

MongoDB & PHP - Welcome

What we'll cover:

- MongoDB quick overview
- Setting up MongoDB + PHP
- Actual PHP code!
(https://github.com/gatesvp/php_get_started)

MongoDB & PHP – Who Am I?

```
{  
  name: 'Gaëtan (Gates) Voyer-Perrault',  
  job_title: 'Customer Success Engineer',  
  email: 'gates@10gen.com',  
  twitter: '@gatesvp',  
  specialties: [ 'php', 'content monetization',  
    'mongodb', 'powershell', ':)' ]  
}
```

What is MongoDB?

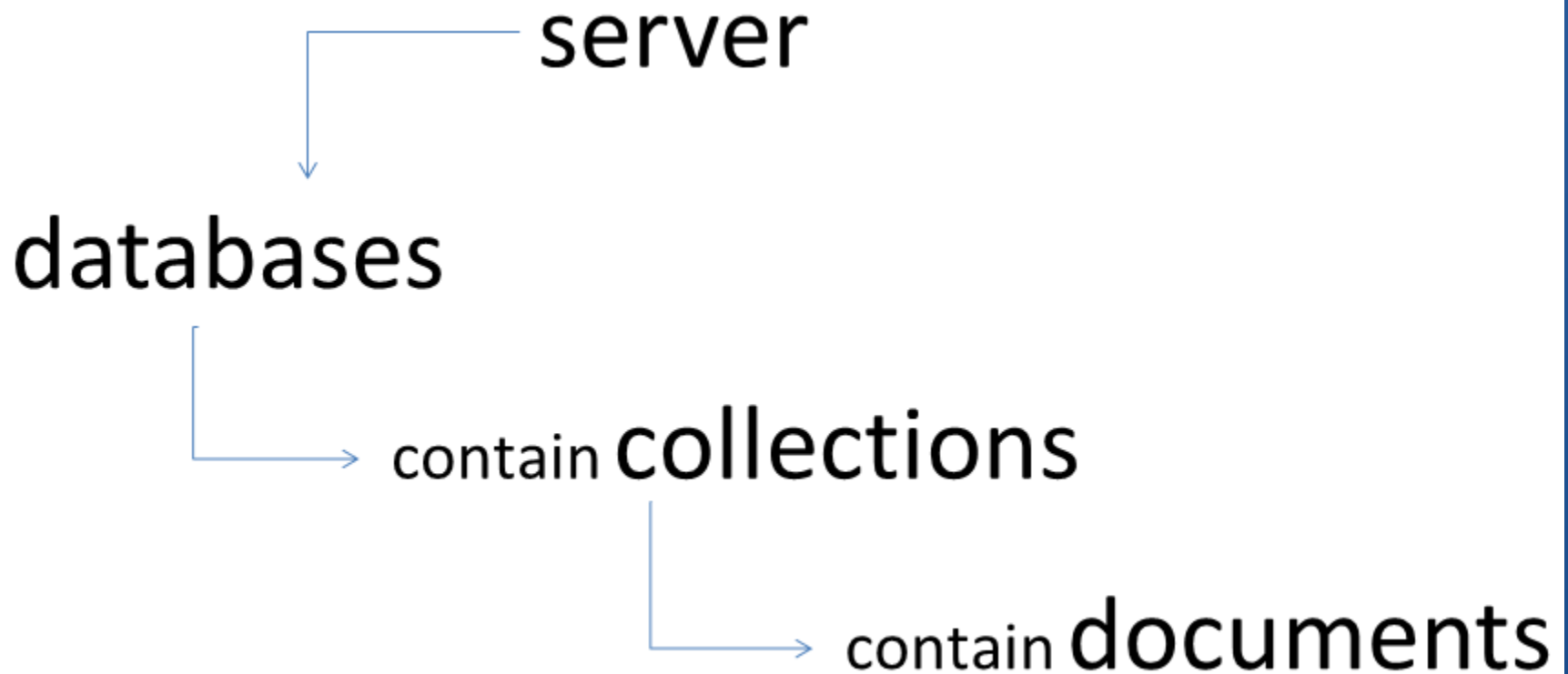
From the website: “*Scalable, high-performance, document-oriented database*”:

So it's a database, with specific features:

- Data is stored in BSON (*think JSON*)
- Native arrays, nested documents
- Indexing for faster queries
- Map / Reduce for aggregation

What is MongoDB?

Document-oriented



What is MongoDB?

Document-oriented

Collections are basically “bags of documents”.
In our case, bags of JSON objects.

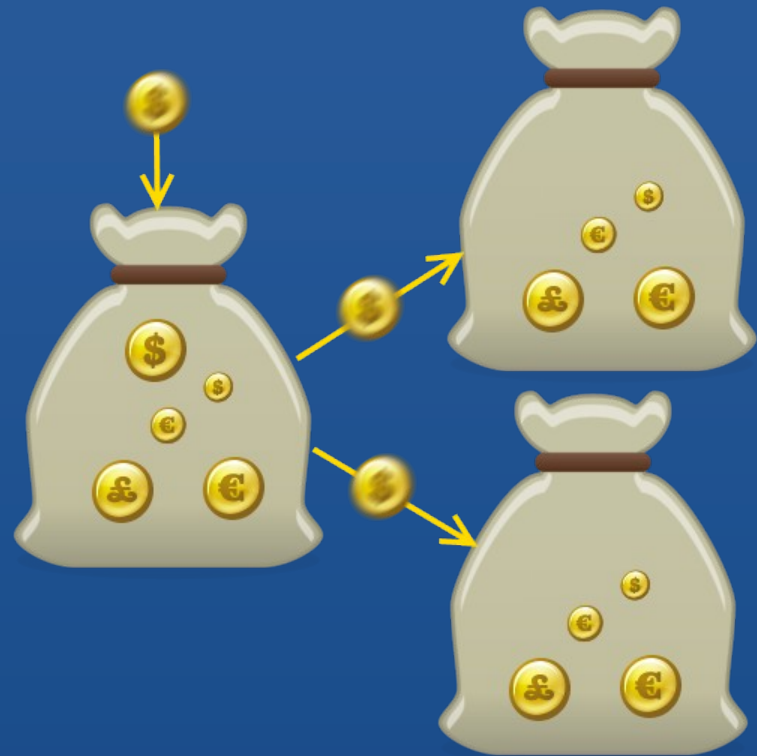
- Different Fields
- Different Sizes
- Indexable



What is MongoDB?

Scalable

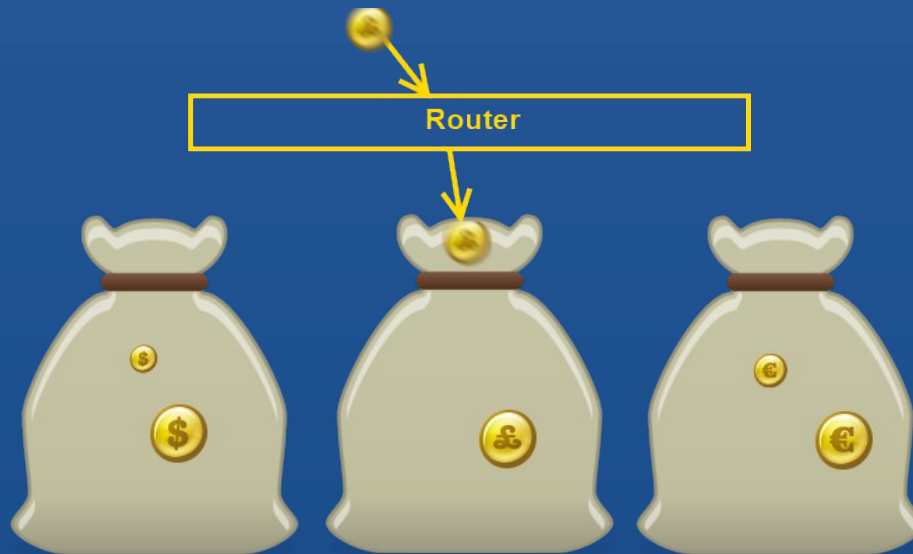
Read scaling and HA
via **Replication**



What is MongoDB?

Scalable

Write scaling via **Sharding**



Installing MongoDB & PHP

Installing MongoDB:

- Download your version
<http://www.mongodb.org/downloads>

On Linux:

```
$ curl http://downloads.mongodb.org/linux/mongodb-linux-x86_64-1.8.1.tgz > mongo.tgz
$ tar xzf mongo.tgz
$ sudo mkdir -p /data/db/
$ sudo chown `id -u` /data/db
$ ./mongodb-xxxxxxx/bin/mongod
```


Installing MongoDB & PHP

Installing PHP Driver:

```
$ sudo apt-get install php5-dev php5-cli php-pear  
$ sudo pecl install mongo
```

Open `php.ini` file and add:
`extension=mongo.so`

Let's start saving documents

Basic Connection:

```
<?php
try {
    $mongo = new Mongo('localhost:27017'); // default
    $db = $mongo->example;
    $collection = $db->test;

    $document = array('x' => 1);
    $collection->insert($document);

    print_r($document);
}
catch(Exception $e) { print($e->getMessage()); }
?>
```

Saving – Some notes

When we insert the document, a couple of “magic” things happen:

- The **example** DB is created
- The **test** collection is created
- An index is created the **_id** field
- An **_id** is created for the document
- The **_id** is added to the document

A more complex document

JSON representation

- Notice the friends array, contact sub-document.

```
{  
  _id: 'gates',  
  name: 'Gaëtan Voyer-Perrault',  
  friends: ['bernie', 'alvin'],  
  followers: 18,  
  contact: {  
    twitter: '@gatesvp',  
    email: 'gates@10gen.com'  
  }  
}
```

A more complex document

PHP representation

- Nested hashtables

```
$doc = array( '_id' => 'gates',  
    'name' => 'Gaëtan Voyer-Perrault',  
    'friends' => array('bernie', 'alvin'),  
    'followers' => 18,  
    'contact' => array( 'twitter' => '@gatesvp',  
        'email' => 'gates@10gen.com')  
    )  
)
```

Some Basic Queries

Queries accept a document / hashtable:

```
// Basic query
```

```
$query = array( '_id' => 'gates' );  
$result = $collection->findOne($query);
```

```
// Query on array
```

```
$query = array( 'friends' => 'alvin' );  
$result = $collection->findOne($query);
```

```
// Query on sub-document
```

```
$query = array( 'contact.twitter' => '@gatesvp' );  
$result = $collection->findOne($query);
```

Some Basic Queries – less fields

Queries can specify only certain fields

```
// Filter fields
```

```
$query = array( '_id' => 'gates' );  
$fields = array( '_id' => 0, 'name' => 1, 'friends' => 1 );  
$result = $collection->findOne($query, $fields);
```

Some Advanced Queries

Mongo has several \$operators:

```
// Greater than ($gt)
```

```
$query = array( 'followers' => array( '$gt' => 10 ) );  
$results = $collection->find($query);
```

```
// IN ($in)
```

```
$query = array( '_id' =>  
    array( '$in' => array('gates','jim') )  
);  
$results = $collection->find($query);
```

```
// also support for $or, $exists, $mod, $type, $size  
// regexes and negated versions of these.
```


Cursoring through results

Result of a find() is cursor.
Cursor works with foreach.

```
$collection->insert(array('x' => 1));  
$collection->insert(array('x' => 2));  
$collection->insert(array('x' => 3));
```

```
$results = $collection->find();
```

```
foreach($results as $r) {  
    print_r($r);  
}
```

Cursoring through results

Alternately works with while loop:

```
$collection->insert(array('x' => 1));  
$collection->insert(array('x' => 2));  
$collection->insert(array('x' => 3));
```

```
$results = $collection->find();
```

```
while($results->hasNext()){  
    $r = $results->getNext();  
    print_r($r);  
}
```

Cursoring through results

Cursor also does counting, skipping, limiting:

```
// Greater than ($gt)
```

```
$collection->insert(array('x' => 1));
```

```
$collection->insert(array('x' => 2));
```

```
$collection->insert(array('x' => 3));
```

```
print($collection->find()->count()); // 3
```

```
$res = $collection->find()->limit(1); // x=>1
```

```
$res2 = $collection->find()->skip(1)->limit(1); // x=>2
```

Updating Documents

Query + Update command

// Does not behave as you would expect

```
$query = array( '_id' => 'gates' );  
$update = array( 'followers' => 19 );  
$collection->update($query, $update);
```

// Instead

```
$query = array( '_id' => 'gates' );  
$update = array( '$set' => array('followers' => 19) );  
$collection->update($query, $update);
```

Updating Documents

Other operators

```
// Instead
$query = array( '_id' => 'gates' );
$update = array(
    '$set' => array('name' => 'GVP',
        'contact.website' => 'http://10gen.com/'),
    '$inc' => array('followers' => 1),
    '$push' => array('friends' => 'antoine'),
    '$unset' => array('contact.twitter' => 1)
);
$collection->update($query, $update);
```

More on updating

- There are several more operators
\$push, \$pop, \$pull, \$addToSet
\$rename
- Operators are atomic within a document.
- Only one operation per field.
You cannot \$pop & \$pull an array in one command.

Updating multiples

By default, only first doc is updated.
We can change this.

```
$collection->insert(array('x'=>1));  
$collection->insert(array('x'=>1));  
$collection->insert(array('x'=>3));
```

```
$query = array('x' => 1);  
$update = array('$inc' => array('x' => 1));  
$options = array('multiple' => true);  
$collection->update($query, $update, $options);
```

Deleting

Very similar to updating.

```
$collection->remove(); // deletes everything!
```

```
$query = array('x' => 1);  
$collection->remove($query); // deletes where x=>1
```

Beware empty query!

Update if not exist

We call this the '**upsert**'

```
// Upsert
$query = array('_id' => 'gates');
$update = array('$inc' => array('followers' => 1));
$options = array('upsert' => true);
$collection->update($query, $update, $options);
```

A word about transactions

MongoDB does **not** have joins.

Likewise:

- ... no transactions across documents
- ... no transactions across collections

If you need these take a look at
`findAndModify` + two-phase commit

A word about exceptions

Catch them.

Especially when using Replica Sets.
Failover is not instant.

More words about exceptions

Timeouts you'll want to check or set.

Connection timeouts:

```
try{
    $connString = 'server1:27017';
    $connOptions = array('timeout' => 3000); // 3 seconds
    $mongo = new Mongo($connString, $connOptions);
}
catch(MongoConnectionException $ex){ // log }
```

More words about exceptions

Timeouts you'll want to check or set.

Cursor timeouts:

```
try{
    ...
    $res = $coll->find($query)->timeout(1000) // 1 second

    while($res->hasNext()){
        $data = $res->getNext();
    }
}
catch(MongoCursorException $ex){ // log }
catch(MongoCursorTimeoutException $ex){ // log }
```

A word about arrays

PHP arrays are a funny beast.
Take the following doc

```
$document = array(  
    'normal' => array('first','second'),  
    'crazy' => array("0" => 'first', '1' => 'second'),  
    'arrayObj' => new ArrayObject(array('first',  
'second')),  
    'object' => array('1' => 'first', '2' => 'second')  
);  
  
$collection->insert($document);
```

A word about arrays

\$push only works on arrays as stored in the DB.

```
$collection->update(array(),  
  // works  
  array('$push' => array('normal' => 'third')),  
  array('$push' => array('crazy' => 'third')),  
  
  // fails  
  array('$push' => array('object' => 'third')),  
  array('$push' => array('arrayObj' => 'third'))  
);
```

Other features

MongoDB has lots of other features.

- DB commands: accessible from PHP
- GridFS: store large files
- Indexing: optimize queries
- Geo-spatial queries



try at try.mongodb.org

A Tiny MongoDB Browser Shell (mini tutorial included)
Just enough to scratch the surface

```
MongoDB browser shell version: 0.1.2
connecting to random database
type "help" for help
type "tutorial" to start the tutorial
>
```

drivers at mongodb.org

mongodb.org Supported

C
C++
C#
Java
Javascript
Perl
PHP
Python
Ruby

Community Supported

REST	node.js
Clojure	Objective C
ColdFusion	PHP
Delphi	PowerShell
Erlang	Blog post
F#	Python
Go: gomongo	Ruby
Groovy	Scala
Haskell	Scheme (PLT)
Javascript	Smalltalk: Dolphin
Lua	Smalltalk

 **download at** mongodb.org

conferences, appearances, and meetups
<http://www.10gen.com/events>



Facebook

<http://bit.ly/mongofb>



| Twitter

[@mongodb](https://twitter.com/mongodb)



| LinkedIn

<http://linkd.in/joinmongo>

support, training, and this talk brought to you by


10gen


10gen

© Copyright 2010 10gen Inc.

 **mongoDB**