University of Washington

iSchool Info 330

# Module 10 – Cloud Databases and Other Technology

In this module, we will look at how database projects are completed and various common options that are applied to database as they mature**.**

## Outline

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

|  |
| --- |
| **Module10: Cloud DBs, Hadoop, and other No-SQL Technologies** |
| Session01 Lectures and Labs < 110 mins |
| Completing Solutions - 50 |
| Host a Database in the Cloud - 30 |
| Lab 1: Host a Database in the Cloud - 20 |
| Session02 - Lab |
| Final Project - Milestore04 |
| Session03 Lectures and Labs < 110 mins |
| Cloud Technologies - 20 |
| No-SQL - 20 |
| Hadoop - 20 |
| Cosmos - 20 |

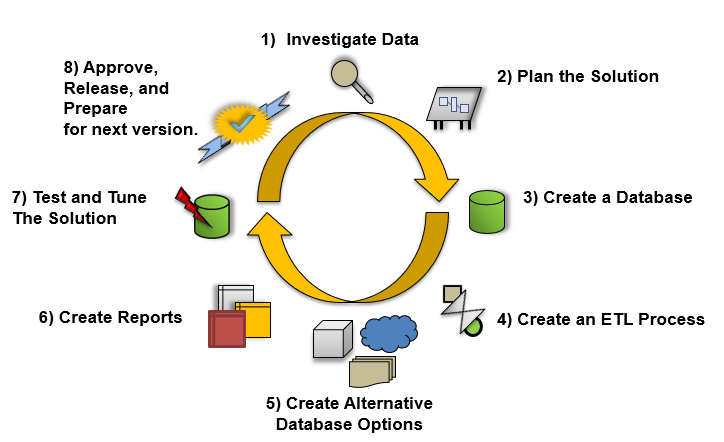
**Note**: Times are only estimates and may change without notice!

# Session01 < 110 mins

In this session, we start to examine how database **solutions and their individual projects end** and some **options to consider for future** versions.

## Completing Solutions - 50

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 and is arguably the most important step, because a solution is worthless if it is not functioning properly.

* 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 is does contain**, and identifying **possible** areas of **improvement**.

When we **began**, we determined the database's contents using an **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 each individual column, attribute, or configuration noted in the documentation and **verify** that what was planned **has indeed 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 an certain input, an appropriate output is always obtained**. This validation is the core of the testing process.

This 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 a number of operations. The **output** would be a **specific value** that is **associated with the algorithm** defined by the operation 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 in order for the test team** to efficiently do their job. This 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 option 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 perform 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 and Update 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 a number of ways, but most **commonly improvement** comes from providing **indexes,** adding **hardware** resources**, or reducing the amount of resources** needed to perform the same action.

#### Hardware Options

**Additional** hardware can include adding more random-access memory (**RAM**), additional **processors**, or additional **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 parts 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 major 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.

## Host a Database in the Cloud - 30

" Simply put, cloud computing is the delivery of computing **services—servers, storage, databases, networking, software, analytics, and more**—over the **Internet** (“the cloud”). Companies offering these computing services are called cloud providers and typically charge for cloud computing services based on usage, **similar to how you’re billed for water or electricity at home**." (<https://azure.microsoft.com/en-us/overview/what-is-cloud-computing>, 2017)

"**Top benefits of cloud computing**

Cloud computing is a big shift from the traditional way businesses think about IT resources. What is it about cloud computing? Why is cloud computing so popular? Here are **6 common reasons organizations are turning to cloud** computing services:

1. **Cost**

Cloud computing **eliminates** the capital expense of **buying hardware and software** and setting up and running on-site datacenters—the racks of servers, the round-the-clock electricity for **power and cooling, the IT experts** for managing the infrastructure. It adds up fast.

2. **Speed**

Most cloud computing services are provided **self service and on demand**, so even vast amounts of computing resources can be provisioned in minutes, typically with **just a few mouse clicks**, giving businesses a lot of flexibility and taking the pressure off capacity planning.

3. **Global** **scale**

The benefits of cloud computing services include the **ability to** **scale elastically**. In cloud speak, that means delivering the right amount of IT resources—for example, **more or less** computing power, storage, bandwidth—right **when its needed**, and from the right geographic location.

4. **Productivity**

On-site datacenters typically require a lot of “racking and stacking”—hardware set up, software patching, and other time-consuming **IT management chores**. Cloud computing removes the need for many of these tasks, so IT teams can spend time on achieving more important business goals.

5. **Performance**

The biggest cloud computing services run on a worldwide network of secure datacenters, which are regularly upgraded to the latest generation of **fast and efficient computing hardware**. This offers several benefits over a single corporate datacenter, including **reduced network latency for applications** and greater economies of scale.

6. **Reliability**

Cloud computing makes data backup, disaster recovery, and business continuity easier and less expensive, because data can be **mirrored at multiple redundant sites** on the cloud provider’s network."

(<https://azure.microsoft.com/en-us/overview/what-is-cloud-computing>, 2017)

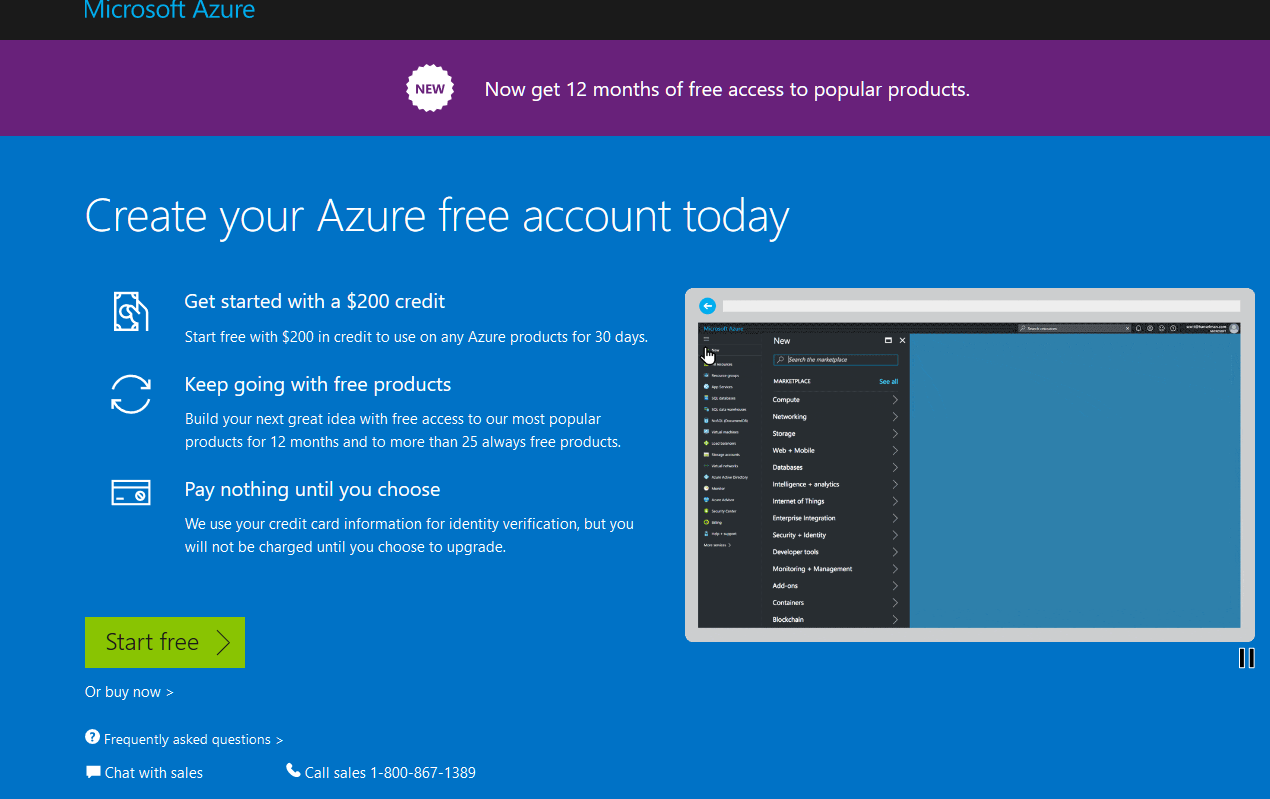
### Microsoft

"**SQL Database** is a general-purpose relational database service **in** Microsoft **Azure** that **supports structures such as relational data, JSON, spatial, and XML**. It delivers dynamically scalable performance and provides options such as columnstore indexes for extreme analytic analysis and reporting, and in-memory OLTP for extreme transactional processing. **Microsoft handles all patching and updating** of the SQL code base seamlessly and abstracts away all management of the underlying infrastructure. +

SQL Database shares its code base with the Microsoft SQL Server database engine. With Microsoft's cloud-first strategy, the newest capabilities of SQL Server are released first to SQL Database, and then to SQL Server itself. This approach provides you with the newest SQL Server capabilities with no overhead for patching or upgrading - and with these new features tested across millions of databases. (<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-technical-overview>, 2017)

**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) **NOTE: Show Students!**



### Amazon

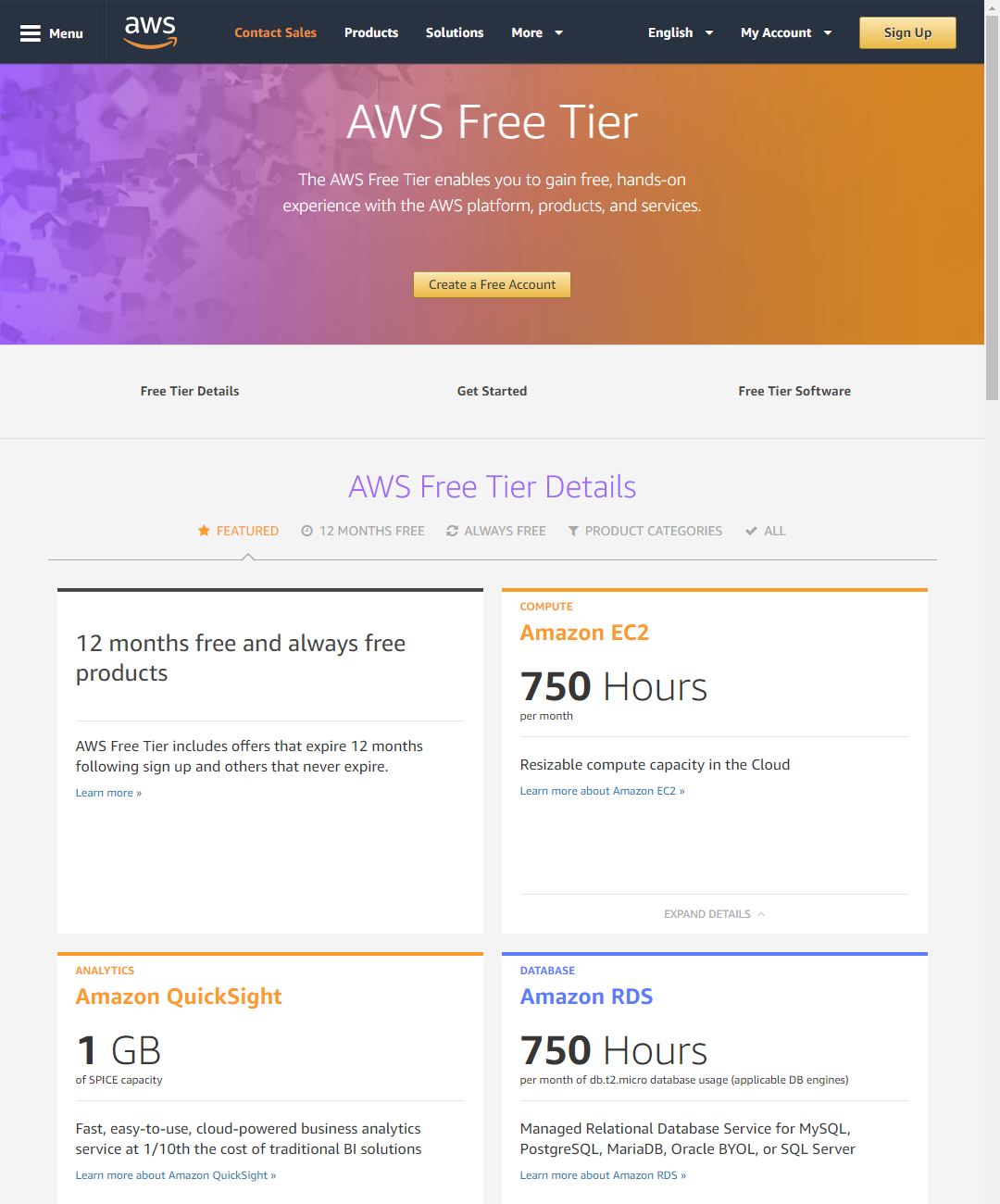
"Amazon Relational Database Service (Amazon RDS) makes it **easy to set up, operate, and scale** a **relational database in the cloud**. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such **as hardware provisioning**, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.

Amazon RDS is available on several database instance types - **optimized for memory, performance or I/O** - and provides you with six familiar database engines to choose from, including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, and Microsoft SQL Server. You can use the AWS Database Migration Service to easily migrate or replicate your existing databases to Amazon RDS."( <https://aws.amazon.com/rds/>, 2017)

**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) **NOTE: Show Students!**



## Lab 1: Hosting a Database in the Cloud - 20

In this lab, you will use create a database using Amazons Cloud services.

**Note**: Since this is a complex process**, your instructor will walk you through it**.

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

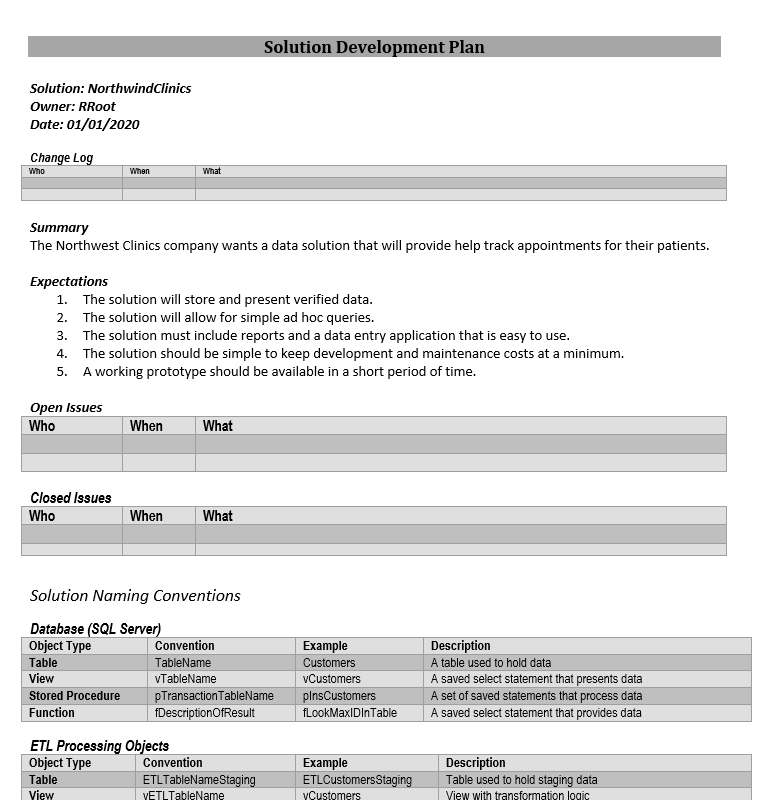
# Session02 Lectures and Labs < 50 mins

Lab 2: Final Project - Milestone04

In this lab you will 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.

In this Milestone you have two tasks to perform. The **first** is the creation of a **formal project development document for your solution**.

I have provided you with a sample of a solution development planning. You can you will need to **remove the existing entries** and **add your entries**, but it should provide you with a convenient format in which to enter in your project's meta-data.



The **second task** requires that you create a **document** about the **lessons you have learned throughout this course**. This can be a **relatively informal overview** of the different things you have learned during the quarter. It does not have to be an exhaustive enumeration of these items but should include at least two or more different aspects of creating and working with a relational database.

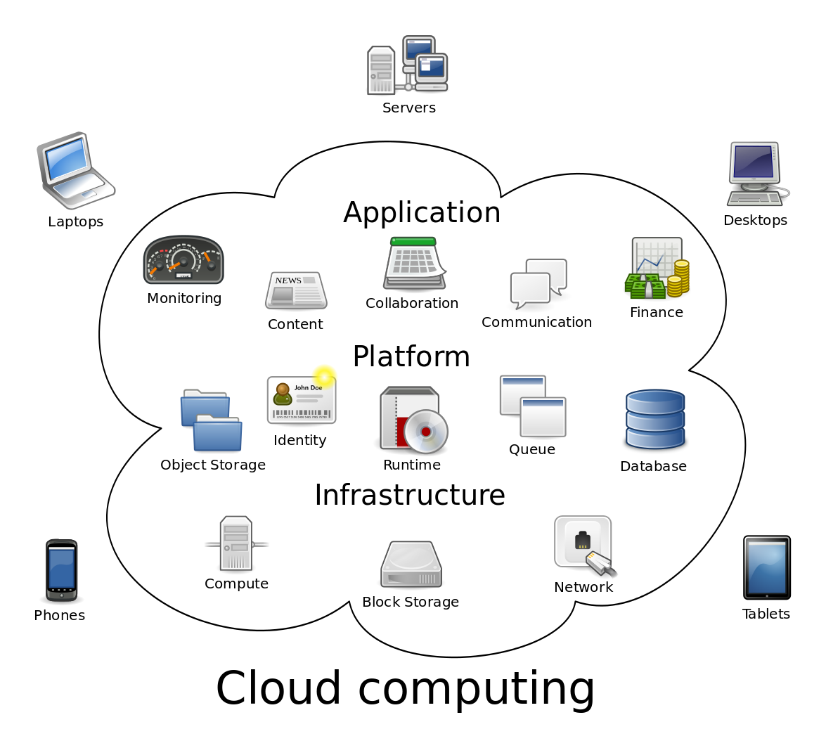
# Session03 Lectures and Labs < 110 mins

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

## Cloud Technologies - 20

*"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**) was created.

**1980s** Private networks are more common than the Internet (APRANet).

**1990s** Private networks are more common than the Internet, but both are used. Tunneling though the internet to connect private networks becomes common. Access speed is slow and not universal.

**2000's** Private networks are more common, but people use the Internet daily. Access speed has increased, but connection is still not universal. Application Service Provides startup are the new "hot" investment, but many fail.

* In 2006, Amazon opens "Elastic Compute Cloud."
* In 2008, Google opens Google App Engine
* In 2010, Microsoft opens Microsoft Azure
* In 2011, IBM opens SmartCloud
* And so on...

It seems clear that as long as we have cheap and near universal access to the Internet that cloud computing is here to stay!

#### Scalability

Ability to meet the demands of varying workloads. Quickly add more resources as needed. This is known as **'scale-up' versus 'scale-out.'**

* Scale-up = Adding additional resources to single server; **CPUs, Memory, Storage**
* Scale-out = Adding **additional servers** to application (more flexible than Scale-up)

Scale-Up are not dynamic and often require downtime when adding components! Scaled-out databases are designed to work across multiple servers seamlessly making adding/dropping servers dynamic, with minimal intervention.

#### Cost

* Hardware
* Software
* Licenses were based on number of concurrent users
* Spikes have a potential for excessive charges

## No-SQL - 10

No-SQL or Not-Only-SQL, is simply another type of structured database.

*"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 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)

## Hadoop - 10

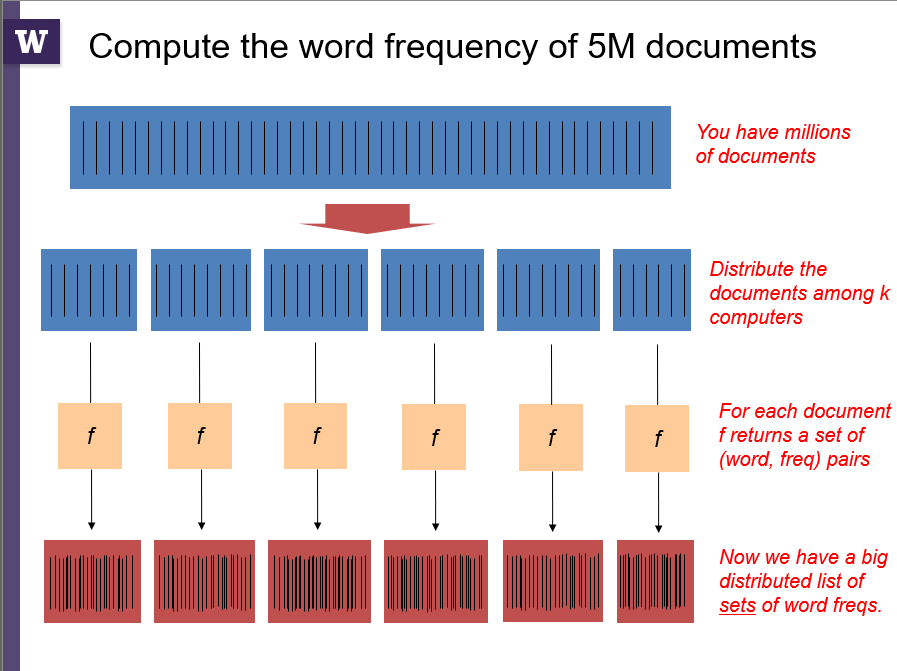
*"Apache Hadoop (*[*/həˈduːp/*](https://en.wikipedia.org/wiki/Help:IPA/English)*) is an*[*open-source*](https://en.wikipedia.org/wiki/Open_source)[*software framework*](https://en.wikipedia.org/wiki/Software_framework)*used for*[*distributed storage*](https://en.wikipedia.org/wiki/Clustered_file_system)*and processing of*[*dataset*](https://en.wikipedia.org/wiki/Dataset)*of*[*big data*](https://en.wikipedia.org/wiki/Big_data)*using the*[*MapReduce*](https://en.wikipedia.org/wiki/MapReduce)[*programming model*](https://en.wikipedia.org/wiki/Programming_model)*. It consists of*[*computer clusters*](https://en.wikipedia.org/wiki/Computer_cluster)*built from*[*commodity hardware*](https://en.wikipedia.org/wiki/Commodity_hardware)*. All the modules in Hadoop are designed with a* ***fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework****.*

*The core of Apache Hadoop consists of a storage part, known as* ***Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model****. Hadoop* ***splits files into large blocks and distributes them across nodes in a cluster****." (*[*https://en.wikipedia.org/wiki/Apache\_Hadoop*](https://en.wikipedia.org/wiki/Apache_Hadoop)*, 2017)*

*"Mapreduce is a framework for* ***processing big data in******two phases Map & Reduce****. Both the phases have key-value pairs as input and output.*

*Map phase implements* ***Mapper function****, in which user-provided code will be executed on* ***each key-value pair*** *(k1, v1) read* ***from the input******files****. The output of the mapper function would be zero or more key-value pairs (k2, v2) which are called intermediate pairs. Here the key is what the data will be grouped on and the value is the information related to the analysis in the reducer.*

***Reduce*** *phase* ***takes mapper output*** *(grouped key-value data) (k2, v2) and runs reduce function on each key-value group. reduce* ***function iterates over the list*** *of values associated with a key and* ***produces outputs like aggregations****,* ***statistics******etc****.. Once the reduce function is done, it sends zero or more key-value pairs (k3, v3) to the final the output file." (*[*http://hadooptutorial.info/mapreduce-programming-model/*](http://hadooptutorial.info/mapreduce-programming-model/)*, 2017)*



From Bill Howe's University of Washington INFX 445 class

## Cosmos - 10

*"Azure Cosmos DB is Microsoft’s proprietary globally-distributed,* ***multi-model database service*** *"for managing data at planet-scale" launched in May 2017.[1] It builds upon and extends the earlier Azure DocumentDB, which was released in 2014.[2] It is schema-less and generally classified as a NoSQL database."*

*...*

*A*[*JavaScript engine*](https://en.wikipedia.org/wiki/JavaScript_engine)*is embedded in Cosmos DB. This is a perfect fit for JSON documents, but it also* ***enables additional functionality****:*

* ***Stored procedures****. Functions that bundle an arbitrarily complex set of operations and logic into an*[*ACID*](https://en.wikipedia.org/wiki/ACID)*-compliant transaction. They are isolated from changes made while the stored procedure is executing and either all write operations succeed or they all fail, leaving the database in a* ***consistent state****. Stored procedures are executed in a single partition. Therefore, the caller must provide a partition key when calling into a partitioned collection. Stored procedures can be used to make up for the lack of certain functionality. For instance, the lack of aggregation capability is made up for by the implementation of an*[*OLAP*](https://en.wikipedia.org/wiki/Online_analytical_processing)*cube as a stored procedure in the open sourced documentdb-lumenize[[5]](https://en.wikipedia.org/wiki/Cosmos_DB" \l "cite_note-5) project.*
* ***Triggers****. Functions that get executed* ***before or after specific operations*** *(like on a document insertion for example) that can either alter the operation or cancel it.*
* ***User-defined functions (UDF)****. Functions that can be called from and augment the SQL query language making up for limited SQL support."*

*(*[*https://en.wikipedia.org/wiki/Cosmos\_DB*](https://en.wikipedia.org/wiki/Cosmos_DB)*, 2017)*

"Cosmos DB also **doesn't force a commitment to a conventional column-style, key/value, or document-based paradigm**. Existing NoSQL systems like MongoDB can use Cosmos DB as a storage back end, or Cosmos DB can be queried by conventional SQL. " (<https://www.infoworld.com/article/3195836/database/why-microsofts-cosmos-db-represents-the-future-of-cloud-databases.html>, 2017)

## MongoDB - 20

**"What is MongoDB?**

MongoDB is **a document database** with the scalability and flexibility that you want with the querying and indexing that you need.

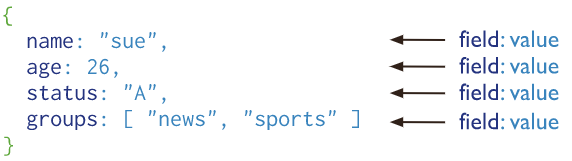
* MongoDB **stores** data in flexible, **JSON-like documents**, meaning fields can vary from document to document and data structure can be changed over time
* The **document model maps to the objects** in your application code, making data easy to work with
* **Ad hoc queries, indexing, and real time aggregation** provide powerful ways to access and analyze your data
* MongoDB is a **distributed database** at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use
* MongoDB is **free and open-source**, published under the GNU Affero General Public License

...

(<https://www.mongodb.com/what-is-mongodb>, 2017)

"Document Database

**A record in MongoDB is a document**, which is a data structure composed of **field and value pairs**. MongoDB documents are similar to JSON objects. The values of **fields may include other documents, arrays, and arrays of documents**."

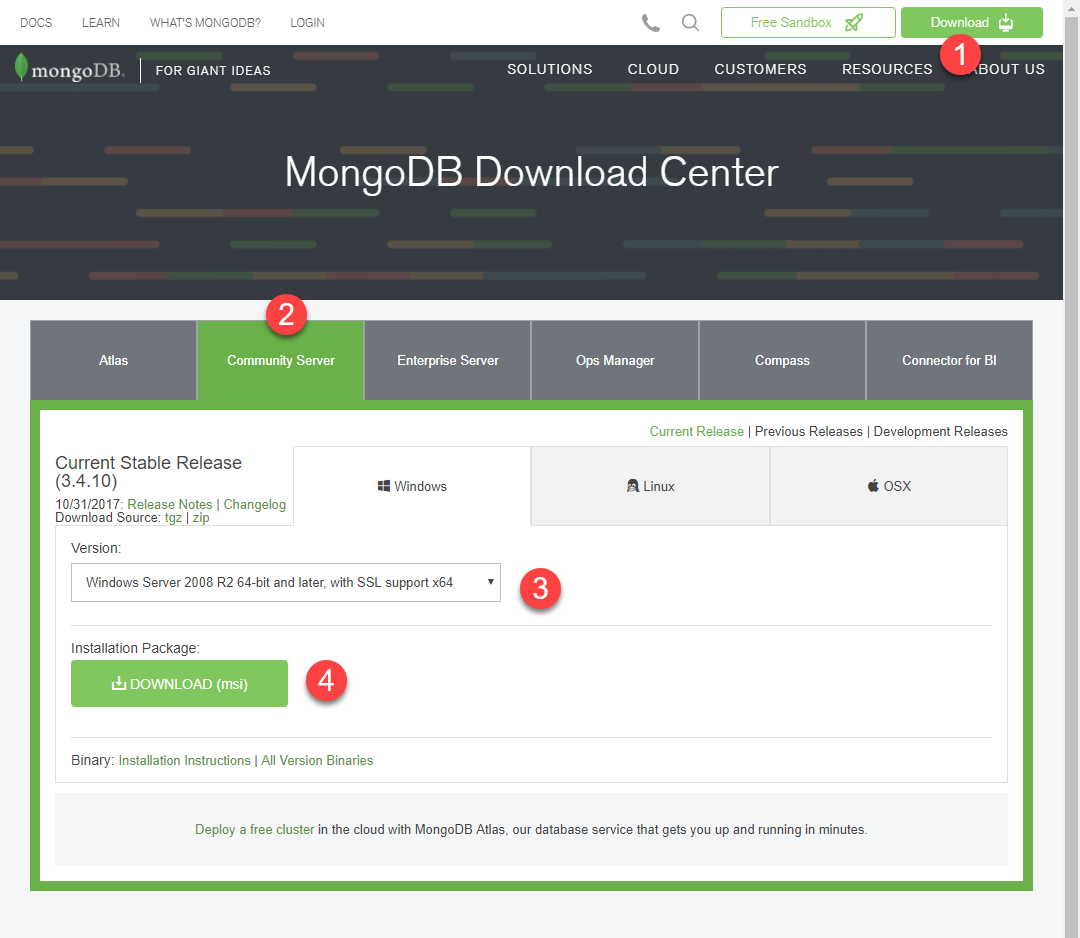


(<https://docs.mongodb.com/manual/introduction/>, 2017)

### Installing MongoDB

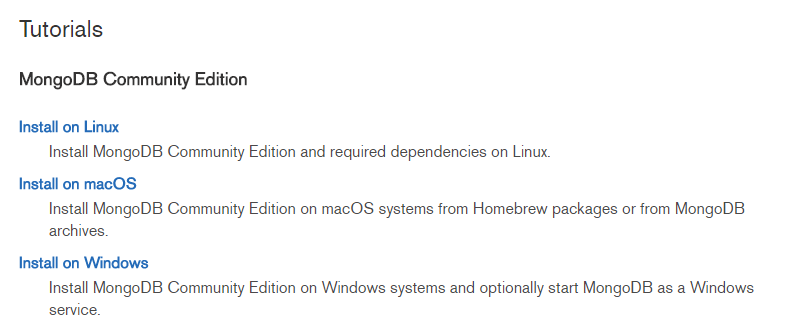
"MongoDB Community Edition requires **Windows** Server **2008** R2, Windows **Vista**, **or** **later**. The .msi **installer includes all other software dependencies** and will automatically upgrade any older version of MongoDB installed using an .msi file." (<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>, 2017)

"You may download MongoDB Community Edition through either the MongoDB Download Center or the popular macOS package manager Homebrew." (<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/>, 2017)



"MongoDB Download Center"

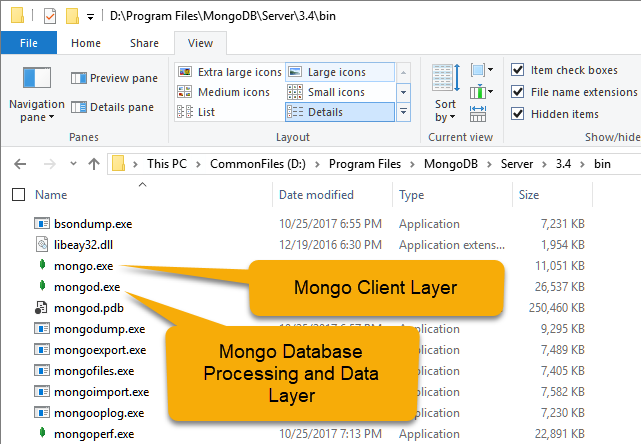
<https://www.mongodb.com/download-center?jmp=nav#community>



"This section of the manual contains tutorials on installation of MongoDB."

<https://docs.mongodb.com/manual/installation/>

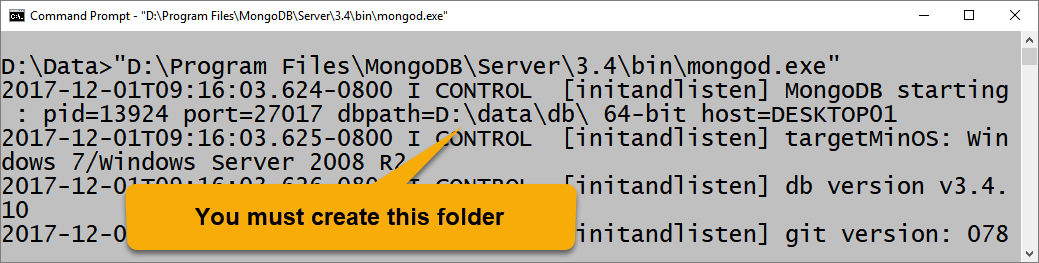
On **Windows** OS, you run an MS Installer package (.msi) and **chose the location** of your installation. You can install it on any drive, but make a **note** of which one you choose. You will need **to locate the mongo.exe and mongoD.exe files** later to create and manage your databases.



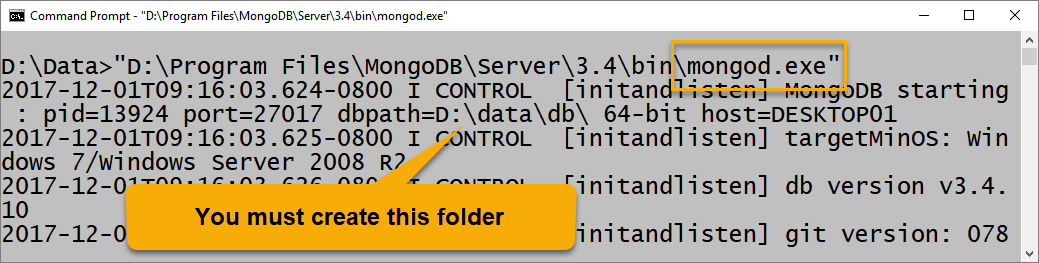
### Installing the Processing and Data Layer

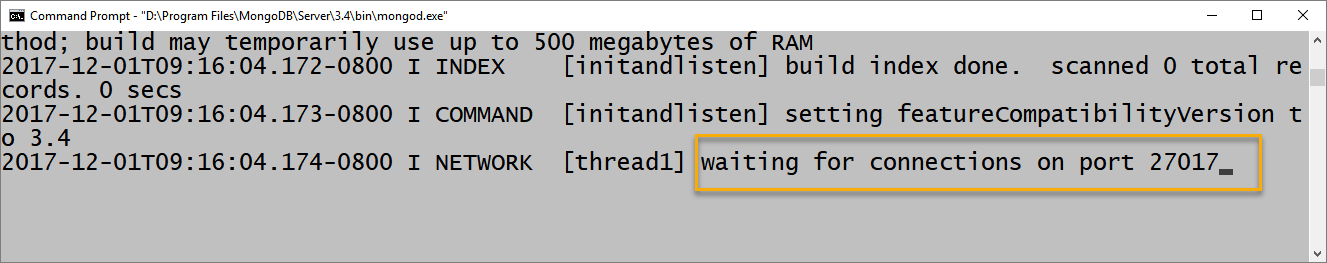
You create a new database file using the relational database management software which represents the **processing layer** in a MongoDB application by using the "**mungod.exe**" file.

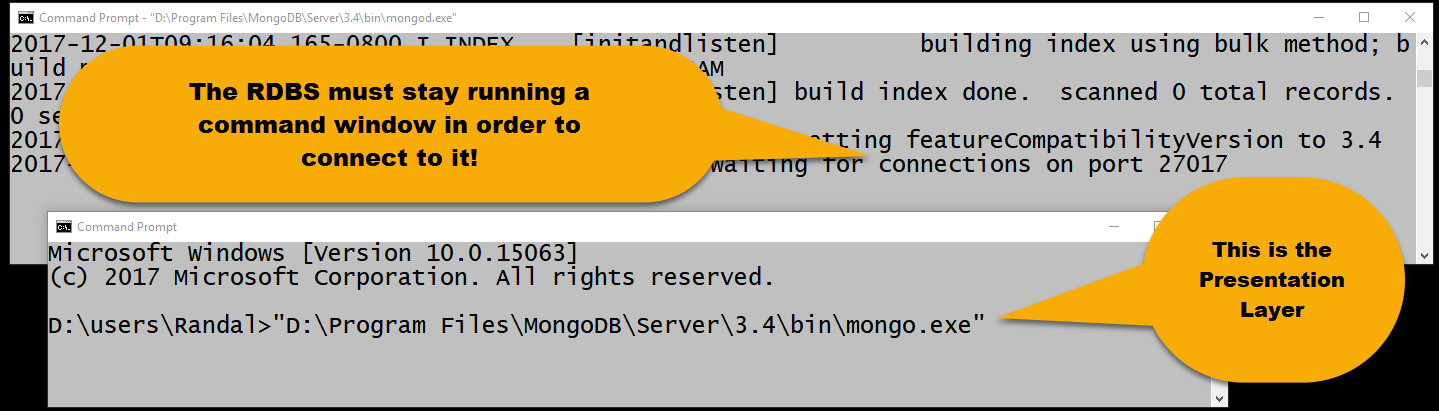
You will **need a folder** for the RDMS to use. The default is a folder called "**data\db**" on the **same drive** as your installation of MongoDB.

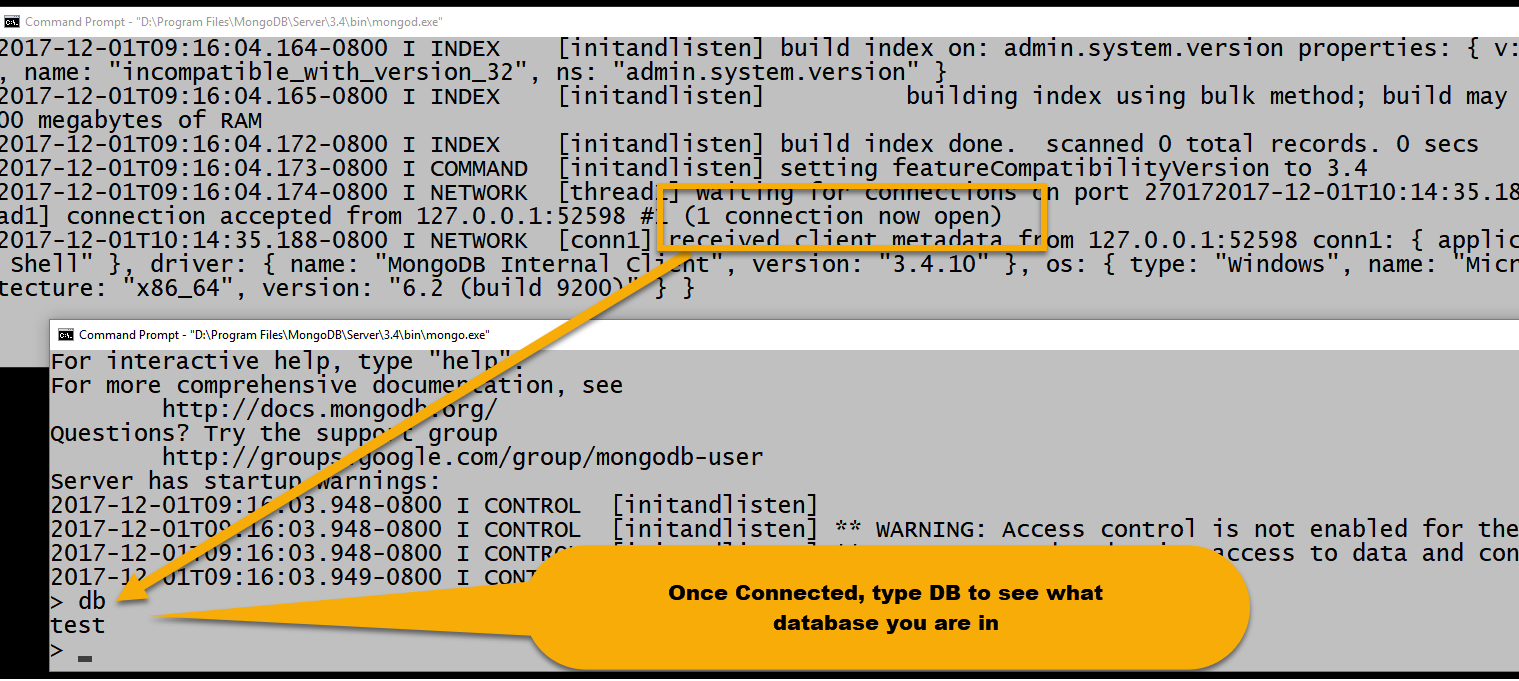


By navigating to the **executable** file and launching it the **software will start running in the background**. This background process is **used for database processing** and will create a test database in a default database folder. Note that this folder must exist in order for the service to start up.









### Creating a Database

Creating a new database in MongoDB is easy! Type in the following command to have it **use an existing database or automatically create one** if it does not already exist.

Use Sales

If later you wish to **delete that database**, you type in this command (while you are in the one you wish dropped!):

db.dropDatabase()



Instead of tables, **data is stored as a collection of documents**. Each document is a **binary version of JSON data** (BSON). To create a new document, you type in code similar to this:

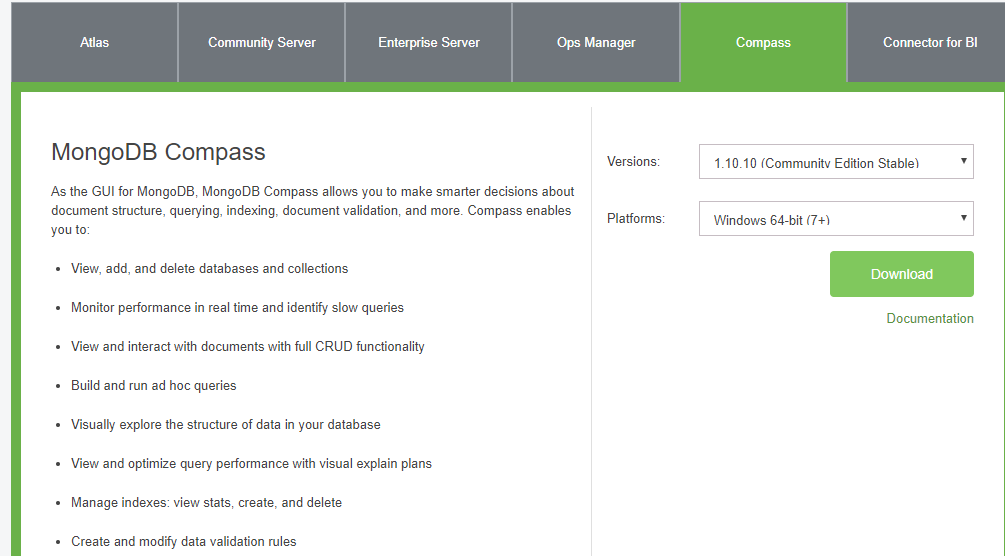
**use** ProjectDB

**db**.**Clinics**.**insertOne**{"ClinicID":101,"ClinicName":"GreenLake","ClinicPhoneNumber":"206-123-1234","ClinicAddress":"7210 Woodlawn Ave NE","ClinicCity":"Seattle","ClinicState":"WA","ClinicZipCode":"98115"}

"The insertOne() operation creates **both the database** *myNewDB* **and the collection** *myNewCollection1* if they do not already exist." (<https://docs.mongodb.com/manual/core/databases-and-collections/>, 2017)

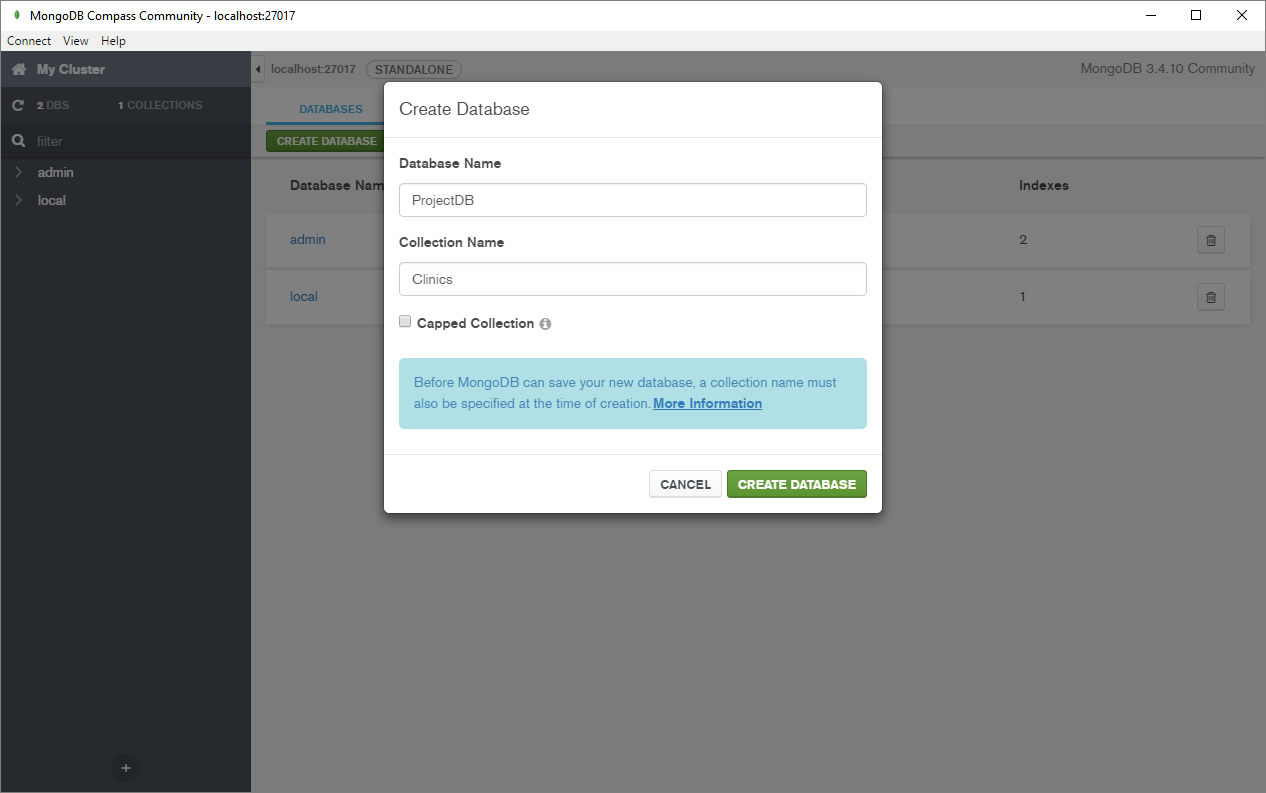
### Managing MongoDB

While working through a command shell window does work, it is not the most comfortable user experience. At some point you will probably download and install an integrated development environment (IDE) and use that instead.



<https://www.mongodb.com/download-center?jmp=nav#compass>

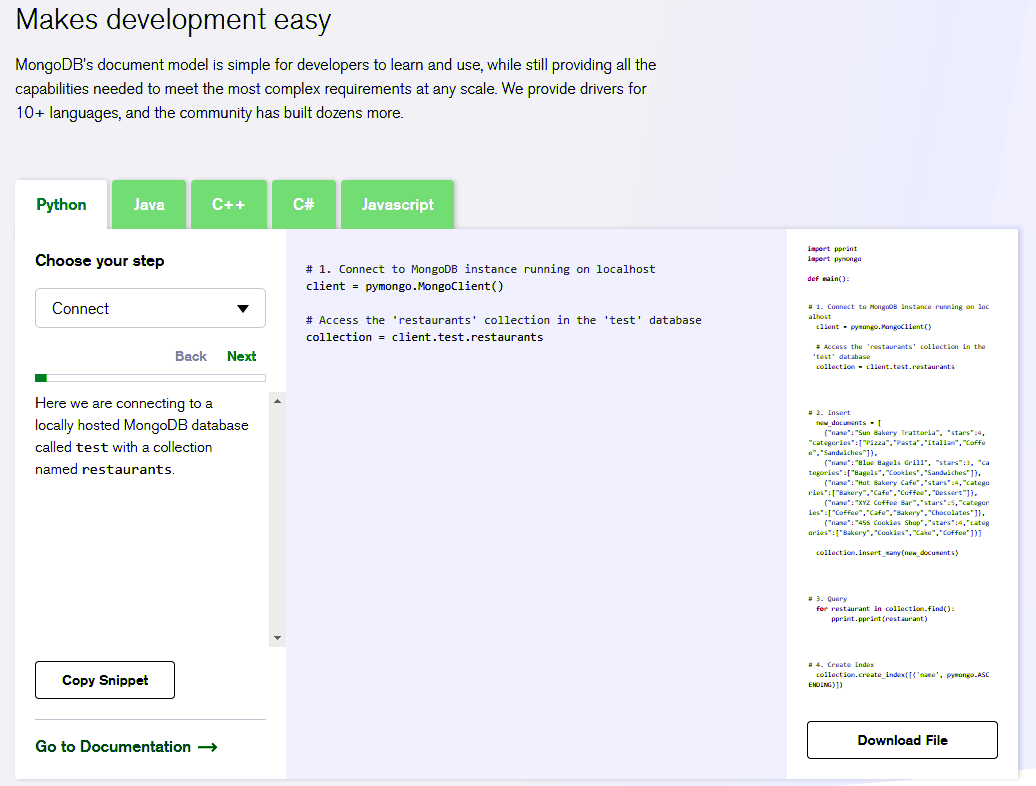
Once the **installation** is finished you will be able to **create and manage new databases and their collection** of objects.



### Creating a MongoDB Applications

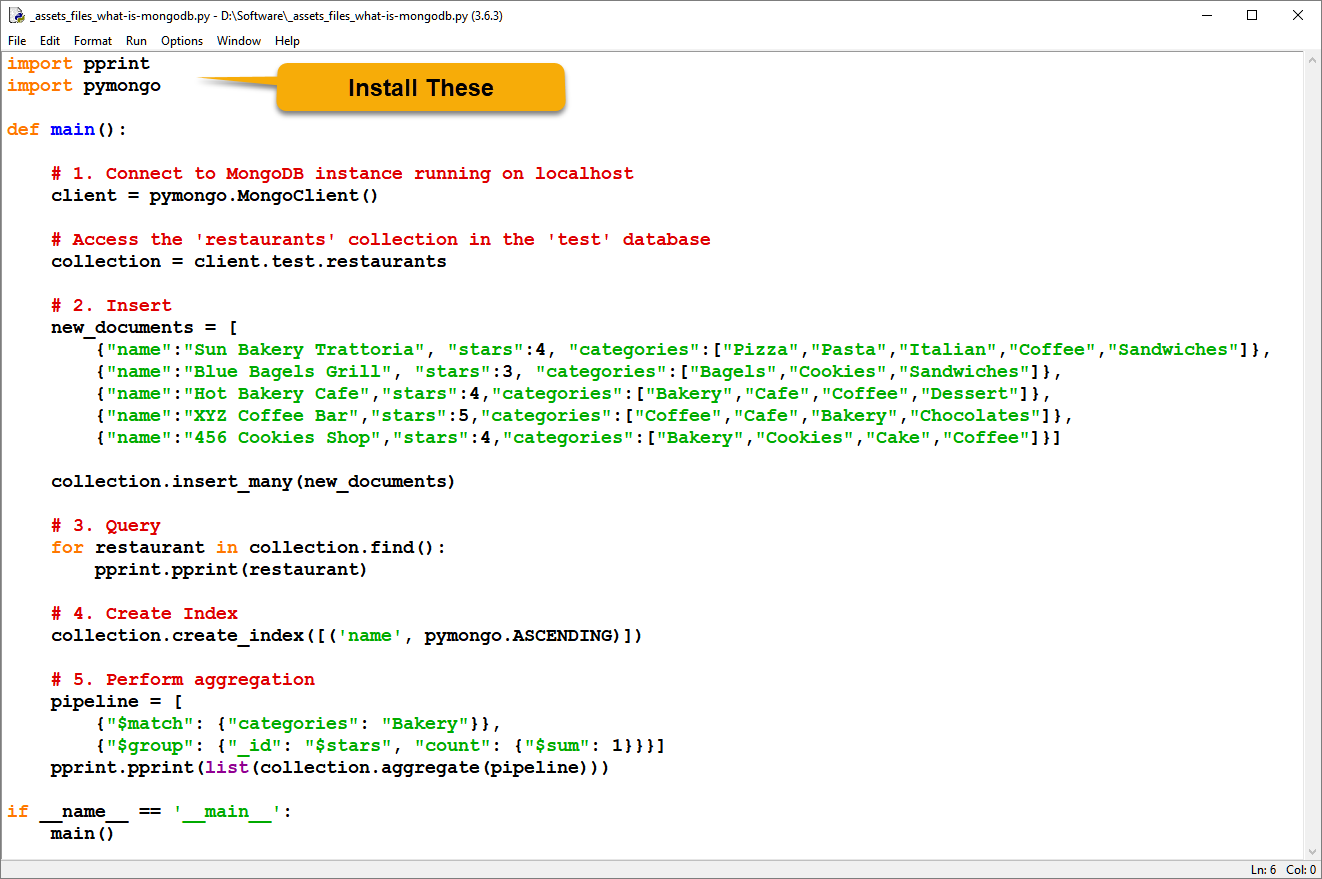
Of course, you can also create your own **custom application using** programming languages like C#, Java, or **Python**. The MongoDB website has **some example code** for each of these languages as well as others.

From the "What is MongoDB?" page on the MongoDB.com website, you can download demonstration files and several languages. Here is an example using the Python language.

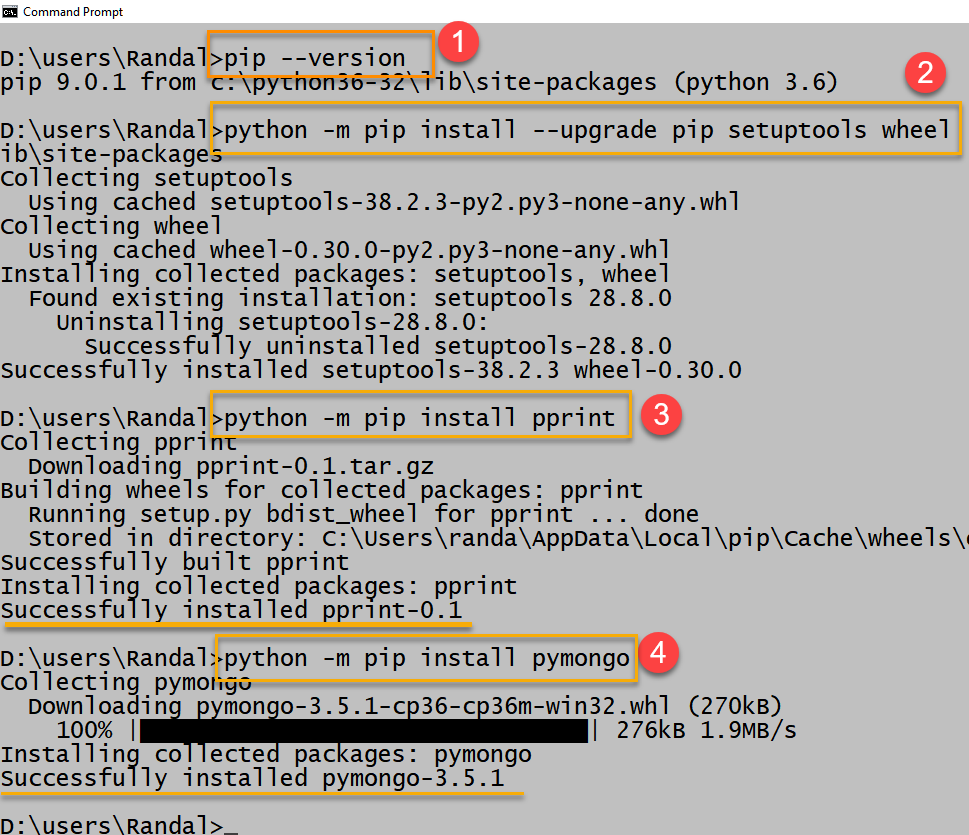


<https://www.mongodb.com/what-is-mongodb>

While you have not much experience with either python or MongoDB just looking at the code sample should be somehow familiar due to our previous discussions in this course



Previously we learned how to create python modules. Other people create python modules all the time and make them available to the Internet community. How to connect to MongoDB, you must download and install additional modules as we did when connecting to Microsoft SQL server.



## INFO 330 Introduction to Relational Database Management Systems

This is a good 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)

Where as 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 you hard work has paid off!

## 