TASK

Introduction to Databases and MongoDB

Visit our website

Introduction

WELCOME TO THE INTRODUCTION TO DATABASES AND MONGODB!

Dynamic web applications rely on data. If you want to make a web application truly dynamic, storing data about your users is a good starting point. In this task, you will learn about the various types of databases. You will also be introduced to MongoDB, a very popular database and how to use MongoDB to create a database as a service.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

The best way to get help is to log in to Discord at **https://discord.com/invite/hyperskills** where our specialist team is ready to support you.

Our team is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

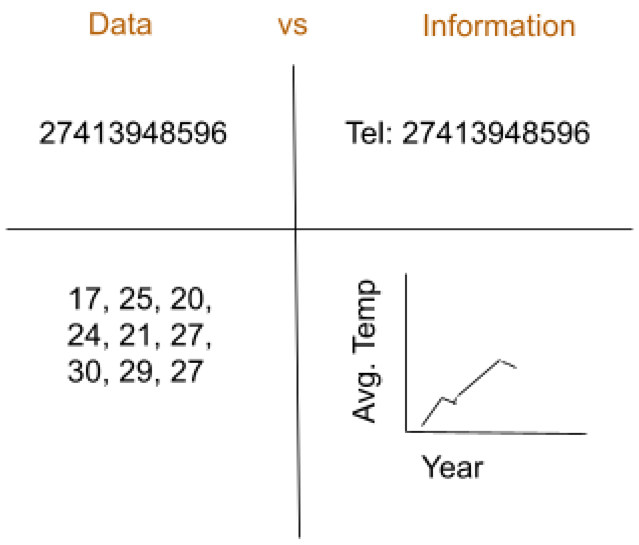
DATABASES

Data is core to software development. Programs are designed to manipulate, create and visualise data. Therefore, it is important for all software developers to be able to access, manipulate, and store data.

Databases are used to store data. A database is simply a large container of data, with the ability to order the data in multiple ways while providing easy access to the data itself. Web developers often have to manipulate the data stored in databases. For example, you may need to store your users' usernames, passwords, names, addresses, and telephone numbers, etc. Therefore, full stack web developers need to be able to work proficiently with databases.

DATA VS. INFORMATION

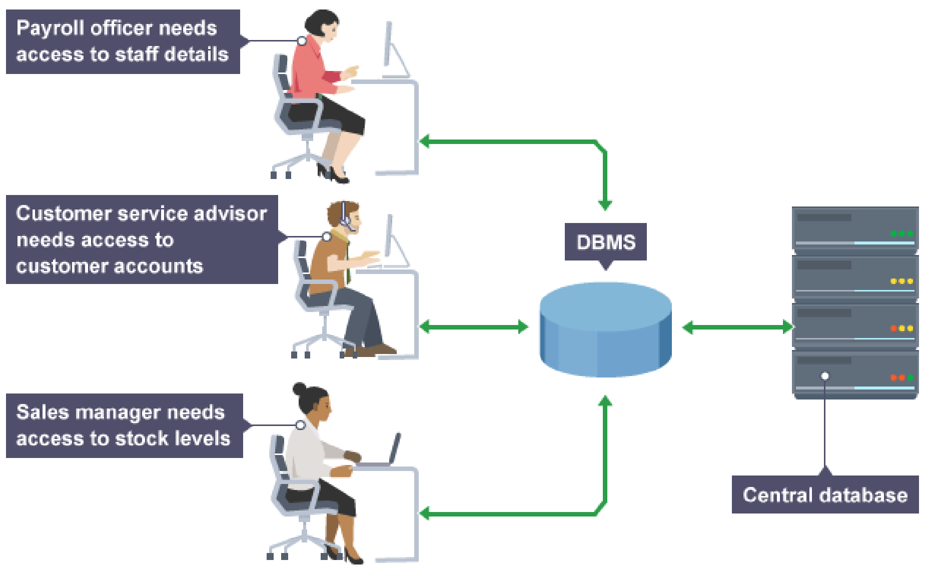
In order to properly understand databases, you must first understand the difference between data and information. Simply put, data are raw facts. The word raw indicates that the facts have not yet been processed to create meaning. Information, on the other hand, is the result of processing raw data to make it meaningful. Data processing can be as simple as organising data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modelling.



The production of accurate, relevant and timely information is the key to good decision-making and, in turn, good decision-making is the key to a business' survival in a competetive global environment. Timely and useful information requires accurate data which must be generated properly and stored in a format that is easy to access and process. The data environment should be carefully managed.

DATABASE MANAGEMENT SYSTEM

A database can be thought of as a well-organised electronic filing cabinet where powerful software, known as a database management system (DBMS), helps manage the contents of the cabinet. A database management system is a collection of programs that manage the database structure and control access to the data stored in the database.



Payroll officer needs access to staff details
Customer service advisor needs access to customer accounts
Sales manager needs access to stock levels
DBMS
Central database

The DBMS manages the interaction between the end user and the database (bbc.co.uk)

The illustration above shows how the DBMS serves as an intermediary between the user and the database. The DBMS receives all application requests and translates them into the complex operations required to fulfil those requests.

Much of the database's internal complexity is hidden from the application programs and end users by the DBMS. There are some very important advantages to having a DBMS between the end user's application and the database. Firstly, the DBMS allows the data in the database to be shared among multiple applications or users. Secondly, the DBMS integrates many different users' views of the data into a single data repository.

The DBMS helps make data management much more efficient and effective and provides advantages such as:

- **Improved data sharing:** the DBMS helps create an environment in which end users have better access to more and better-managed data.
- **Better data integration:** an integrated view of the organisation's operations and a clearer view of the big picture is promoted by wider access to well-managed data.
- **Minimised data inconsistency:** data inconsistency occurs when different versions of the same data appear in different places. A properly designed database greatly reduces the probability of data inconsistency.
- **Improved data access:** a query is a specific request for data manipulation (e.g. to read or update the data) sent to the DBMS. The DBMS makes it possible to produce quick answers to spur-of-the-moment queries.
- **Improved decision-making:** better-quality information (on which to base decisions) is generated, due to better-managed data and improved data access.
- **Increased end-user productivity:** the availability of data and the tools that transform data into usable information encourages end users to make quick, informed decisions.

SPOT CHECK 1

Let's see what you can remember from this section.

1. What is the difference between data and information?
2. What are 3 advantages of using a database management system?

TYPES OF DATABASES

There are many different types of databases. These databases can be classified according to the number of users supported, where the data are located, the type of data stored, the intended data usage and the degree to which the data are structured.

A database can be classified as either **single-user** or **multi-user**. A database that only supports one user at a time is known as a single-user database. With a single-user database, if user A is using the database, users B and C must wait until user A is done. A desktop database is a single-user database that runs on a personal computer. A multi-user database, on the other hand, supports multiple users at the same time. A workgroup database is a multi-user database that supports a relatively small number of users (usually less than 50) or a specific department within an organisation. When a multi-user database supports many users (more than 50) and is used by the entire organisation, across many departments, it is known as an enterprise database.

A database can also be classified based on location. A **centralised database** is a database that supports data located at a single site, while a **distributed database** supports data distributed across several different sites.

A popular way of classifying databases is based on how they will be used and on the time sensitivity of the information gathered from them. An example of this is an **operational database**, which is a database that is designed to primarily support a company's day-to-day operations. Operational databases are also known as online transaction processing (OLTP), transactional, or production databases.

The degree to which data is structured is another way of classifying databases. Data that exist in their original, or raw, state are known as **unstructured data**. In other words, they are in the format in which they were collected. **Structured data** are the result of formatting unstructured data to facilitate the storage, use, and generation of information. You apply structure based on the type of processing that you intend to perform on the data. For example, imagine that you have a stack of printed invoices. If you just want to store these invoices so that you are able to retrieve them or display them later, you can scan them and save them in a graphical format. However, if you want to derive information from them, such as monthly totals or average sales, having the invoices in a graphical format will not be useful. You could instead store the invoice data in a structured spreadsheet format so that you can perform the desired computations.

Analytical databases focus on storing historical data and business metrics used exclusively for tactical or strategic decision making. They typically comprise two components; a data warehouse and online analytical processing (OLAP) front end. Analytical databases allow the end-user to perform advanced data analysis of business data using sophisticated tools. A data warehouse, on the other hand, focuses on storing data used to generate the information required to make tactical or strategic decisions.

There are many ways of storing data, each with its own benefits and limitations. Therefore, databases can be designed differently to meet different needs. We are going to consider two of the most popular types of databases: relational databases and NoSQL databases.

Relational databases

Relational databases store information about different entities and the relationships between them. The image below is an example of an entity-relationship diagram (ERD) that is used to describe the relationships between certain entities.

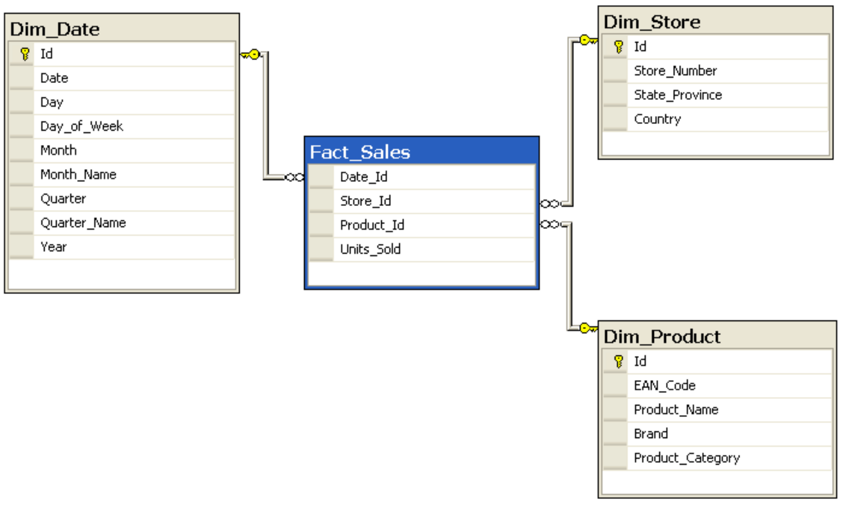


Image source:
https://en.wikipedia.org/wiki/Star_schema#/media/File:Star-schema-example.png

In a relational database, an entity is a table that stores all the data about a certain thing. For example, you may want to store information about all your customers in a database. You would then create a customer table (customer entity), which could look something like the one shown below:

C_NAME	C_PHONE	C_ADDRESS	C_POSTCODE	A_NAME	A_PHONE	TP	AMT	REN

Alfred Smith	082 345 2341	207 Willow St, Port Elizabeth	6390	Leah Hahn	084 259 2073	T1	R100.00	05-Apr-2021
Kathy Dunne	083 567 9012	556 Bad St, Cape Town	7100	Alex Alby	085 785 3938	S2	R250.00	16-Jun-2021
Paul Farris	076 782 1232	2148 High St,Benoni	1522	Leah Hahn	084 259 2073	T2	R850.00	22-sep-2021

Customer entity key

C_NAME	= customer name
C_PHONE	= customer phone
C_ADDRESS	= customer address
C_POSTCODE	= customer postcode
A_NAME	= agent name
A_PHONE	= agent phone
TP	= insurance type
AMT	= insurance policy amount in thousands of R
REN	= Insurance renewal date

The CUSTOMER table contains 3 records. Each record is composed of 9 fields: C_NAME, C_PHONE, C_ADDRESS, C_POSTCODE, A_NAME, A_PHONE, TP, AMT, and REN. Each record describes a specific customer; each customer is an instance of the customer entity.

If you were designing a database for a store, you might also have a product entity that stores all the information about the products you sell. Your database would then contain a product table and a customer table. The database would also save information about the relationship between the two entities. For example, it would store information about which products a particular customer bought on a particular date. How this is implemented in a relational database, however, is beyond the scope of this course.

NoSQL databases

A problem with relational databases is that their performance degrades as the volume of data increases. Many web applications have to store massive amounts of data. Imagine the amount of data that companies like Amazon and Google have to store, for example. Addressing this problem led to the development of NoSQL databases. You are using a NoSQL database every time you search for a product on

Amazon, watch a video on Youtube, or send a message to a friend on Facebook. NoSQL databases generally have the following characteristics:

- They are not based on the relational model
- They support distributed database architectures i.e. servers in different areas
- They provide high scalability, high availability, and fault tolerance
- They support very large amounts of sparse data
- They are geared toward performance rather than transaction consistency

There are several types of NoSQL databases. Some of these are briefly described below:

1. **Key-value store databases**: This is the simplest form of NoSQL database. Every item in the database is stored as a key (used to identify the value) and its value.

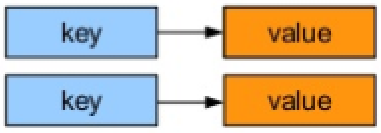


Image source:
https://www.slideshare.net/arangodb/introduction-to-column-oriented-databases

2. **Column-oriented databases**: Like key-value store databases, a key is used to identify values but instead of the key identifying only one value, it can be used to identify multiple values.

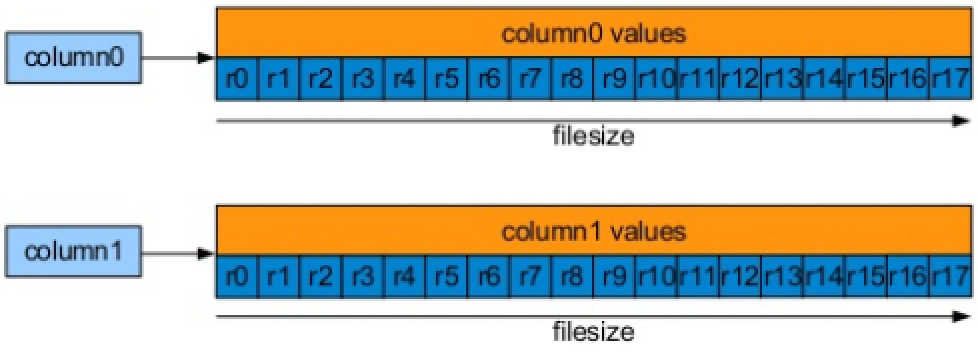

column0
column0 values
r0 r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r13 r14 r15 r16 r17
filesize
column1
column1 values
r0 r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r13 r14 r15 r16 r17
filesize

Image source:
https://www.slideshare.net/arangodb/introduction-to-column-oriented-databases

3. **Document store databases**: With this type of database, a key is used to identify a particular document. Documents are stored in recognised formats like XML, JSON, PDF, etc.

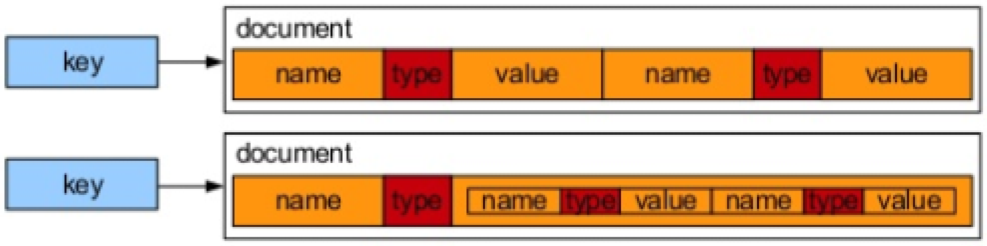
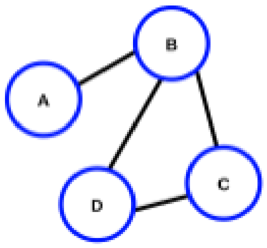


Image source:
[***https://www.slideshare.net/arangodb/introduction-to-column-oriented-databases***](https://www.slideshare.net/arangodb/introduction-to-column-oriented-databases)

4. **Graph databases**: This database uses the graph data structure to store data. Graphs contain nodes (A, B, C, and D in the image below) and edges (the lines connecting the nodes, so AB, BD, BC, and DC in the image below). In graph databases, nodes are objects and edges are the relationships between these objects. Social networking applications, like Facebook, often store data using graph databases because this model is good for tracking the relationships between objects.



5. **Object-oriented databases**: These databases combine object-oriented programming (OOP) and database principles. These databases are tied to specific programming languages.

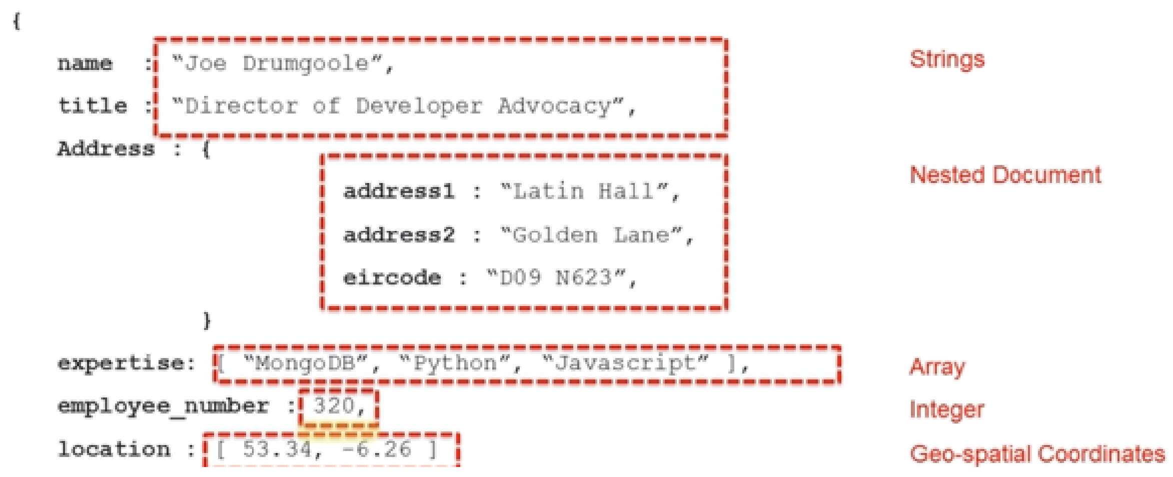
MONGODB

In this bootcamp you will be working with MongoDB, a document store and NoSQL database. A MongoDB is made up of collections and documents.

- **Collection:** A collection is a group of documents. It is similar to an entity or table when working with relational databases.

- **Documents:** In relational databases, records are stored in tables. An example of a record in the CUSTOMER table we considered earlier would be Alfred Smith. MongoDB uses documents instead of records (or rows in a table) to store data (i.e. with a MongoDB database, Alfred Smith's data would be stored in a document instead of in a row in the CUSTOMER table).

MongoDB uses BSON documents. BSON stands for Binary JSON. BSON uses JSON files and stores type information. JSON files are just text information; this means that it has to be parsed if you want to program with it. Since BSON stores type information it is quicker and more efficient to use than JSON (see an example below).



```
{
    name  : "Joe Drumgoole",
    title : "Director of Developer Advocacy",
    Address : {
                    address1 : "Latin Hall",
                    address2 : "Golden Lane",
                    eircode : "D09 N623",
              }
    expertise: [ "MongoDB", "Python", "Javascript" ],
    employee_number : 320,
    location : [ 53.34, -6.26 ]
```

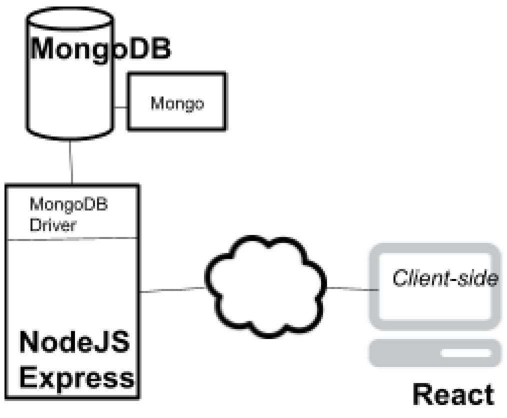
Strings
Nested Document
Array
Integer
Geo-spatial Coordinates

Extra resource

Watch **this video** where MongoDB's CTO and Co-Founder, Eliot Horowitz, gives a comprehensive introduction to MongoDB in under 5 minutes.

MONGODB IN A FULL STACK WEB APPLICATION

Before we start using MongoDB, it is important to first see how MongoDB fits into full-stack web development and what other tools are needed to get it to work properly.



Consider the image above. As you already know, to use your app, clients will interact with a web server that will be running Node.js. Node.js will route all requests and perform whatever server-side logic is needed by our app. If our user wants to access, add, or change information that needs to persist, they will need access to the MongoDB database. An important component of MongoDB is Mongo, its administrative shell. Mongo is a C++ program that allows you to execute instructions on the database from a command line interface. Mongo allows you to use the MongoDB query language.

For Node.js to be able to communicate with MongoDB, it also makes use of *MongoDB drivers*. The official MongoDB driver can be installed using NPM (more on this later).

MONGODB AS A SERVICE

A few years ago, in order to run a database, you had to have a server (or servers) with all the necessary software installed and configured. You would also need someone who would act as the database administrator. In many organisations, this is still the case and this may be a justified expense and effort. Today though, there is an attractive alternative: database as a service.

Since cloud computing has become more popular, there are more cloud-based options for developers. You can host your web app on the cloud (HEROKU, Azure, etc) instead of on your own dedicated server. You can also use a database hosted by a cloud service provider (as in the case of MongoDB's Atlas), rather than having the hassle of setting up and maintaining your own database server. Below are some key benefits of this approach:

1. It is often cheaper than having your own database server because you only pay for what you use.
2. The cloud service provider deals with all the hassle of ensuring the configuration, backup, maintenance, and security of the database server.
3. It is quick and easy to start using a database with minimal configuration.

In this bootcamp, we will be using MongoDB's database as a service solution: Atlas.

In this task, you are going to create your first database using MongoDB. Before you can do this though, you are first going to:

1. download and install MongoDB on your local machine so that you can use Mongo, the administrative shell,

2. use Atlas to create and host a MongoDB on the cloud, and
3. use Mongo to access and manipulate your database cluster on Atlas.

INSTALL MONGODB

We will be installing MongoDB's free Community Server for this course. Your other main option is the Enterprise Server which you can download as a free trial. Follow the steps linked below to get started with MongoDB.

- **MongoDB Community Edition Installation Steps**

The steps to set up and configure MongoDB differ slightly depending on the OS you are using. The installation instructions given above contain instructions for running MongoDB on your specific operating system.

Your Mongo shell is now ready to be used to connect to a database server. In this bootcamp, we will be using MongoDB Atlas which will provide the platform and infrastructure we need for a database server on the cloud.

Take note:

Mongo is MongoDB's administrative shell. It is a C++ program that allows you to execute instructions on the database from a command line interface. Mongo allows you to use the MongoDB query language.

SETUP MONGODB ATLAS

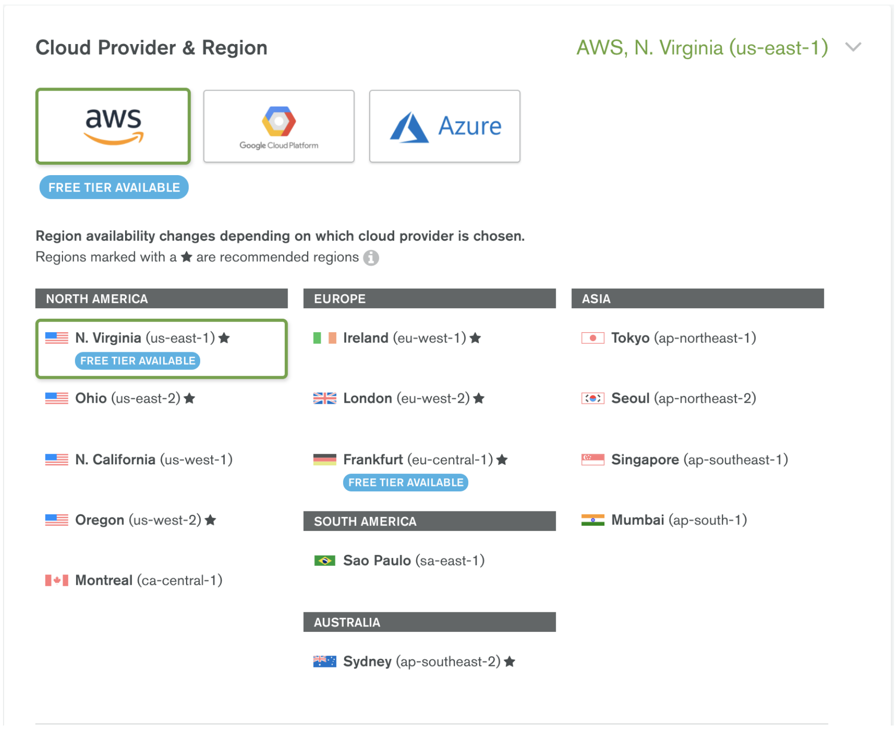
In this bootcamp, we will be using MongoDB's Database as a service solution: Atlas. To get a quick (2-minute) overview of what Atlas is and why we are using it, watch this **short video**.

To configure MongoDB Atlas, do the following:

- Go **here** and enter your information to get started with Atlas.

- You will then be taken to the 'Create New Cluster' page.

    - Under Cloud provider & Region select AWS and any free tier region.

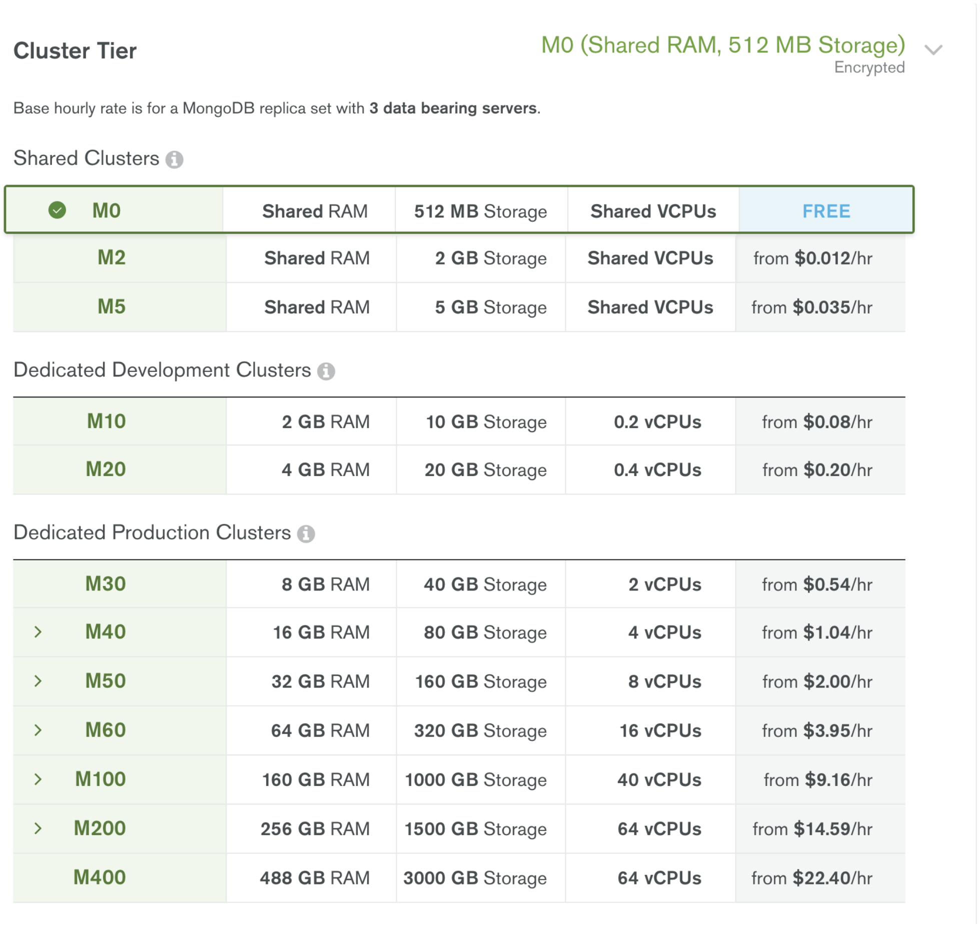

Cloud Provider & Region

AWS, N. Virginia (us-east-1)

Region availability changes depending on which cloud provider is chosen.
Regions marked with a ★ are recommended regions

NORTH AMERICA	EUROPE	ASIA
N. Virginia (us-east-1) ★ FREE TIER AVAILABLE	**Ireland** (eu-west-1) ★	**Tokyo** (ap-northeast-1)
Ohio (us-east-2) ★	**London** (eu-west-2) ★	**Seoul** (ap-northeast-2)
N. California (us-west-1)	**Frankfurt** (eu-central-1) ★ FREE TIER AVAILABLE	**Singapore** (ap-southeast-1)
Oregon (us-west-2) ★	SOUTH AMERICA	**Mumbai** (ap-south-1)
Montreal (ca-central-1)	**Sao Paulo** (sa-east-1)	
	AUSTRALIA	
	Sydney (ap-southeast-2) ★	

- Under ‘Cluster Tier’ select the free M0 option.


Cluster Tier

M0 (Shared RAM, 512 MB Storage)
Encrypted

Base hourly rate is for a MongoDB replica set with **3 data bearing servers**.

Shared Clusters

✓ **M0**	**Shared** RAM	**512 MB** Storage	**Shared VCPUs**	FREE
M2	**Shared** RAM	**2 GB** Storage	**Shared VCPUs**	from **$0.012**/hr
M5	**Shared** RAM	**5 GB** Storage	**Shared VCPUs**	from **$0.035**/hr

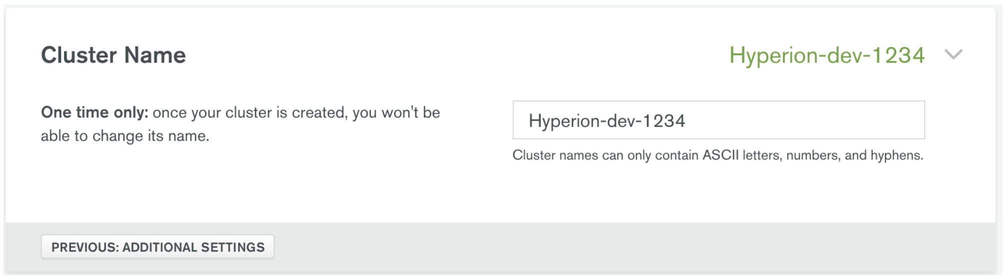
Dedicated Development Clusters

M10	**2 GB** RAM	**10 GB** Storage	**0.2 vCPUs**	from **$0.08**/hr
M20	**4 GB** RAM	**20 GB** Storage	**0.4 vCPUs**	from **$0.20**/hr

Dedicated Production Clusters

M30	**8 GB** RAM	**40 GB** Storage	**2 vCPUs**	from **$0.54**/hr
› **M40**	**16 GB** RAM	**80 GB** Storage	**4 vCPUs**	from **$1.04**/hr
› **M50**	**32 GB** RAM	**160 GB** Storage	**8 vCPUs**	from **$2.00**/hr
› **M60**	**64 GB** RAM	**320 GB** Storage	**16 vCPUs**	from **$3.95**/hr
› **M100**	**160 GB** RAM	**1000 GB** Storage	**40 vCPUs**	from **$9.16**/hr
› **M200**	**256 GB** RAM	**1500 GB** Storage	**64 vCPUs**	from **$14.59**/hr
M400	**488 GB** RAM	**3000 GB** Storage	**64 vCPUs**	from **$22.40**/hr

- You can rename your cluster under ‘Cluster Name’.


Cluster Name Hyperion-dev-1234

One time only: once your cluster is created, you won't be able to change its name.

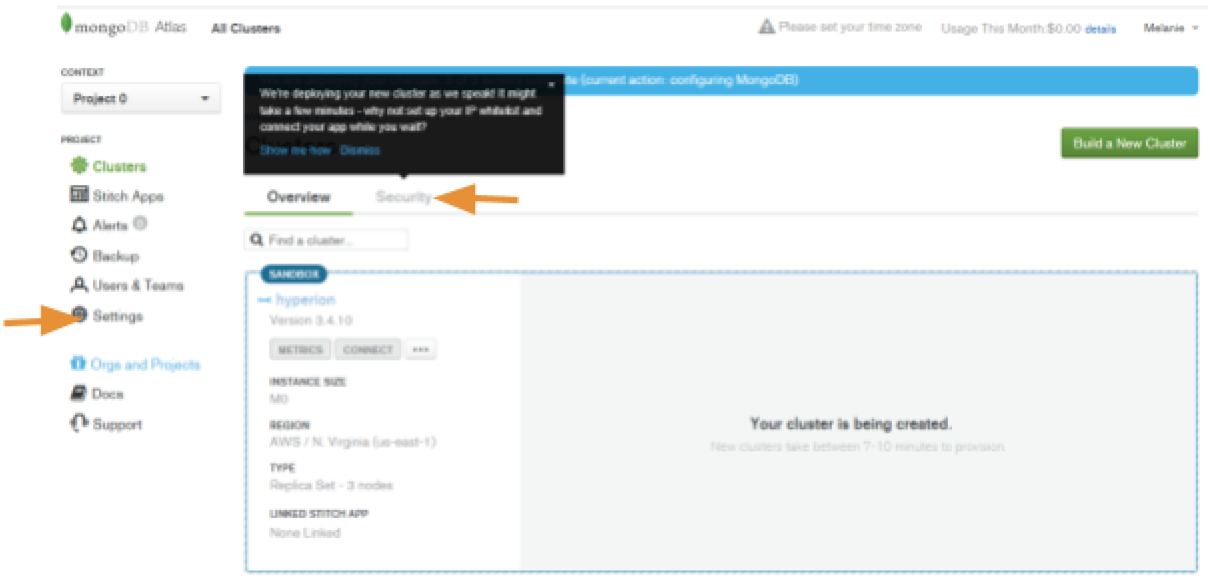
Hyperion-dev-1234

Cluster names can only contain ASCII letters, numbers, and hyphens.

PREVIOUS: ADDITIONAL SETTINGS

- Click on the ‘Create Cluster’ button at the bottom of the page to create your cluster.

Once you have created your cluster, you will be taken to a page similar to the one shown below. You can change the name and other settings for your default project by selecting *settings* as shown in the image below.



We're deploying your new cluster as we speak! It might take a few minutes - why not set up your IP whitelist and connect your app while you wait?

(current action: configuring MongoDB)

Version 3.4.10

INSTANCE SIZE M0

REGION AWS / N. Virginia (us-east-1)

TYPE Replica Set - 3 nodes

LINKED STITCH APP None Linked

Your cluster is being created.

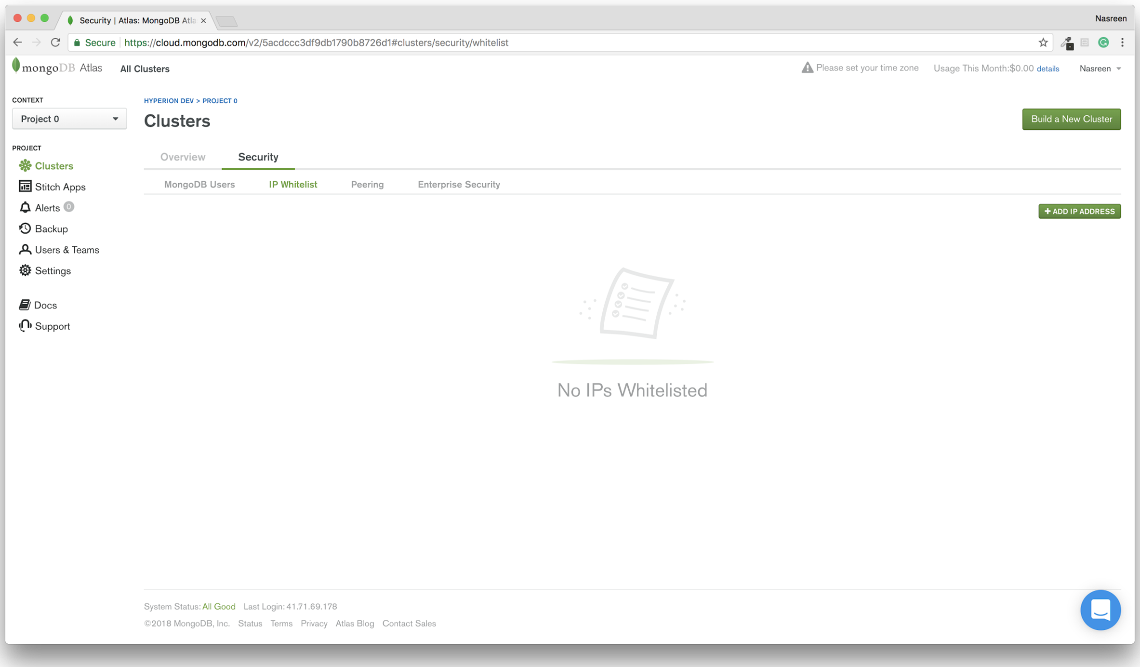
New clusters take between 7-10 minutes to provision.

You have now successfully created a database cluster. You will now use the administrative shell, Mongo, to add a database to this cluster. Before you do this though, there are some security settings you need to tweak to make sure that you can access the Atlas cluster from your computer.

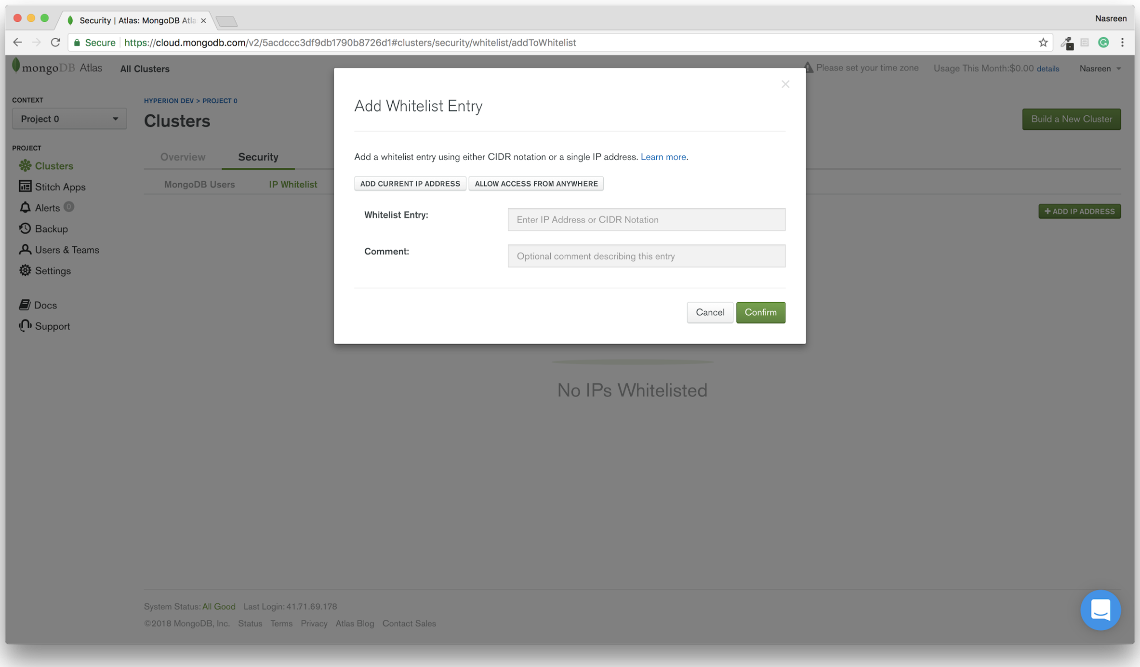
Security settings

One of the ways that MongoDB ensures security is by only allowing certain machines (IP addresses) access to your cluster. The IPs that are allowed to access your cluster are listed under the ‘Security’ tab in the ‘IP Whitelist’. Click on the

‘Security’ tab and select ‘IP Whitelist’ from the menu, then click on the ‘+ Add IP Address’ button.



In the ‘Add Whitelist Entry’ popup window, click on the ‘Allow Access From Anywhere’ button and then click ‘Confirm’.


Add Whitelist Entry
Add a whitelist entry using either CIDR notation or a single IP address. Learn more.
ADD CURRENT IP ADDRESS
ALLOW ACCESS FROM ANYWHERE
Whitelist Entry:
Enter IP Address or CIDR Notation
Comment:
Optional comment describing this entry
Cancel
Confirm

It is not good practice to allow all IP addresses to access your database for obvious security reasons, but for the purposes of the next few tasks, we are going to ask you to do this. The reason for this is that we are going to ask you to give a mentor access to your database and you won't know their IP address ahead of time. In practice, however, it is advisable to have a limited IP whitelist.

Take note:

You should remember to configure this IP whitelist when you deploy your app. Remember that you will ultimately write code to access your database in your Express app (which you will soon learn to do). Therefore, your application server will be making requests to your database. If you deploy your back-end app to the cloud (e.g. deploy to Heroku) you do not necessarily know what the IP address of your web server will be! This could obviously be a problem - if your web server IP address isn't added to the Atlas IP whitelist, your Express app won't be allowed to communicate with your database!

To address this problem you can create a Heroku Private Space. According to an **article by Heroku**, "Private Spaces are dedicated environments for running dynos and certain types of add-ons within an isolated network. Access to apps in a Private Space can be controlled at the network level. Outbound requests from apps in a Private Space originate from a set of stable IP addresses, which allows you to securely communicate with IP white-listed services on-premise or on other networks."

We recommend that you read **this guide** by MongoDB to see how to integrate Atlas with Heroku. At the time of writing this task, this was not possible with a free Atlas account. In such a case you would have to allow access to your database from any IP address to allow your back-end deployed on Heroku to communicate with your database on MongoDB Atlas.

Although Heroku provides **add-ons** for accessing databases hosted by **mLab** (another Database as a Service for MongoDB), mLab was, at the time of writing, migrating to MongoDB Atlas. Thus the decision was made to base the content on MongoDB Atlas instead of mLab.

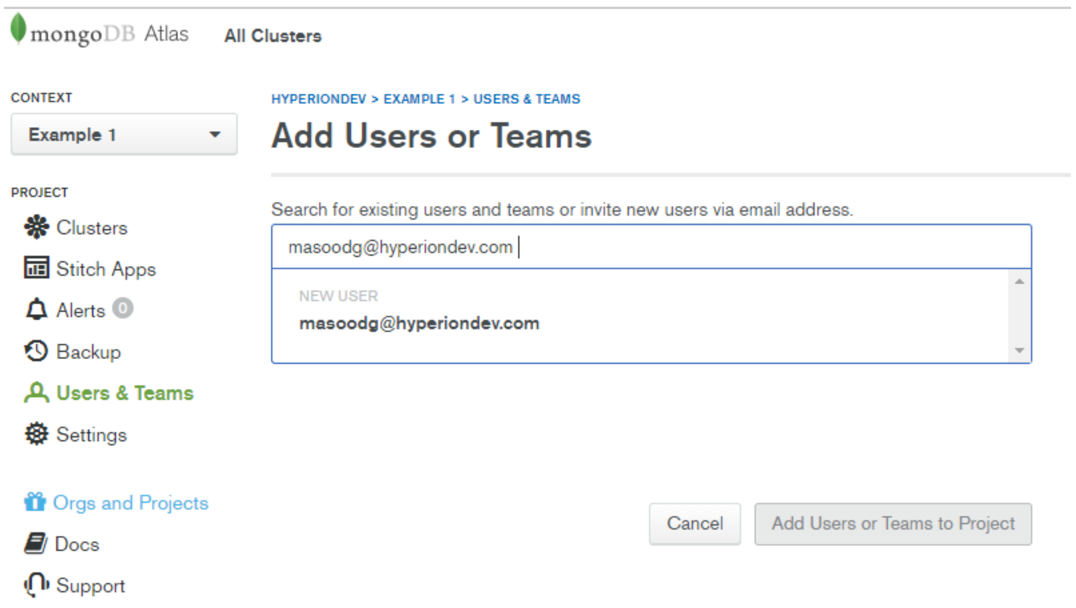
If your machine is protected by a firewall, you also have to ensure that this doesn't block access to Atlas. Atlas servers run on ***port 27017*** on Amazon AWS. Check **here** to see if this port is blocked on your machine or not. If **this page** doesn't load, your firewall is probably blocking port 27017. If it is blocked, make sure to unblock it before you proceed. How this is done will depend on the firewall you are using.

Google the appropriate instruction to unblock port 27017 for the firewall you are running.

From the Security tab, you can also add users and manage the rights of the users you allow to access your database.

Manage users and teams

As the database administrator, you have to manage who is able to access your database and what they can do with your database. For your next MongoDB tasks, you are required to give a mentor access to your database. To do this, select “Users and teams” and then click on the “Add users and teams” button. You can then invite a mentor to be a user of your database by entering an email address as indicated in the image below. Please use the email address **allreviewers@hyperiondev.com**.



Search for existing users and teams or invite new users via email address.

masoodg@hyperiondev.com

NEW USER

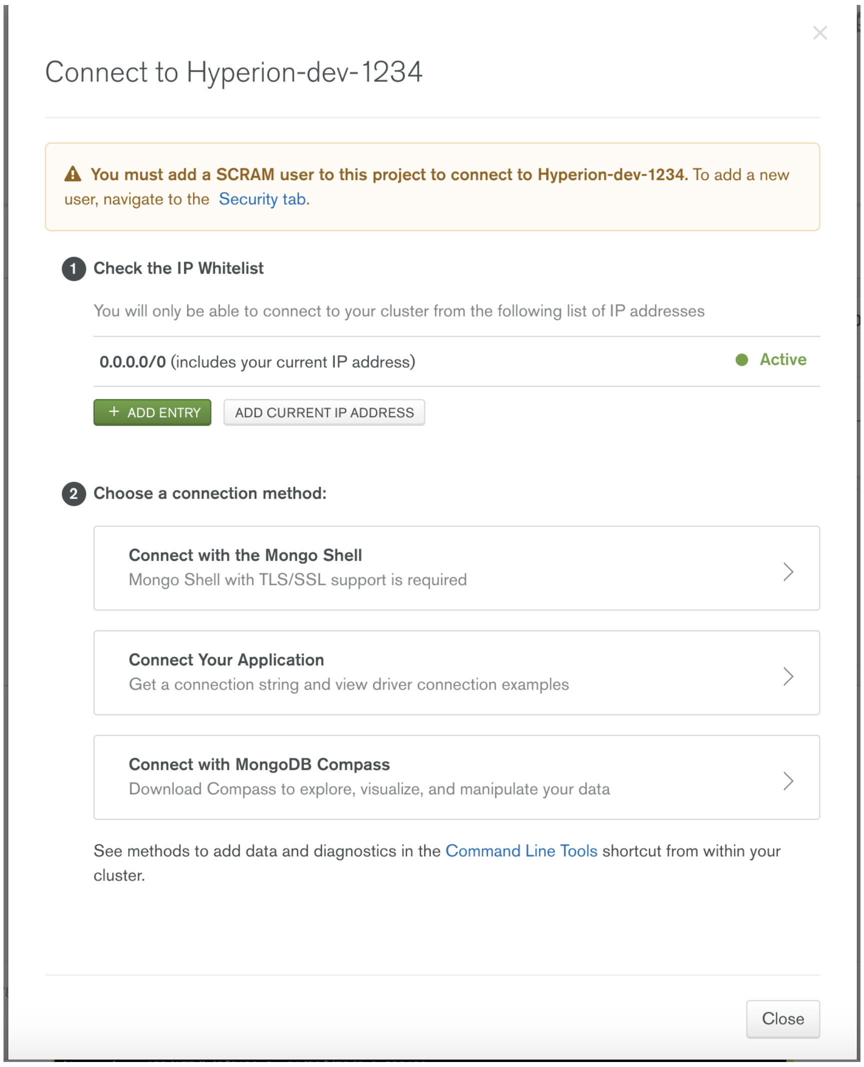
masoodg@hyperiondev.com

ACCESS THE DATABASE ON THE CLOUD USING THE MONGO SHELL

You are now ready to access the database server you have configured on MongoDB Atlas using the Mongo shell on your local machine. Remember that Mongo is the administrative shell used to run instructions on your MongoDB server.

To connect to the database server using Mongo you need a connection string that specifies everything needed for this connection. Atlas provides this connection string for you. Select "Connect" as shown below to find the connection string.

The following popup window will appear:



Connect to Hyperion-dev-1234

You must add a SCRAM user to this project to connect to Hyperion-dev-1234. To add a new user, navigate to the Security tab.

1 Check the IP Whitelist

You will only be able to connect to your cluster from the following list of IP addresses

0.0.0.0/0 (includes your current IP address) ● Active

2 Choose a connection method:

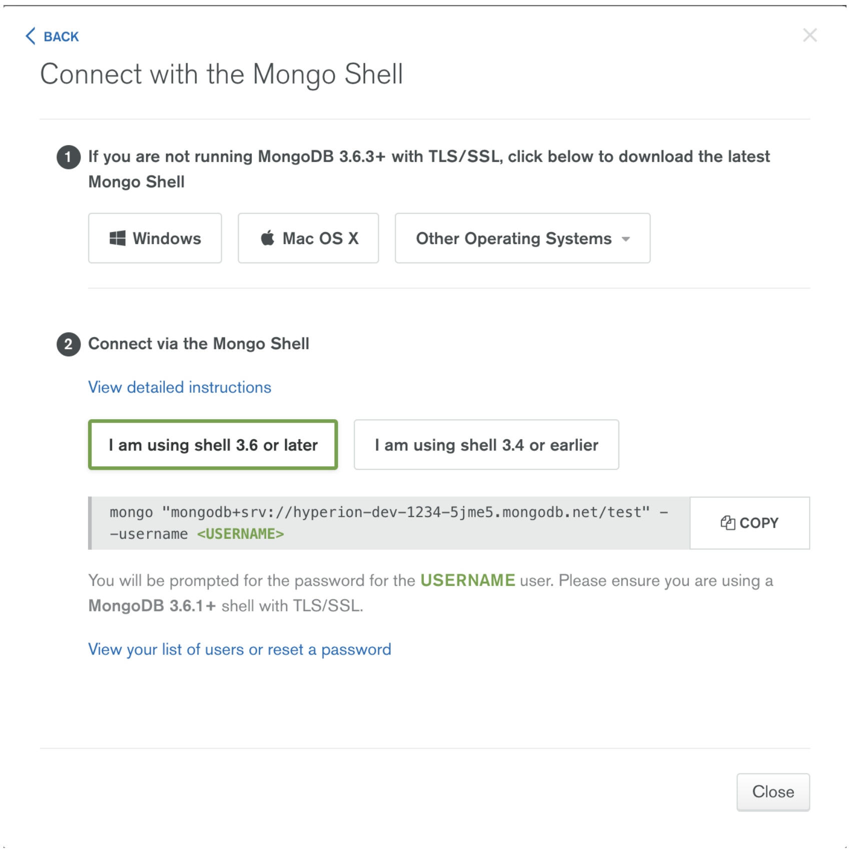
Connect with the Mongo Shell
Mongo Shell with TLS/SSL support is required

Connect Your Application
Get a connection string and view driver connection examples

Connect with MongoDB Compass
Download Compass to explore, visualize, and manipulate your data

See methods to add data and diagnostics in the Command Line Tools shortcut from within your cluster.

From the window above select 'Connect with the Mongo shell'.



Connect with the Mongo Shell

1 If you are not running MongoDB 3.6.3+ with TLS/SSL, click below to download the latest Mongo Shell

Windows
Mac OS X
Other Operating Systems

2 Connect via the Mongo Shell

View detailed instructions

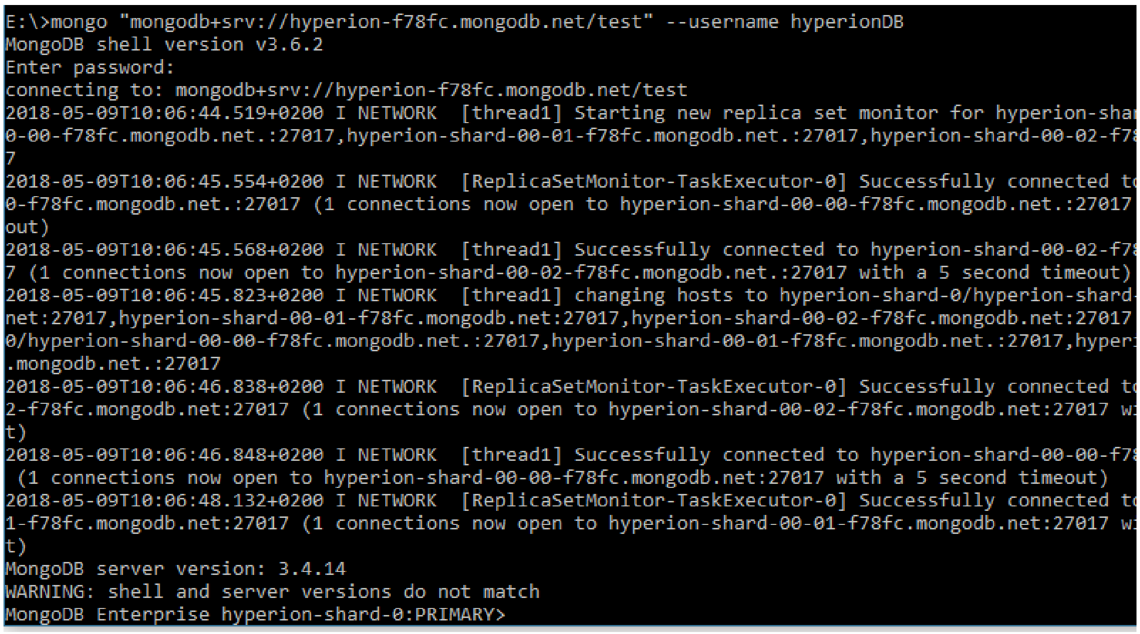
I am using shell 3.6 or later
I am using shell 3.4 or earlier

```
mongo "mongodb+srv://hyperion-dev-1234-5jme5.mongodb.net/test" –
–username <USERNAME>
```

You will be prompted for the password for the **USERNAME** user. Please ensure you are using a **MongoDB 3.6.1+** shell with TLS/SSL.

View your list of users or reset a password

The window above should then appear. Select 'I am using shell 3.6 or later', and copy and paste the connection string that appears there, into your command line interface. You should see output similar to that shown in the image below:



```
E:\>mongo "mongodb+srv://hyperion-f78fc.mongodb.net/test" --username hyperionDB
MongoDB shell version v3.6.2
Enter password:
connecting to: mongodb+srv://hyperion-f78fc.mongodb.net/test
2018-05-09T10:06:44.519+0200 I NETWORK  [thread1] Starting new replica set monitor for hyperion-sha
0-00-f78fc.mongodb.net.:27017,hyperion-shard-00-01-f78fc.mongodb.net.:27017,hyperion-shard-00-02-f7
7
2018-05-09T10:06:45.554+0200 I NETWORK  [ReplicaSetMonitor-TaskExecutor-0] Successfully connected t
0-f78fc.mongodb.net.:27017 (1 connections now open to hyperion-shard-00-00-f78fc.mongodb.net.:27017
out)
2018-05-09T10:06:45.568+0200 I NETWORK  [thread1] Successfully connected to hyperion-shard-00-02-f7
7 (1 connections now open to hyperion-shard-00-02-f78fc.mongodb.net.:27017 with a 5 second timeout)
2018-05-09T10:06:45.823+0200 I NETWORK  [thread1] changing hosts to hyperion-shard-0/hyperion-shard
net:27017,hyperion-shard-00-01-f78fc.mongodb.net:27017,hyperion-shard-00-02-f78fc.mongodb.net:27017
0/hyperion-shard-00-00-f78fc.mongodb.net.:27017,hyperion-shard-00-01-f78fc.mongodb.net.:27017,hyper
.mongodb.net.:27017
2018-05-09T10:06:46.838+0200 I NETWORK  [ReplicaSetMonitor-TaskExecutor-0] Successfully connected t
2-f78fc.mongodb.net:27017 (1 connections now open to hyperion-shard-00-02-f78fc.mongodb.net:27017 w
t)
2018-05-09T10:06:46.848+0200 I NETWORK  [thread1] Successfully connected to hyperion-shard-00-00-f7
 (1 connections now open to hyperion-shard-00-00-f78fc.mongodb.net:27017 with a 5 second timeout)
2018-05-09T10:06:48.132+0200 I NETWORK  [ReplicaSetMonitor-TaskExecutor-0] Successfully connected t
1-f78fc.mongodb.net:27017 (1 connections now open to hyperion-shard-00-01-f78fc.mongodb.net:27017 w
t)
MongoDB server version: 3.4.14
WARNING: shell and server versions do not match
MongoDB Enterprise hyperion-shard-0:PRIMARY>
```

You have connected to your MongoDB server hosted by Atlas! You are now able to use the Mongo shell to create and modify databases on your server.

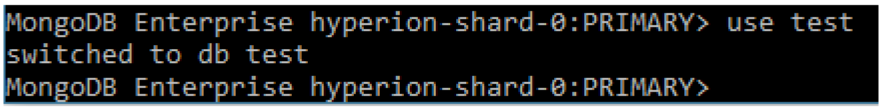
Extra resource

Watch **this short video** to see how easily you can connect to your Atlas database with the Mongo shell.

CREATE A DATABASE

Once you can access your database server (run by Atlas), you can issue instructions using Mongo to change your database. We are going to create a database.

To do this type the following using the Mongo shell: `use test` where *test* is the name of the database. If the database does not already exist, this instruction will create it.



```
MongoDB Enterprise hyperion-shard-0:PRIMARY> use test
switched to db test
MongoDB Enterprise hyperion-shard-0:PRIMARY>
```

MONGODB COMPASS

You may have noticed that when you installed MongoDB, MongoDB Compass was also installed. Compass allows you to interface with your database. You should be able to connect to your database using Compass too. Give it a try.

QUIT MONGO

To quit mongo, type `quit()` into the Mongo shell.

Compulsory Task 1

Follow these steps:

- Install MongoDB. See appropriate detailed installation instructions on MongoDB's download centre for details.
- Create a cluster on MongoDB Atlas.
- Add a mentor or code reviewer as a user to your cluster on Atlas, using the email address **allreviewers@hyperiondev.com** .
- Ensure that your firewall isn't blocking access to MongoDB.
- Connect to your cluster using the Mongo shell.
- Make a database called "test"
- Create a document called "myMongoDB" in which you include the following:
    - A screenshot that shows that you have added a mentor or code reviewer as a MongoDB user to your Atlas cluster.
    - A screenshot of your command line interface that shows how you have used the Mongo shell to connect to your MongoDB Atlas cluster.
    - A screenshot of your command line interface that shows that you have successfully created a database called "test" on your MongoDB Atlas cluster.

Completed the task(s)?

Ask an expert code reviewer to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.

SPOT CHECK 1 ANSWERS

1. Data are raw facts that have not yet been processed to reveal their meaning. Information, on the other hand, is the result of processing the raw data to reveal its meaning. Data processing can be as simple as organising data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modelling.
2. (any 3)
   a. Improved data sharing
   b. Better data integration
   c. Minimised data inconsistency
   d. Improved data access
   e. Improved decision-making
   f. Increased end-user productivity