

# Introduction to MongoDB

---

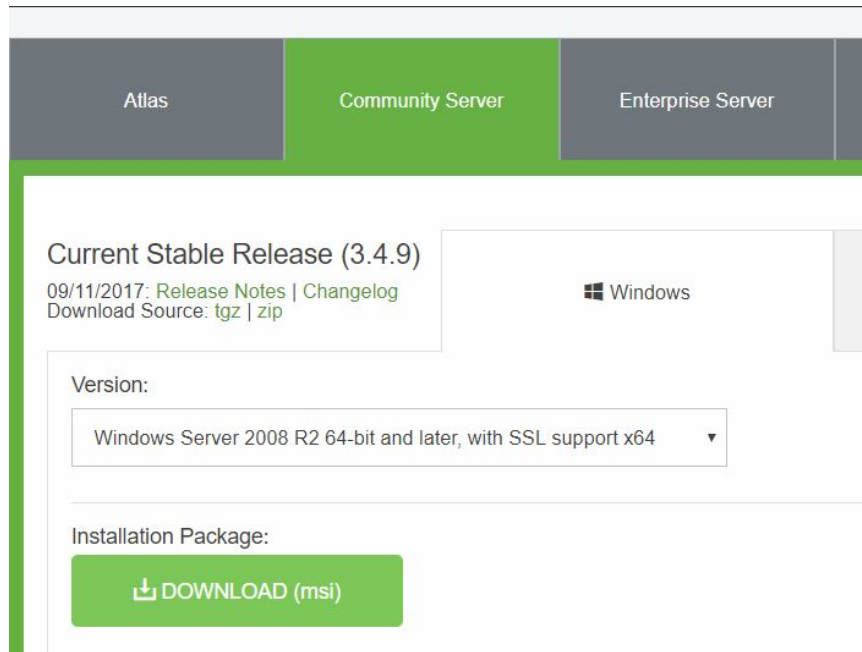
26/03/2018

# What is MongoDB

- It's a free NoSQL document oriented database.
  - Not only SQL, since SQL queries are supported.
- It uses JSON-like documents to save data.
  - Flexible schemas - Allows missing fields.
- High performance, high availability, and automatic scaling.
  - Used by large companies like Amazon.

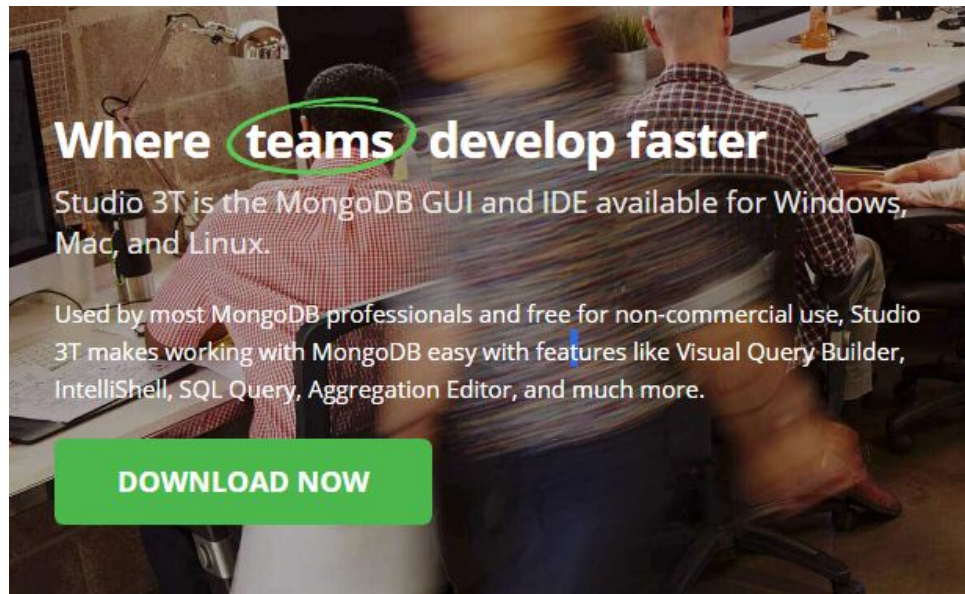
# Installation

## MongoDB community server



The screenshot shows the MongoDB download page for the Community Server. At the top, there are three tabs: 'Atlas', 'Community Server' (which is selected and highlighted in green), and 'Enterprise Server'. Below the tabs, the 'Current Stable Release (3.4.9)' is displayed, along with the date '09/11/2017' and links to 'Release Notes' and 'Changelog'. The 'Download Source' is listed as 'tgz' or 'zip'. A 'Windows' icon is visible. Under the 'Version:' section, a dropdown menu is open, showing 'Windows Server 2008 R2 64-bit and later, with SSL support x64'. At the bottom, under 'Installation Package:', there is a green button with a download icon and the text 'DOWNLOAD (msi)'.

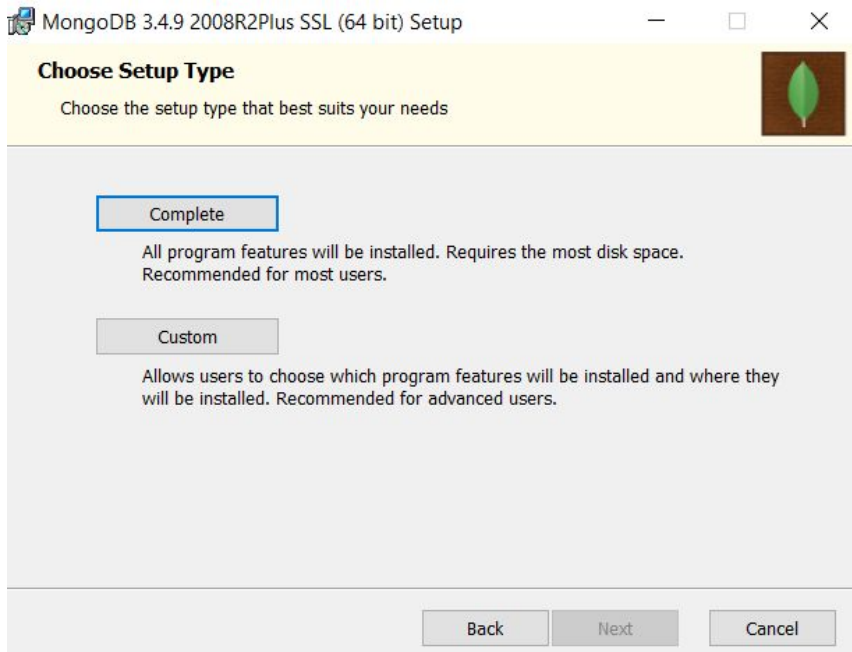
## Studio 3T



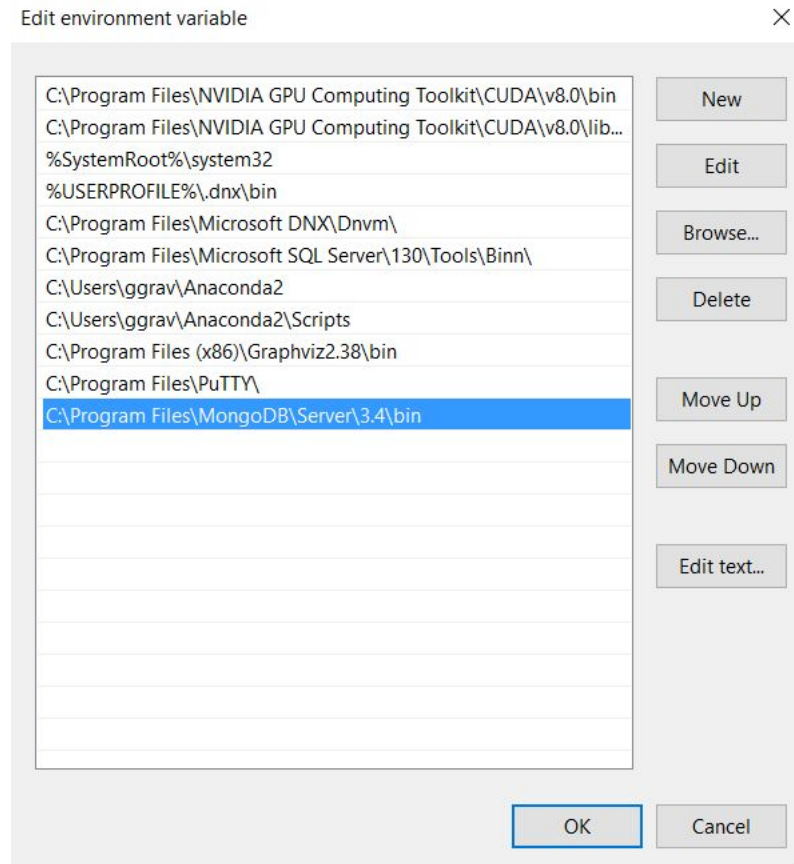
The advertisement for Studio 3T features a background image of two people working at a desk with multiple monitors. The text 'Where teams develop faster' is prominently displayed at the top, with the word 'teams' circled in green. Below this, it states 'Studio 3T is the MongoDB GUI and IDE available for Windows, Mac, and Linux.' Further down, it mentions 'Used by most MongoDB professionals and free for non-commercial use, Studio 3T makes working with MongoDB easy with features like Visual Query Builder, IntelliShell, SQL Query, Aggregation Editor, and much more.' At the bottom, there is a large green button with the text 'DOWNLOAD NOW'.

# Installation

## Step 1



## Step 2



# Execution

Step 3 -> Create folder "C:\data\db\" -> Open cmd -> run "mongod"

```

C:\Users\ggrav>cmd
Command Prompt - mongod

(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\ggrav>mongod
2017-10-13T03:50:03.878-0700 I CONTROL [initandlisten] MongoDB starting : pid=17672 port=27017 dbpath=C:\data\db\ 64-bit host=
2017-10-13T03:50:03.878-0700 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten] db version v3.4.9
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten] git version: 876ebee8c7dd0e2d992f36a848ff4dc50ee6603e
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1u-fips 22 Sep 2016
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten] allocator: tcmalloc
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten] modules: none
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten] build environment:
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten]     distmod: 2008plus-ssl
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten]     distarch: x86_64
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten]     target_arch: x86_64
2017-10-13T03:50:03.880-0700 I CONTROL [initandlisten] options: {}
2017-10-13T03:50:03.882-0700 I - [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine
the active storage engine to 'wiredTiger'.
2017-10-13T03:50:03.882-0700 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=7611M,session_max=20000,evicti
threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=
0000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-10-13T03:50:04.217-0700 I CONTROL [initandlisten]
2017-10-13T03:50:04.217-0700 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-10-13T03:50:04.218-0700 I CONTROL [initandlisten] **           Read and write access to data and configuration is unrestric
2017-10-13T03:50:04.218-0700 I CONTROL [initandlisten]
2017-10-13T13:50:04.469+0300 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/c
2017-10-13T13:50:04.472+0300 I NETWORK [thread1] waiting for connections on port 27017
```

# What is a document

- A sum of key / value pairs

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```



← field: value  
← field: value  
← field: value  
← field: value

# What is a collection

- A grouping of MongoDB **documents**.
- A **collection** is the equivalent of an RDBMS table.
- A **collection** exists within a single database.
- **Collections** do not enforce a schema.
- **Documents** within a collection can have different fields.
- **Collections** are grouped into **databases**.

Example: **db** = tweets, **collections** = greek\_tweets, english\_tweets, etc.

# Operations

- Run / Connect to mongo
  - Import documents
  - Insert documents
  - Create queries
  - Quick data lookups
  - Indexing
-



# Connection

Studio 3T:

- Studio 3T -> Connect -> New connection -> (enter name) -> Save -> Connect
- Right click -> Add database -> (enter name) -> ok

Python:

```
1      import pymongo
2
3      client = pymongo.MongoClient('localhost', 27017)
4      db = client['yelp']
```

# Import dataset

Studio 3T:

- Connect -> Select DB -> Import collections -> JSON -> Add sources -> Rename target collections -> Rename collections\* -> Start import

\* Rename collections to “restaurants”, “reviews” and “users”.

# Insert documents

Python:

```
38     # Find the coordinates for each restaurant and
39     # save them to an external collection
40     all_restaurants = find_all_restaurants()
41     for restaurant in all_restaurants:
42         json_obj = {
43             'name': restaurant['name'],
44             'business_id': restaurant['business_id'],
45             'longitude': restaurant['longitude'],
46             'latitude': restaurant['latitude']
47         }
48     insert_to_db(json_obj, 'restaurants_coordinates')
```

# Querying

Studio 3T:

- Right click collection -> Open Intellishell
- Examples:
  - `db.restaurants.find({})`
  - `db.restaurants.find({"neighborhood": "Downtown"})`
  - `db.restaurants.find({ $and: [{"neighborhood": "Southeast"}, {"city": "Las Vegas"} ]})`

Python:

```
24 db['restaurants'].find({'neighborhood': neighborhood})
25 db['restaurants'].find_one({'business_id': restaurant_id})
26 db['reviews'].find({"$and": [{"business_id": restaurant_id}, {"stars": 5}]})
```

# Data lookups - For quick data checking

Studio 3T:

- Right click collection -> Open Intellishell
  - Ex: `db.collection_name.find( { "Search_Field": "value" }, { "Field_to_display": 1 } )`
  - `db.restaurants.find( { "neighborhood": "Downtown" }, { "name": 1 } )`
  - `db.restaurants.find( { "name": /. *pollos.*/ }, { "text": 1 } )`
    - /. \* is the regex equivalent for: “any single character, 0 or more times”

Python:

```
27     def find_reviews_that_contain_a_word(word):
28         return db['reviews'].find({'text': {'$regex': '.*' + word + '.*'}})
29
```

"this lecture isn't mandatory,  
however it will offer you valuable  
informa-"



# Indexing

- Basic indexing
  - Speeds up queries on specific fields
  - Mostly used fields should be indexed
  - Index responsibly : Too many indices might slow down the database
- Text indexing
  - Faster text search queries on string content
  - Term lookup
  - Exact phrase lookup
  - Automatic relevance sorting

# How to - Text indexing

The screenshot shows the Studio 3T interface with the 'existing\_customers' collection selected in the left sidebar. A context menu is open over the collection, listing various actions. The 'Add Index...' option is highlighted. The main window displays a table of documents from the 'existing\_customers' collection, showing fields like '\_id' and 'name'.

**Context Menu Options:**

- Open Collection Tab
- Open Collection With Custom Page Size...
- Open IntelliShell
- Open SQL
- Open Aggregation Screen
- Open Map-Reduce
- Export Collection...
- Import Data...
- Copy Collection
- Rename Collection...
- Clear Collection
- Drop Collection
- Add Index...**
- Add/Edit Validator...
- Add View Here...
- Analyze Schema...
- Compare To...
- Current Operations
- Collection Statistics
- Server Info
- Refresh Selected Item
- Refresh All
- Choose Color
- Disconnect

**Document Table:**

Value	Type
{ 14 fields }	Document
{ 13 fields }	Document
{ 14 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 15 fields }	Document
{ 14 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 12 fields }	Document
{ 14 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 12 fields }	Document
{ 13 fields }	Document
{ 14 fields }	Document
{ 13 fields }	Document
{ 13 fields }	Document
{ 14 fields }	Document



# How to - Text indexing

The screenshot shows the Studio 3T for MongoDB interface. The left sidebar displays the database structure, with the 'test-db' database selected and the 'existing\_customers' collection highlighted. The main window is titled 'Add Index' and shows the 'Name' field set to 'compound\_text\_index'. The 'Indexed Fields' table is empty, and the 'Index Options' section shows checkboxes for 'Unique', 'Sparse', 'Drop Duplicates', 'Expire After', and 'Create In Background'. The 'Create Index' button is at the bottom right.

Studio 3T for MongoDB

Connect Collection IntelliShell SQL Aggregate Map-Reduce Export Import Users Roles Schema Compare Feedback

Search Open Connections (Cmd+F)

alice Studio 3T ReplicaSet (3 servers) test-db existing\_customers

Name: compound\_text\_index

Fields Text Options Geo Options Partial Index Options JSON Collation

Indexed Fields:

Field	Index Type

Add Field(s)...  
Remove Field  
Move Up  
Move Down

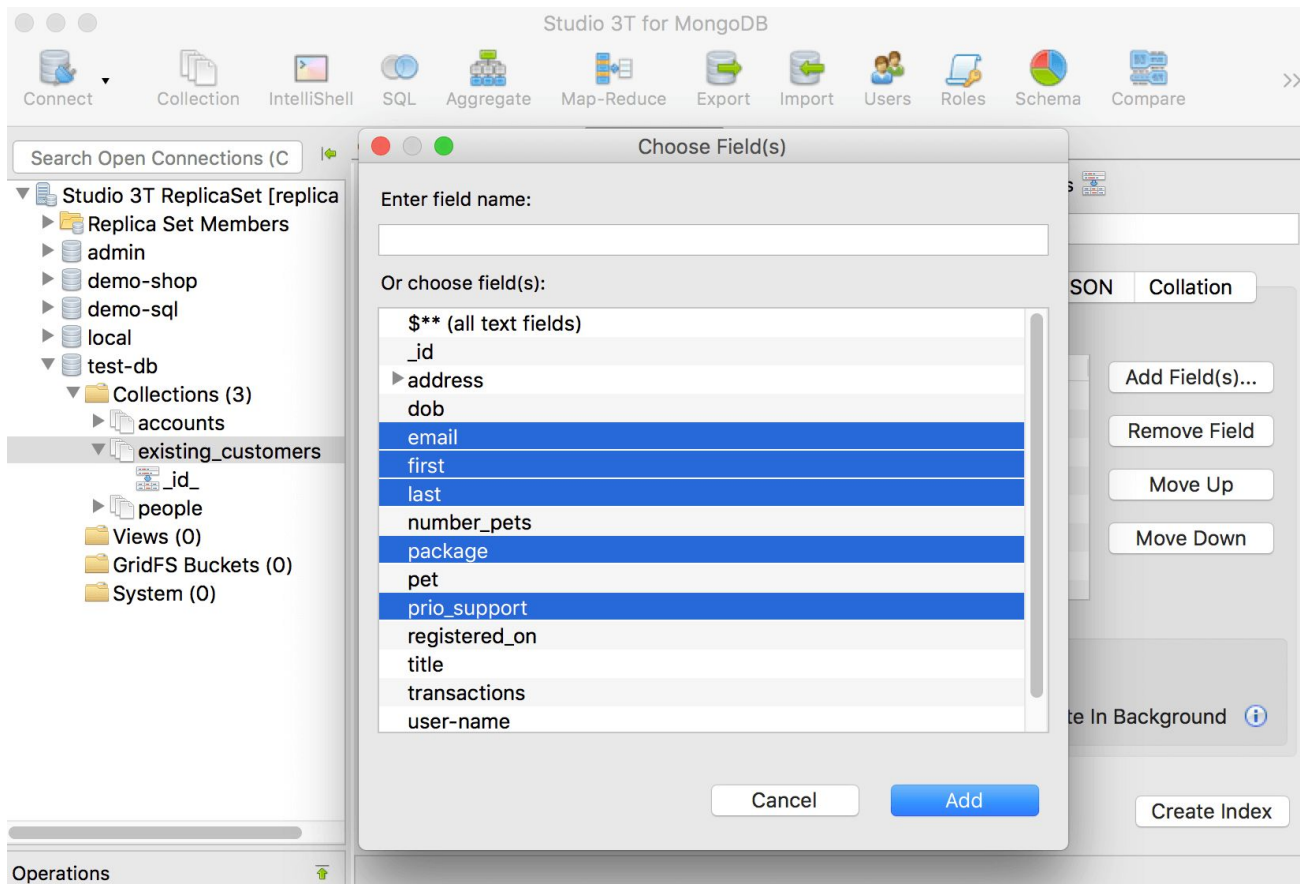
Index Options

☐ Unique ☐ Sparse ☐ Drop Duplicates  
☐ Expire After  Seconds ☐ Create In Background

Create Index

Operations

# How to - Text indexing



# How to - Text indexing

Studio 3T for MongoDB

Connect Collection IntelliShell SQL Aggregate Map-Reduce Export Import Users Roles Schema Compare Feedback

Search Open Connections (Cmd+F)

▼ Studio 3T ReplicaSet [replica set]

- ▶ Replica Set Members
- ▶ admin
- ▶ demo-shop
- ▶ demo-sql
- ▶ local
- ▼ test-db
  - ▶ Collections (3)
    - ▶ accounts
    - ▶ existing\_customers
    - ▶ people
  - ▶ Views (0)
  - ▶ GridFS Buckets (0)
  - ▶ System (0)

alice Studio 3T ReplicaSet (3 servers) test-db existing\_customers

Name: compound\_text\_index

Fields Text Options Geo Options Partial Index Options JSON Collation

Indexed Fields:

Field	Index Type
email	text
first	ascending
last	descending
package	hashed
prio_support	text
	2dsphere
	2d
	geoHaystack

Add Field(s)... Remove Field Move Up Move Down

Index Options

☐ Unique ☐ Sparse ☐ Drop Duplicates

☐ Expire After  Seconds ☐ Create In Background

Create Index

Operations

# How to - Text indexing

Studio 3T for MongoDB

Connect Collection IntelliShell SQL Aggregate Map-Reduce Export Import Users Roles Schema Compare Feedback

Search Open Connections (Cmd+F) Add Index

alice Studio 3T ReplicaSet (3 servers) test-db existing\_customers

Name: compound\_text\_index

Fields Text Options Geo Options Partial Index Options JSON Collation

Indexed Fields:

Field	Index Type
email	text
first	text
last	text
package	text
prio_support	text

Add Field(s)... Remove Field Move Up Move Down

Index Options

☐ Unique ☐ Sparse ☐ Drop Duplicates

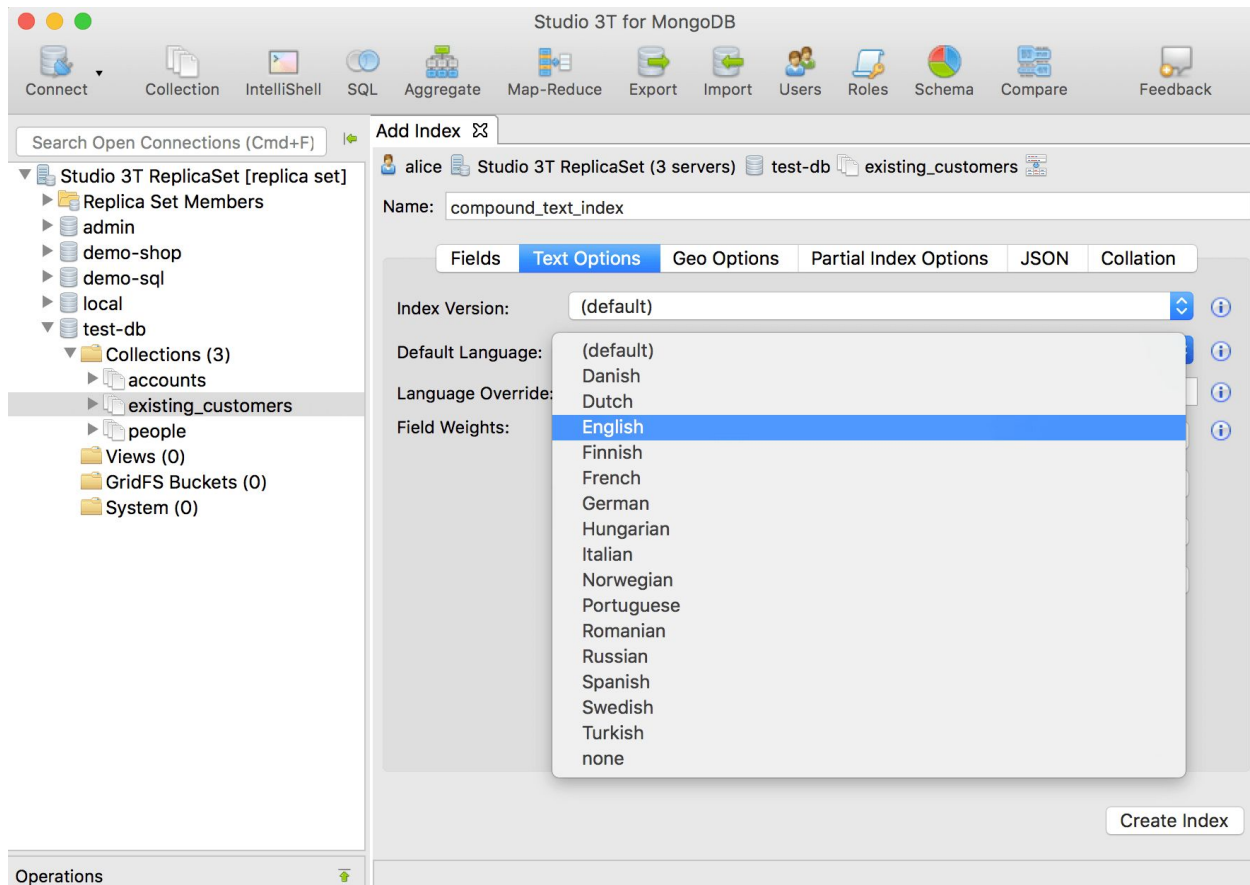
☐ Expire After  Seconds

☒ Create In Background

Create Index

Operations

# How to - Text indexing

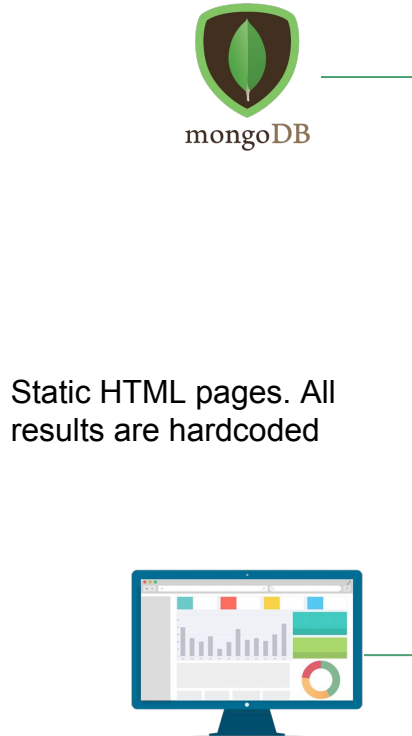


# Querying with text indices

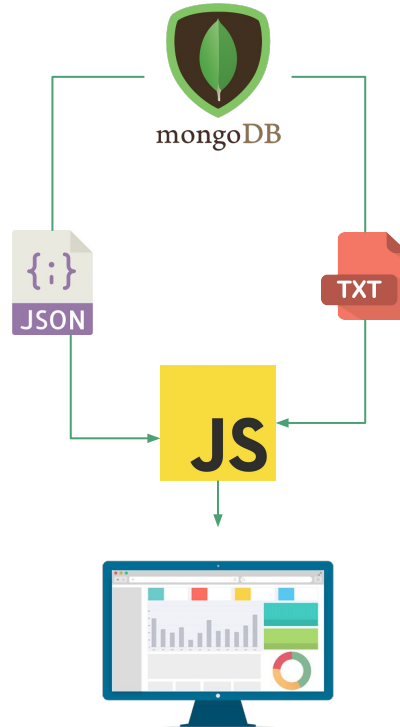
- `db.stores.find( { $text: { $search: "java coffee shop" } } )`
  - `$text` tokenizes the search string and performs a logical OR
  - Will search on all indexed fields
  - Results include a relevance score for each record
- `db.stores.find( { $text: { $search: "java \"coffee shop\"" } } )`
  - Will match exact phrases “java” OR “coffee shop”
- `db.stores.find( { $text: { $search: "java shop -coffee" } } )`
  - Will match (“java” OR “shop”) AND NOT(“coffee”)

# Web app development

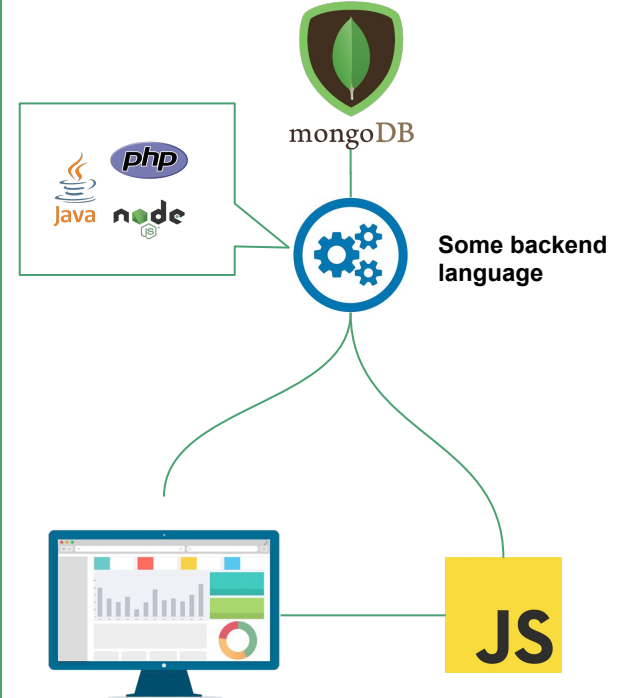
## Possibility #1



## Possibility #2



## Possibility #3





A python-based microframework, suitable for small-scale applications.

How to create a simple Flask App:

#### app.py

```
from flask import Flask, render_template

app = Flask(__name__, template_folder='views')

@app.route("/")
def home():
    uni_name = "Aristotle University of Thessaloniki"
    return render_template('some_html_file.html', uni_name=uni_name)

if __name__ == "__main__":
    app.run(debug=True, host='127.0.0.1', port=5110)
```



## some\_html\_file.html

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <p>Welcome to your first Flask application!</p>
    <!-- Variable is passed, using the jinja2 templating engine -->
    <h2>{{ uni_name }}</h2>
  </body>
</html>
```



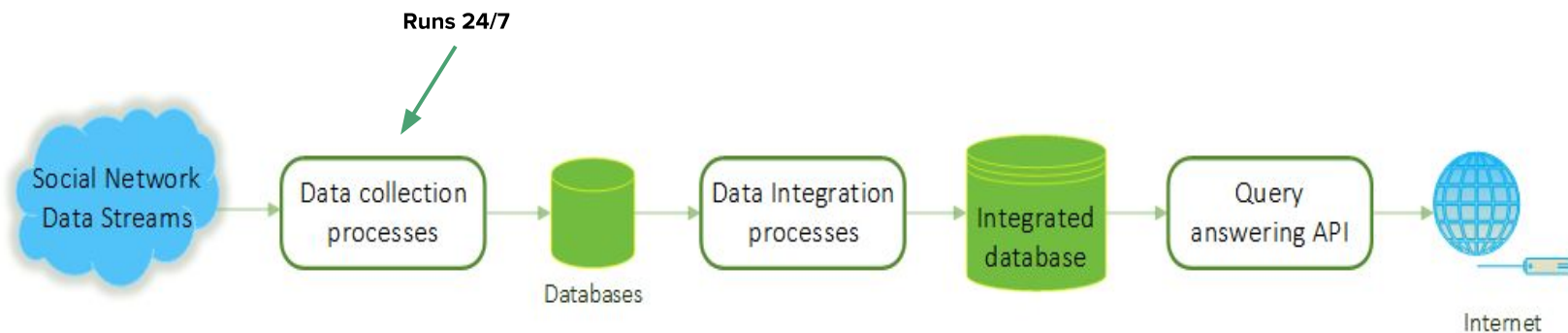
Welcome to your first Flask application!

**Aristotle University of Thessaloniki**

# Diligent - A Social Media Data Integration platform

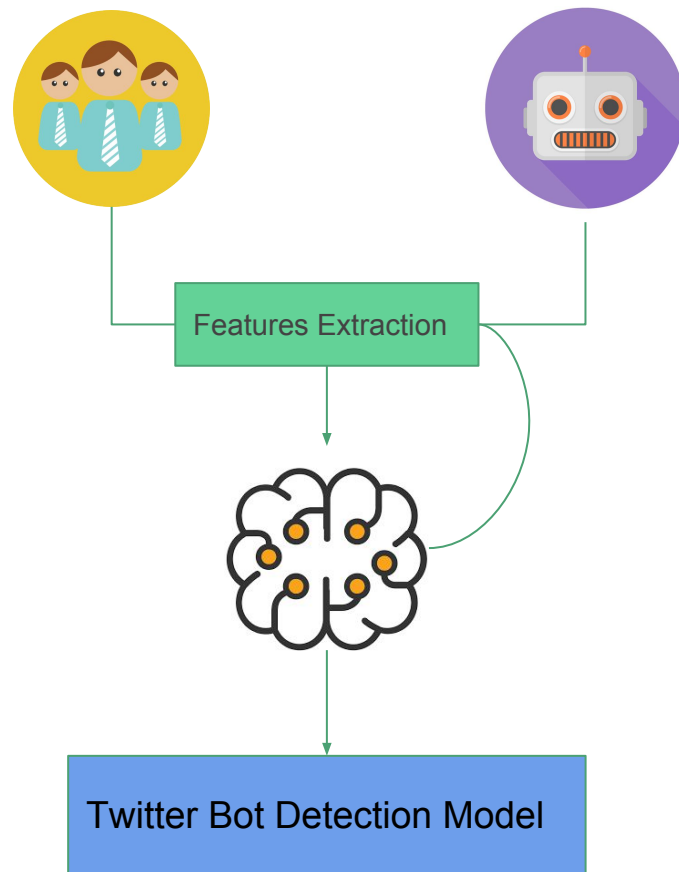
- Social media data explosion - Information Era
  - All data and metadata are saved into databases
  - Including private information (fingerprint, face, political views, etc.)
- Various data sources can be Integrated
  - Extract valuable insights
  - Personalized advertising
  - Sentiment analysis
  - -> Build a complete image - Increase data value \$\$
- Machine Learning opportunities
  - Integrated data as input

# Diligent - System pipeline

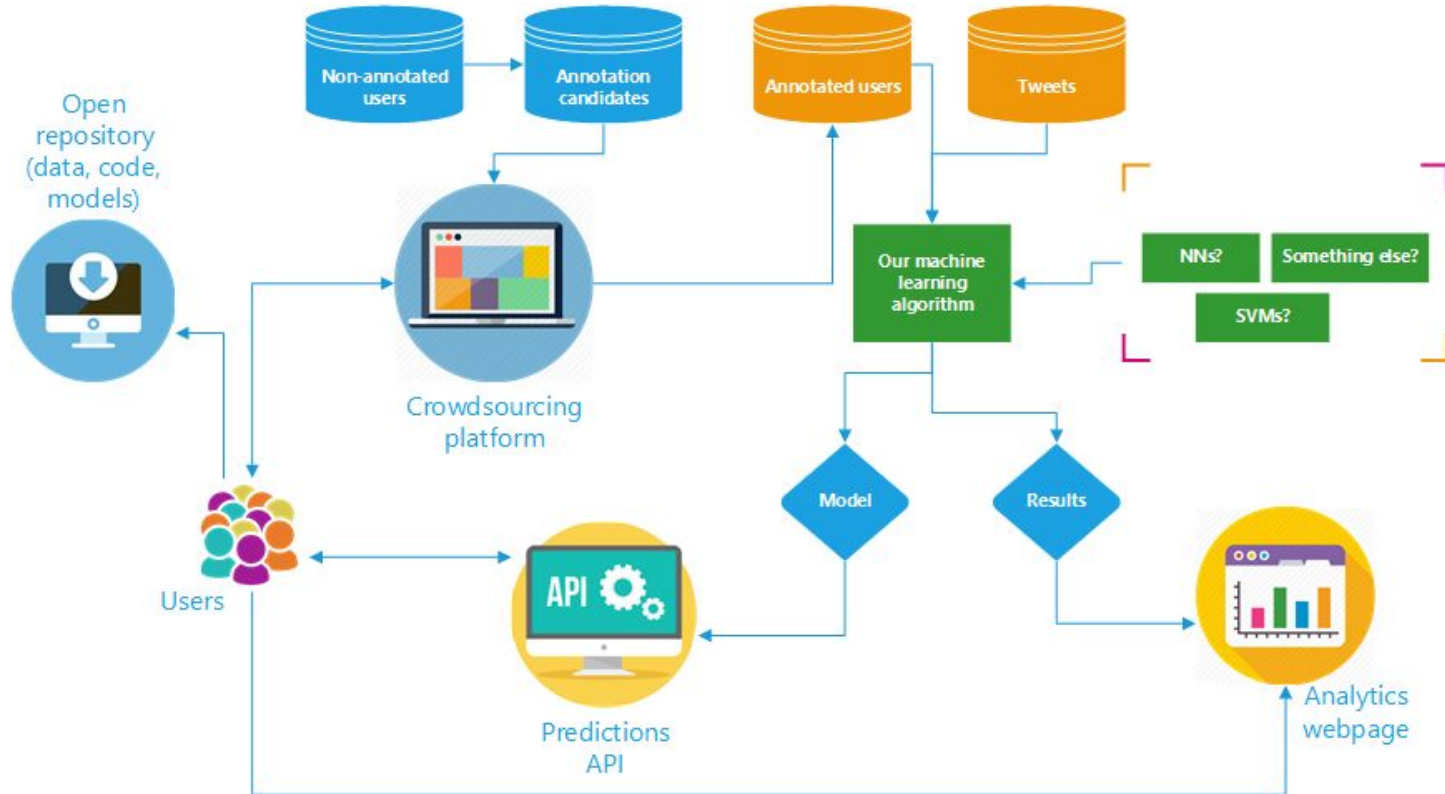


# Twitter bots detection

- Own a set of pre-annotated users as bots or humans
- Extract features from each user (focus on sentiment)
  - E.g. “num of tweets”, “tweets entropy”, “average sentiment polarity”
- Feed a machine learning algorithm all the features vectors
- Evaluate and extract model



# Full-scale architecture



# Resources

- [Dataset jsons](#)
- [Github link](#)
- [MongoDB](#)
- [Flask](#)
- [Flask-RESTful](#)
- [Python](#)