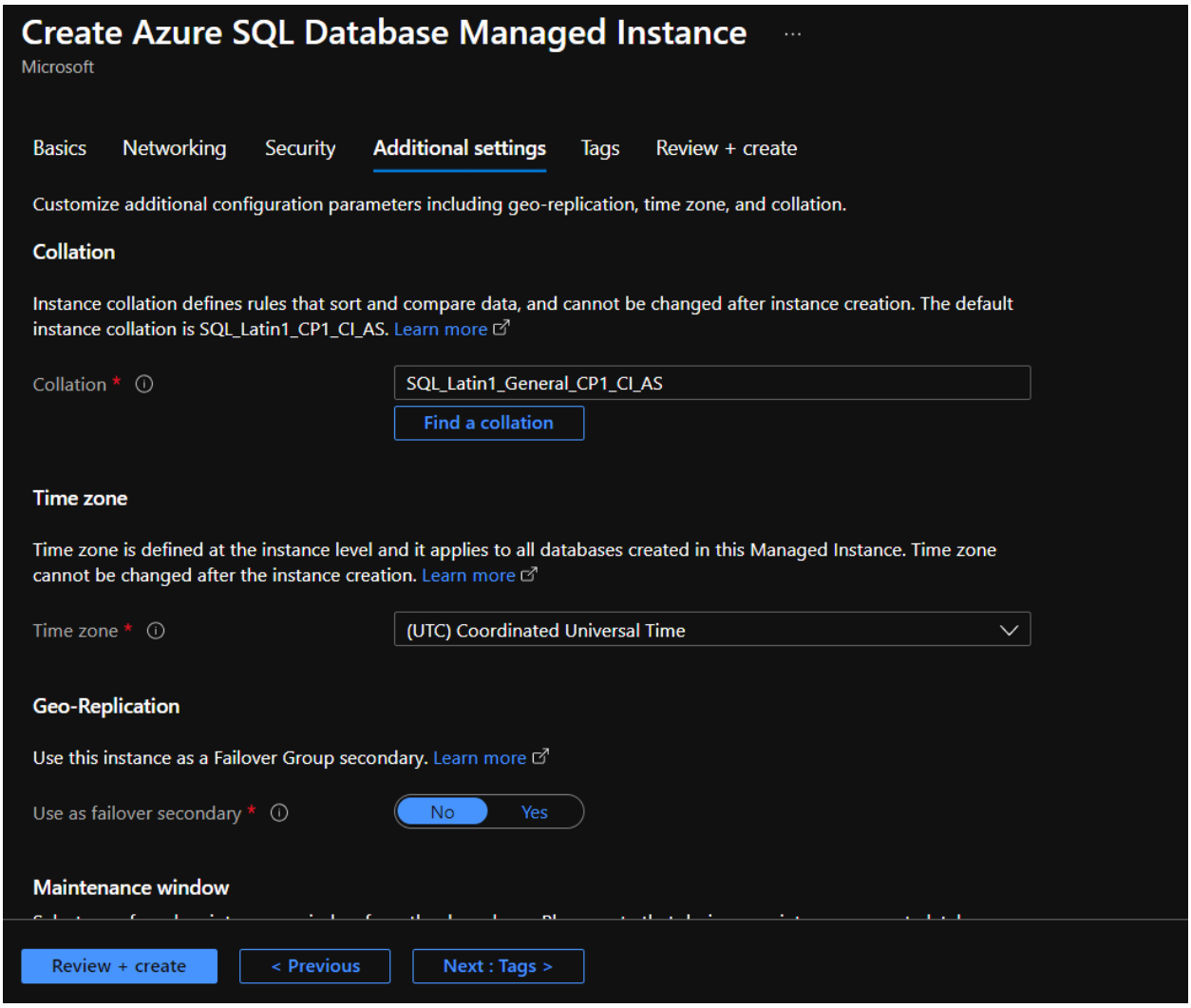
On the other hand, we will keep the Public endpoint (data) in disable. This makes the engine unable to be consumed from the Internet:



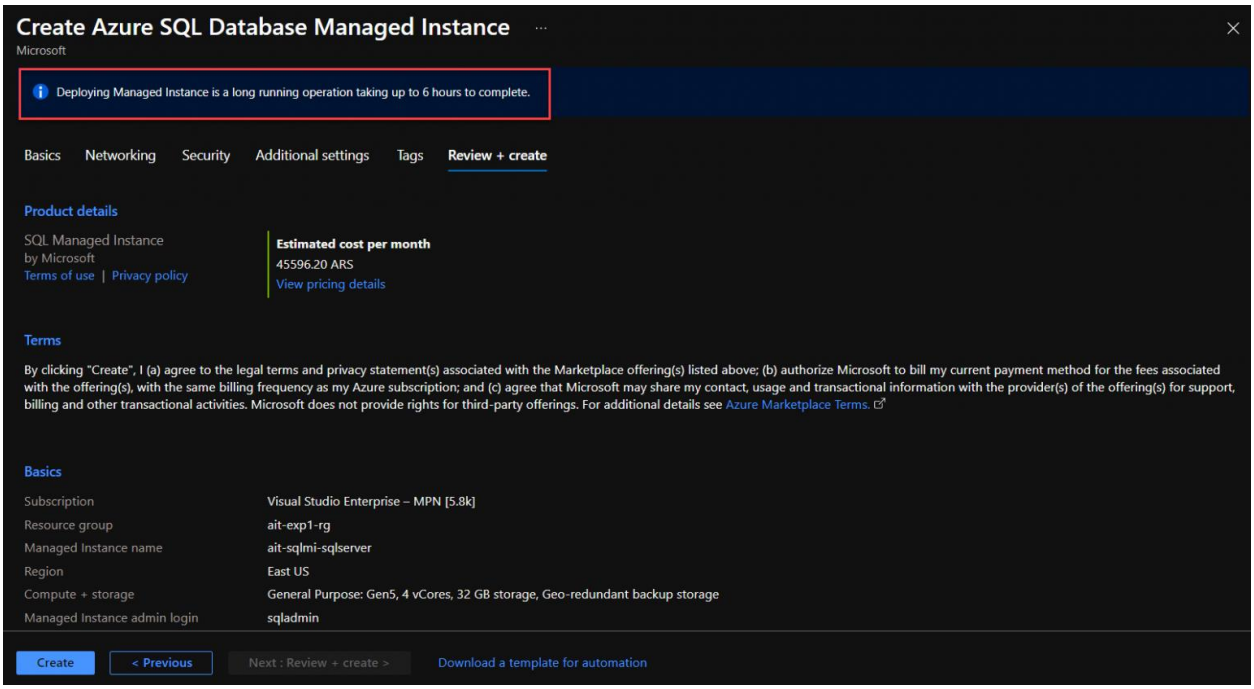
Regarding the security tab, at this time we will not configure any of the options:



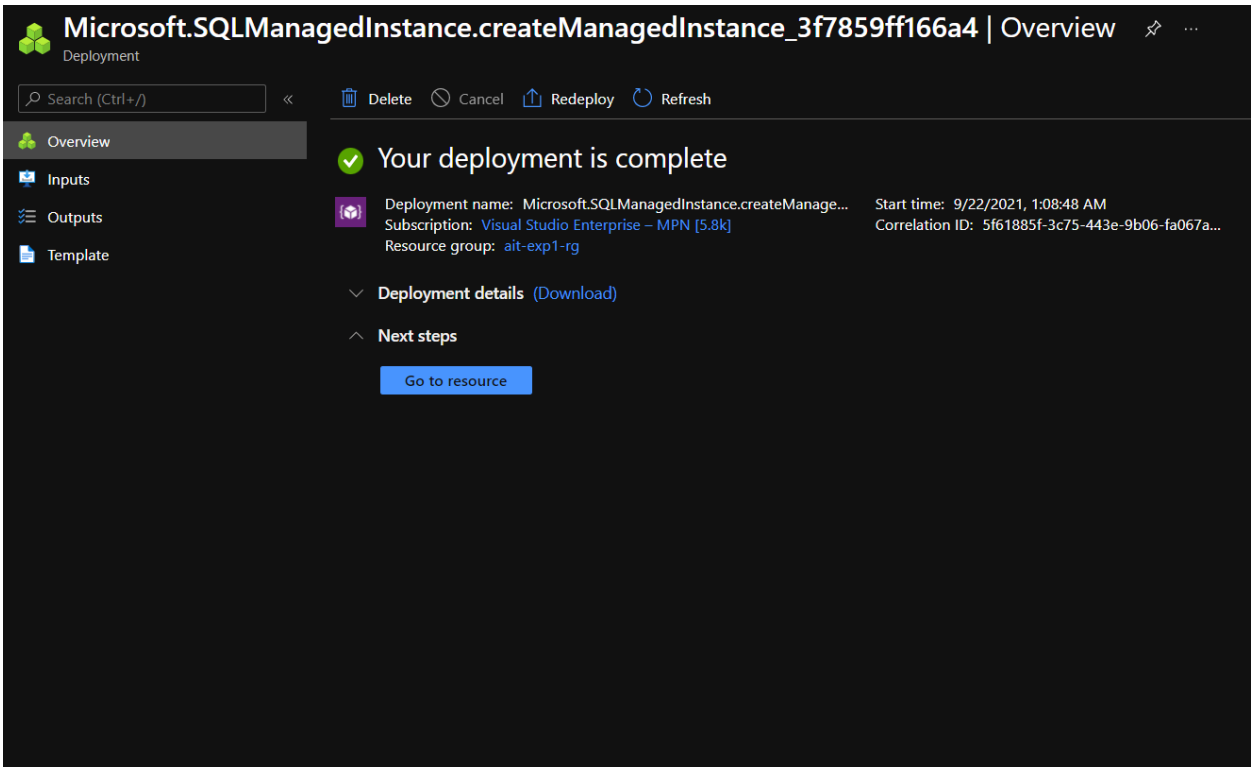
In Additional settings, we will discover that we can select the collection, configure time zone aspects of our server, and configure Geo-replication. We will configure everything as seen in the image:



We come to the last step: we must start with the creation. Remember that this type of resource can take up to 6 hours from the start:

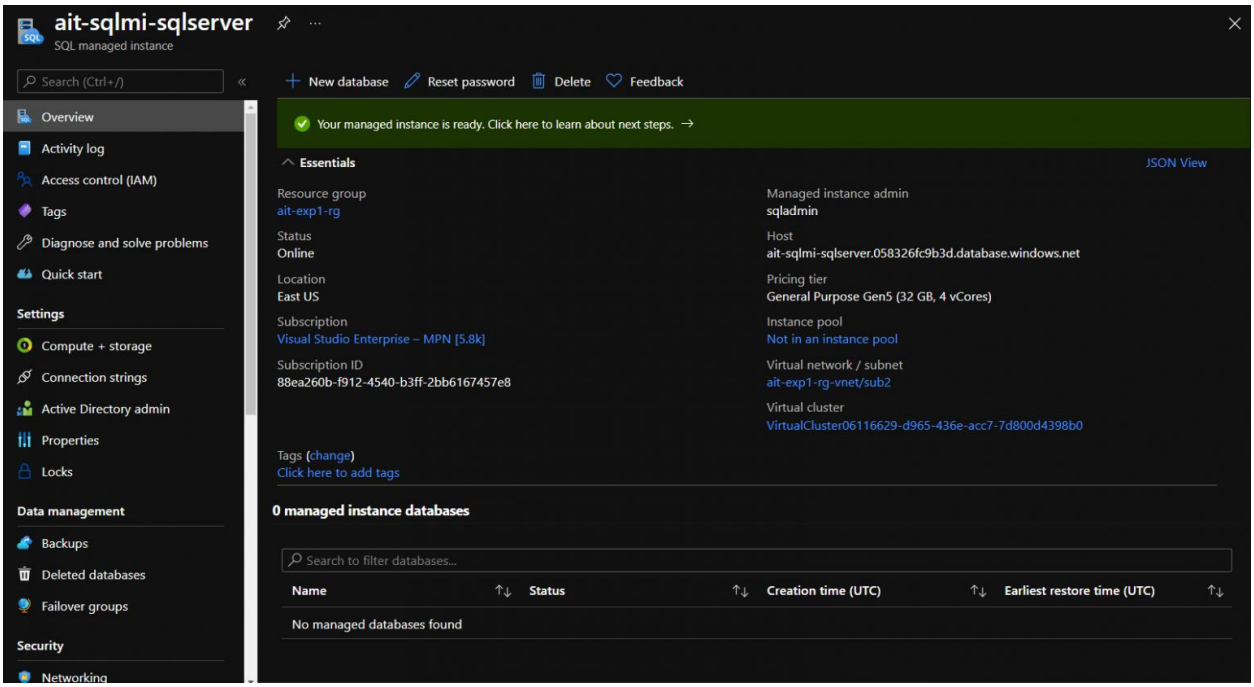


After a few hours have passed, the resource will be created:

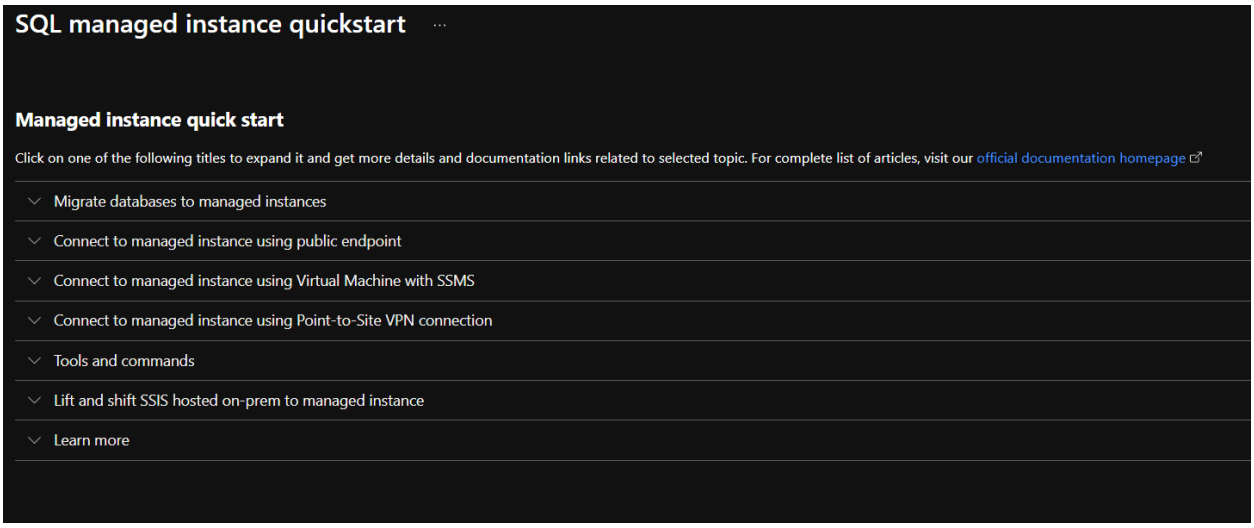


Access to Azure SQL Managed Instance resource

We can go into the resource called “ait-sqlmi-<string>-sqlserver” to explore what the console looks like:



And we can even explore the quickstart on it:

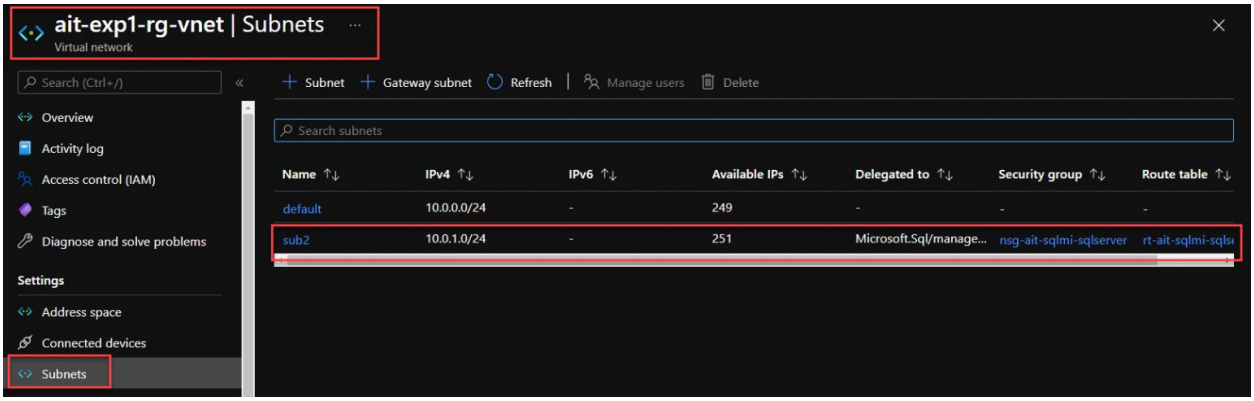


Very good work! Remember that this resource has a high economic value. It is important that you continue the experience and then delete it, so you do not excessively consume your credit in this exercise.

Database creation in SQL MI

Network Validation

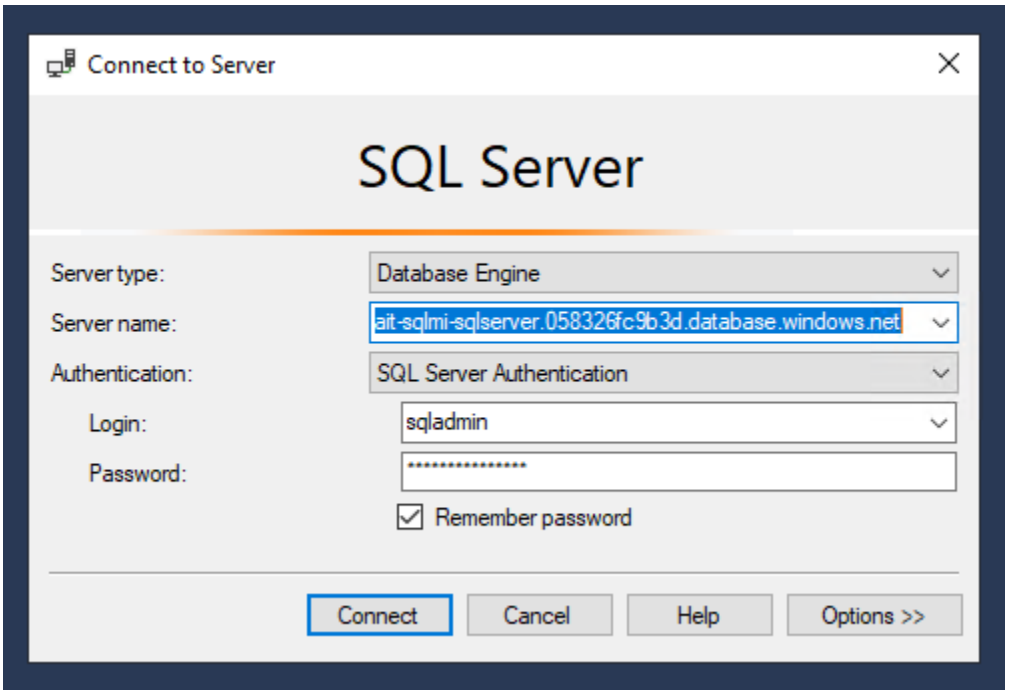
After having created the SQL Managed Instance, we can verify that in sub2 (subnet of our virtual network) we have a Security Group applied (layer 4 filtering rules) and a routing table applied. Also, we have a service endpoint to Microsoft.Sql managed instance:



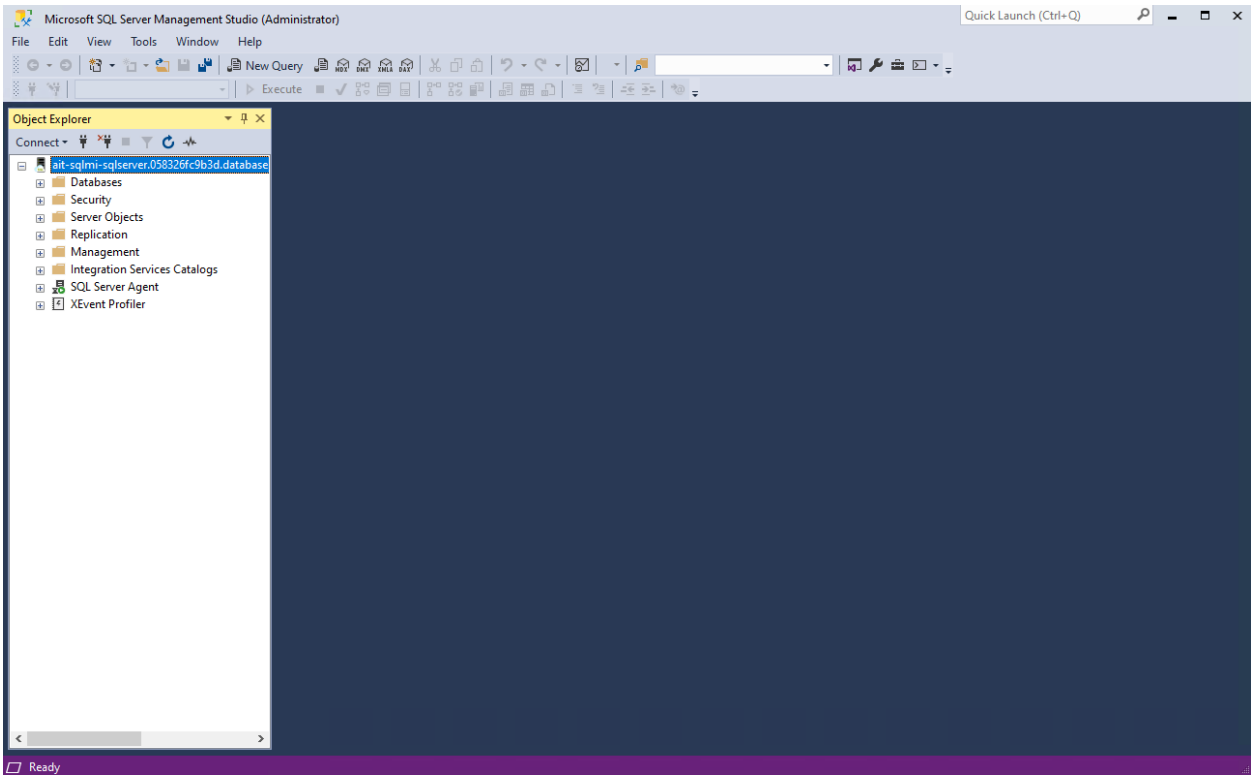
This will allow local network traffic originating from the vNet (as we've seen before) to seamlessly connect to the managed instance.

Connection to SQL Management Studio

To connect to the instance, we are going to use the VM generated in experience 1. There we will open SQLMGMStudio and put the public URL that we get from the SQL Managed Instance Overview, obviously with its credentials:



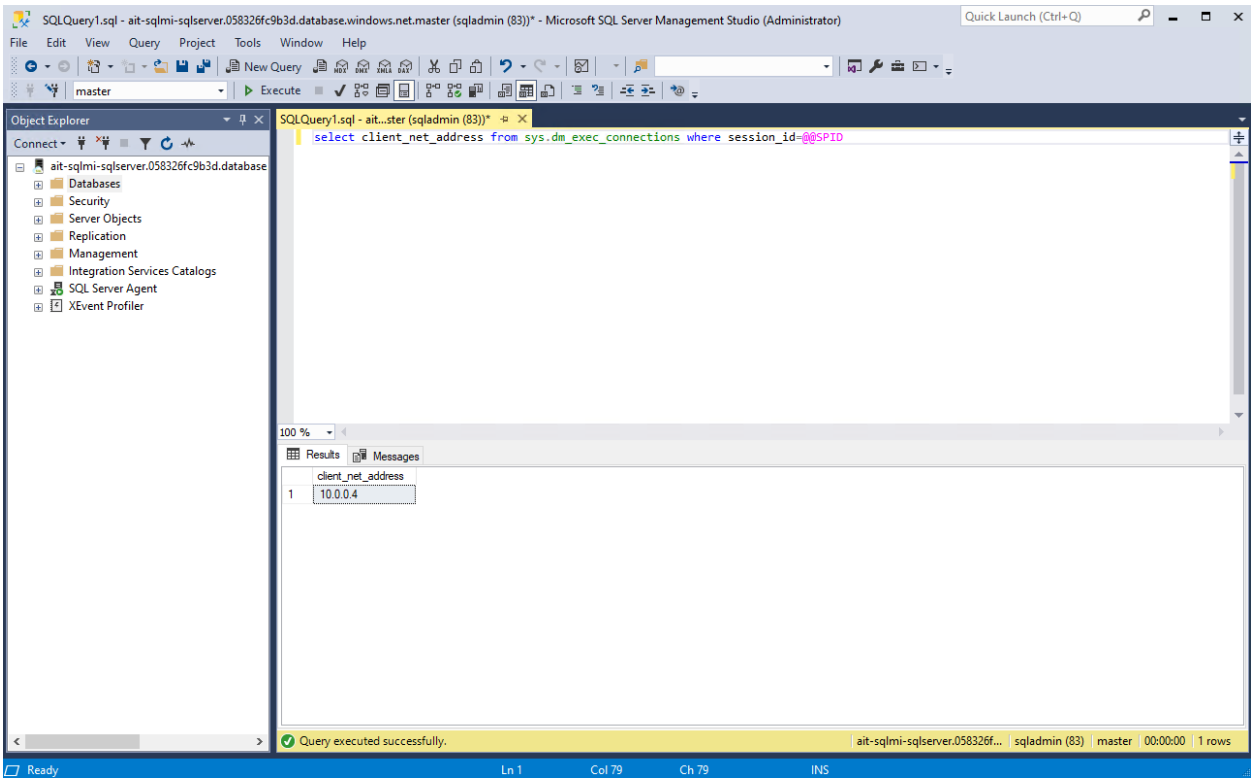
No problem, we can connect to the instance:



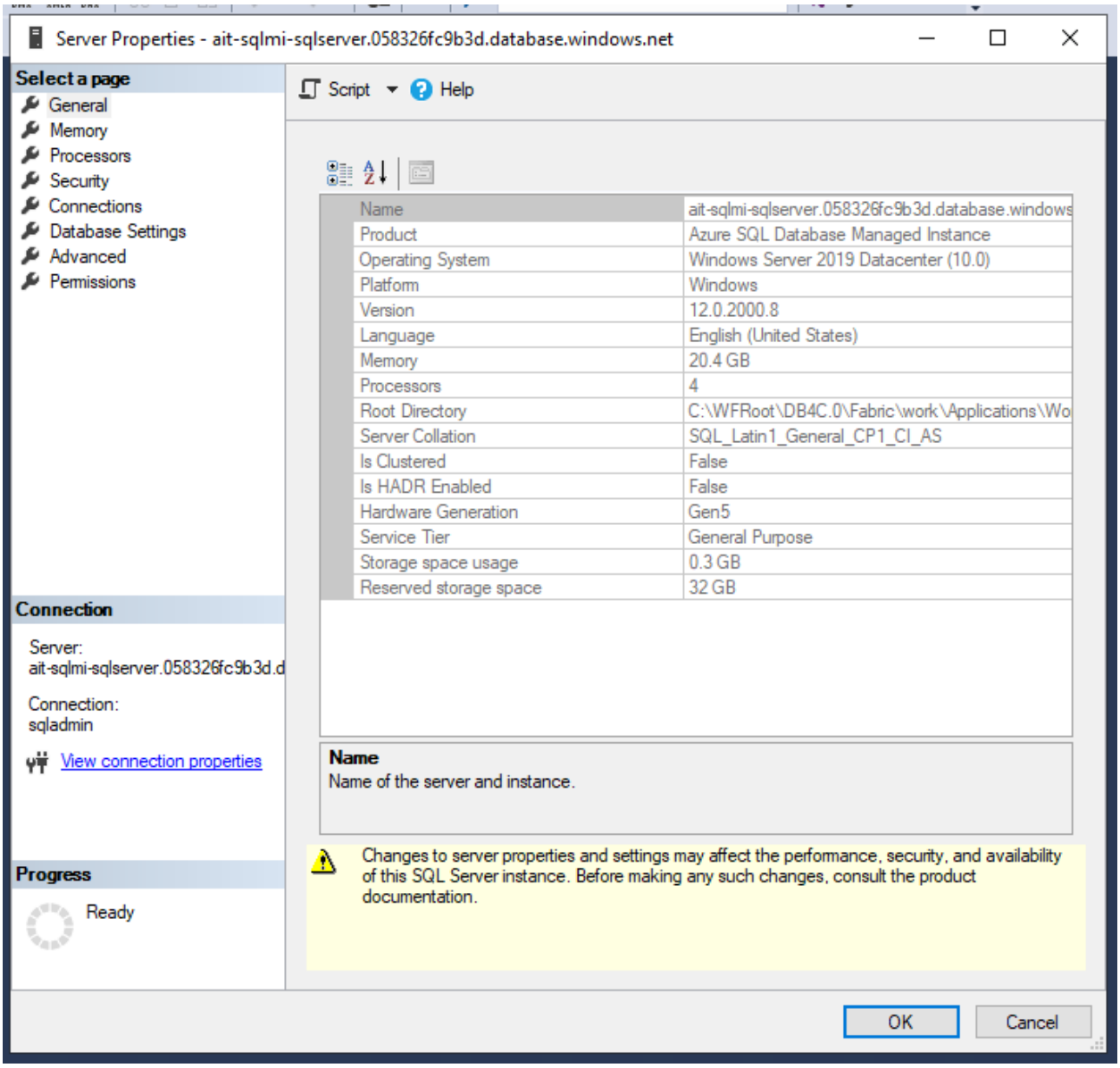
To validate that the traffic originates from the internal network, we execute the same command from Experience 3:

select client_net_address from sys.dm_exec_connections where session_id=@@SPID

We see the results on the screen:

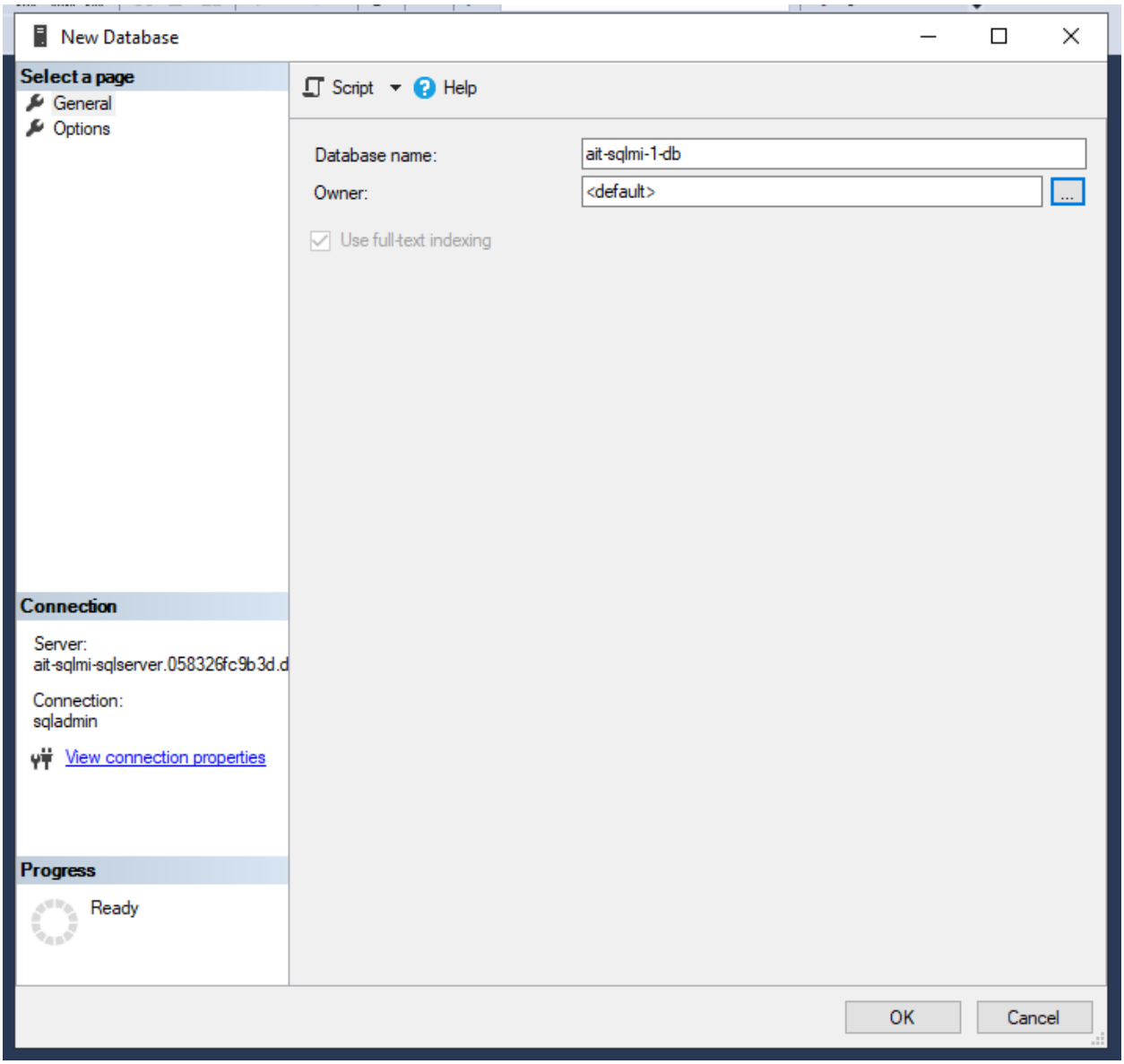


Server Validation:



Creating a Test Database

We can generate a database, just as we would on a local SQL Server:


New Database
Select a page
General
Options
Script
Help
Database name:
ait-sqlmi-1-db
Owner:
<default>
Use full-text indexing
Connection
Server:
ait-sqlmi-sqlserver.058326fc9b3d.d
Connection:
sqladmin
View connection properties
Progress
Ready
OK
Cancel

Great job!

Exercise 5: Data migration

This section will be completed on the second day.

Links of interest

This section will be completed on the second day.