**MongoDB**MongoDB is what is known as a NoSQL database. Basically what this means is that your output from it is going to be JSON rather than SQL. Mongo is generally used in combination with Node.JS to create web applications and is also part of Graylog, a popular log managing system used to store and view logs that have been forwarded from servers or client computers.

**Installation**The installation process for MongoDB will vary with each OS.  
  
Debian/Ubuntu:  
sudo apt install mongodb  
  
RHEL (You might need to add the repo file to you system, I’ll describe how to do that at the bottom of this document, for if the install fails):  
sudo yum install mongodb-org  
  
Gentoo:  
sudo emerge dev-db/mongodb  
  
Arch:  
sudo pacman -S mongodb  
  
To my knowledge, there is no mysql\_secure\_installation type thing to use with mongodb, at least not an official one.

**Making a Database**Now that we have MongoDB installed, we can connect to it and get to work on creating and managing our database. Since this guide is being written from a WRCCDC point of view, I will make that the theme of our database. MongoDB is the easiest database to get into:  
  
mongo  
  
Right off the bat, we run into quite a few differences between Mongo and the other two databases covered in this chapter. First, by default anyone can access it and see every database as well as create new ones. Second, the commands are quite different, due to it being a NoSQL database. Third, it’s used for different applications. To my knowledge, there aren’t currently any web apps that support using both, only one or the other.  
  
With that out of the way, let’s first list the existing databases, and then create our own:  
show dbs  
use wrccdc  
  
With Mongo, if you try to use a database and it doesn’t exist, it creates it. While this does simplify the creation of databases, it can also be annoying if you fat finger a name and end up creating data on a new database rather than the existing one you intended to use. To view database you are currently using, you can type:  
db  
  
MongoDB does not have tables, it has collections. MongoDB does not have rows, it has documents. MongoDB does not have columns, it has fields. When creating a collection, you can do with or without giving it a starting document. To create it without giving it any info:  
db.createCollection(“teams”)  
  
Alternatively, if you try to give a document to a non-existent collection, Mongo will create the collection and then give it the information. To insert a document into a collection:  
db.teams.insert({ color: “Blue”, role: “Defend” })  
  
Honestly, Mongo does not care what if you pass to a collection as long as you format it correctly, which is why you don’t need to format them at creation, as one would with a table. For example, right after to creating the Blue team document, we can now create one for Red team and give it extra fields and Mongo will still accept it.:  
db.teams.insert({ color: “Red”, role: “Attack”, reason: “They are rude” })  
  
Just like with SQL databases, Mongo can update information within a collection using an update command. Let’s pretend Red team breaks through our amazing defenses and messes with our MongoDB. They might change the reason field in the Red team document. They could do this by doing:  
db.teams.update({ “reason”: “They are rude”}, {$set:{“reason”: “No we are not”} })  
  
How evil. They could also delete that document entirely by using the remove command like so:  
db.teams.remove({ “reason”: “No we are not” })

**Other Things to Look Into**Some other to things that you may want to research from here are:

* Encrypting a database (Good to do in real world, not worth it in CCDC)
* Password protecting databases and permissions within MongoDB
* Graphical front-ends like Mongo-Express or MongoUI