University of Washington

iSchool Info 330

# Module 10 – Cloud Databases and No-SQL Technology

In this module, we will look at cloud technology and how database projects are completed**.**

## Outline

Here is a general outline of what we will be doing this module:

|  |
| --- |
| **Module10: Cloud DBs and No-SQL Technologies** |
| Session01 Lectures and Labs < 110 mins |
| Cloud Technologies |
| Host a Database in the Cloud |
| Host a Database in the Cloud |
| Session02 - Lab |
| Final Project - Milestore04 |
| Session03 Lectures and Labs < 110 mins |
| No-SQL Database |
| Document Databases (MongoDB)  Completing Solutions |
|  |

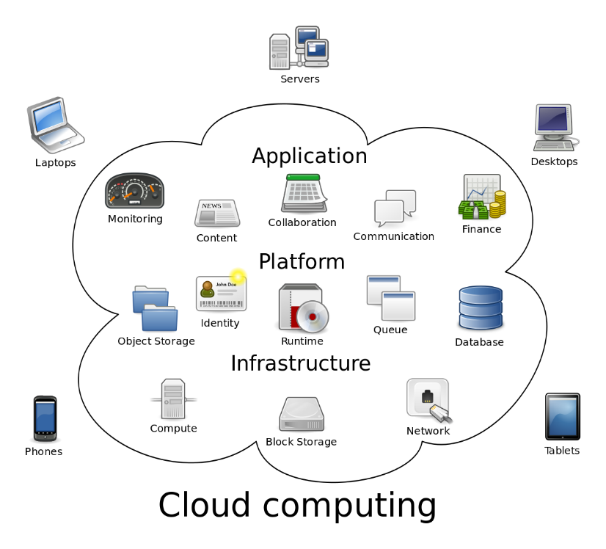
# Session01 < 110 mins

In this session, you will learn **more about cloud-based databases** and Relational database alternatives (**No-SQL** or Not Only SQL **databases**).

## Cloud Technologies

*"Cloud computing is an information technology (IT)* ***paradigm*** *that enables ubiquitous access to shared pools of configurable system resources and higher-level services that can be rapidly provisioned with minimal management effort, often over the Internet. Cloud computing relies on* ***sharing of resources to achieve coherence and economy*** *of scale, similar to a utility.*

*Third-party clouds enable organizations to focus on their core businesses instead of expending resources on computer infrastructure and maintenance.* ***Advocates*** *note that cloud computing allows* ***companies to avoid or minimize up-front IT infrastructure costs****. Proponents also claim that cloud computing allows enterprises to get their* ***applications up and running faster, with improved manageability and less maintenance****, and that it enables IT teams to more rapidly adjust resources to meet fluctuating and unpredictable business demand. Cloud providers typically use a "pay-as-you-go" model, which can lead to* ***unexpected operating expenses if administrators are not familiarized with cloud-pricing models****."*



(<https://en.wikipedia.org/wiki/Cloud_computing>, 2017)

### History

**1970s**

* The **Advanced Research Projects Agency Network** (**ARPANET**) created

**1980s**

* **Private networks are more common** than the Internet (APRANet)

**1990s**

* **Private networks** use **the Internet to tunnel** to other private networks**,** **speed slow, connections not universal**

**2000's**

* **Private networks still prevalent**, **people use the Internet daily**, **speed better**, **connections not universal**

**Mid 2000's**

* Private networks still prevalent, people use the Internet daily, **speed much better**, **connections near-universal**
* **Application Service Providers** (ASP) are a **new way to host server** software
* In 2006, Amazon "**Elastic Compute Cloud**" now **hosts server** software (will become **AWS**)
* In 2008, Google opens **Google App Engine** now **hosts server** software
* In 2010, Microsoft opens **Microsoft Azure** now **hosts server** software
* In 2011, IBM opens **SmartCloud** now **hosts server** software

**What changed?** We now have **affordable, fast, and near-universal access** to the Internet!

### Advantages of Cloud Computing

There are **two main benefits** to cloud computing: **performance and cost**.

**Performance** is its ability to quickly add resources as needed is known as **'scale-up' or 'scale-out.'**

* **Scale-up** = Adding **additional resources to** a **single** **server**; **CPUs, Memory, Storage**
  + **Often requires downtime** when **adding components**
  + **Simpler management and software development**
* **Scale-out** = Creating applications that can **use multiple servers** to an application (more flexible than Scale-up)
  + **Complex management and software development**
  + **Seldom requires downtime** when **adding servers**

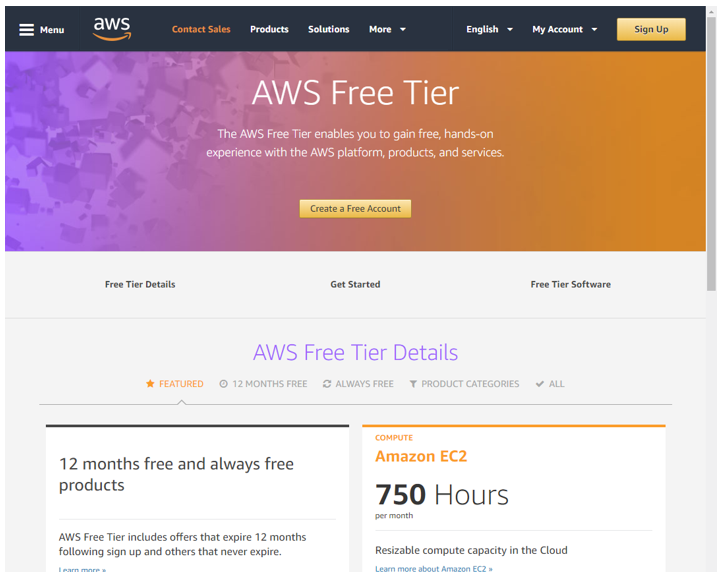
**Costs are reduced by changing for only the resources used**. This is like the difference between rent a car or buying a car. If you are not going to use a car much it is cheaper to rent as need then it is to buy it. **For some databases, it is cheaper to pay for only the time the database is used instead of buying a full license.**

### Amazon's Web Services (AWS)

**Create and Connect to a Microsoft SQL Server Database with Amazon RDS**

"In this tutorial, you will learn how to create a Microsoft SQL Server database Instance (we call this a 'DB instance'), connect to the database, and delete the DB instance. We will do this using Amazon Relational Database Service (Amazon RDS) and everything done in this tutorial is free tier eligible."

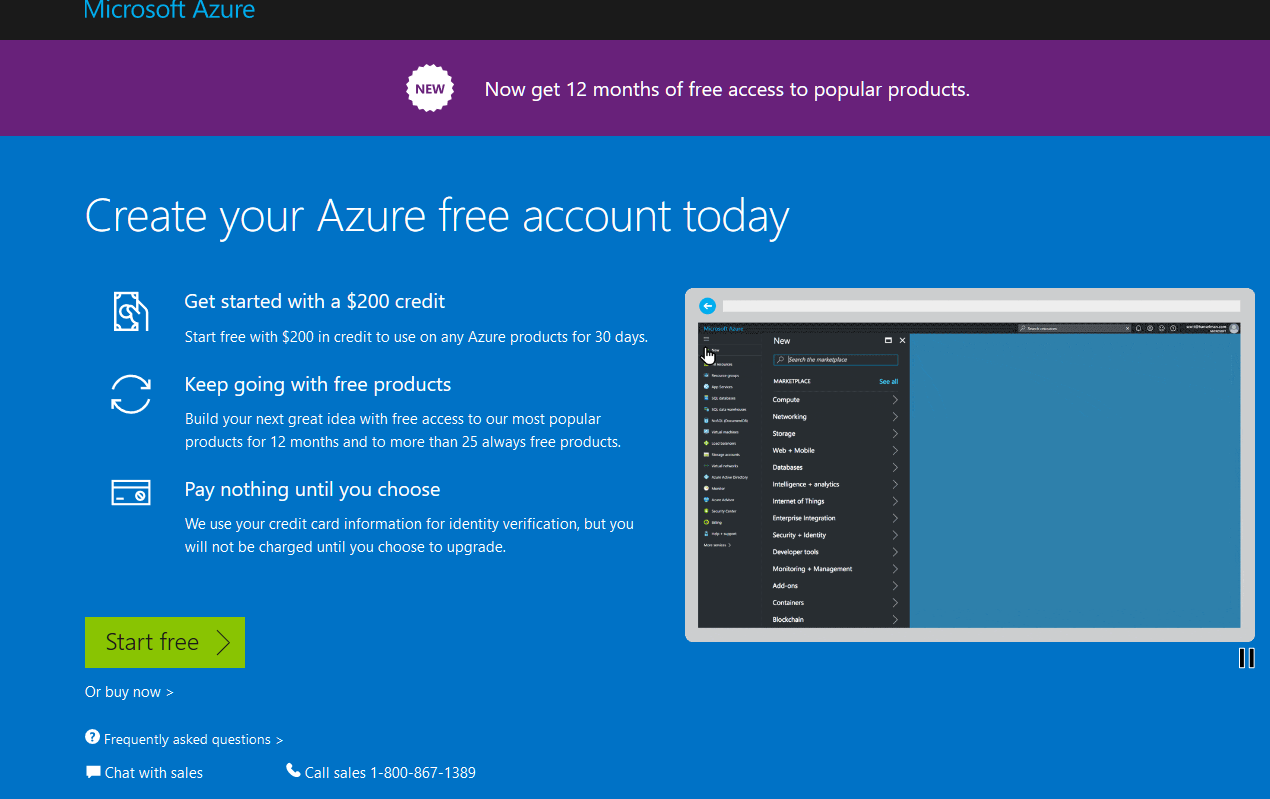
( <https://aws.amazon.com/getting-started/tutorials/create-microsoft-sql-db/>, 2017)



### Microsoft's Azure

**Create an Azure SQL database in the Azure portal**

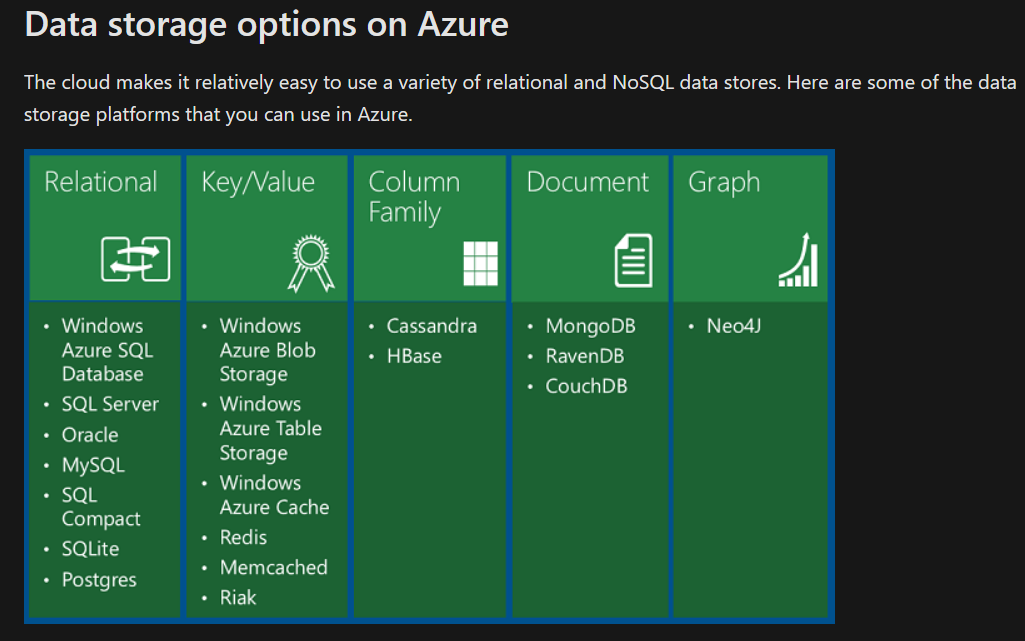
"This **quick start tutorial** walks through how to create a SQL database in Azure. Azure SQL Database is a "Database-as-a-Service" offering that enables you to run and scale highly available SQL Server databases in the cloud. This quick start shows you how to get started by creating a SQL database using the Azure portal." (<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-get-started-portal>, 2017)



## Azure Databases

"There is **no single best data management choice for all data storage tasks**; different data management solutions are optimized for different tasks. Most real-world cloud apps have a variety of data storage requirements and are often served best by a combination of multiple data storage solutions.

…"

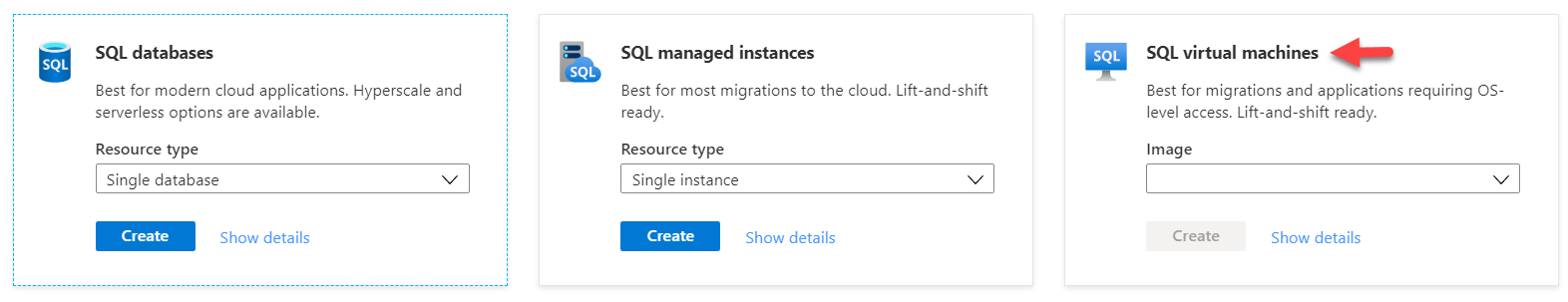


(<https://docs.microsoft.com/en-us/aspnet/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/data-storage-options>, 2020)

### Azure for SQL Server

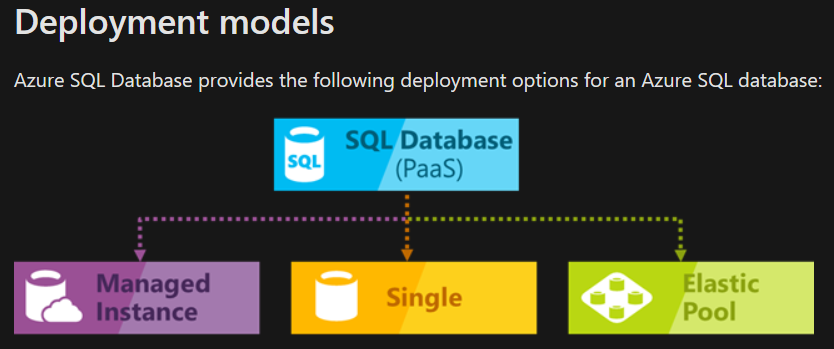
Since we have been working with Microsoft's stack, we're going to **use Microsoft's Azure as our cloud platform**. Let's look at what the options are and how they are used.

**First**, you have several choices of how you can deploy a SQL database on Azure. The traditional approach is to use a **Server Based Virtual Machine,** which is **renting a** **hosted Operating System** on which you can **install SQL Server if you own a license**.



**The second option is the Serverless "Platform as a Service." There are three basic choices in this category.**

* [**Managed Instance**](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-managed-instance) Rent and **manage an instance of the** [**Microsoft SQL Server Database Engine**](https://docs.microsoft.com/sql/sql-server/sql-server-technical-documentation?toc=/azure/sql-database/toc.json). **Like On-Premises server**, you can create as many databases on the server as like, but **without having to pay for a license first**.
* [**Single Database**](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-single-database)**:** Rent and **manage a single database**. Rent is **based on Data Transaction Units** (DTU), which reflect how much CPU power, Memory access, and Disk space are used.
* [**Elastic pool**](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-pool)**:** Rent **one or more databases as a single** collection. Rent is **also** **based on Data Transaction Units** (DTU), which reflect how much CPU power, Memory access, and Disk space are used. However, **DTUs are flexible,** and rent lowers or raises depending on use.

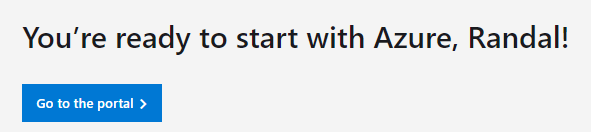
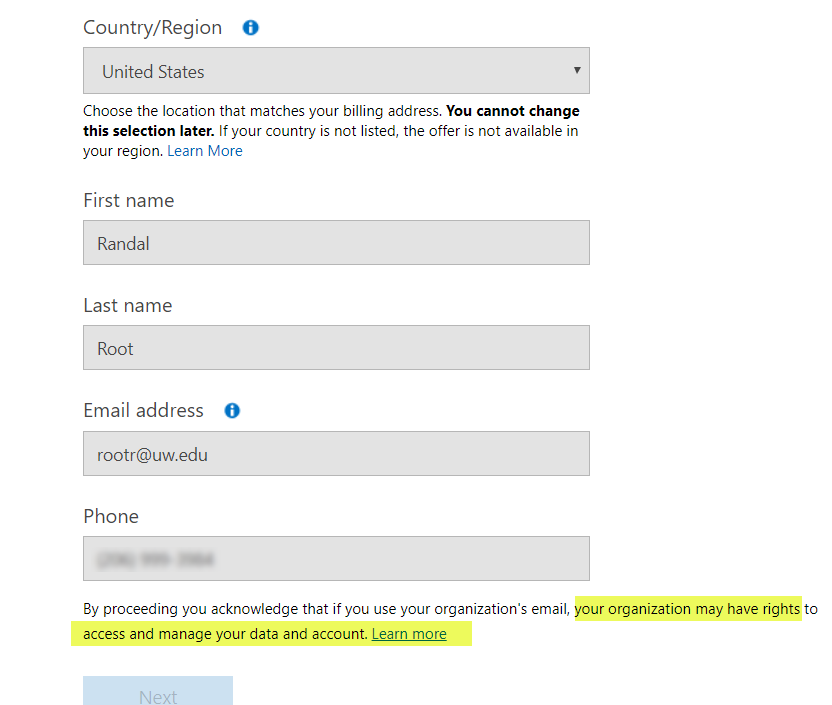


(<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-technical-overview>, 2020)

### Creating an Azure SQL Server Database

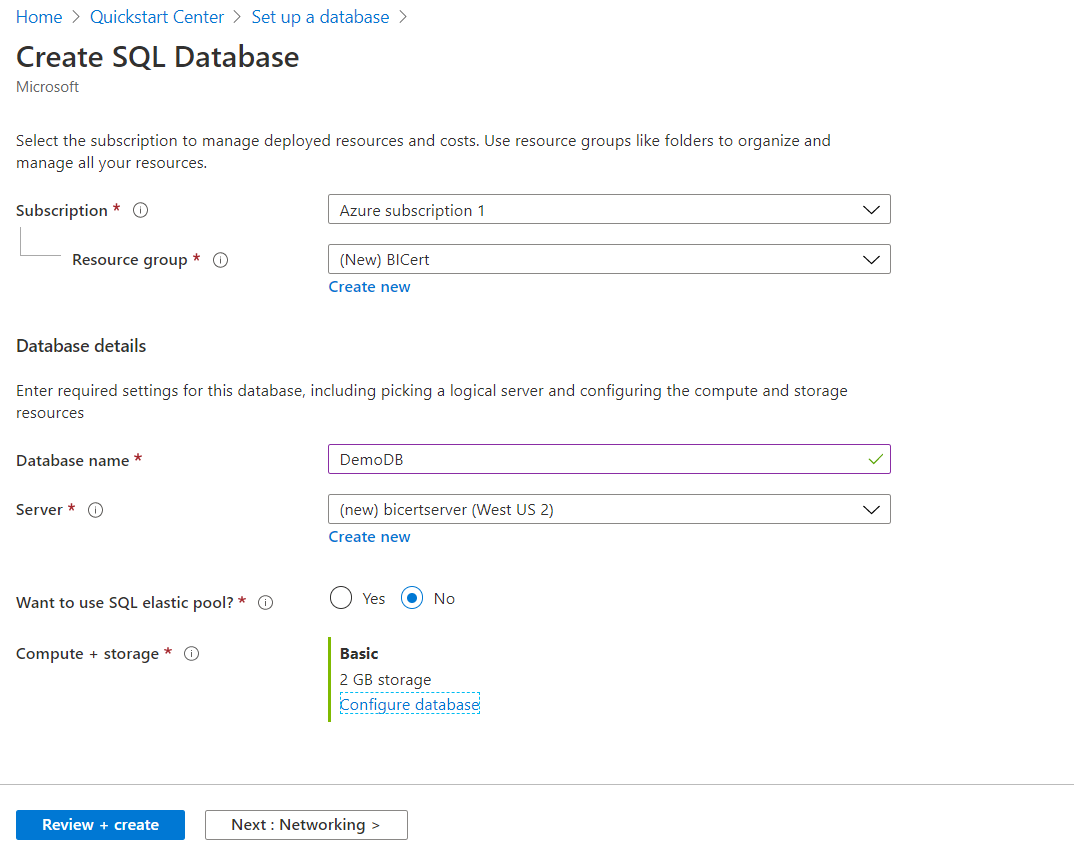
To get started with Azure, you will need an account. You can use your UW email account to create a new login and receive **$200** of "credit" to test things **for 30 days**. **After** that, **some things** will be **charged** for, but **other things** are free for one **year**, and **some** **others are free "forever."**

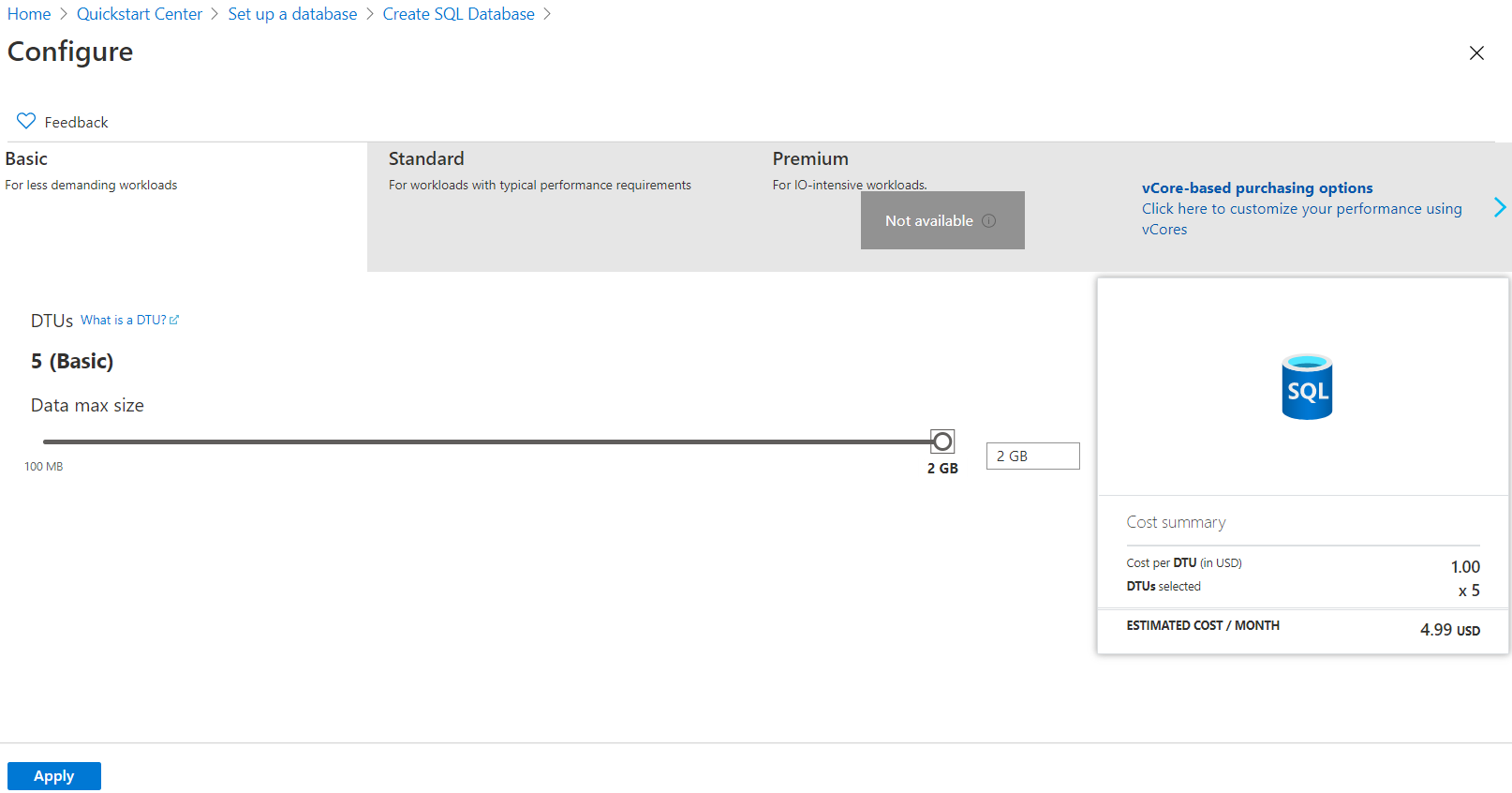
<https://azure.microsoft.com/en-us/free/>



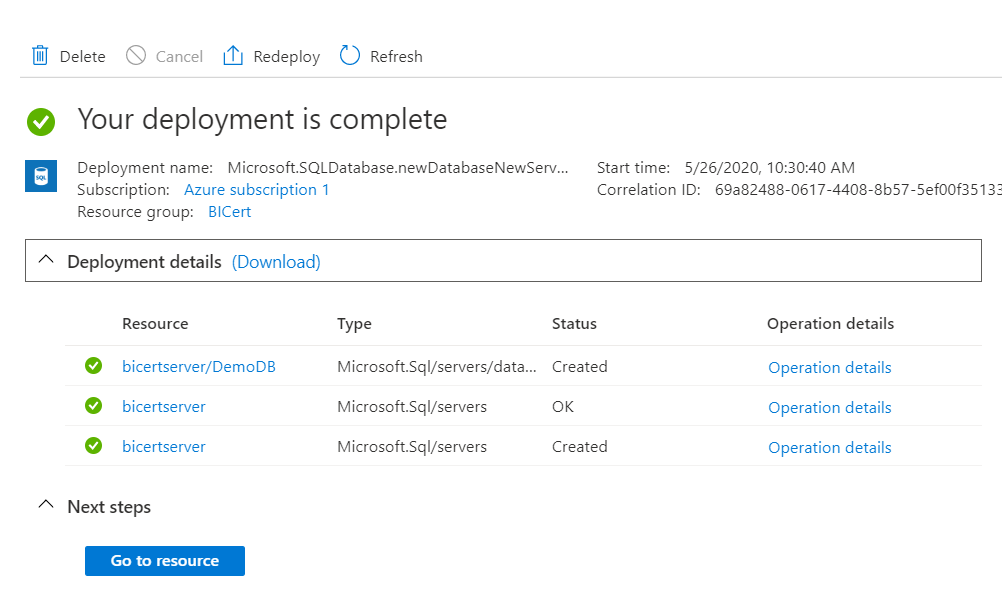
After a while, you will see a button to access your personal portal.

From the portal, you can make a SQL Database. **Currently**, the **lowest priced database is about $5 per month**, since **they have removed their "always free" edition**.

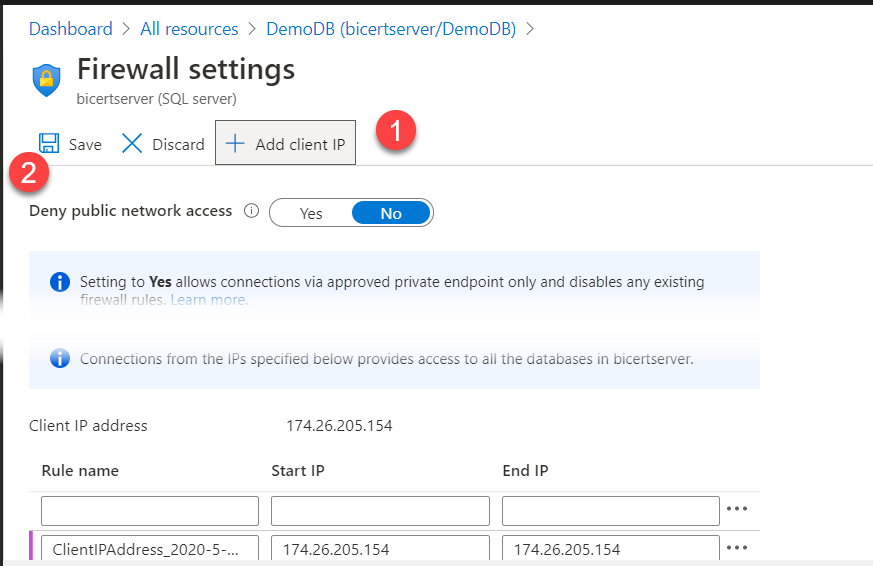




Wait for the deployment to complete and then access your server by clicking the **"Go to resource" button**.



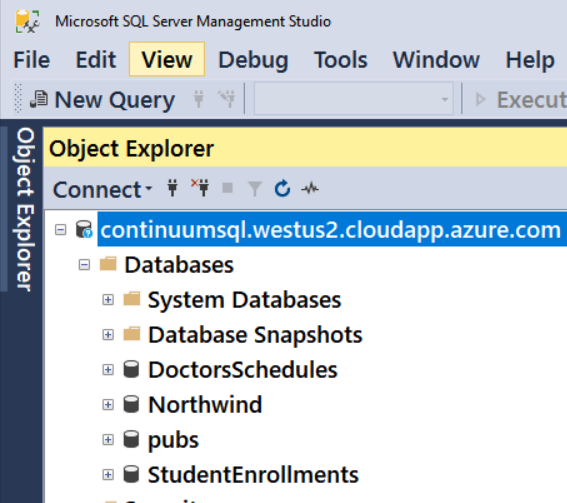
After deployment, you need to **add your IP address** before you can access your database from your computer, outside of the web portal.



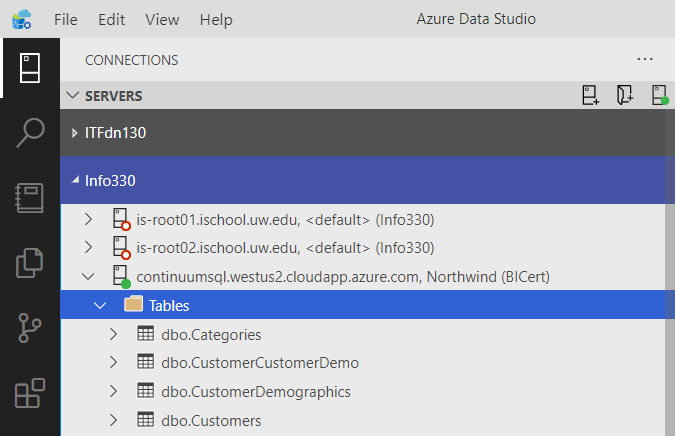
### Connecting to an Azure Server

If your organization has a SQL Server hosted on Azure Cloud, you **can connect to it with many client applications**.

#### SQL Server Management Studio



#### Azure Data Studio



#### Custom Applications

If you want more customization, you can create applications using multiple languages. Here is an example of a console application using Microsoft's C#:

static void Main(string[] args)

{

//Create a connection string

SqlConnectionStringBuilder objCSB;

objCSB = new SqlConnectionStringBuilder();

objCSB.DataSource = "continuumsql.westus2.cloudapp.azure.com";

objCSB.InitialCatalog = "StudentEnrollments";

objCSB.UserID = "Info330";

objCSB.Password = "Info330";

//Open a connection

SqlConnection objCon = new SqlConnection();

objCon.ConnectionString = objCSB.ConnectionString;

objCon.Open();

//Issue a command

SqlCommand objCmd = new SqlCommand();

objCmd.Connection = objCon;

objCmd.CommandText = "Select StudentID, StudentEmail From Students";

//Process the Results

SqlDataReader objDR = objCmd.ExecuteReader();

while (objDR.Read())

{

Console.WriteLine(objDR[0] + ": " + objDR[1]);

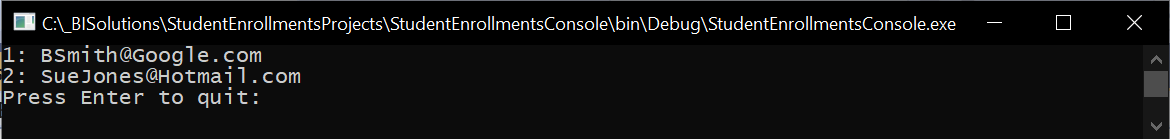
}

objCon.Close();

Console.Write("Press Enter to quit: ");

Console.Read();

}



Of course, you do not have to use only Microsoft applications and programming languages with SQL Server (either on Azure or locally)!

*"In this quickstart, you* ***use Python to connect to an Azure SQL database*** *and use T-SQL statements to query data.*

*…*

*To install Python, the ODBC driver and SQLCMD, and the* ***Python driver for SQL Server****, see*[*configure an environment for* ***pyodbc*** *Python development*](https://docs.microsoft.com/en-us/sql/connect/python/pyodbc/step-1-configure-development-environment-for-pyodbc-python-development#windows)*.*

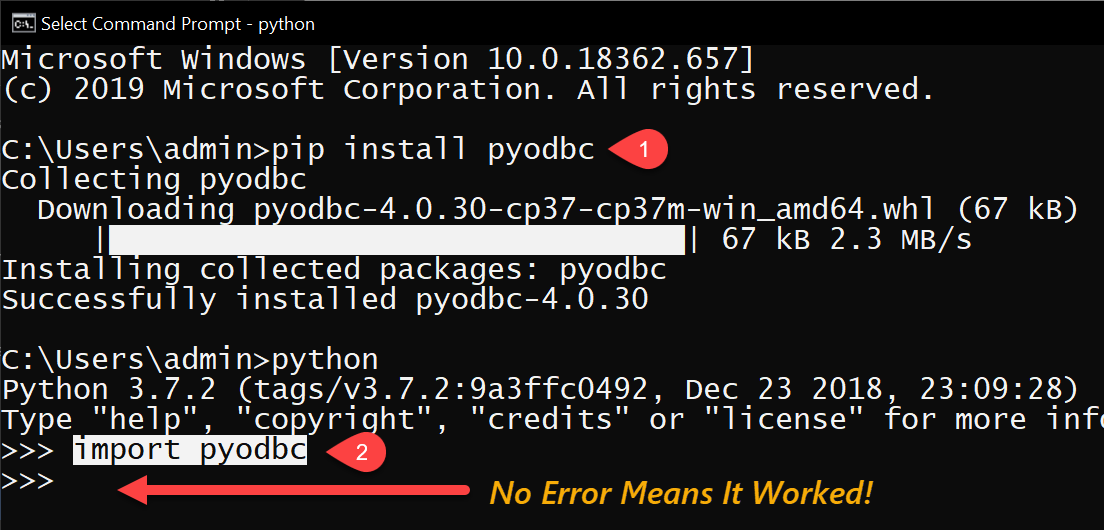
*…*

*For further information, see*[***Microsoft ODBC Driver***](https://docs.microsoft.com/en-us/sql/connect/odbc/microsoft-odbc-driver-for-sql-server)*."*

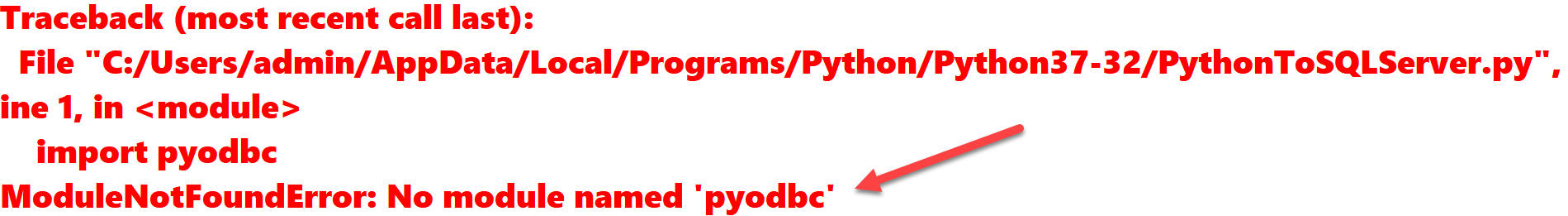
(<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-connect-query-python?tabs=windows>, 2020)

A few years ago, a developer named Michael Kleehammer wrote a connection module for Python to SQL communication.

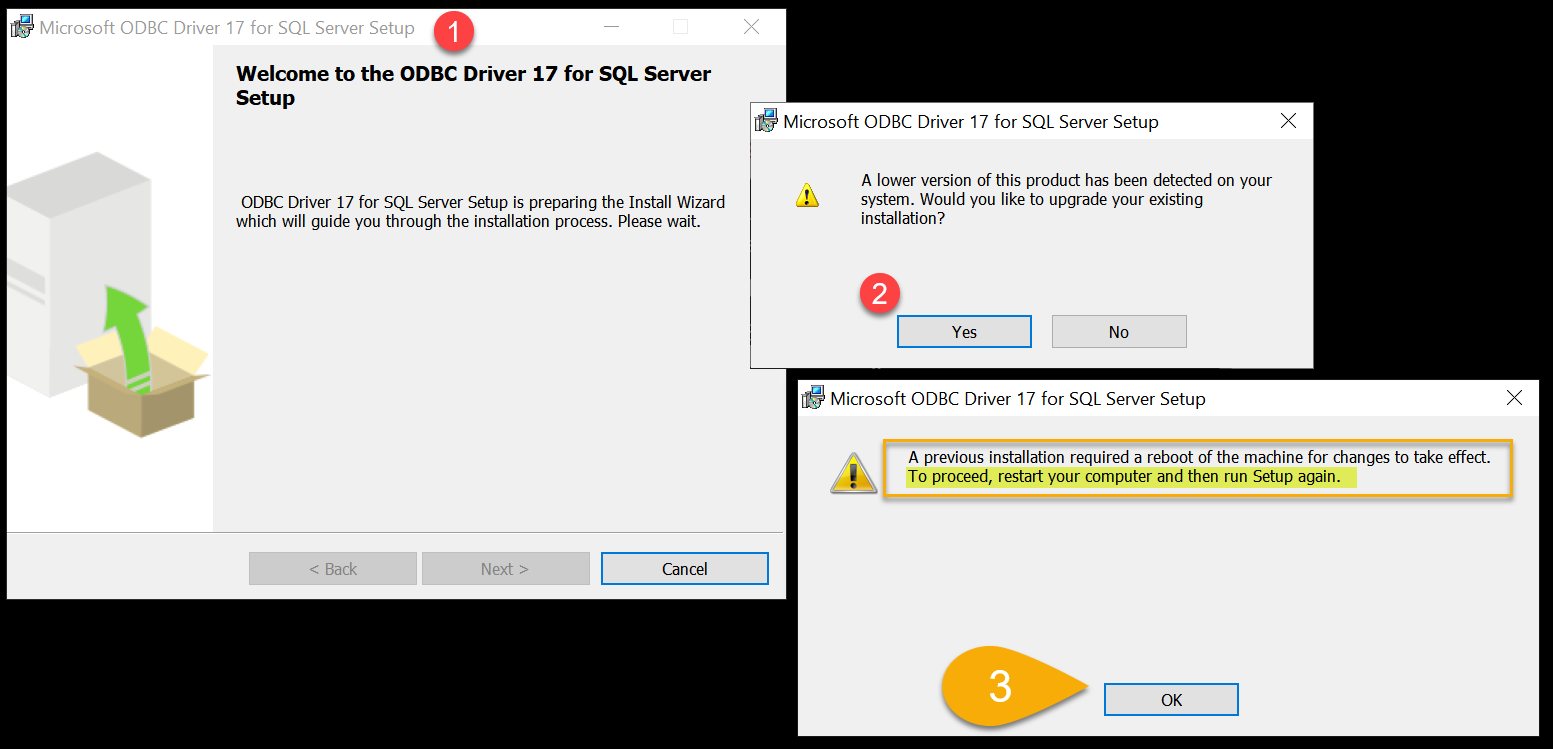
<http://mkleehammer.github.io/pyodbc/>



**Note:** If you this work in the command window, but not in IDLE, **close and reopen the IDLE** environment.



NOTE: On **Windows, ODBC should already be on your computer**, but if not, you can install it. Just be aware that it may require a restart**. On Mac, it needs to be installed**.



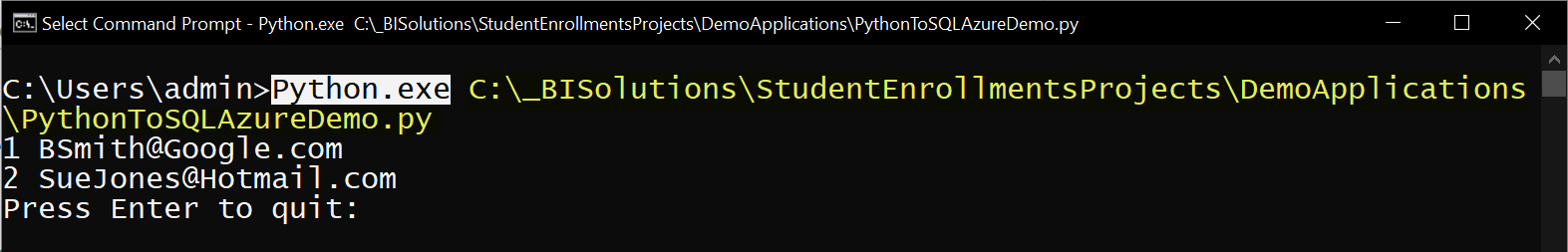
Once you have **both the** **PyODBC and the SQL ODBC driver installed**, use the following code to make a connection:

**import** pyodbc  
  
**""" Making a Connection """**

server = **'continuumsql.westus2.cloudapp.azure.com'**database = **'StudentEnrollments'**username = **'Info330'**password = **'Info330'**driver = **'{ODBC Driver 17 for SQL Server}'**con = pyodbc.connect(**'DRIVER='** + driver  
 + **';SERVER='** + server  
 + **';PORT=1433'** + **';DATABASE='** + database  
 + **';UID='**+ username  
 + **';PWD='**+ password)

**""" Issuing a Command """**cursor = con.cursor()  
cursor.execute(**"SELECT StudentID, StudentEmail FROM [dbo].[Students]"**)  
row = cursor.fetchone()  
  
**""" Processing the Results """  
while** row:  
 print (str(row[0]) + **" "** + str(row[1]))  
 row = cursor.fetchone()

print(**"Press Enter to quit: "**)  
input()



# Session02 Lectures and Labs < 50 mins

In this session, you continue to work on the final. You will work on your own for the whole 50 minutes of this lab but may ask questions whenever you would like help.

# Session03 Lectures and Labs < 110 mins

## No-SQL Databases

**No-SQL or Not-Only-SQL, is simply a non-relational database that does not require you to use the SQL language.**

*"A NoSQL (originally referring to "non SQL" or "non relational") database provides a mechanism for storage and retrieval of* ***data that is modeled in means other than the tabular relations*** *used in relational databases. Such databases have* ***existed since the late 1960s****, but did not obtain the "NoSQL" moniker until a* ***surge of popularity*** *in the early twenty-first century, triggered by the needs of Web 2.0 companies such as Facebook, Google, and Amazon.com. NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also* ***sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages.****"* (<https://en.wikipedia.org/wiki/NoSQL>, 2017)

### What is No-SQL?

* A movement **away from** SQL **relational** databases
* Came out of Internet Age with **cheap connectivity being expected**!
* Broad set of technologies to **address shortfalls of relational databases**
* **Dynamic** Schemas, **auto-sharding**, LARGE data **storage** on **multiple computers**

**No-SQL** databases **do not often enforce structure**

* Address a range of data management problems
* Cross-industry / application
* Structure will change quickly and often
* Fixed table structure not required

**Relational** databases **often enforce structure**:

* Schema ('shape') of the table is defined ahead of time
* Few/no schema changes
* Few NULL data values (well-designed/appropriate schema)

Unstructured data **may contain consistency issues**! Eric Brewer at UC Berkeley states that it is only possible to have 2 out of the 3: **C**onsistency, **A**vailability, **P**artition Tolerance.

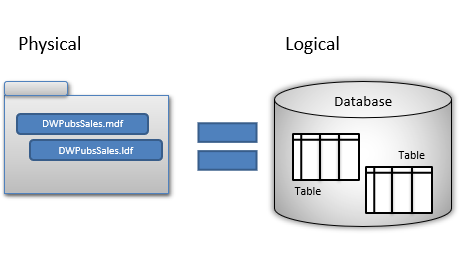
"In some sense, the NoSQL movement is about creating choices that **focus on availability first** and **consistency second**" (<https://www.computer.org/cms/Computer.org/ComputingNow/homepage/2012/0512/T_CO2_CAP12YearsLater.pdf>, Eric Brewer, 2012)

## Cube Databases

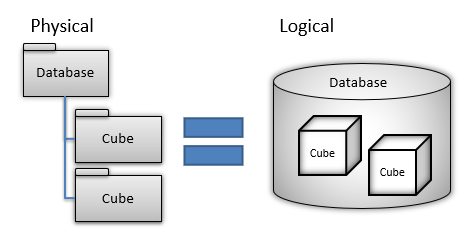
"The purpose of [*OLAP Cubes]* is to provide high-performance reporting data. **Reports created on a multidimensional cube run faster compared to reports built upon a set of relational tables**. As usual, faster performance means more complexity. Creating SSAS cubes **demands** you to understand dimensional data in a completely new way. It requires that **you to understand the difference between SSAS cubes and dimensions**. It even requires you to learn new development and administrative tools, plus four programming languages (**MDX, XMLA, DMX, and DAX**), if you want to master SSAS." ***(***Pro SQL Server 2012 BI Solutions, Root and Mason, Apress, 2012***)***

**One of the reasons for this increase performance** is the inherent ability of SSAS to create and **store aggregate values**. **Another** reason for this is the actual **design structure of the database**.

A SQL Server **relational database** the tables and database itself are logical constructs representing actual physical files on the hard drive. **Each database has at least two files**, but the SQL server engine makes it appear as if it is one object. **Each database can hold many tables,** and each SQL server can host many databases.



**Analysis Server** does something very similar, but the **structure is different**. In SSAS, each **database is a folder** on the hard drive, and **each cube a subfolder** within the database folder. Each **Database can have many cubes** and each **Analysis Server can host many databases**.



## Dimension and Measures

A cube is like **a multidimensional array the organizes a set of subjects** such as authors and titles, and publishers and stores multiplied by its measured values. **Each subject represents a dimension of the cube,** and **each measured value is cross multiplied to provide a distinct aggregate value for each combination of attribute and measure**, the product of which is a multi-dimensional cube.

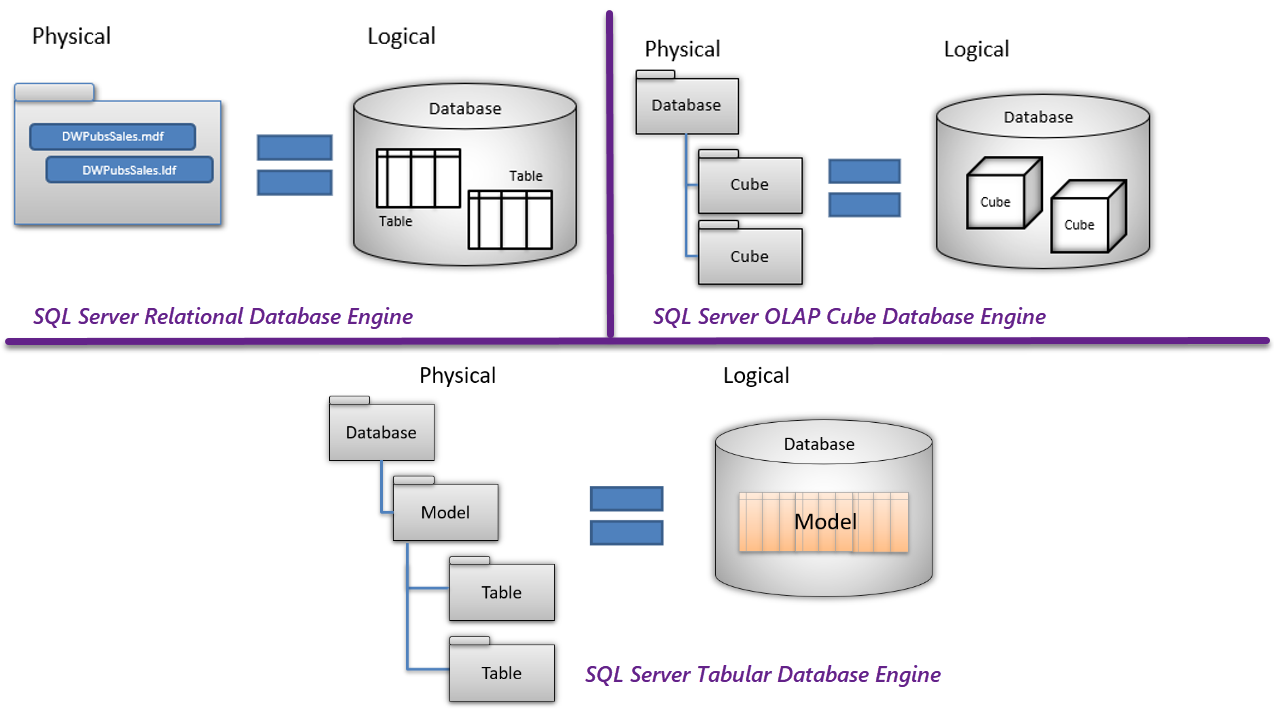


## Tabular Databases

With SQL Server's **relational database,** the tables and database itself are logical constructs representing **at least two files**, but the SQL server engine makes it appear as if it is one object. **Each database can hold many tables,** and each SQL server can host many databases.

**In Analysis Server Cube Server,** each **database is a folder** on the hard drive, and **each cube a subfolder** within the database folder. Each **Database can have MANY cubes,** and each **Analysis Server can host many databases**.

**In Analysis Server Tabular Server,** each **database is a folder** on the hard drive, with **subfolders representing the different tables of the tabular data model**. Each **Database can have only ONE model, but each Analysis Server can host many databases**.



## Tabular Models vs. OLAP Cubes vs. Reporting Tables

**Tabular model are similar to combined set of reporting tables**, but there are **differences**. Most companies decide that using reporting tables is easier and provide sufficient performance for their needs, but may move to a Tabular model base reporting system later. Each option **has advantages and disadvantages**.

Tabular Models:

* **Best performance for data retrieval**, but not transactional statements
* **Less** Joins and Subqueries
* **Storage of aggregate values is encouraged**
* You access the Model **using** the **MDX or DAX, and** programming languages
* You configure the Model **using** the **relatively unknown XMLA, JSON, or Tabular Model Scripting Language (TMSL)**
* **Many-to-many** relationships in tabular models **required a workaround before SSAS 2019**

**Cubes:**

* Has **excellent performance for data retrieval**, but not transactional statements
* **Less** Joins and Subqueries
* **Storage of aggregate values is encouraged** in SSAS Cubes
* You access Cubes **using** the **relatively unknown** **MDX and XMLA** programming languages

**Reporting Tables:**

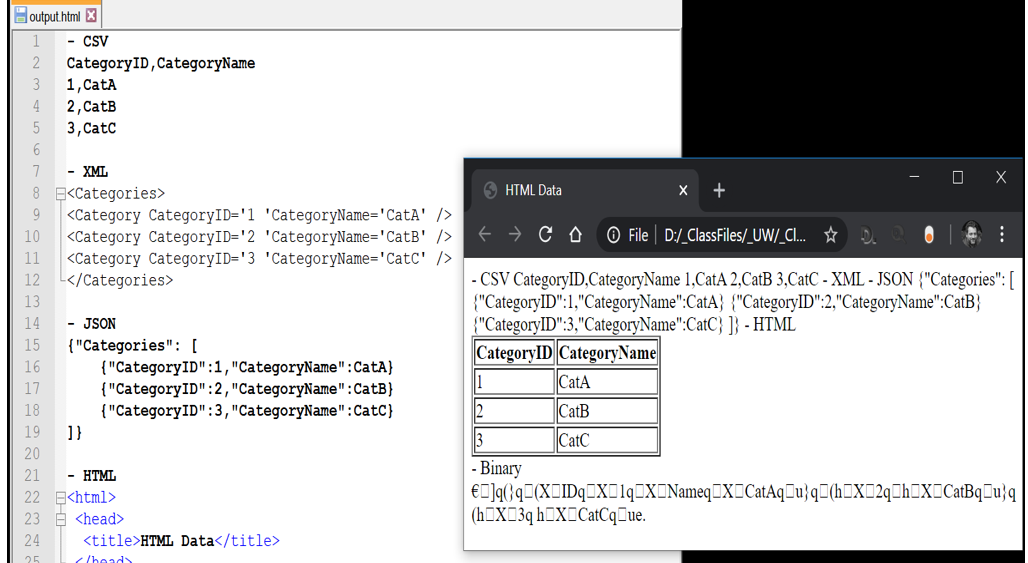
* Has **good enough performance** for **both data retrieval and transaction statements**
* **More** Joins and Subqueries
* **Storage of aggregate values is discouraged**
* You access tables **using the commonly known** **SQL** programming language

## Document Databases

Once you have generated data documents, you need someplace to store them. You can use a file server, but **searching** for data within the files **can be difficult**. The industry has been creating **databases designed especially for managing data files.**

### Data Documents

Over the years, data has been placed in documents using a **variety of formats**. The current popular ones include **CSV, XML, JSON, HTML, and binary**. **Some formats are useful in one application, but not in another**. Which one you use is **often determined by which application** you want to use the data with.



### Creating Documents

When people think of **documents**, they often think of **using a text editor or browsing PDF files**. However, more **often**, **applications create documents without a user ever knowing** aboutthe underlying program's code. This automation is convenient for most users, but **developers and administrators frequently need more control**.

Of course, the output can be created using many languages and technologies. Here is **an example using Microsoft's Transact SQL code**.

Begin Try

Use Master;

If Exists(Select Name From SysDatabases Where Name = 'OutputDemos')

Begin

Alter Database [OutputDemos] set Single\_user With Rollback Immediate;

Drop Database [OutputDemos];

End

Create Database [OutputDemos];

End Try

Begin Catch

Print Error\_Message();

End Catch

go

use OutputDemos;

go

Create Table Table1 (ID varchar(10), Name varchar(100))

go

Insert Into Table1("ID", "Name")

Values ('1', 'CatA'), ('2','CatB'), ('3','CatC')

go

With [CSV]

As (

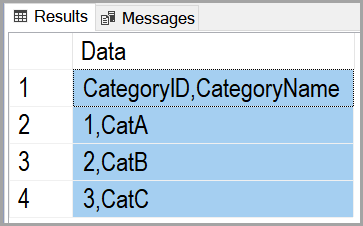
Select 'CategoryID' + ',' + 'CategoryName' as [Data], 1 as [Sort]

Union

Select "ID" + ',' + "Name", 2 as [Sort] From Table1 )

Select [Data] From [CSV] Order By [Sort];

go



With [XML]

As (

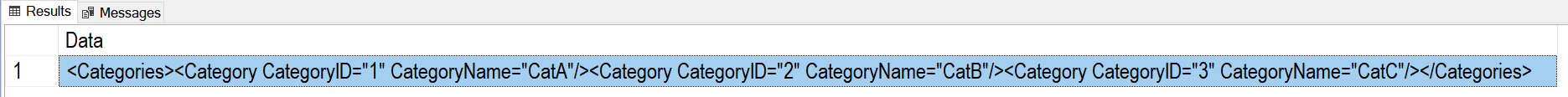
Select (Select "ID" as [CategoryID], "Name" as [CategoryName]

From Table1

For xml raw('Category'), root('Categories')) as [Data]

) Select [Data] From [XML];

go



With [JSON]

As (

Select '{"Categories":' as [Data], 1 as [Sort]

Union

Select (Select "ID" as [CategoryID], "Name" as [CategoryName]

From Table1

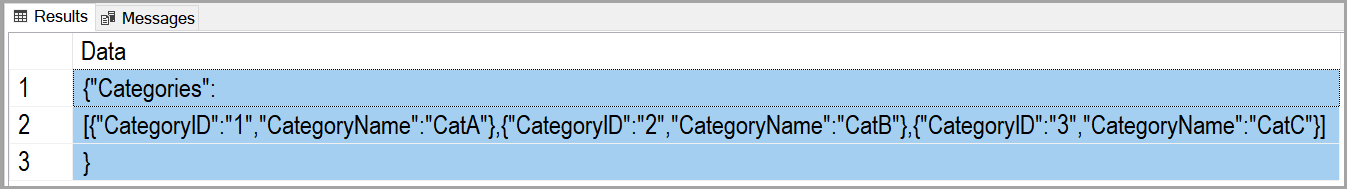
For JSON path), 2 as [Sort]

Union

Select '}', 3

) Select [Data] From [JSON] Order by [Sort]

go



With HTML

AS

(

Select '<html>' as [Data], 1 as [Sort]

Union

Select '<head><title>HTML Data</title></head>', 2 as [Sort]

Union

Select '<body>', 3 as [Sort]

Union

Select '<Table border="1">', 4 as [Sort]

Union

Select '<tr><th>CategoryID</th><th>CategoryName</th></tr>', 5 as [Sort]

Union

Select '<tr><td>' + "ID" + '</td><td>' + "Name" + '</td></tr>', 6 as [Sort]

From Table1

Union

Select '</table>', 7 as [Sort]

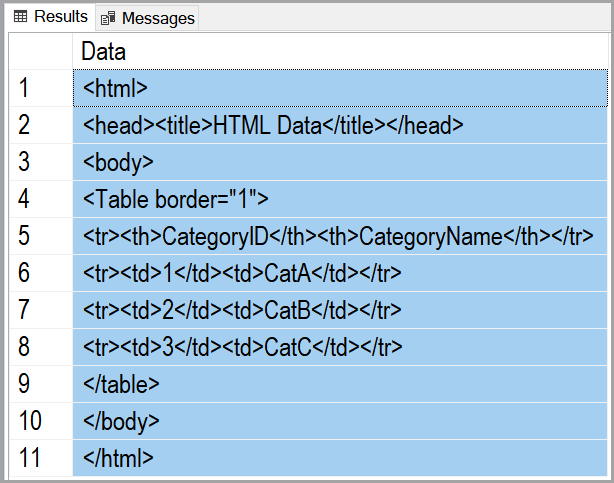
Union

Select '</body>', 8 as [Sort]

Union

Select '</html>', 9 as [Sort]

)Select [Data] From HTML Order by [Sort]



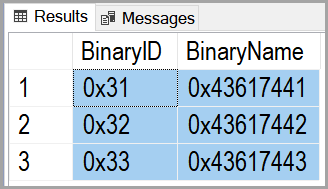
Declare @BinaryData Table([BinaryID] varbinary(10), [BinaryName] varbinary(100))

Insert Into @BinaryData([BinaryID], [BinaryName])

Select cast([ID] as varbinary(10)), cast([Name] as varbinary(100)) From Table1

Select [BinaryID], [BinaryName] From @BinaryData

go



### 

### MongoDB

MongoDB is **a document database that simple to use and free!**

* MongoDB is **open-source**
* MongoDB is **a document database** designed for OLAP
* MongoDB allows inserts, updates, and deletes, but **does not include strong transaction** **support**
* MongoDB **stores** data in **JSON-like documents**
* **Fields can vary from document to document**
* MongoDB can use **distributed databases** to speedup read performance

### Working with MongoDB

In this course, we will **use MongoDB as our example of a document database**. Here are some **common terms** you need to know as we get started.

## Terms

|  |  |
| --- | --- |
| Relational Databases | Mongo DB Document Databases |
| Database | ***Database*** |
| Table | ***Collection*** |
| Row | ***Document*** |
| Column/Key | ***Field*** |
| Joins | ***Nested fields in one document*** |
| Server | ***Software managing a set of databases*** |

### Databases

Each database stores a **set of collections**. The database always has **ONE Primary server** but can have a chain of **secondary servers for READ performance**. The **Primary is responsible for the logging changes**.

**Primary - > Secondary -> Secondary -> Secondary**

### Collections

Collections store **one or more documents**. Although Collections are similar to tables, you **cannot query collections in other databases**.

### Documents

**"**documents are a **set of fields**, **similar to JSON** objects. These fields **can be nested to store "one to many" related data**.

{

"\_id" : ObjectId("5e5c0ea333e82c880b5bb977"),

"ClinicID" : 101,

"ClinicName" : "GreenLake",

"ClinicPhoneNumber" : "206-123-1234",

"ClinicAddress" : "7210 Woodlawn Ave NE",

"ClinicCity" : "Seattle",

"ClinicState" : "WA",

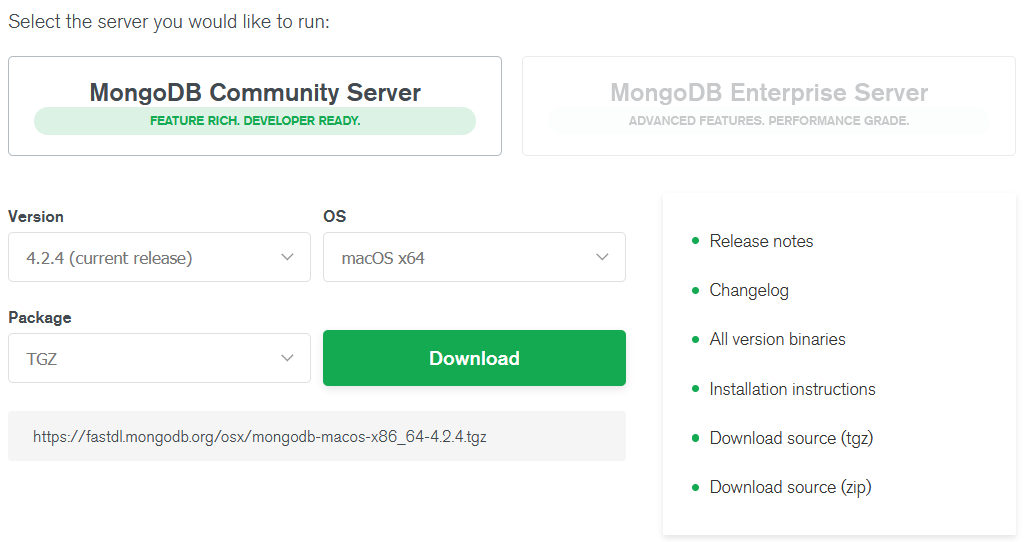
"ClinicZipCode" : "98115"

}

## How to get MongoDB

* **Mongo Atlas is their cloud-based server service**
* MongoDB **Community Edition and Enterprise are their on-site servers**
* MongoDB **Community Edition and Enterprise are both free (Enterprise recommend for tutorial)**
* Installs on **Windows, Mac, and other operating systems**
* **MongoDB can run as a service since version 4.0**

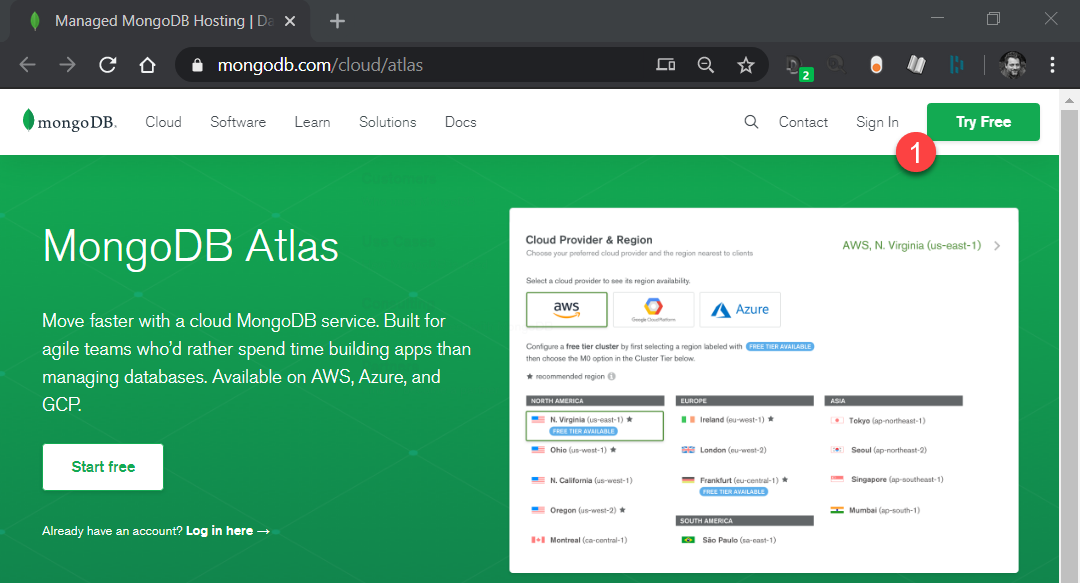
Installing MongoDB on **Windows** is easy since it includes a **simple installer program**. Installing on **Mac** is a bit **more complex** since the **installation is managed** using the popular **macOS package manager Homebrew**." (<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/>, 2017)



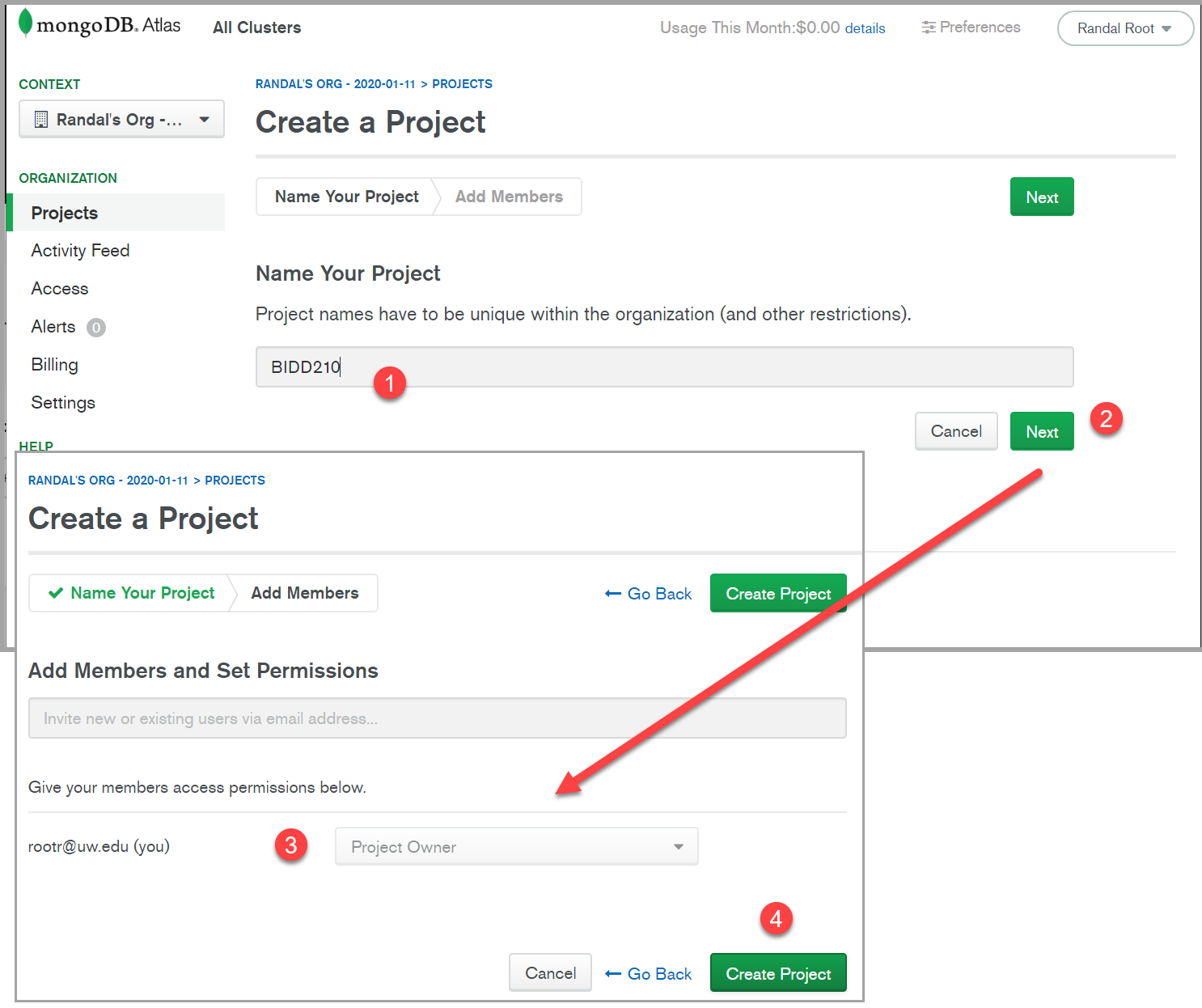
## MongoDB Atlas

*"MongoDB Atlas provides an* ***easy way to host and manage your data in the cloud****. This tutorial guides you through creating an Atlas cluster, connecting to it, inserting data, and querying data." (*[*https://docs.atlas.mongodb.com/getting-started/*](https://docs.atlas.mongodb.com/getting-started/)*, 2020)*

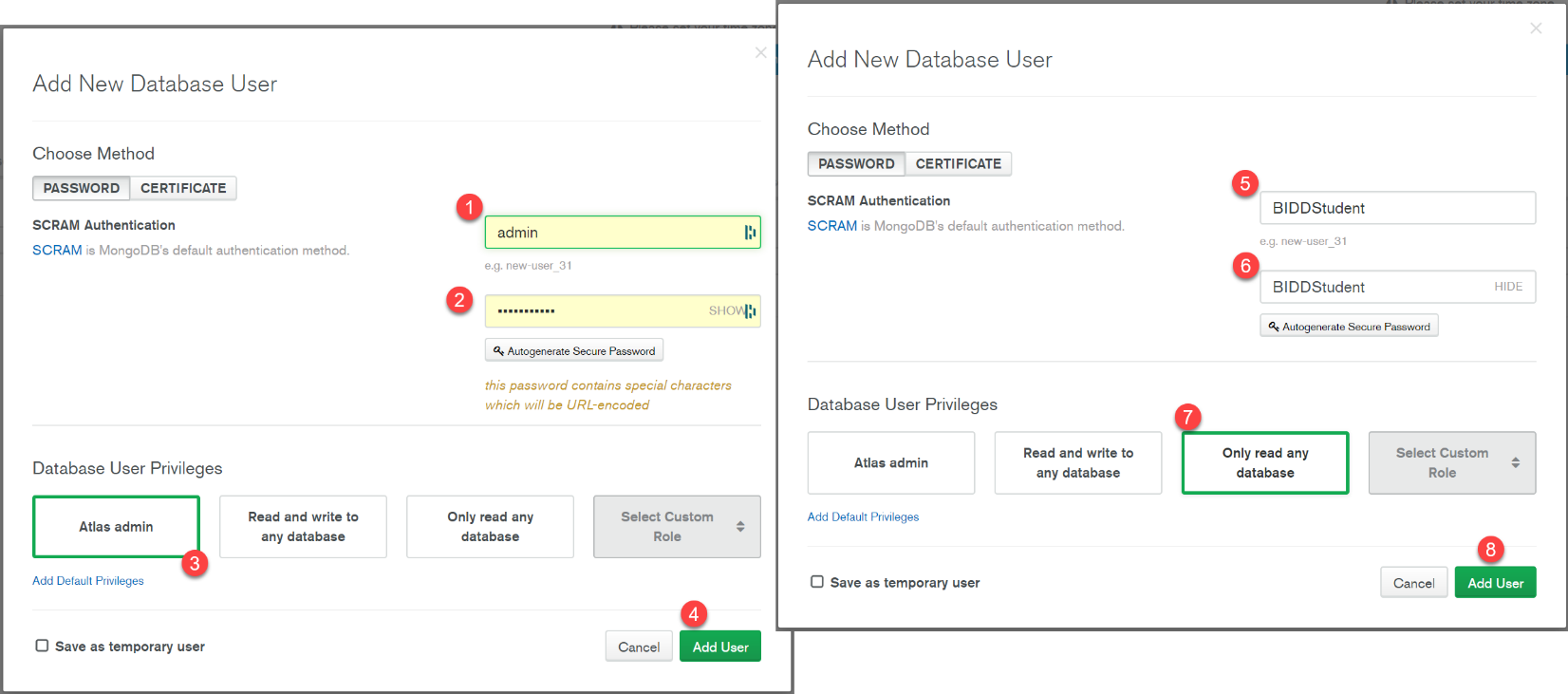
1) Login to MongoDB Atlas: <https://www.mongodb.com/cloud/atlas>



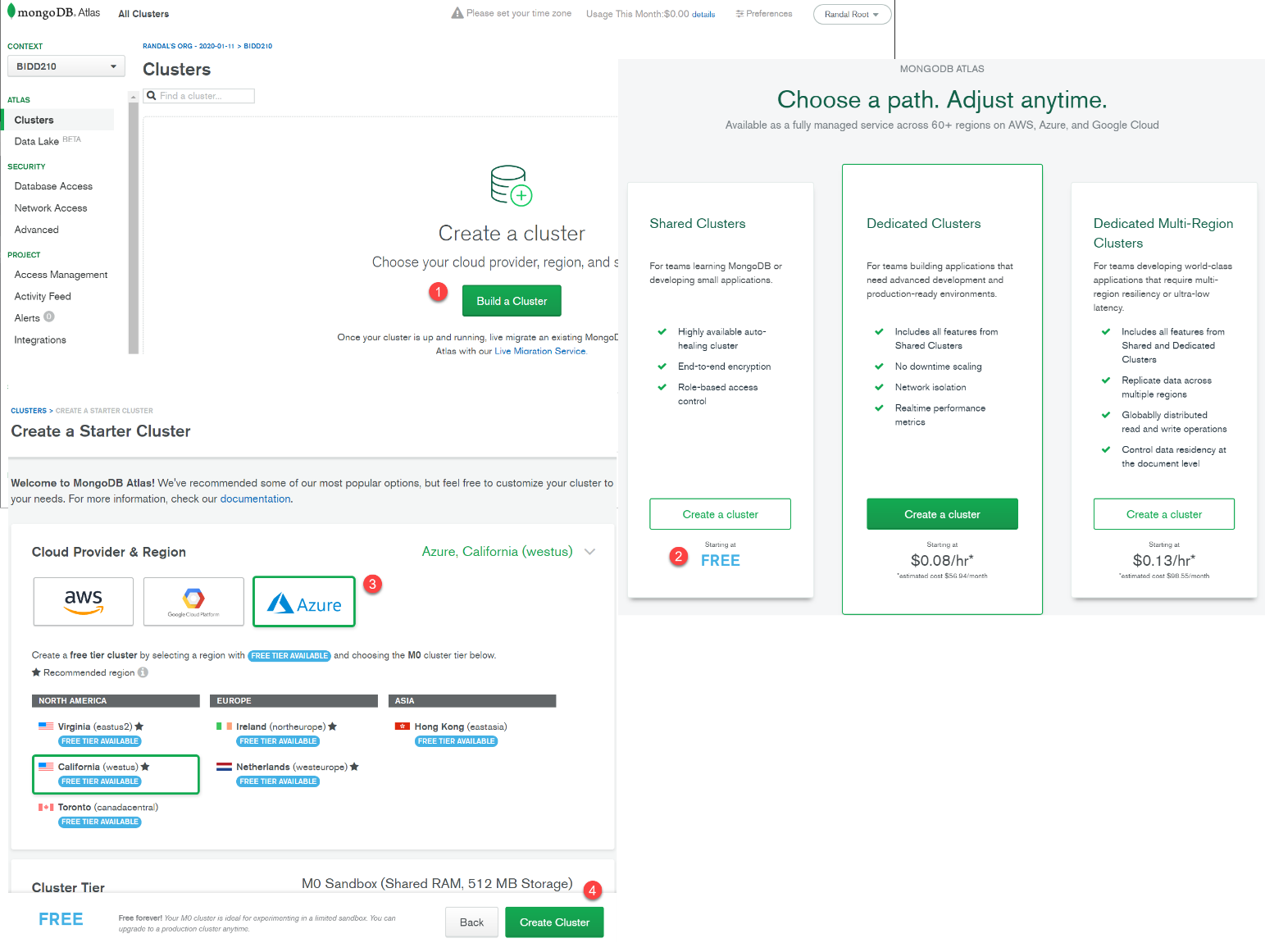
2. Create a new Project



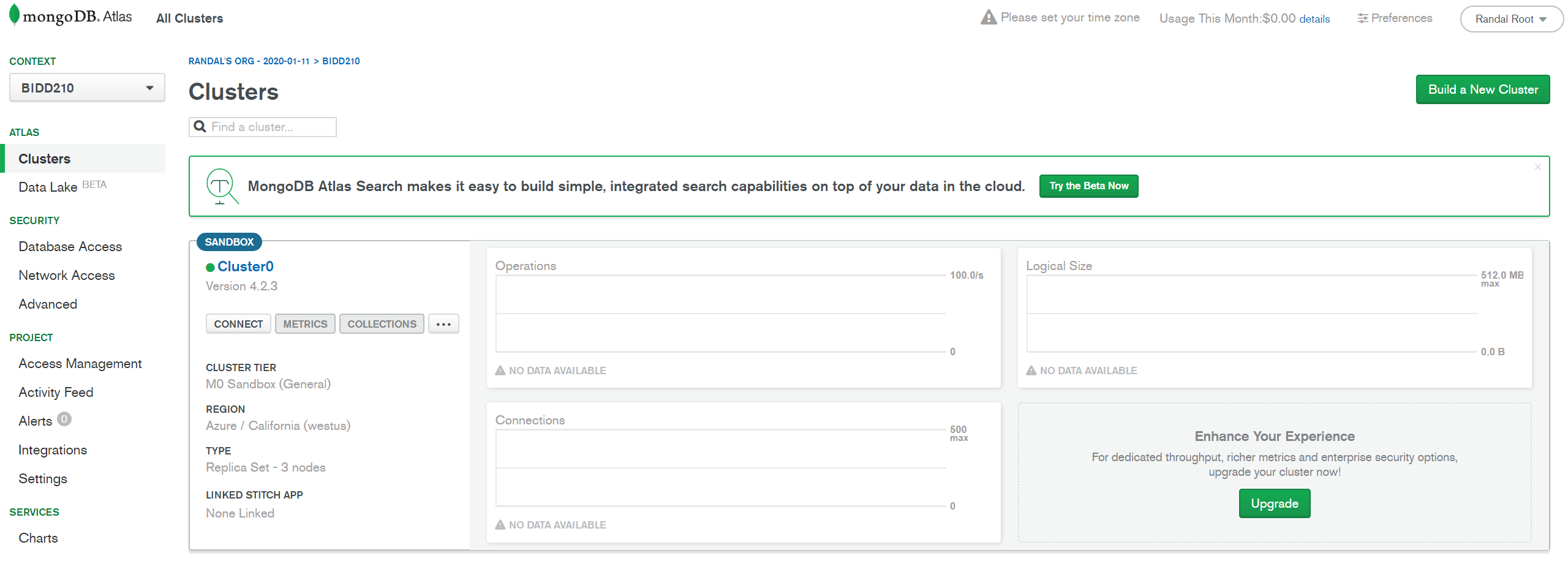
3. Add Members.

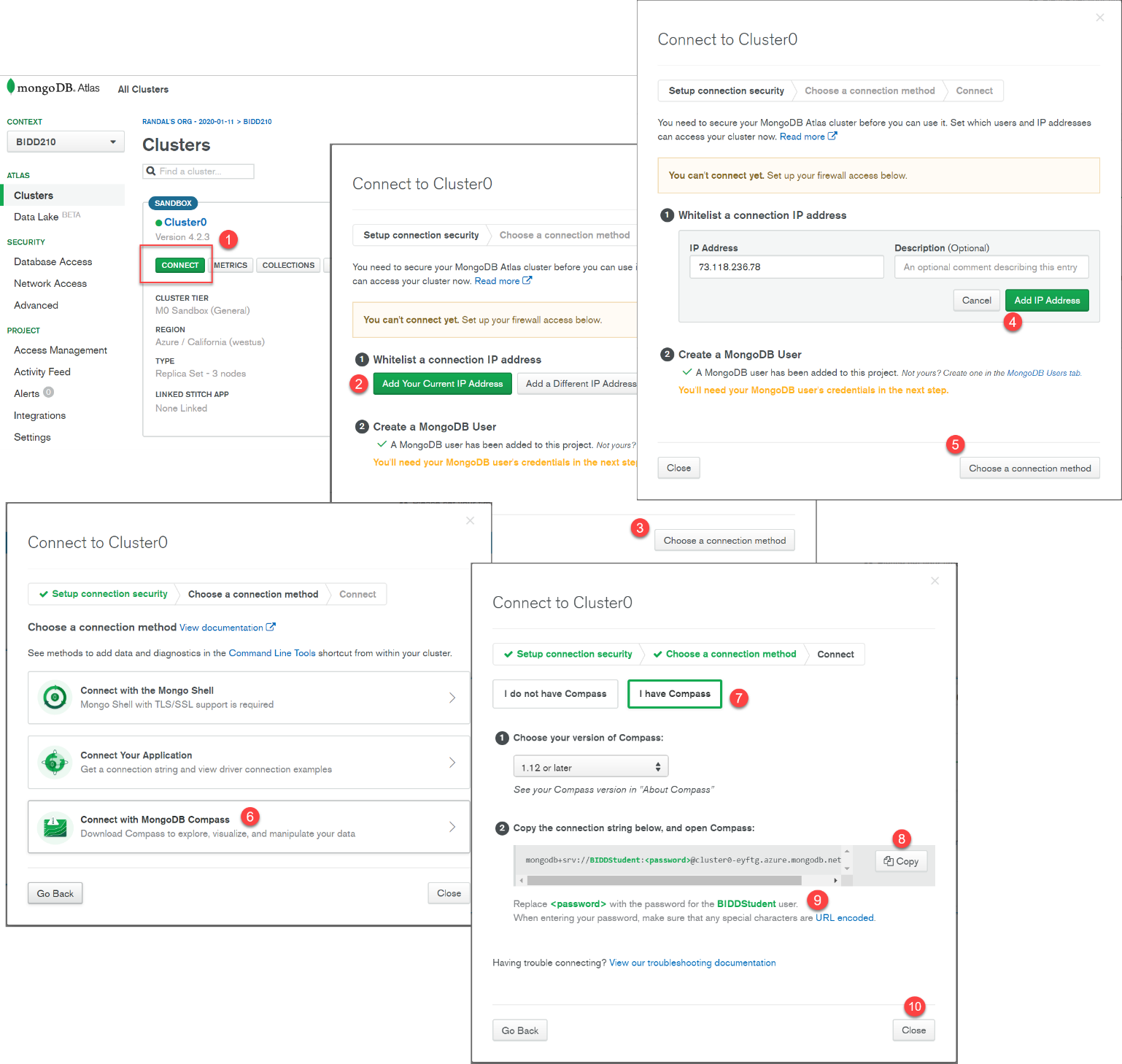


4. Choose which cloud platform will host the new database's Cluster.



5. Wait for the Sandbox window to appear.

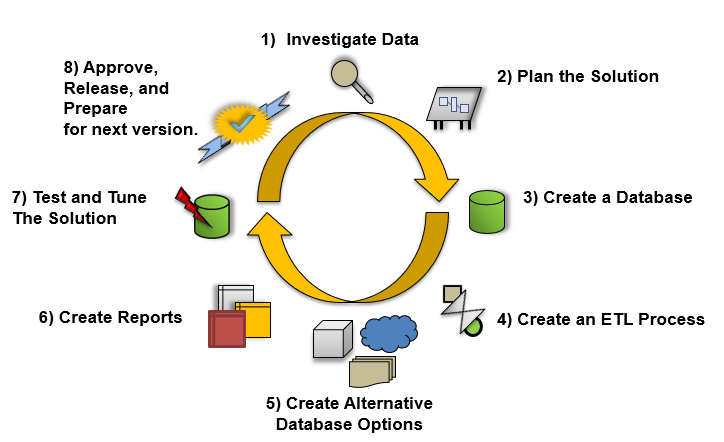


6. Add your IP Address for access to the database's Cluster.

**Important:** You can open connection from Any IP, but make sure you never do so for your admin account or you could end out with $$$$ charges!

## Completing Solutions

The **process** of creating a database solution can be seen as a **series of eight steps**. In this course, we are finally approaching the end of these.



The practice of **testing and tuning constitutes a final step** in this process. It is arguably the most crucial step because a solution is worthless if it is not functioning correctly.

* For **testing**, the goal is to **validate** the current solution through objective verification.
* For **tuning**, the goal is to provide possibilities for enhancements by **benchmarking** the current performance, **identifying** poorly performing components, and **recommending** improvements.

### Testing

Testing a solution involves documenting what the solution **should contain**, verifying **what it contains**, and identifying **possible** areas of **improvement**.

We have often **determined the database's design using a metadata** **Excel spreadsheet**. Now that the solution has been created, we could **pass the Excel spreadsheet onto the test team for validation**. The test team would then go through the columns and configurations noted in the documentation, **verifying** that everything had **been implemented**.

In most cases, solutions start without much input from testers. But, an **effort should be made to formalize an ongoing test process** so that the test team can become more experienced with the solutions you are making. Doing so will improve each version as it iterates through the solution process. **Ideally, the test team would have been brought in during the architectural phase to give input**. The more experienced the test team is with this type of software, the more useful their input is during the architectural phase of your solution.

When **developers think of testing**, they often think of a person or a piece of software that examines the input versus the output **to ensure validity**. For a **database solution**, the input is the **values inserted** into the columns of the database that are later **output via reports**. **The tester's goal is to ensure that, given a certain input, an appropriate output is always obtained**. This validation is the core of the testing process.

This testing is not always as straightforward as it sounds. **How are the testers to know what a valid output of a given input is**? Is it implicit through the type of data that is used? Is the output explicitly indicated with some sort of documentation that can be compared to the results?

Many output results can be determined by examining the values that were input into the columns within a database, but that is not always the case. For **example**, if the **input** is an author's **first and last names** and the **output is a concatenation**, the operation would change the author's first and last names into one singular name. **Incorrect output** might display as follows:

* Leaving **no space** between the first and last names
* The last name **preceding** the first
* The first or last name **missing**

In **this** **example**, your test team should be able to **easily** establish what the input and output should be and then verify what is needed by providing the correct information when the reports are generated.

Now let's think of something a **little bit more difficult**. What about a **complex computation** that provides statistical deviations against sales data? In a situation like this, the equation can consist of three or four numerical values along with several operations. The **output** would be a **specific value** that is **associated with the algorithm** defined by the process for this calculation. In these cases, **proofs must be provided by people that specialize** in this kind of analytics.

Of course, this means that more **documentation must be generated for the test team** to efficiently do their job. Documenting can become an issue as management tries to **coordinate the documentation effort in conjunction with the testing effort**.

Keep in mind that ***something* is much better than *nothing***. All too often, testing is left out of the process because it is believed to be too costly to implement on small solutions. From our example, you can see that this is a mistaken assumption. **Something as simple** as an Excel spreadsheet will work:



### Objective Verification

Objective verification is **possible** **only if** the **objectives** of the solution have been **clearly defined**. In a perfect world, the development process clearly identifies the goals at the beginning of the solution. In **reality**, **documentation will likely be rather sketchy and incomplete**.

Most team members realize that **good documentation** will make the test teams' job that much more effective, but time and resources can limit the chances of this taking place.

Here are some **suggestions** **to keep consistency and verify objectives**:

* **Create** a **standard template** for documentation
* **Make** sure the standard template is **easy to follow**
* **Make** sure the standard template **does not take long to complete**
* **Make** sure that developers **update the documents with lessons learned**
* **Have** **testers** **review** the documentation **before implementation** begins, where possible
* **Have** a professional **technical writer** **review** and **enhance the documentation**

### Performance Options

Performance options may improve a database solution's ability to **perform its functionality in a minimal amount of time**. Ideally, each process, from **running the ETL code** to **rendering the reports**, should be performed as **quickly as possible** so that anyone requesting information will not have to wait.

The ETL operation extracts data from one location, transforms it, and loads it in another location. In general, tuning **ETL performance** revolves around making **Select, Insert, Update, and Delete statements more efficient**. **Report** **performance** revolves around making **Select more efficient**. So, to tune a solution, **start by increasing Select performance.**

**Select performance** can be improved in several ways. Still, the most **common improvement** comes from providing **indexes,** adding **hardware** resources**, or reducing the number of resources** needed to perform the same action.

#### Hardware Options

**Additional** hardware can include adding more random-access memory (**RAM**), additional **processors**, or extra **hard drives**.

Of the three, **RAM is the most vital**. Whenever SQL Server selects data from a table, it must pull the table's data into RAM before it returns the result set. If there is **insufficient RAM**, SQL Server will have to **temporarily use the hard drive as a storage base**, which is substantially slower than if RAM was available. While SQL Server performs these actions quite efficiently, adding RAM will provide you with a quick and easy way to improve your select performance.

Adding additional **processors** can **sometimes help** SQL Server improve select performance as well. Whether this happens is **dependent upon the complexity** of the SQL query **and what other software** is on the same computer as your SQL Server.

Adding **additional hard drives** is a common technique for improving performance. The practice of configuring two or more hard drives to act as a single hard drive is referred to as a Redundant Array of Independent Disks (**RAID**). It is usually referred to as a RAID array.

The **three common types** of RAID configurations are striping (**RAID 0**), mirroring (**RAID 1**), and mirroring with striping **(RAID 1+0**).



RAID is a collection of **two or more hard drives** that are made to **look like a single hard drive**. Your **software**, such as SQL Server, **does not know there are multiple hard drives involved**.

* A **RAID 0 stripe** places some parts of your data on one drive and other components on a different hard drive. **If one of the drives fails, you have only part of your data**!
* A **RAID 1 mirror** keeps a real-time copy of your data. **If a drive fails, you still have the copy**.
* **RAID 1+0 combines both mirroring and striping** to give you the best of both worlds! You are not restricted to only two drives; in fact, you need at least four to start with, and it includes the real-time copy of your data.

This simple addition will increase performance, but **it can be improved on**. For instance, you can buy more hard drives, create an additional RAID array, and **move the database to a different drive than the one hosting the operating system**. The idea is that the OS has one set of hard drives, and the database has another set of hard drives, and they are not contending for resources.



#### Indexing Options

Adding indexes to your database tables can increase select performance. Because report data is often retrieved from many tables using a **SQL JOIN statement**, creating indexes on columns commonly used in these joins makes sense. **Typically, these columns are foreign keys columns**, but not exclusively. It is best to **keep track of which columns are queried** in the reports.

#### Archiving Stored Data

One of the more effective ways of increasing performance is **reducing the amount of data stored** in either the database. Failing to do so is one of the significant performance bottlenecks in any solution. The basic concept is simple; as time goes by, data that was **once important is no longer and can be safely achieved**. For example, consider a solution that focuses on sales data and stores data from the year 1990 to today. **Sales trends from the 1990s are no longer relevant to today's market**, so while the best choice may not be to delete the data, this question arises: "Do I really need this data in the database?" If the answer is no, then it is a good idea to archive the data from the solution.

#### Caching Report Data

Reporting software may include **caching options** that allow you to **retrieve report data directly from the cache without having to repeatedly re-create the reports**. These options provide a great degree of performance and should be considered whenever report performance is slow.

## INFO 330 Introduction to Relational Database Management Systems

Now is an excellent place to end our discussion of the nature of databases. While there is so much more to learn, you currently should have **enough foundational knowledge to begin** that journey, if you so desire.

**We've learned** about databases from a practical standpoint with Hands-On activities and assignments throughout the quarter. We've touched on the theoretical and conceptional topics as needed, but our focus was always on **how you use data in your daily life and how databases affect you**.

### Course Description

"Introduction to relational database management systems focused on relational theory and the application of conceptual, logical, and physical database modeling. Key topics include the **relational model, SQL, entity-relationship modeling, three-tier architectures, implementation of database applications, and related topics** in information systems." (<http://www.washington.edu/students/crscat/info.html#info340>, 2017)

Whereas before this class, the following diagram would not have made much sense to you, **please take a moment to consider it now and realize how all your hard work has paid off**!

## 