NODE AND AZURE

Server Structure

FILE AND STRUCTURE

על מנת לעבוד עם שרתים ושירותים חיצוניים,
עלינו לעבוד עם שיטה מסוימת המתאימה לפלטפורמה.
היום נלמד על קבצים חשובים במבנה השרת

EXPRESS SERVER

The easiest way create an Express server is using Express's server creator

We just type "express <name of the app>" into CMD
And a base client server will be created for us!
(never forget *to npm install –g express*)

Lets dive into its structure…

```
C:\Users\itamar\Downloads>express test

   create : test
   create : test/package.json
   create : test/app.js
   create : test/public/images
   create : test/public/javascripts
   create : test/public
   create : test/public/stylesheets
   create : test/public/stylesheets/style.css
   create : test/routes
   create : test/routes/index.js
   create : test/routes/users.js
   create : test/views
   create : test/views/index.jade
   create : test/views/layout.jade
   create : test/views/error.jade
   create : test/bin
   create : test/bin/www

   install dependencies:
     > cd test && npm install

   run the app:
     > SET DEBUG=test:* & npm start
```

EXPRESS SERVER - THINGS TO REMEMBER…

A server is built mainly from three parts:

1. Routes – anything related to API calls
2. Logic – anything related to the server itself
3. DataBase

When dealing with Node,

we take the DataBase out from the server

(as we saw in MongoDb)

EXPRESS SERVER

- /bin
  - www
- /routes
- /views
  - /js
  - /css
  - …
  - /views
- app.js

Bin Folder: the server settings folder

WWW: (no JS extension!)

- The main server file
- Holds the server settings
- Used to store and create object there are not route oriented

EXPRESS SERVER

- /bin
  - www
- /routes
- /views
  - /js
  - /css
  - …
  - /views
- app.js

Routes Folder:

- This is where the routes usually are (but it not mandatory)
- It is accustomed that each route category has its own JS file

App.js: (or whichever name you'll want)

- The main routing file
- Holds the routing settings
- Used to store and manage the routes in the server

EXPRESS SERVER

- /bin
  - www
- /routes
- /views
- /public
  - /js
  - /css
  - …
  - /templates
- app.js

Views Folder: *(in use with Jade or EJS)*

- Holds the main server views
- We can change the default folder with:

```
app.set('views', __dirname + '/yourViewDirectory');
```

Public Folder:

- This is where the client side files are
- It is accustomed that each file category has its own folder

CONFIG.JSON

This is the configuration JSON file (see what I did here? hehe)

<u>What is does:</u>

- loads the default configuration file;
- loads environment specific configuration file and overrides defaults;

<u>and then</u>:

- uses environment variables;
- and command-line arguments to override data from configuration files.

How to install:

- npm install config.json

How to use:

- *https://www.npmjs.com/package/config.json*

EXPRESS SERVER – WWW FILE EXAMPLE

```
var app = require('../app');
var debug = require('debug')('example:server');
var http = require('http');
```

Lets use the app file, in app.js we did `module.exports = app;`

```
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);
```

We start our server and listen to the port we want

process.env.PORT – tells the system to try and use it default port

EXPRESS SERVER - APP.JS EXAMPLE

We require the routing files

```
var routes = require('./routes/index');
var users = require('./routes/users');
var home = require('./routes/home');
```

Then we define when to route to each

```
app.use('/', routes);
app.use('/', users);
app.use('/home', home);
```

Home.js example

```
/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index.ejs');
});

module.exports = router;
```

MONGO AND AZURE

We need a way to run Mongo backend

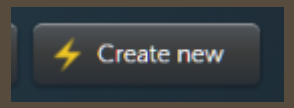
There are two ways:

1. mLab – a host for your Database, free up to 0.5Gb
2. Azure VM – running it inside a VM

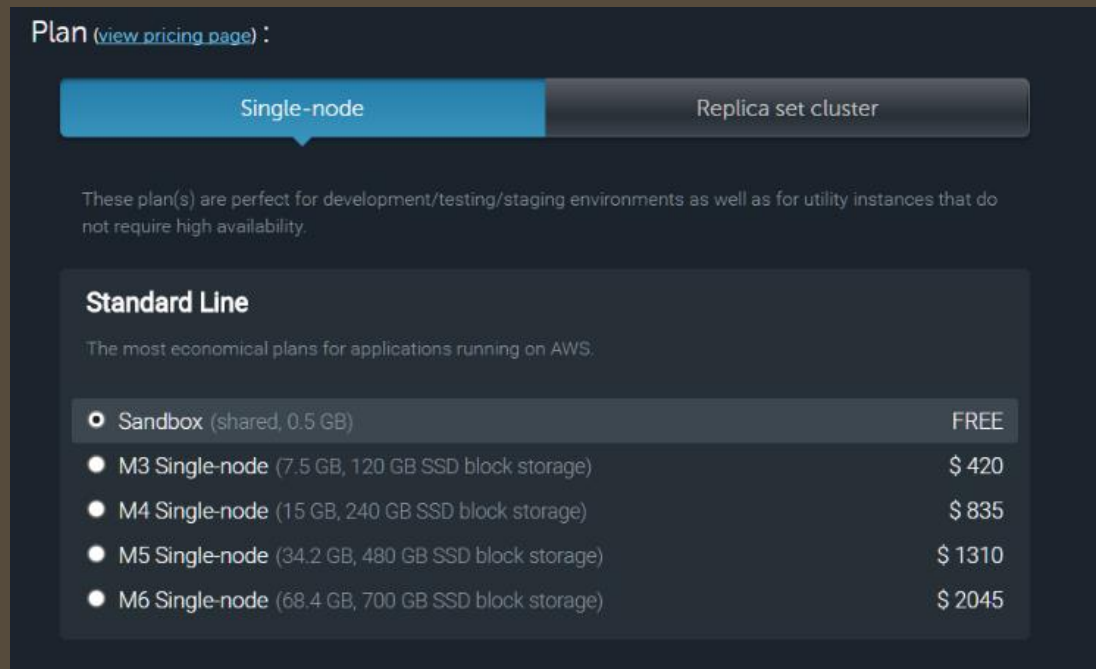
MLAB

mLab is a mongo service

To use mLab:

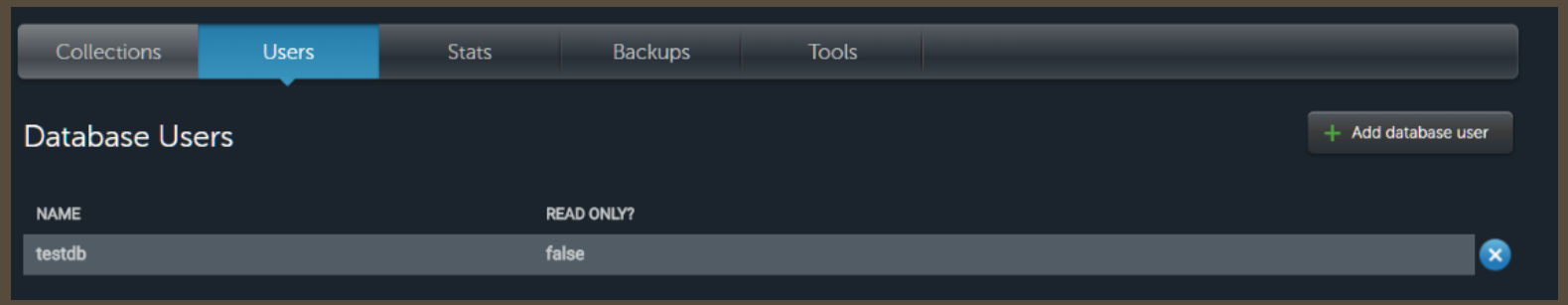
- Go to mLab site
- Login
- Click Create New 

- Select free single plan



MLAB

- Add db user



- In Server, connect to

To connect using a driver via the standard MongoDB URI (what's this?):

```
mongodb://<dbuser>:<dbpassword>@ds032319.mlab.com:32319/jce16
```

```
var connection = mongoose.createConnection('mongodb://testdb:testdb@ds032319.mlab.com:32319/jce16');//connect to the db server
```