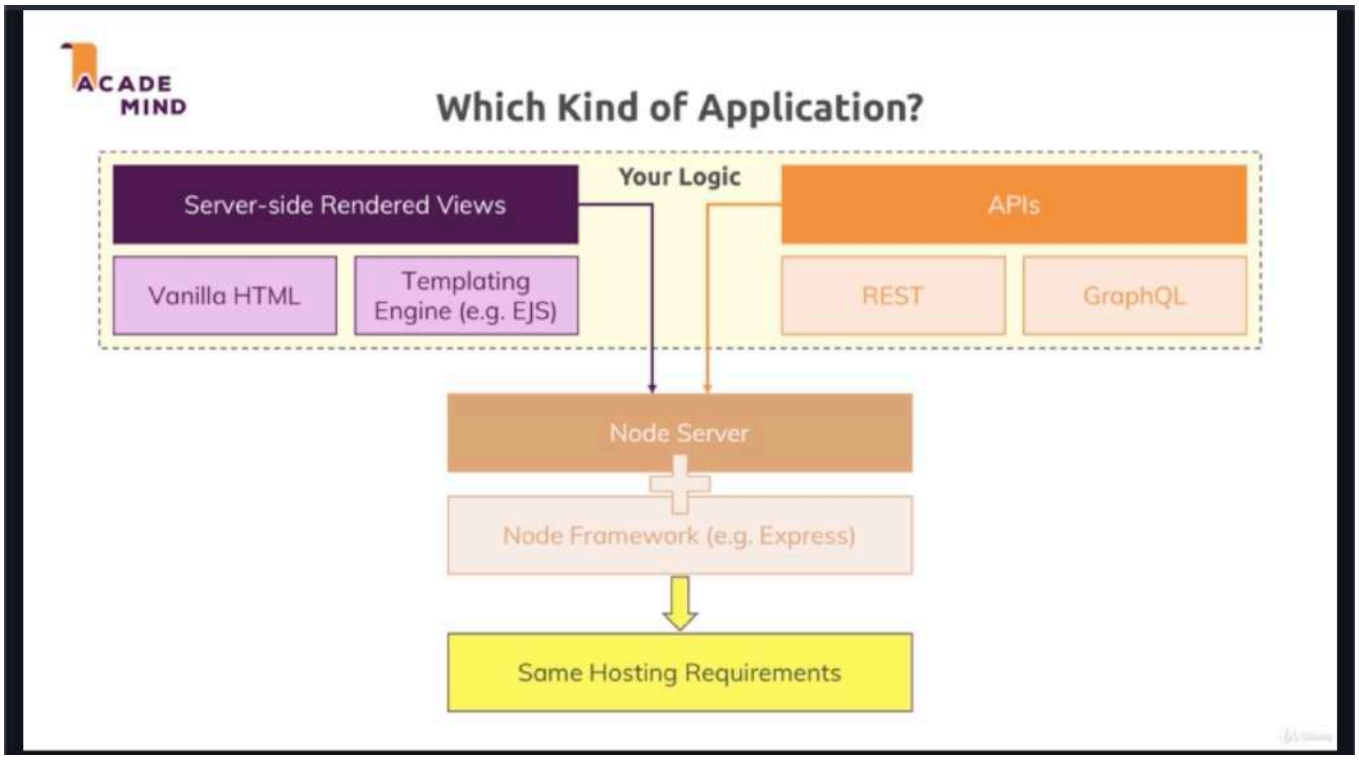
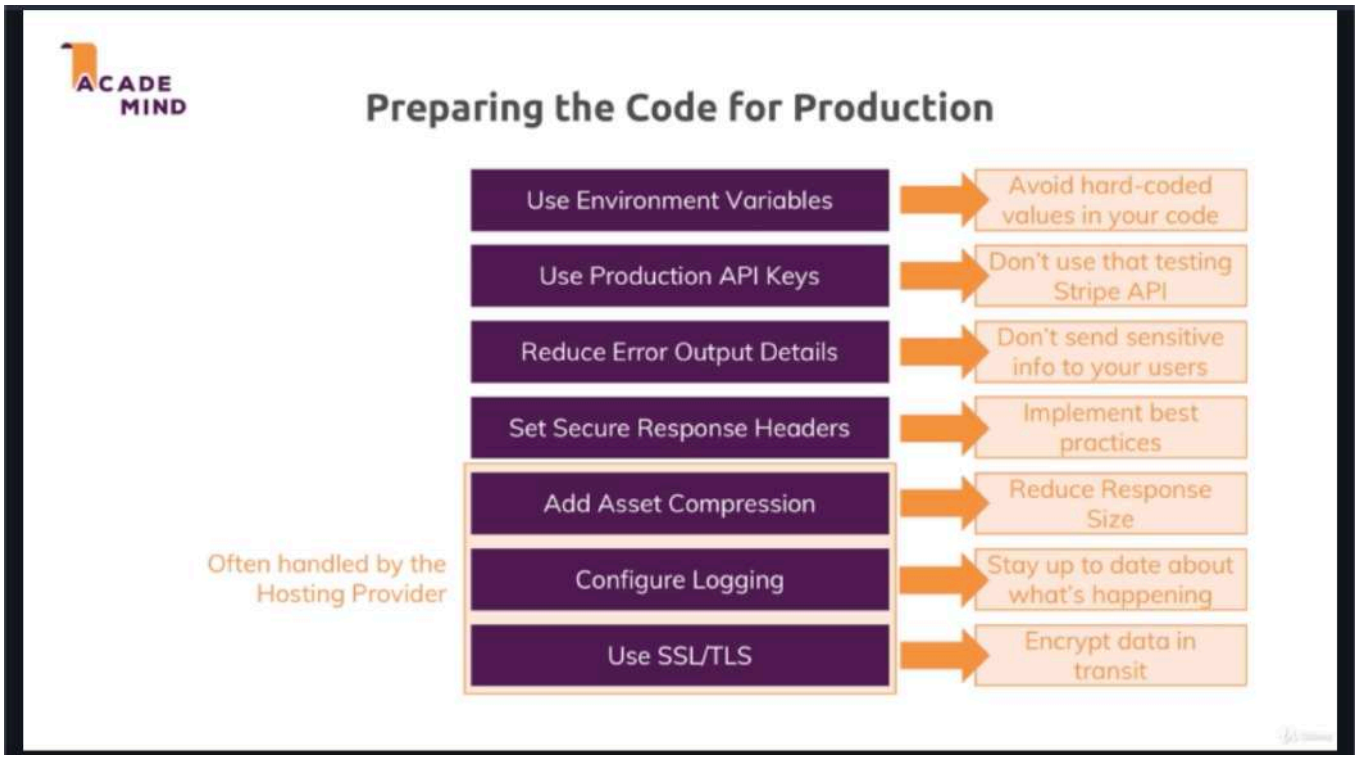
29. Deploying Our App

* Chapter 443: Module Introduction

* Chapter 444: Deploying Different Kinds Of Apps



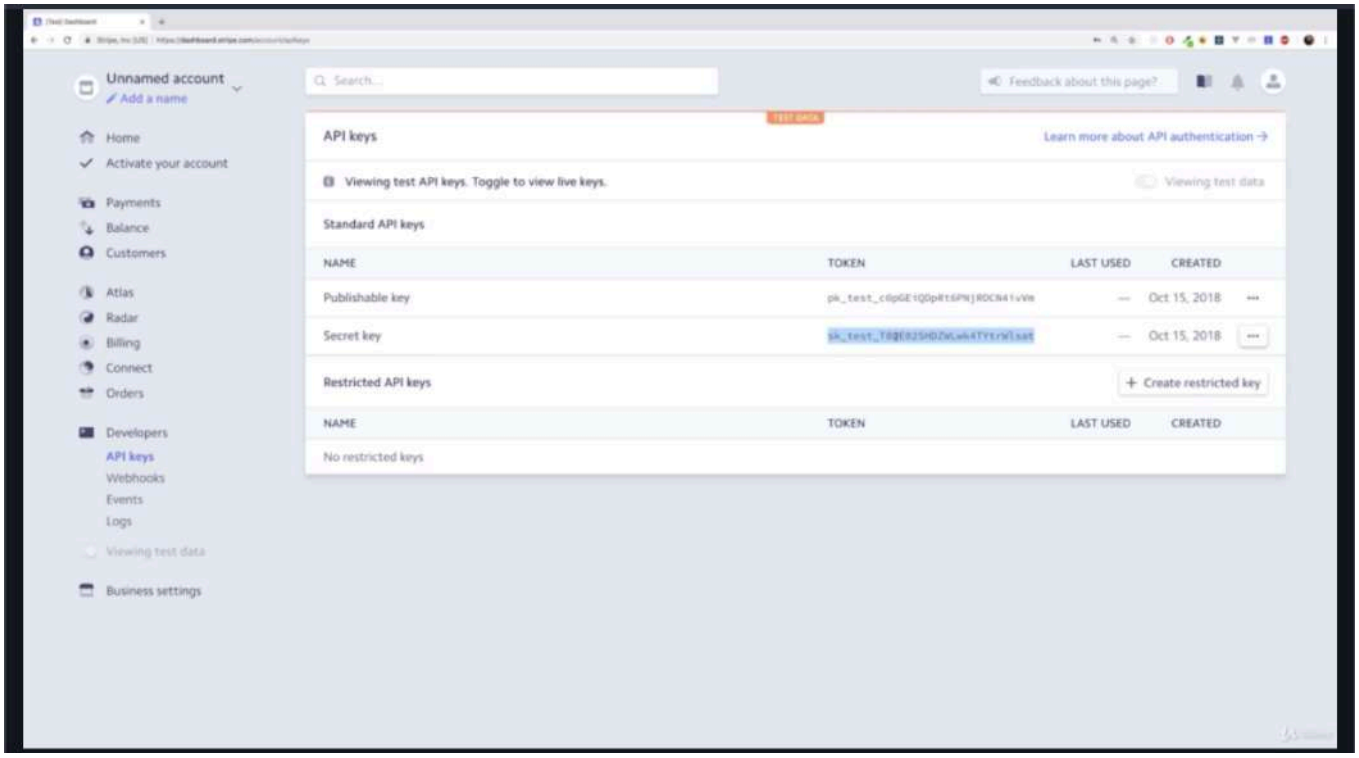
* Chapter 445: Deployment Preparations

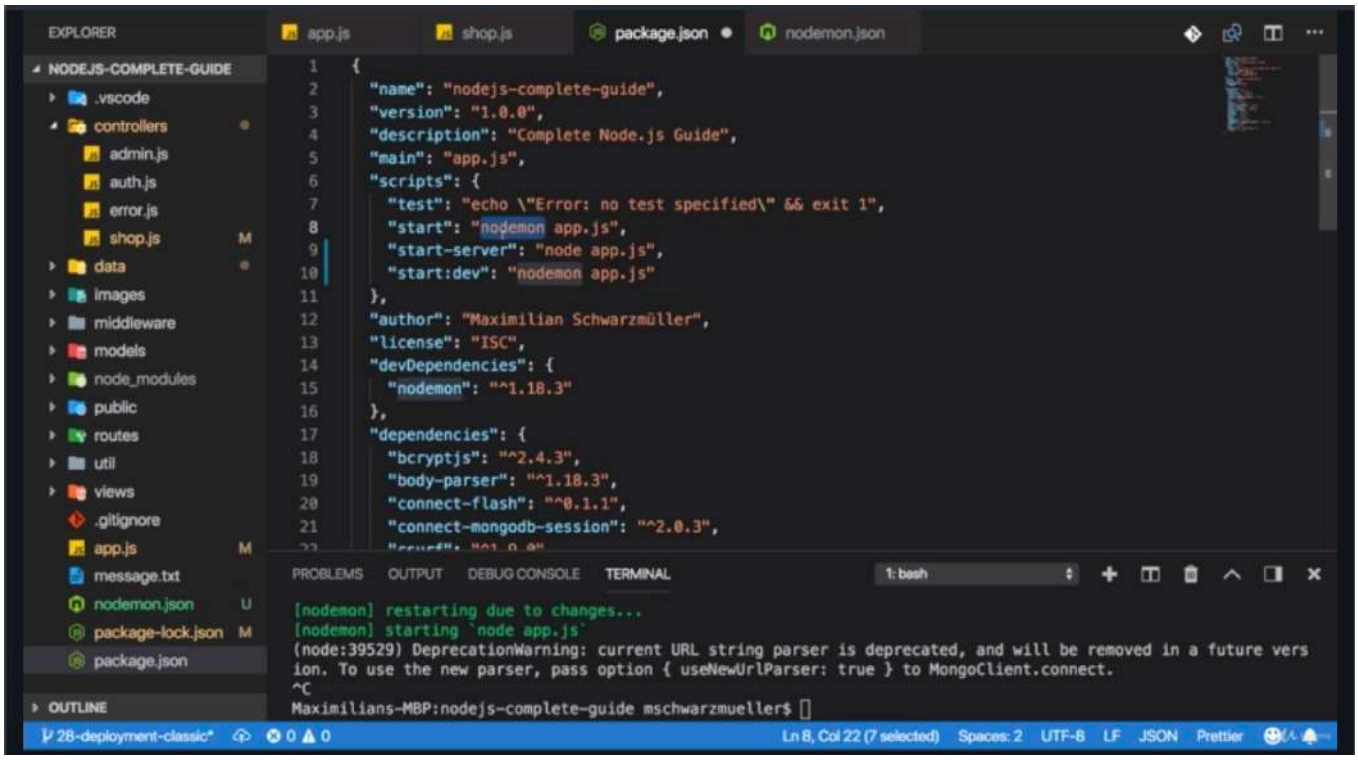


* Chapter 446: Using Environment Variables

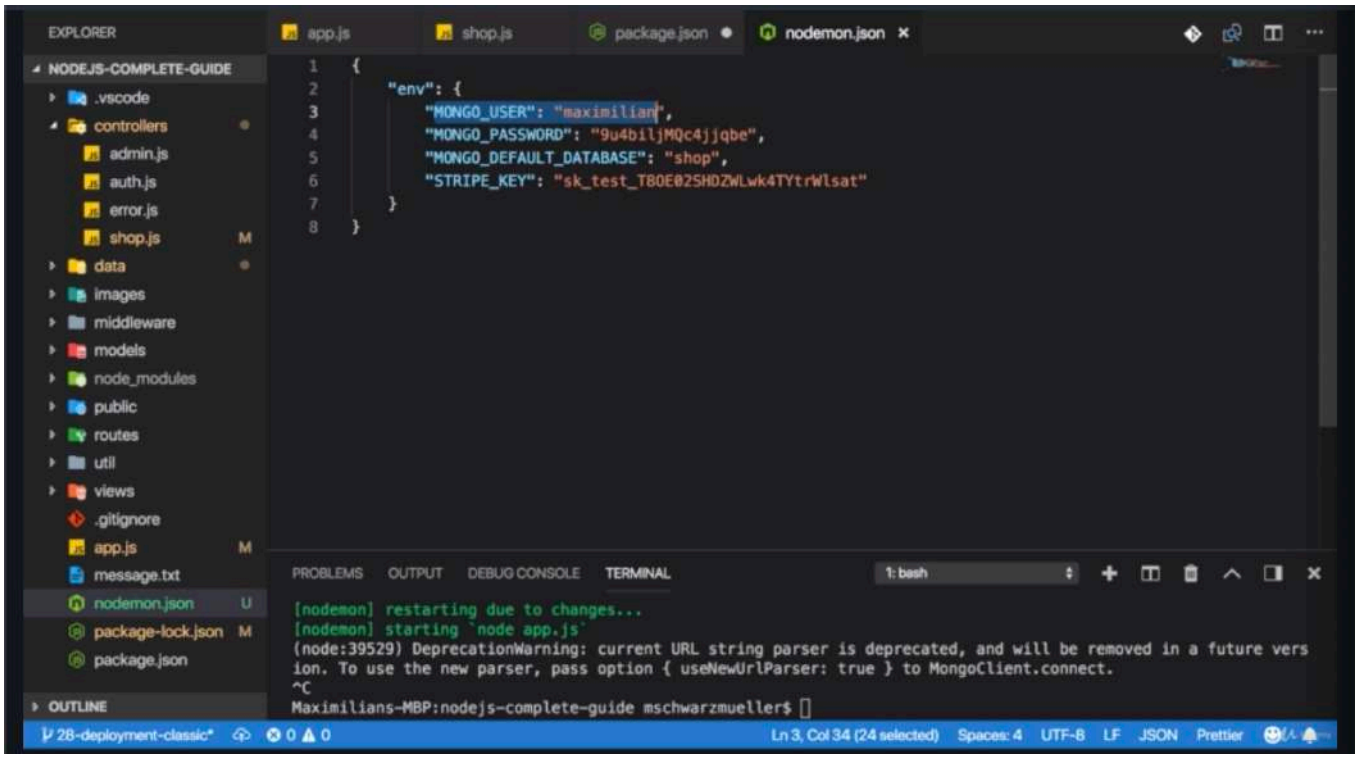
1. update
- app.js
- ./controllers/shop.js
- nodemon.js
- package.json

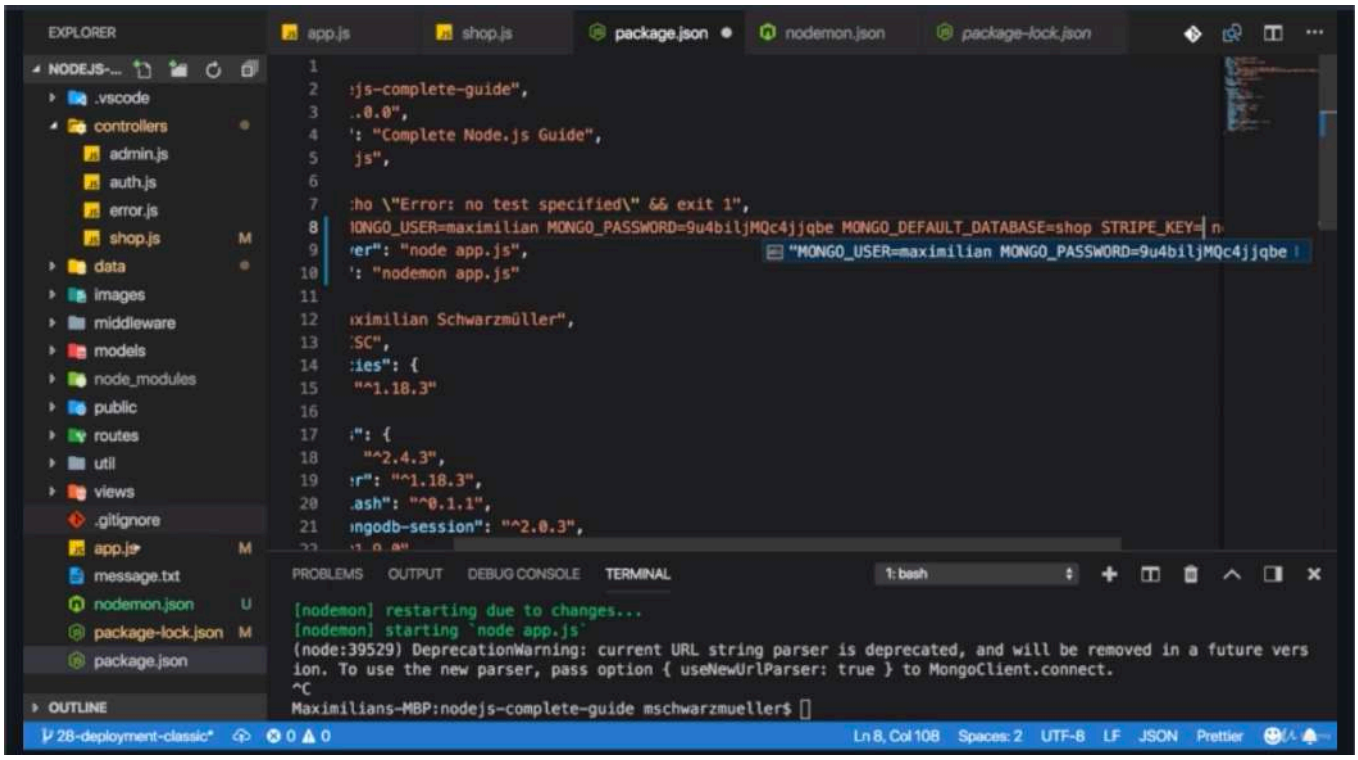

stripe

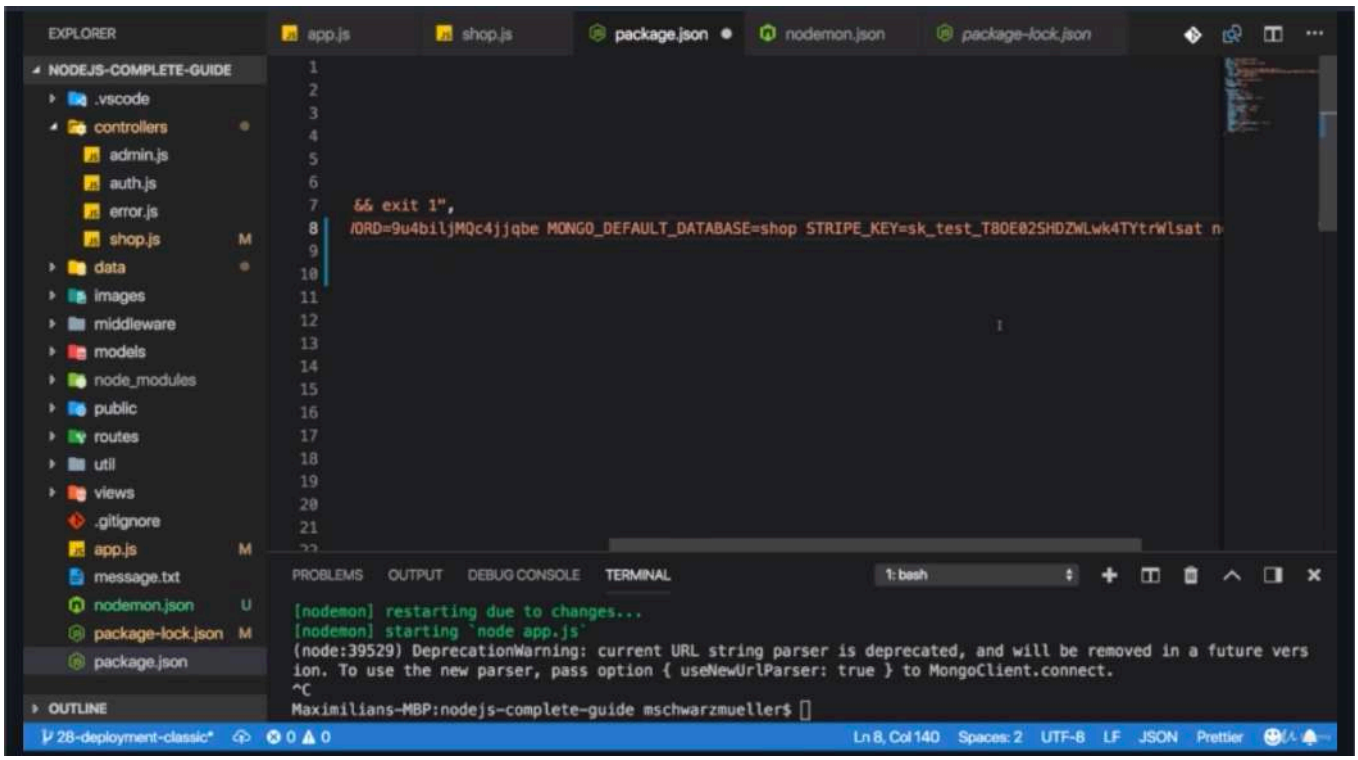

Unnamed account
Add a name
Search...
Feedback about this page?
Home
Activate your account
Payments
Balance
Customers
Atlas
Radar
Billing
Connect
Orders
Developers
API keys
Webhooks
Events
Logs
Viewing test data
Business settings
API keys
Learn more about API authentication →
Viewing test API keys. Toggle to view live keys.
Viewing test data
Standard API keys
NAME
TOKEN
LAST USED
CREATED
Publishable key
Oct 15, 2018
Secret key
Oct 15, 2018
Restricted API keys
+ Create restricted key
NAME
TOKEN
LAST USED
CREATED
No restricted keys



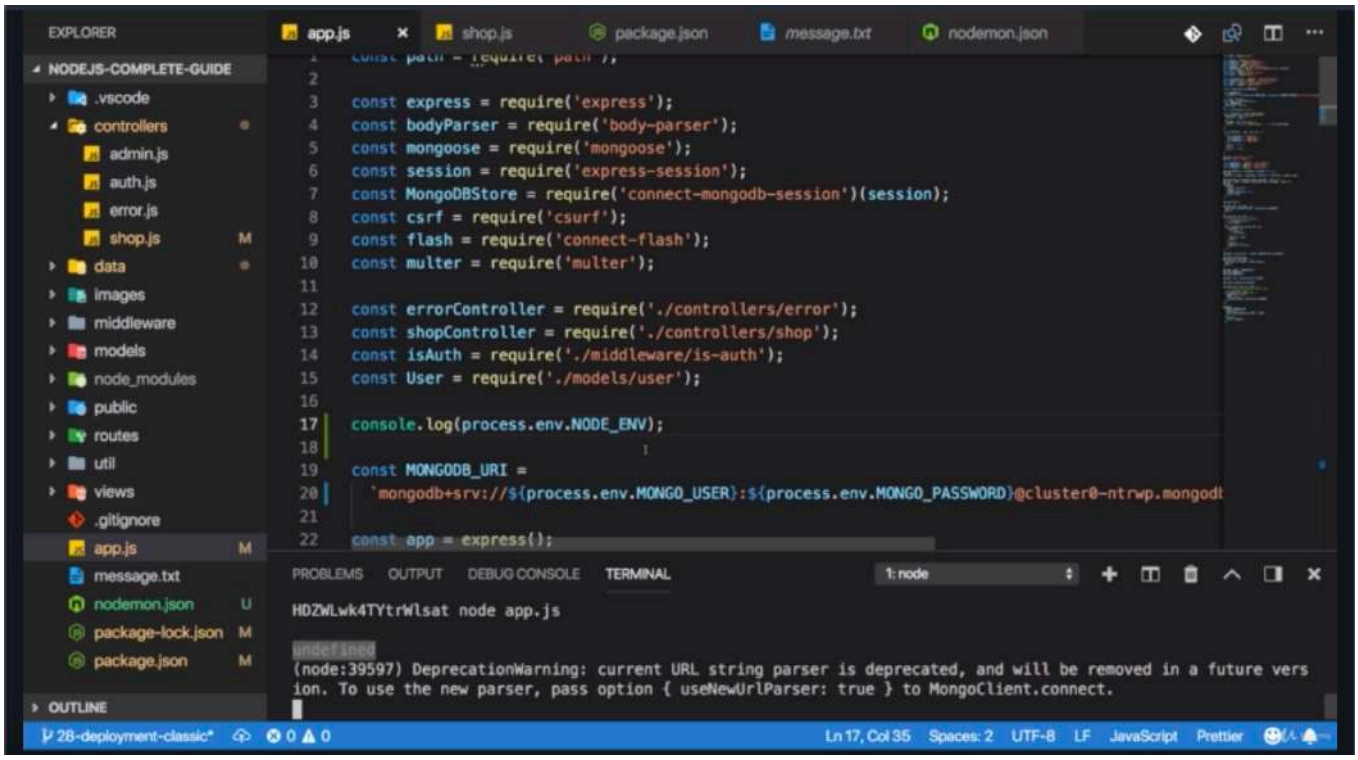
- in the “start”, this will not used at ’nodemon app.js’ but 'node app.js’ so if you wanna pass an environment variables here, you also got different solutions. and when using a hosting provider, you can set up the environment variables in the dashboard of your hosting provider.

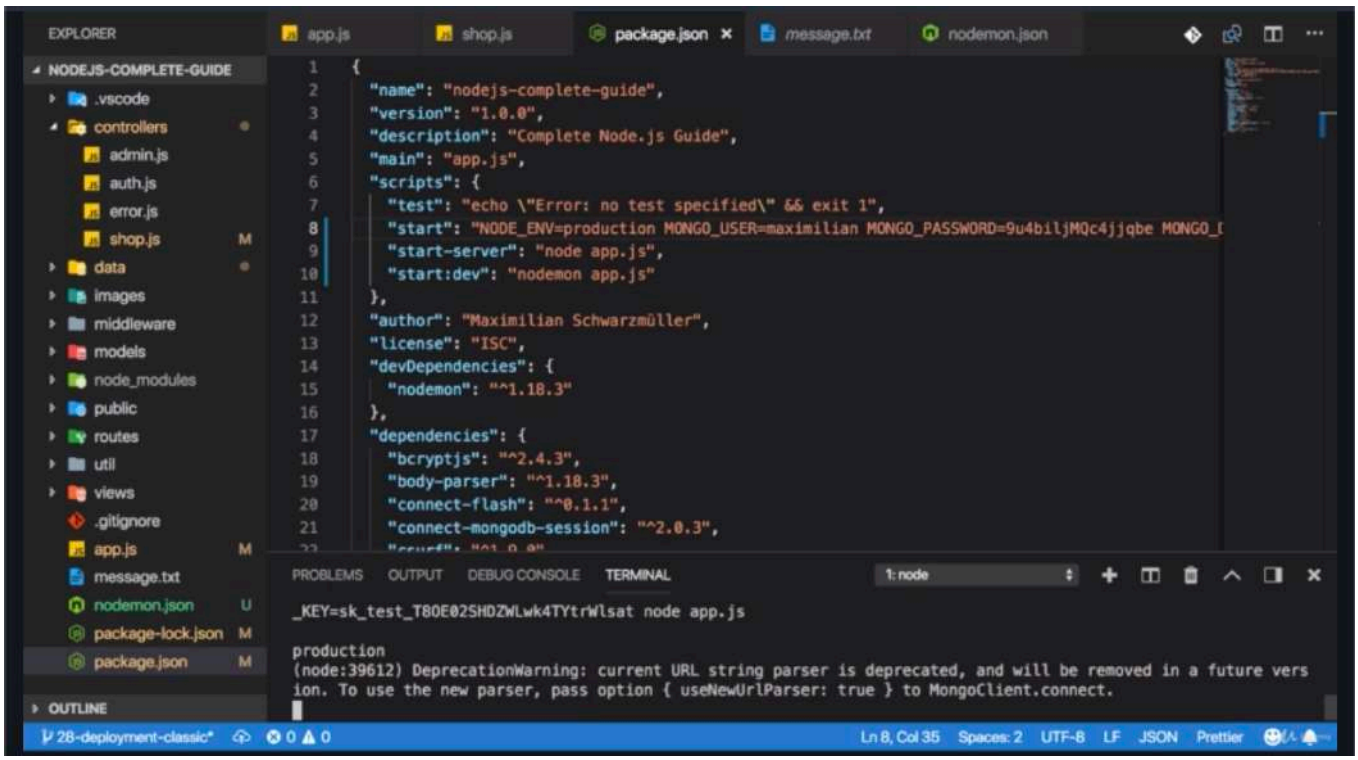






- as a simple solution, take the key value pairs you wanna set up and add them in your package.json file on the “start” script.
- we are running now this with our environment variables”MONGO_USER” “MONGO_PASSWORD” “MONGO_DEFAULT_DATABASE” “STRIPE KEY" here.





- you see ‘undefined’ here. now again this will be set automatically by hosting providers. you can also set it on your own and hosting providers will set this to production. “NODE_ENV=production” which is a special environment variables even though it’s not set by default because express.js will use this by default to determine the environment mode and if you set that to production, express.js will change certain things for example it will reduce the details for errors it frozes and general optimized some thigns for deployment.

```
//app.js

const path = require('path');

const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const session = require('express-session');
```

```
9 const MongoDBStore = require('connect-mongodb-session')(session);
10 const csrf = require('csurf');
11 const flash = require('connect-flash');
12 const multer = require('multer');
13
14 const errorController = require('./controllers/error');
15 const shopController = require('./controllers/shop');
16 const isAuth = require('./middleware/is-auth');
17 const User = require('./models/user');
18
19 /**'process' object is an object not defined by us
20  * this is globally abailable in the node app
21  * it's part of the node core runtime
22  *
23  * 'env' property is an object with all the environment variables
24  * this node process knows there are a bunch of default environment variables
25  * but we can also set our own ones.
26  */
27 const MONGODB_URI =
28   `mongodb+srv://${process.env.MONGO_USER}:${process.env.MONGO_PASSWORD}@cluster0-
z3vlk.mongodb.net/${process.env.MONGO_DEFAULT_DATABASE}`;
29
30 const app = express();
31 const store = new MongoDBStore({
32   uri: MONGODB_URI,
33   collection: 'sessions'
34 });
35 const csrfProtection = csrf();
36
37 const fileStorage = multer.diskStorage({
38   destination: (req, file, cb) => {
39     cb(null, 'images');
40   },
41   filename: (req, file, cb) => {
42     cb(null, new Date().toISOString() + '-' + file.originalname);
43   }
44 });
45
46 const fileFilter = (req, file, cb) => {
47   if (
48     file.mimetype === 'image/png' ||
49     file.mimetype === 'image/jpg' ||
50     file.mimetype === 'image/jpeg'
51   ) {
52     cb(null, true);
53   } else {
54     cb(null, false);
55   }
56 };
57
58 app.set('view engine', 'ejs');
59 app.set('views', 'views');
60
61 const adminRoutes = require('./routes/admin');
62 const shopRoutes = require('./routes/shop');
63 const authRoutes = require('./routes/auth');
```

```

app.use(bodyParser.urlencoded({ extended: false }));
app.use(
  multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
);
app.use(express.static(path.join(__dirname, 'public')));
app.use('/images', express.static(path.join(__dirname, 'images')));
app.use(
  session({
    secret: 'my secret',
    resave: false,
    saveUninitialized: false,
    store: store
  })
);

app.use(flash());

app.use((req, res, next) => {
  res.locals.isAuthenticated = req.session.isLoggedIn;
  next();
});

app.use((req, res, next) => {
  // throw new Error('Sync Dummy');
  if (!req.session.user) {
    return next();
  }
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
    .catch(err => {
      next(new Error(err));
    });
});

app.post('/create-order', isAuth, shopController.postOrder);

app.use(csrfProtection);
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});

app.use('/admin', adminRoutes);
app.use(shopRoutes);
app.use(authRoutes);

app.get('/500', errorController.get500);

app.use(errorController.get404);
```

```

app.use((error, req, res, next) => {
  // res.status(error.httpStatusCode).render(...);
  // res.redirect('/500');
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
});

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    /**most hosting providers or all those providers
     * that managed it for us
     * will automatically inject the PORT environment variable
     * so most of time we can rely on that being set
     * and for local development we will still foward back to 3000
     * because there will not be set
     */
    app.listen(process.env.PORT || 3000);
  })
  .catch(err => {
    console.log(err);
  });

```

```
//./controllers/shop.js

const fs = require('fs');
const path = require('path');

const PDFDocument = require('pdfkit');
const stripe = require('stripe')(process.env.STRIPE_KEY);

const Product = require('../models/product');
const Order = require('../models/order');

const ITEMS_PER_PAGE = 2;

exports.getProducts = (req, res, next) => {
  const page = +req.query.page || 1;
  let totalItems;

  Product.find()
    .countDocuments()
    .then(numProducts => {
      totalItems = numProducts;
      return Product.find()
        .skip((page - 1) * ITEMS_PER_PAGE)
        .limit(ITEMS_PER_PAGE);
    })
    .then(products => {
      res.render('shop/product-list', {
        prods: products,
        pageTitle: 'Products',
```

```
30        path: '/products',
31        currentPage: page,
32        hasNextPage: ITEMS_PER_PAGE * page < totalItems,
33        hasPreviousPage: page > 1,
34        nextPage: page + 1,
35        previousPage: page - 1,
36        lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
37      });
38    })
39    .catch(err => {
40      const error = new Error(err);
41      error.httpStatusCode = 500;
42      return next(error);
43    });
44 };
45
46 exports.getProduct = (req, res, next) => {
47   const prodId = req.params.productId;
48   Product.findById(prodId)
49     .then(product => {
50       res.render('shop/product-detail', {
51         product: product,
52         pageTitle: product.title,
53         path: '/products'
54       });
55     })
56     .catch(err => {
57       const error = new Error(err);
58       error.httpStatusCode = 500;
59       return next(error);
60     });
61 };
62
63 exports.getIndex = (req, res, next) => {
64   const page = +req.query.page || 1;
65   let totalItems;
66
67   Product.find()
68     .countDocuments()
69     .then(numProducts => {
70       totalItems = numProducts;
71       return Product.find()
72         .skip((page - 1) * ITEMS_PER_PAGE)
73         .limit(ITEMS_PER_PAGE);
74     })
75     .then(products => {
76       res.render('shop/index', {
77         prods: products,
78         pageTitle: 'Shop',
79         path: '/',
80         currentPage: page,
81         hasNextPage: ITEMS_PER_PAGE * page < totalItems,
82         hasPreviousPage: page > 1,
83         nextPage: page + 1,
84         previousPage: page - 1,
85         lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
```

```
    });
  })
  .catch(err => {
    const error = new Error(err);
    error.httpStatusCode = 500;
    return next(error);
  });
};

exports.getCart = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items;
      res.render('shop/cart', {
        path: '/cart',
        pageTitle: 'Your Cart',
        products: products
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postCart = (req, res, next) => {
  const prodId = req.body.productId;
  Product.findById(prodId)
    .then(product => {
      return req.user.addToCart(product);
    })
    .then(result => {
      console.log(result);
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postCartDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  req.user
    .removeFromCart(prodId)
    .then(result => {
      res.redirect('/cart');
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
```

```
    });
};

exports.getCheckout = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      const products = user.cart.items;
      let total = 0;
      products.forEach(p => {
        total += p.quantity * p.productId.price;
      });
      res.render('shop/checkout', {
        path: '/checkout',
        pageTitle: 'Checkout',
        products: products,
        totalSum: total
      });
    })
    .catch(err => {
      const error = new Error(err);
      error.httpStatusCode = 500;
      return next(error);
    });
};

exports.postOrder = (req, res, next) => {
  // Token is created using Checkout or Elements!
  // Get the payment token ID submitted by the form:
  const token = req.body.stripeToken; // Using Express
  let totalSum = 0;

  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      user.cart.items.forEach(p => {
        totalSum += p.quantity * p.productId.price;
      });

      const products = user.cart.items.map(i => {
        return { quantity: i.quantity, product: { ...i.productId._doc } };
      });
      const order = new Order({
        user: {
          email: req.user.email,
          userId: req.user
        },
        products: products
      });
      return order.save();
    })
    .then(result => {
      const charge = stripe.charges.create({
        amount: totalSum * 100,
```

```
198         currency: 'usd',
199         description: 'Demo Order',
200         source: token,
201         metadata: { order_id: result._id.toString() }
202       });
203       return req.user.clearCart();
204     })
205     .then(() => {
206       res.redirect('/orders');
207     })
208     .catch(err => {
209       const error = new Error(err);
210       error.httpStatusCode = 500;
211       return next(error);
212     });
213 };
214
215 exports.getOrders = (req, res, next) => {
216   Order.find({ 'user.userId': req.user._id })
217     .then(orders => {
218       res.render('shop/orders', {
219         path: '/orders',
220         pageTitle: 'Your Orders',
221         orders: orders
222       });
223     })
224     .catch(err => {
225       const error = new Error(err);
226       error.httpStatusCode = 500;
227       return next(error);
228     });
229 };
230
231 exports.getInvoice = (req, res, next) => {
232   const orderId = req.params.orderId;
233   Order.findById(orderId)
234     .then(order => {
235       if (!order) {
236         return next(new Error('No order found.'));
237       }
238       if (order.user.userId.toString() !== req.user._id.toString()) {
239         return next(new Error('Unauthorized'));
240       }
241       const invoiceName = 'invoice-' + orderId + '.pdf';
242       const invoicePath = path.join('data', 'invoices', invoiceName);
243
244       const pdfDoc = new PDFDocument();
245       res.setHeader('Content-Type', 'application/pdf');
246       res.setHeader(
247         'Content-Disposition',
248         'inline; filename="' + invoiceName + '"'
249       );
250       pdfDoc.pipe(fs.createWriteStream(invoicePath));
251       pdfDoc.pipe(res);
252
253       pdfDoc.fontSize(26).text('Invoice', {
```

```
      underline: true
    });
    pdfDoc.text('-----------------------');
    let totalPrice = 0;
    order.products.forEach(prod => {
      totalPrice += prod.quantity * prod.product.price;
      pdfDoc
        .fontSize(14)
        .text(
          prod.product.title +
            ' - ' +
            prod.quantity +
            ' x ' +
            '$' +
            prod.product.price
        );
    });
    pdfDoc.text('---');
    pdfDoc.fontSize(20).text('Total Price: $' + totalPrice);

    pdfDoc.end();
    // fs.readFile(invoicePath, (err, data) => {
    //   if (err) {
    //     return next(err);
    //   }
    //   res.setHeader('Content-Type', 'application/pdf');
    //   res.setHeader(
    //     'Content-Disposition',
    //     'inline; filename="' + invoiceName + '"'
    //   );
    //   res.send(data);
    // });
    // const file = fs.createReadStream(invoicePath);

    // file.pipe(res);
  })
  .catch(err => next(err));
};

```

```
//nodemon.json

{
    "env": {
        "MONGO_USER": "maximilian",
        "MONGO_PASSWORD": "rldnjs12",
        "MONGO_DEFAULT_DATABASE": "shop",
        "STRIPE_KEY": "sk_test_dAPYh6CKcipsTX13kbjOFklT00NICMWhhg"
    }
}
```

```
{
  "name": "nodejs-complete-guide",
  "version": "1.0.0",
  "description": "Complete Node.js Guide",
  "main": "app.js",
  "scripts": {
```

```
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "NODE_ENV=production MONGO_USER=maximilian MONGO_PASSWORD=rldnjs12 MONGO_DEFAULT_DATABASE=shop STRIPE_KEY=sk_test_dAPYh6CKcipsTX13kbjOFklT00NICMWhhg node app.js",
    "start-server": "node app.js",
    "start:dev": "nodemon app.js"
  },
  "author": "Maximilian Schwarzmüller",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^1.18.3"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.18.3",
    "connect-flash": "^0.1.1",
    "connect-mongodb-session": "^2.0.3",
    "csurf": "^1.9.0",
    "ejs": "^2.6.1",
    "express": "^4.16.3",
    "express-handlebars": "^3.0.0",
    "express-session": "^1.15.6",
    "express-validator": "^5.3.0",
    "lodash": "^4.17.11",
    "mongodb": "^3.1.6",
    "mongoose": "^5.2.17",
    "multer": "^1.4.0",
    "mysql2": "^1.6.1",
    "nodemailer": "^4.6.8",
    "nodemailer-sendgrid-transport": "^0.2.0",
    "pdfkit": "^0.8.3",
    "pug": "^2.0.3",
    "sequelize": "^5.0.0-beta.11",
    "stripe": "^6.12.1"
  }
}

```

* Chapter 447: Using Production API Keys

Unnamed account
Add a name

Home
Activate your account

Payments
Balance
Customers

Atlas
Radar
Billing
Connect
Orders

Developers
API keys
Webhooks
Events
Logs
Viewing test data

Business settings

Search...

Feedback about this page?

TEST DATA

API keys

Learn more ab

Please activate your account to access live data

Viewing test API keys. Toggle to view live keys.

Viewing test data

Standard API keys

NAME	TOKEN	LAST USED	CREATED
Publishable key	pk_test_c0pGE1Q0pRt6PNjROCN41vVm	—	Oct 15, 2018
Secret key	sk_test_T8OE02SHDZWLwk4TYtrWlsat	—	Oct 15, 2018

Restricted API keys

+ Create restricted key

NAME	TOKEN	LAST USED	CREATED

No restricted keys

Unnamed account
Add a name

Home
Activate your account

Payments
Balance
Customers

Atlas
Radar
Billing
Connect
Orders

Developers
Viewing test data

Business settings

Search...

Feedback about this page?

Telefon

DE +49 1234 567890

Bankverbindung

Wir akzeptieren Bankkonten in den folgenden Währungen: GBP, EUR, USD, SEK, NOK, DKK and CHF. Währungen, für die kein Bankkonto bei Stripe hinterlegt ist, werden umgerechnet und auf Ihr Hauptkonto überwiesen. Bankkonten in weiteren Währungen können Sie jederzeit später hinzufügen. Mehr erfahren →

Währung

EUR

Land des Bankkontos

Germany

IBAN

DE89370400440532013000

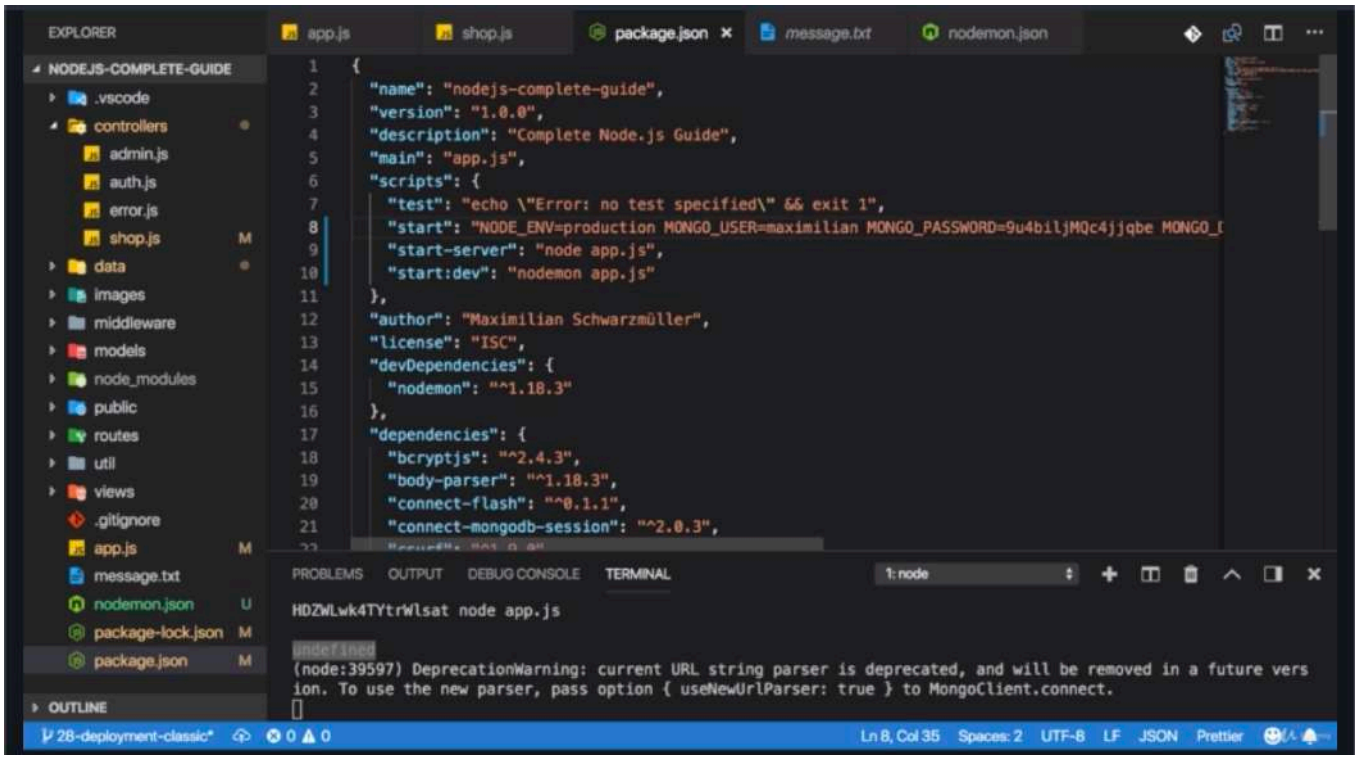
IBAN erneut eingeben

DE89370400440532013000

By activating your account, you agree to our Services Agreement.

Activate account

Save for later



- if you wanna deploy this into production, you need to fill it all out and once you filled it out, you will be able to switch that toggle here at the top and see your life data your production ready keys and you should use these keys in your code. so wherever you are using the test key right now, you would have to replace them with the production ready keys and data is just stripe example. other API might use similar behavior.

* Chapter 448: Setting Secure Response Headers With Helmet

1. update
- app.js

Google

node helmet

Sign in

All Shopping Videos Images News More Settings Tools

About 5,510,000 results (0,53 seconds)

GitHub - helmetjs/helmet: Help secure Express apps with various ...
https://github.com/helmetjs/helmet
Helmet. npm version npm dependency status Build Status FOSSA Status. Helmet helps you secure your Express apps by setting various HTTP headers. It's not a ...
Helmetjs · README.md · Looking for a changelog?

Helmet
https://helmetjs.github.io/
Helmet helps you secure your Express apps by setting various HTTP headers. It's not a silver bullet, but it can help! Quick start. First, run npm install helmet ...
HSTS · Frameguard · noCache · Content Security Policy

Security Best Practices for Express in Production - Express.js
https://expressjs.com/en/advanced/best-practice-security.html
Install Helmet like any other module: $ npm install --save helmet. Then to use it in your code: // ... var helmet = require('helmet') app.use(helmet()) // .
Use TLS · Use Helmet · Use cookies securely

Putting the helmet on – Securing your Express app - Twilio
https://www.twilio.com/blog/2017/11/securing-your-express-app.html
Nov 21, 2017 - Learn how you can improve the security of your Node.js & Express web application with ... Now add the helmet middleware to your application.

Fasten your Helmet.js (Part 1): Securing your Express HTTP headers ...
https://www.ca.com/.../fasten-your-helmet-js-part-1-securing-your-express-http-heade...
Jul 24, 2018 - Helmet.js is a useful Node.js module that helps you secure HTTP headers returned by your Express apps. HTTP headers are an important part ...

Protect Your Node App's Noggin With Helmet - Scott Smith

HELMET

Express.js security with HTTP headers. Latest version: **3.14.0**

Docs Contributors See also npm GitHub

Helmet helps you secure your Express apps by setting various HTTP headers. *It's not a silver bullet*, but it can help!

Quick start

First, run `npm install helmet --save` for your app. Then, in an Express app:

```
const express = require('express')
const helmet = require('helmet')

const app = express()

app.use(helmet())

// ...
```

That's it! Helmet will set various HTTP headers to help protect your app.

How it works

EXPLORER

NODEJS-COMPLETE-GUIDE

- .vscode
- controllers
  - admin.js
  - auth.js
  - error.js
  - shop.js
- data
- images
- middleware
- models
- node_modules
- public
- routes
- util
- views
- .gitignore
- app.js
- message.txt
- nodemon.json
- package-lock.json
- package.json

OUTLINE

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm install --save helmet
[                    ] ⸨ rollbackFailedOptional: verb npm-session 098f0aab6d04b4f9
```

28-deployment-classic 0 0 Prettier

Quick start

First, run `npm install helmet --save` for your app. Then, in an Express app:

```
const express = require('express')
const helmet = require('helmet')

const app = express()

app.use(helmet())

// ...
```

That's it! Helmet will set various HTTP headers to help protect your app.

How it works

Helmet is a collection of 14 smaller middleware functions that set HTTP headers. Running `app.use(helmet())` will not include all of these middleware functions by default.

You can see more in the documentation.

Module	Default?
contentSecurityPolicy for setting Content Security Policy	
crossdomain for handling Adobe products' crossdomain requests	

```
  ) {
    cb(null, true);
  } else {
    cb(null, false);
  }
};

app.set('view engine', 'ejs');
app.set('views', 'views');

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');
const authRoutes = require('./routes/auth');

app.use(helmet());

app.use(bodyParser.urlencoded({ extended: false }));
app.use(
  multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
);
app.use(express.static(path.join(__dirname, 'public')));
```

```
+ helmet@3.14.0
added 19 packages from 9 contributors and audited 2376 packages in 6.093s
found 1 low severity vulnerability
  run `npm audit fix` to fix them, or `npm audit` for details
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm run start:dev
```

- i can see that on this response. i have a couple of headers being set and there are some special headers which were added by helmet and you can always check that by temporarily commenting out helmet

EXPLORER

app.js

NODEJS-COMPLETE-GUIDE

- .vscode
- controllers
  - admin.js
  - auth.js
  - error.js
  - shop.js
- data
- images
- middleware
- models
- node_modules
- public
- routes
- util
- views
- .gitignore
- app.js M
- message.txt
- nodemon.json
- package-lock.json M
- package.json M

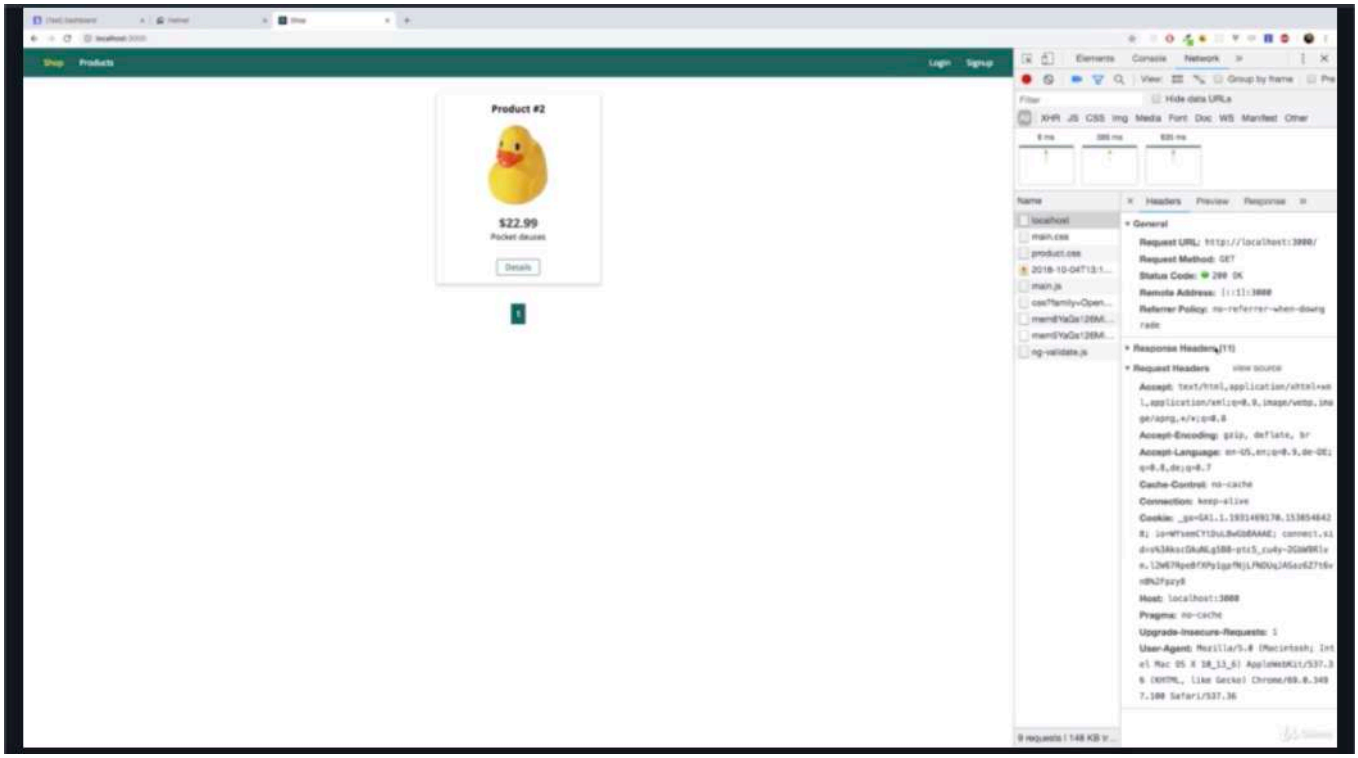
OUTLINE

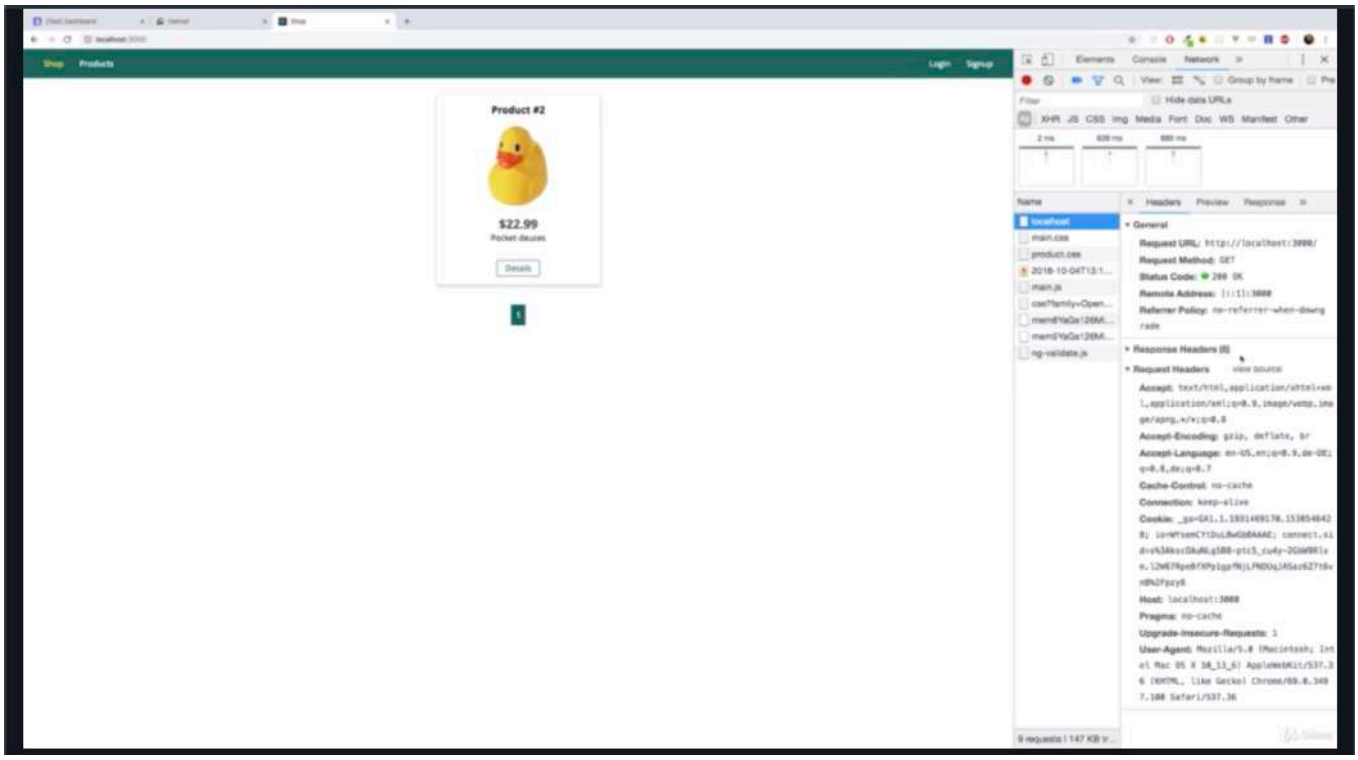
```
    file.mimetype === 'image/jpeg'
  ) {
    cb(null, true);
  } else {
    cb(null, false);
  }
};

app.set('view engine', 'ejs');
app.set('views', 'views');

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');
const authRoutes = require('./routes/auth');

// app.use(helmet());

app.use(bodyParser.urlencoded({ extended: false }));
app.use(
  multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
);
```

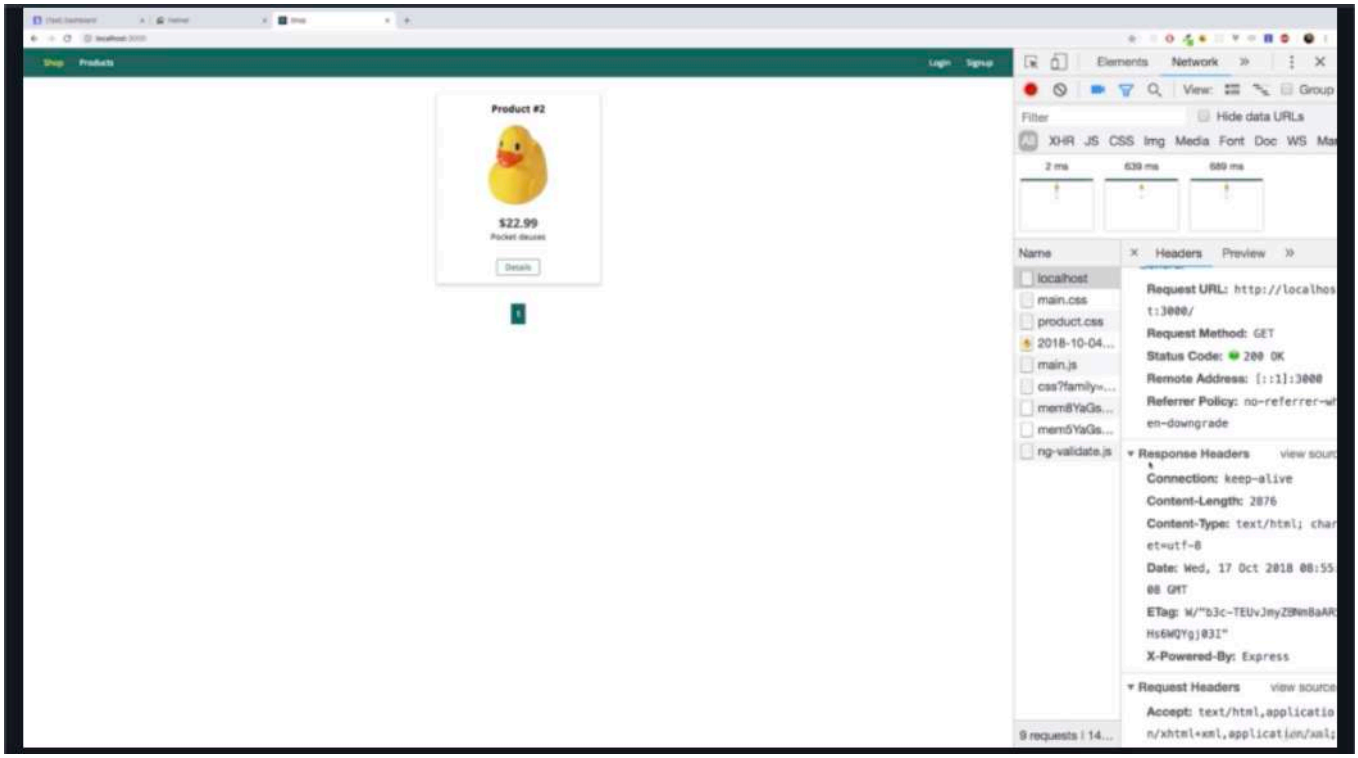
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node

```
[nodemon] starting `node app.js`
(node:39826) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future vers
ion. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```

28-deployment-classic* 0 0 Ln 56, Col 22 Spaces: 2 UTF-8 LF JavaScript Prettier







- we had 11 response headers before. now i reload this, we just have 6 because now some special headers are missing and that is something you should consider doing.

```
//app.js

const path = require('path');

const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const session = require('express-session');
const MongoDBStore = require('connect-mongodb-session')(session);
const csrf = require('csurf');
const flash = require('connect-flash');
const multer = require('multer');
```

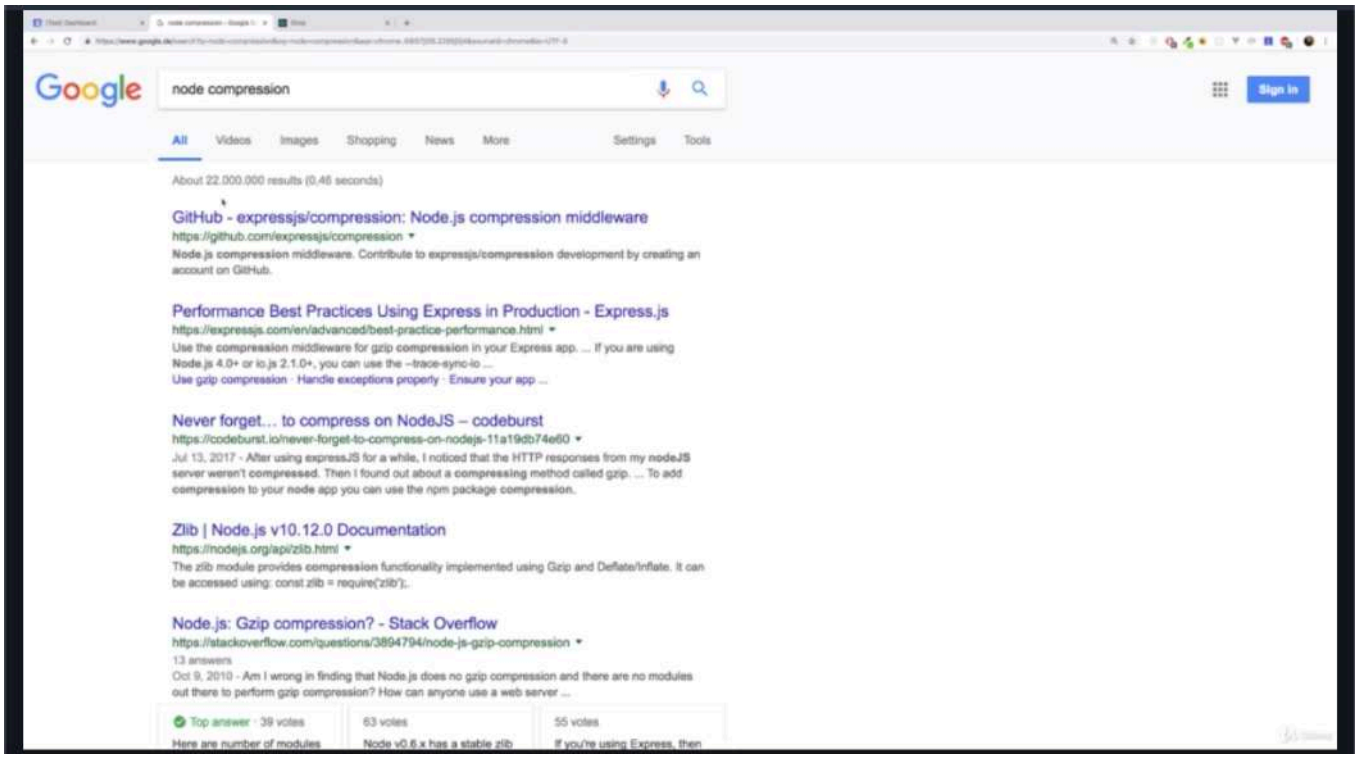
```
13 const helmet = require('helmet');
14
15 const errorController = require('./controllers/error');
16 const shopController = require('./controllers/shop');
17 const isAuth = require('./middleware/is-auth');
18 const User = require('./models/user');
19
20 const MONGODB_URI =
21   `mongodb+srv://${process.env.MONGO_USER}:${process.env.MONGO_PASSWORD}@cluster0-
z3vlk.mongodb.net/${process.env.MONGO_DEFAULT_DATABASE}`;
22
23 const app = express();
24 const store = new MongoDBStore({
25   uri: MONGODB_URI,
26   collection: 'sessions'
27 });
28 const csrfProtection = csrf();
29
30 const fileStorage = multer.diskStorage({
31   destination: (req, file, cb) => {
32     cb(null, 'images');
33   },
34   filename: (req, file, cb) => {
35     cb(null, new Date().toISOString() + '-' + file.originalname);
36   }
37 });
38
39 const fileFilter = (req, file, cb) => {
40   if (
41     file.mimetype === 'image/png' ||
42     file.mimetype === 'image/jpg' ||
43     file.mimetype === 'image/jpeg'
44   ) {
45     cb(null, true);
46   } else {
47     cb(null, false);
48   }
49 };
50
51 app.set('view engine', 'ejs');
52 app.set('views', 'views');
53
54 const adminRoutes = require('./routes/admin');
55 const shopRoutes = require('./routes/shop');
56 const authRoutes = require('./routes/auth');
57
58 app.use(helmet());
59
60 app.use(bodyParser.urlencoded({ extended: false }));
61 app.use(
62   multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
63 );
64 app.use(express.static(path.join(__dirname, 'public')));
65 app.use('/images', express.static(path.join(__dirname, 'images')));
66 app.use(
67   session({
```

```
    secret: 'my secret',
    resave: false,
    saveUninitialized: false,
    store: store
  })
);

app.use(flash());

app.use((req, res, next) => {
  res.locals.isAuthenticated = req.session.isLoggedIn;
  next();
});

app.use((req, res, next) => {
  // throw new Error('Sync Dummy');
  if (!req.session.user) {
    return next();
  }
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
    .catch(err => {
      next(new Error(err));
    });
});

app.post('/create-order', isAuth, shopController.postOrder);

app.use(csrfProtection);
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});

app.use('/admin', adminRoutes);
app.use(shopRoutes);
app.use(authRoutes);

app.get('/500', errorController.get500);

app.use(errorController.get404);

app.use((error, req, res, next) => {
  // res.status(error.httpStatusCode).render(...);
  // res.redirect('/500');
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
```

```
});

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    app.listen(process.env.PORT || 3000);
  })
  .catch(err => {
    console.log(err);
  });

```

* Chapter 449: Compressing Assets

1. update
- app.js



Search or jump to... Pull requests Issues Marketplace Explore

expressjs / compression

Watch 48 Star 1,588 Fork 126

Code Issues 11 Pull requests 10 Projects 0 Insights

Node.js compression middleware

compression javascript nodejs expressjs middleware gzip deflate

299 commits 2 branches 35 releases 12 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

dougwilson 1.7.3 Latest commit beccc1c on 15 Jul

test	tests: use after module for flow control	7 months ago
.eslintignore	lint: use standard style	3 years ago
.eslintrc.yml	build: use yaml eslint configuration	8 months ago
.gitignore	build: add test coverage	4 years ago
.travis.yml	build: Node.js@9.11	6 months ago
HISTORY.md	1.7.3	3 months ago
LICENSE	docs: update license	3 years ago
README.md	docs: fix threshold documentation to match implementation	3 months ago
index.js	Use safe-buffer for improved Buffer API	a year ago
package.json	1.7.3	3 months ago

README.md

The following compression codings are supported:

- deflate
- gzip

Install

This is a Node.js module available through the npm registry. Installation is done using the `npm install` command:

```
$ npm install compression
```

API

```
var compression = require('compression')
```

compression([options])

Returns the compression middleware using the given `options`. The middleware will attempt to compress response bodies for all request that traverse through the middleware, based on the given `options`.

This middleware will never compress responses that include a `Cache-Control` header with the `no-transform` directive, as compressing will transform the body.

Options

`compression()` accepts these properties in the options object. In addition to those listed below, zlib options may be passed in to the options object.

chunkSize

The default value is `zlib.Z_DEFAULT_CHUNK`, or `16384`.

ipardlon the usage

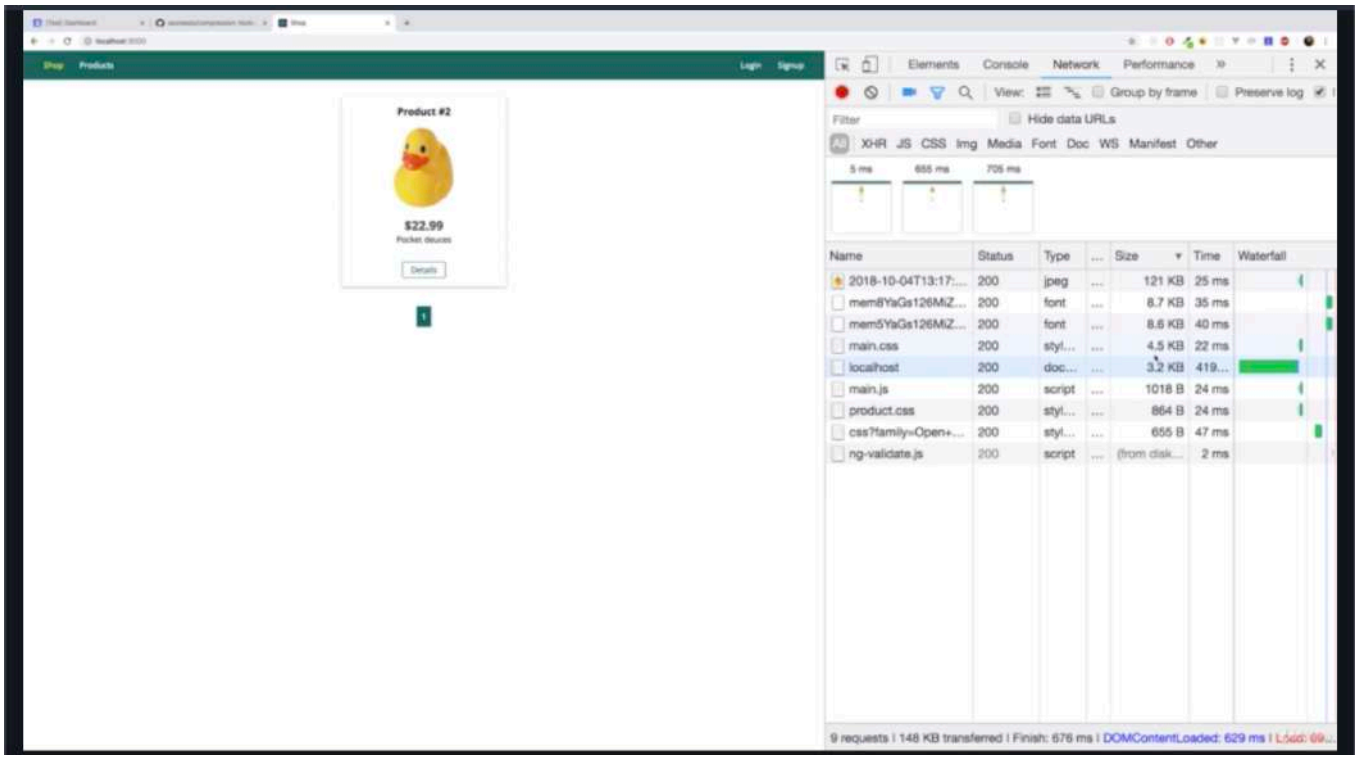
```
    file.mimetype === 'image/jpeg'
  ) {
    cb(null, true);
  } else {
    cb(null, false);
  }
};

app.set('view engine', 'ejs');
app.set('views', 'views');

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');
const authRoutes = require('./routes/auth');

app.use(helmet());

app.use(bodyParser.urlencoded({ extended: false }));
app.use(
  multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
);
```

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm install --save compression
(                    ) ⸨ rollbackFailedOptional: verb npm-session 15bfa9d85682e708
```

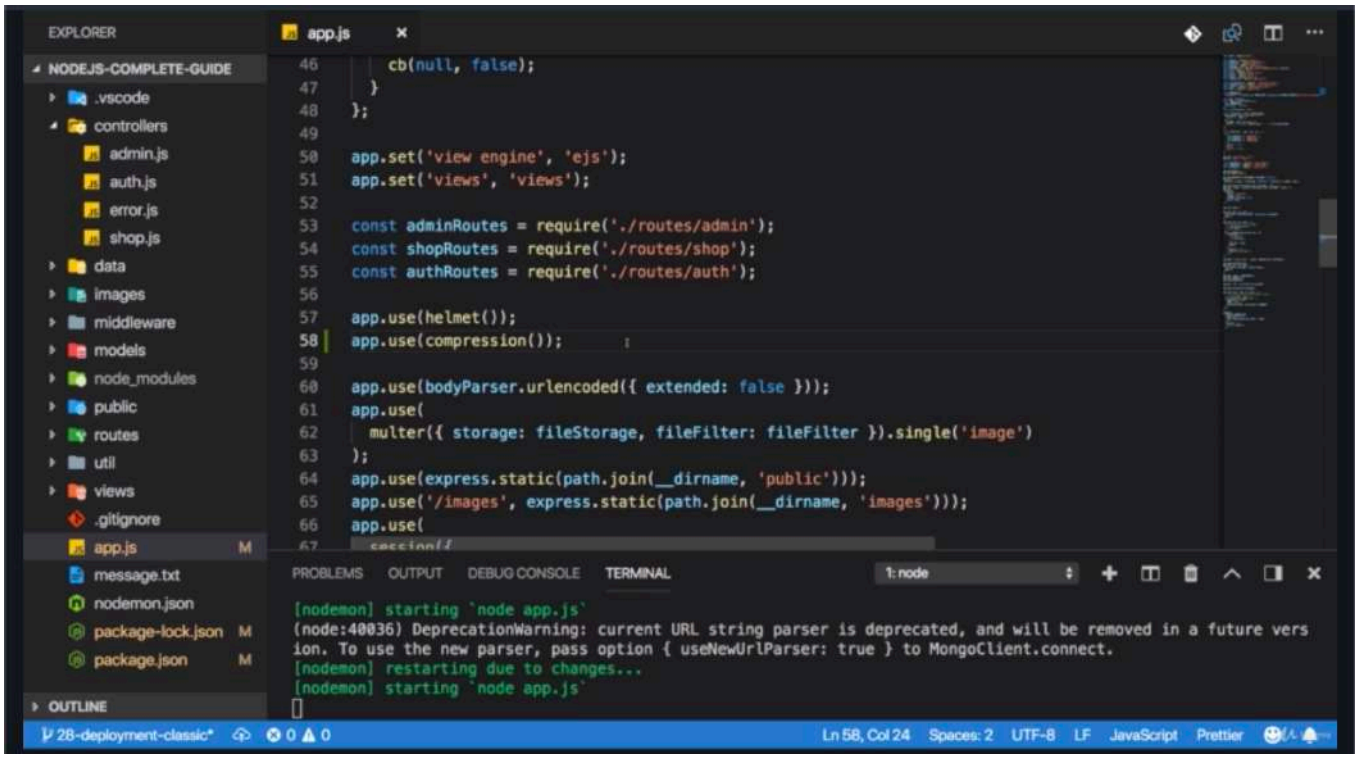
- now with response headers added, let's make sure we serve optimized assets annd for that we can use another package called 'node compression'

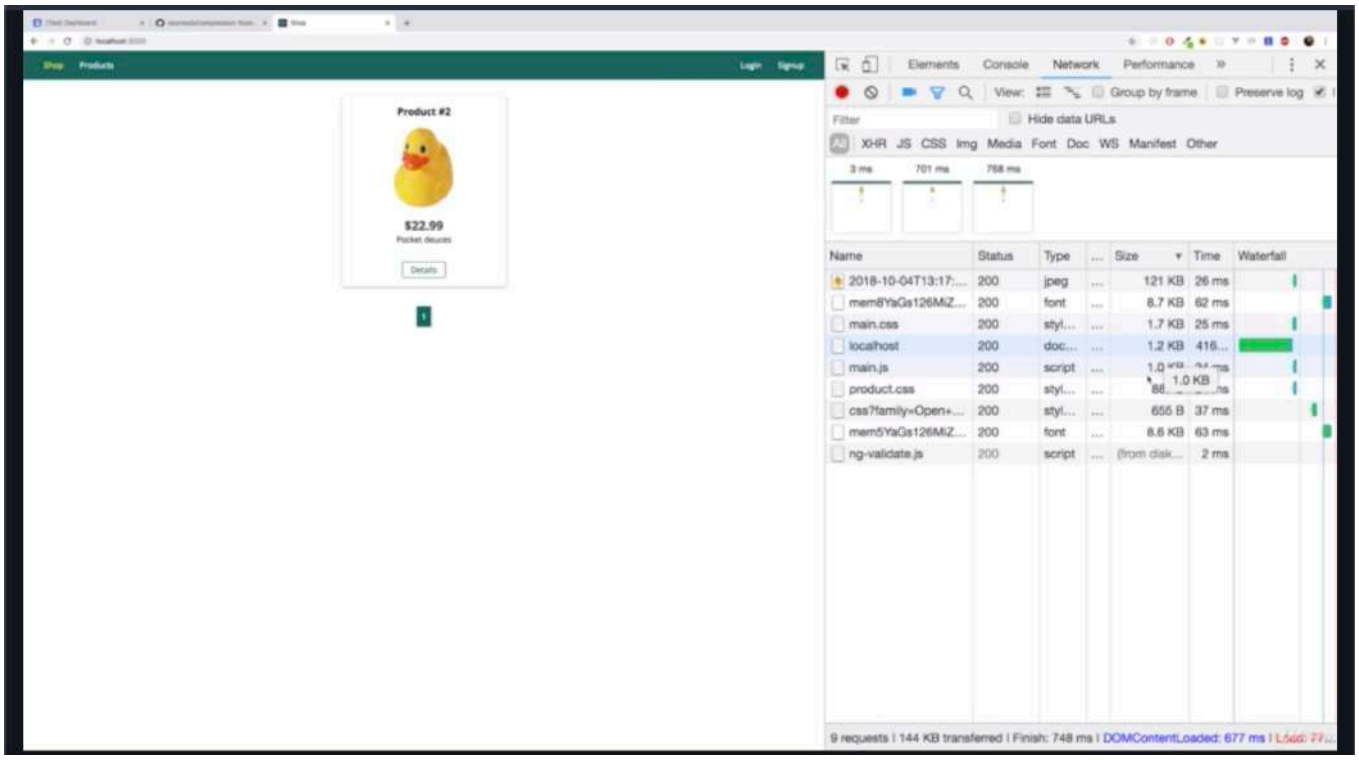

```
    cb(null, false);
  }
};

app.set('view engine', 'ejs');
app.set('views', 'views');

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');
const authRoutes = require('./routes/auth');

app.use(helmet());
// app.use(compression());

app.use(bodyParser.urlencoded({ extended: false }));
app.use(
  multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
);
app.use(express.static(path.join(__dirname, 'public')));
app.use('/images', express.static(path.join(__dirname, 'images')));
app.use(
```

```
ion. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
(node:40014) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future vers
ion. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
```



- after commenting out, these are the size of the assets we are downloading especially have a look at main.css and main.js. obviously these are not super big but still this is the size of the asset as we downloading them by default.





- after commenting, these got a bit smaller and this will matter more if you have more content assets in your application that you need to search.

```
//app.js

const path = require('path');

const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const session = require('express-session');
const MongoDBStore = require('connect-mongodb-session')(session);
const csrf = require('csurf');
const flash = require('connect-flash');
const multer = require('multer');
const helmet = require('helmet');
const compression = require('compression');

const errorController = require('./controllers/error');
const shopController = require('./controllers/shop');
const isAuth = require('./middleware/is-auth');
const User = require('./models/user');

const MONGODB_URI =
  `mongodb+srv://${process.env.MONGO_USER}:${process.env.MONGO_PASSWORD}@cluster0-z3vlk.mongodb.net/${process.env.MONGO_DEFAULT_DATABASE}`;

const app = express();
const store = new MongoDBStore({
  uri: MONGODB_URI,
  collection: 'sessions'
});
const csrfProtection = csrf();

const fileStorage = multer.diskStorage({
  destination: (req, file, cb) => {
```

```
33      cb(null, 'images');
34    },
35    filename: (req, file, cb) => {
36      cb(null, new Date().toISOString() + '-' + file.originalname);
37    }
38  });
39
40  const fileFilter = (req, file, cb) => {
41    if (
42      file.mimetype === 'image/png' ||
43      file.mimetype === 'image/jpg' ||
44      file.mimetype === 'image/jpeg'
45    ) {
46      cb(null, true);
47    } else {
48      cb(null, false);
49    }
50  };
51
52  app.set('view engine', 'ejs');
53  app.set('views', 'views');
54
55  const adminRoutes = require('./routes/admin');
56  const shopRoutes = require('./routes/shop');
57  const authRoutes = require('./routes/auth');
58
59  app.use(helmet());
60  app.use(compression());
61
62  app.use(bodyParser.urlencoded({ extended: false }));
63  app.use(
64    multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
65  );
66  app.use(express.static(path.join(__dirname, 'public')));
67  app.use('/images', express.static(path.join(__dirname, 'images')));
68  app.use(
69    session({
70      secret: 'my secret',
71      resave: false,
72      saveUninitialized: false,
73      store: store
74    })
75  );
76
77  app.use(flash());
78
79  app.use((req, res, next) => {
80    res.locals.isAuthenticated = req.session.isLoggedIn;
81    next();
82  });
83
84  app.use((req, res, next) => {
85    // throw new Error('Sync Dummy');
86    if (!req.session.user) {
87      return next();
88    }
```

```
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
    .catch(err => {
      next(new Error(err));
    });
});

app.post('/create-order', isAuth, shopController.postOrder);

app.use(csrfProtection);
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});

app.use('/admin', adminRoutes);
app.use(shopRoutes);
app.use(authRoutes);

app.get('/500', errorController.get500);

app.use(errorController.get404);

app.use((error, req, res, next) => {
  // res.status(error.httpStatusCode).render(...);
  // res.redirect('/500');
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
});

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    app.listen(process.env.PORT || 3000);
  })
  .catch(err => {
    console.log(err);
  });

```

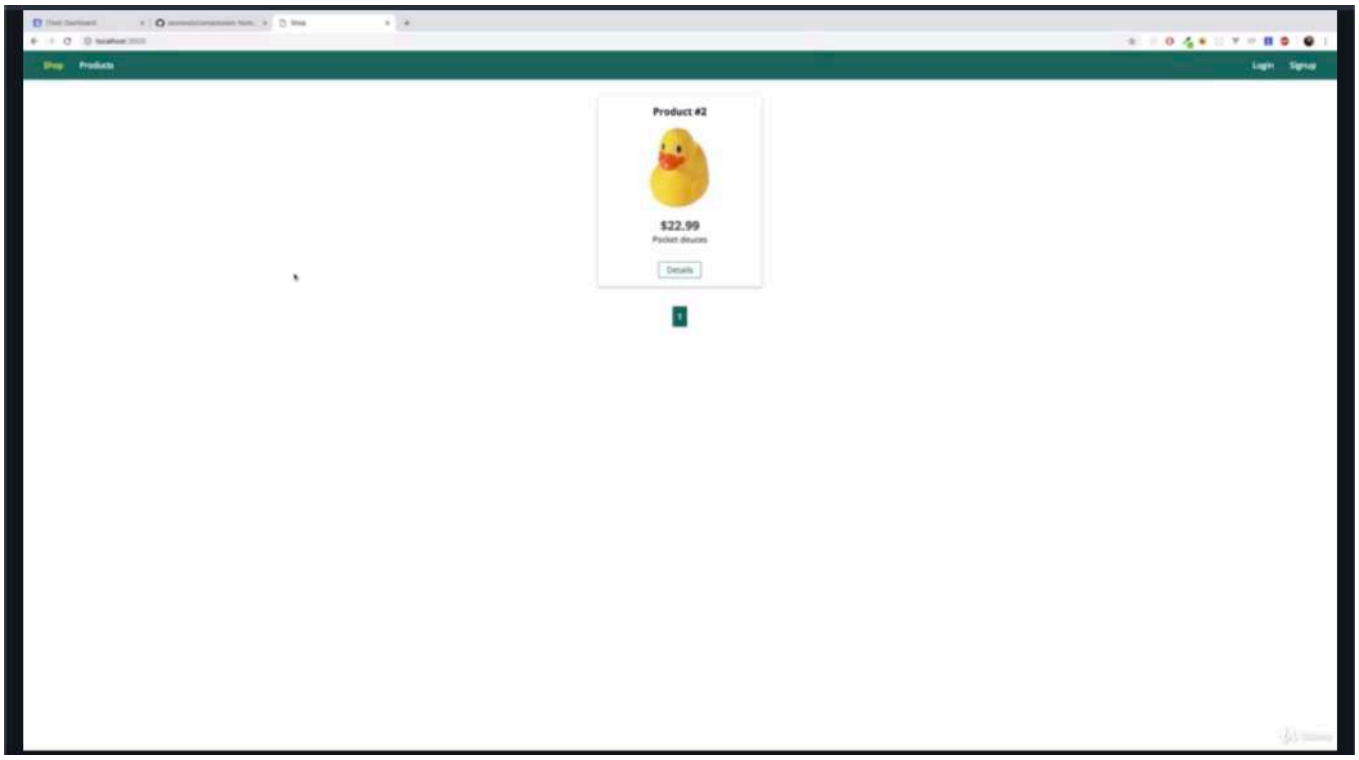
* Chapter 450: Setting Up Request Logging

1. update
- app.js

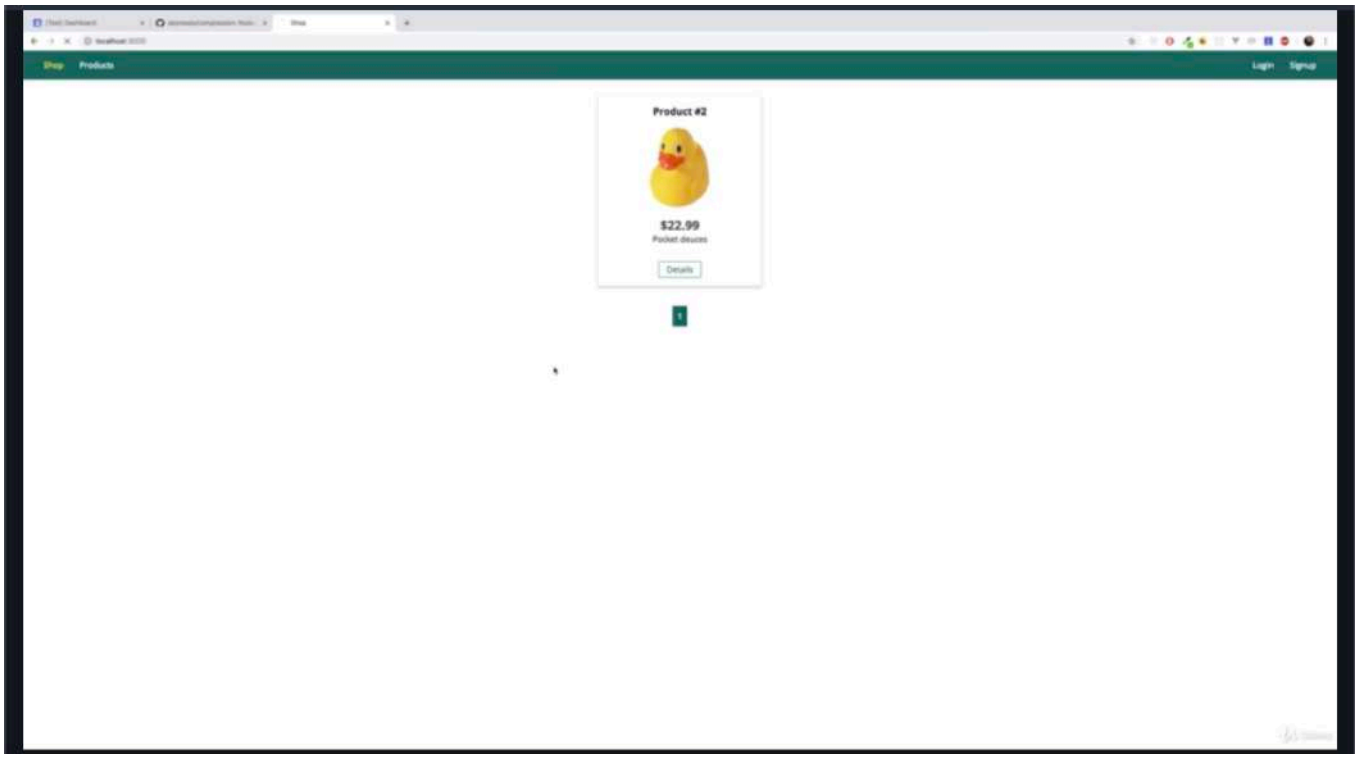

```
EXPLORER                                app.js
NODEJS-COMPLETE-GUIDE
  .vscode
  controllers
    admin.js
    auth.js
    error.js
    shop.js
  data
  images
  middleware
  models
  node_modules
  public
  routes
  util
  views
  .gitignore
  app.js                                M
  message.txt
  nodemon.json
  package-lock.json                     M
  package.json                          M
OUTLINE
28-deployment-classic*  0 ▲ 0           Ln 57, Col 19  Spaces: 2  UTF-8  LF  JavaScript  Prettier

46      cb(null, false);
47    }
48  };
49
50  app.set('view engine', 'ejs');
51  app.set('views', 'views');
52
53  const adminRoutes = require('./routes/admin');
54  const shopRoutes = require('./routes/shop');
55  const authRoutes = require('./routes/auth');
56
57  app.use(helmet());
58  app.use(compression());
59
60  app.use(bodyParser.urlencoded({ extended: false }));
61  app.use(
62    multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
63  );
64  app.use(express.static(path.join(__dirname, 'public')));
65  app.use('/images', express.static(path.join(__dirname, 'images')));
66  app.use(
67    session({

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                    1: node

Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ npm install --save morgan
[                    ] ⸨ rollbackFailedOptional: verb npm-session be67ef687713ef4e
```

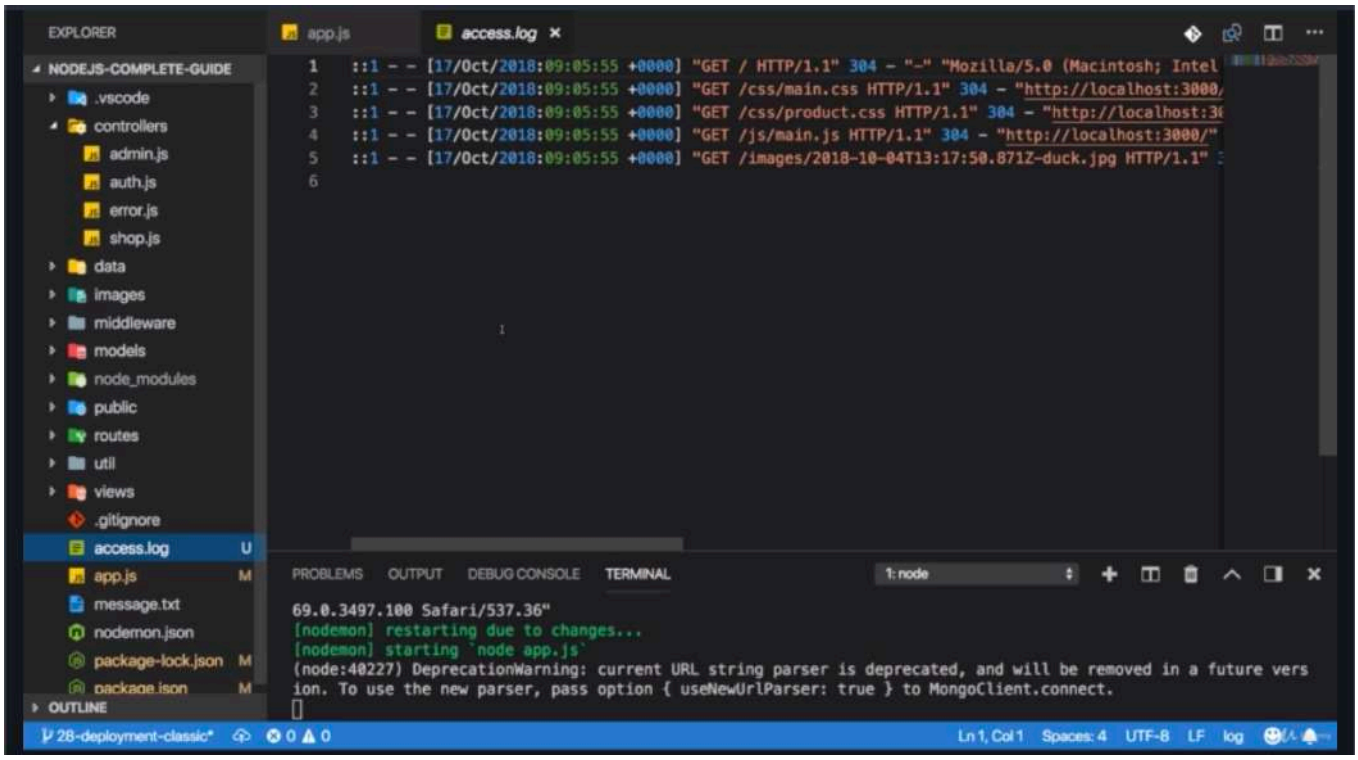
- 'npm install —save morgan' is the package that makes logging request data really simple.
- 'npm run start:dev' is my nodemon server


```
EXPLORER                                app.js
NODEJS-COMPLETE-GUIDE
  .vscode
  controllers
    admin.js
    auth.js
    error.js
    shop.js
  data
  images
  middleware
  models
  node_modules
  public
  routes
  util
  views
  .gitignore
  app.js                                M
  message.txt
  nodemon.json
  package-lock.json                     M
  package.json                          M
OUTLINE
28-deployment-classic*  0 ▲ 0           Ln 60, Col 29  Spaces: 2  UTF-8  LF  JavaScript  Prettier

44    ) {
45      cb(null, true);
46    } else {
47      cb(null, false);
48    }
49  };
50
51  app.set('view engine', 'ejs');
52  app.set('views', 'views');
53
54  const adminRoutes = require('./routes/admin');
55  const shopRoutes = require('./routes/shop');
56  const authRoutes = require('./routes/auth');
57
58  app.use(helmet());
59  app.use(compression());
60  app.use(morgan('combined'));
61
62  app.use(bodyParser.urlencoded({ extended: false }));
63  app.use(
64    multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
65  );

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                    1: node

::1 - - [17/Oct/2018:09:03:23 +0000] "GET /images/2018-10-04T13:17:50.871Z-duck.jpg HTTP/1.1" 304 - "http://lo
calhost:3000/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
69.0.3497.100 Safari/537.36"
::1 - - [17/Oct/2018:09:03:24 +0000] "GET /favicon.ico HTTP/1.1" 404 - "http://localhost:3000/" "Mozilla/5.0 (
Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36"
```



- if i reload, you will find some logging data here in the console. you see detailed logging information about the incoming request that we had a GET request which browser i used which operating system and so on.
- we typically don't wanna see that in the console when we deploy our application. instead some files would be nice and to log that 2 files, we just have to add something here.





- now if i reload, we don't have a log here but we have this new 'access.log' file and here we see the log data and that is how we would wanna log that.

```
//app.js

const path = require('path');
const fs = require('fs');

const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const session = require('express-session');
const MongoDBStore = require('connect-mongodb-session')(session);
const csrf = require('csurf');
const flash = require('connect-flash');
const multer = require('multer');
const helmet = require('helmet');
const compression = require('compression');
const morgan = require('morgan');

const errorController = require('./controllers/error');
const shopController = require('./controllers/shop');
const isAuth = require('./middleware/is-auth');
const User = require('./models/user');

const MONGODB_URI =
  `mongodb+srv://${process.env.MONGO_USER}:${process.env.MONGO_PASSWORD}@cluster0-z3vlk.mongodb.net/${process.env.MONGO_DEFAULT_DATABASE}`;

const app = express();
const store = new MongoDBStore({
  uri: MONGODB_URI,
  collection: 'sessions'
});
const csrfProtection = csrf();

```

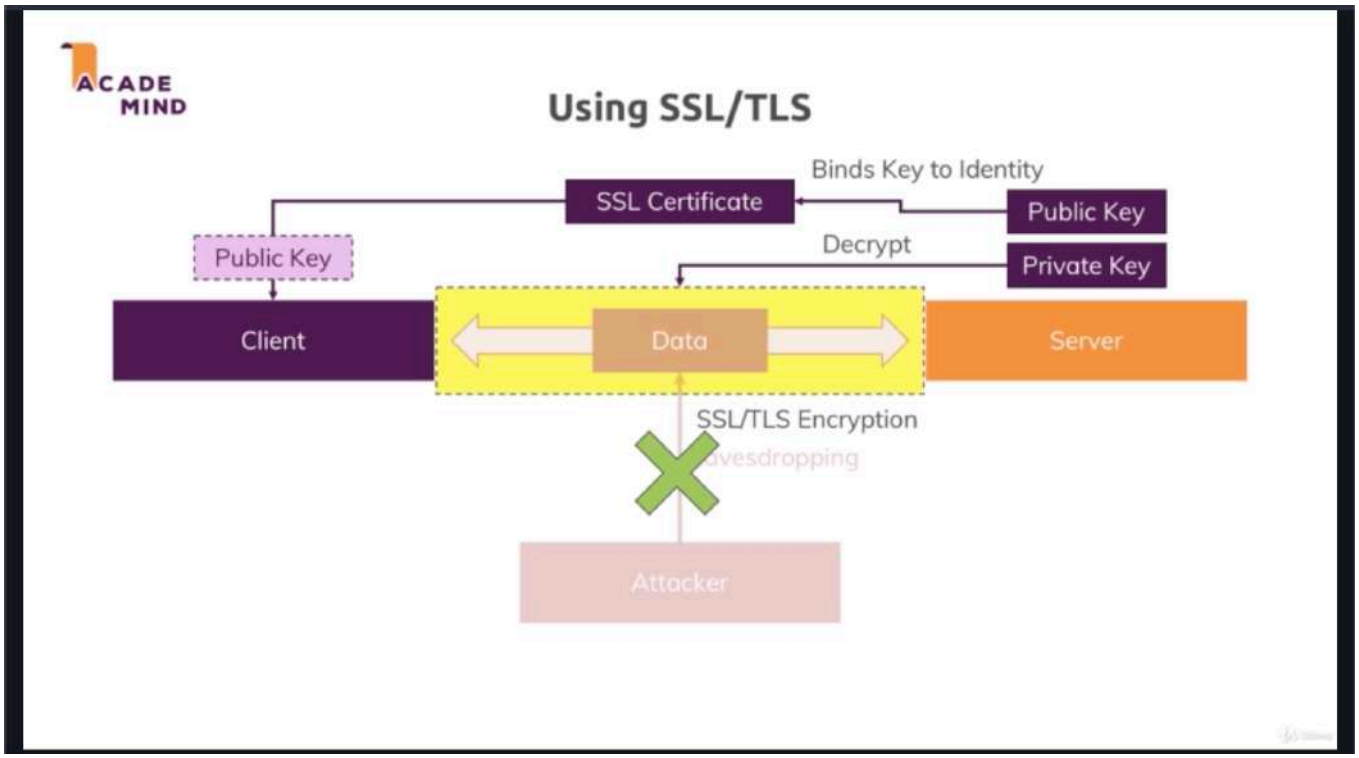
```
33 const fileStorage = multer.diskStorage({
34   destination: (req, file, cb) => {
35     cb(null, 'images');
36   },
37   filename: (req, file, cb) => {
38     cb(null, new Date().toISOString() + '-' + file.originalname);
39   }
40 });
41
42 const fileFilter = (req, file, cb) => {
43   if (
44     file.mimetype === 'image/png' ||
45     file.mimetype === 'image/jpg' ||
46     file.mimetype === 'image/jpeg'
47   ) {
48     cb(null, true);
49   } else {
50     cb(null, false);
51   }
52 };
53
54 app.set('view engine', 'ejs');
55 app.set('views', 'views');
56
57 const adminRoutes = require('./routes/admin');
58 const shopRoutes = require('./routes/shop');
59 const authRoutes = require('./routes/auth');
60
61 /**we write to log with 'access.log' into this filel
62  * '{ flags: 'a' }' means append
63  * so new data will be appended to that file
64  * and not overwrite the existing file
65  * but added at the end of file
66  * so that we don't have log statement in the file
67  * but we continuously add them to the file
68  *
69  * this 'writeStream' can be used by 'morgan'
70  * and we passed this Stream to morgan
71 */
72 const accessLogStream = fs.createWriteStream(
73   path.join(
74     __dirname,
75     'access.log'),
76     { flags: 'a' }
77   );
78
79 app.use(helmet());
80 app.use(compression());
81 /**in parentheses,
82  * pass the information on how to log this into this function
83  * now you fnd more in the official docs
84  * that defines which data is being locked
85  * and how it's formatted
86  * i will go with 'combined'
87  *
88  * 2nd argument will be used to log our requests
```

```
 * and they offer it with a reload */
app.use(morgan('combined', { stream:accessLogStream }));

app.use(bodyParser.urlencoded({ extended: false }));
app.use(
  multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
);
app.use(express.static(path.join(__dirname, 'public')));
app.use('/images', express.static(path.join(__dirname, 'images')));
app.use(
  session({
    secret: 'my secret',
    resave: false,
    saveUninitialized: false,
    store: store
  })
);

app.use(flash());

app.use((req, res, next) => {
  res.locals.isAuthenticated = req.session.isLoggedIn;
  next();
});

app.use((req, res, next) => {
  // throw new Error('Sync Dummy');
  if (!req.session.user) {
    return next();
  }
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
    .catch(err => {
      next(new Error(err));
    });
});

app.post('/create-order', isAuth, shopController.postOrder);

app.use(csrfProtection);
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});

app.use('/admin', adminRoutes);
app.use(shopRoutes);
app.use(authRoutes);

app.get('/500', errorController.get500);
```

```
app.use(errorController.get404);

app.use((error, req, res, next) => {
  // res.status(error.httpStatusCode).render(...);
  // res.redirect('/500');
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
});

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    app.listen(process.env.PORT || 3000);
  })
  .catch(err => {
    console.log(err);
  });

```

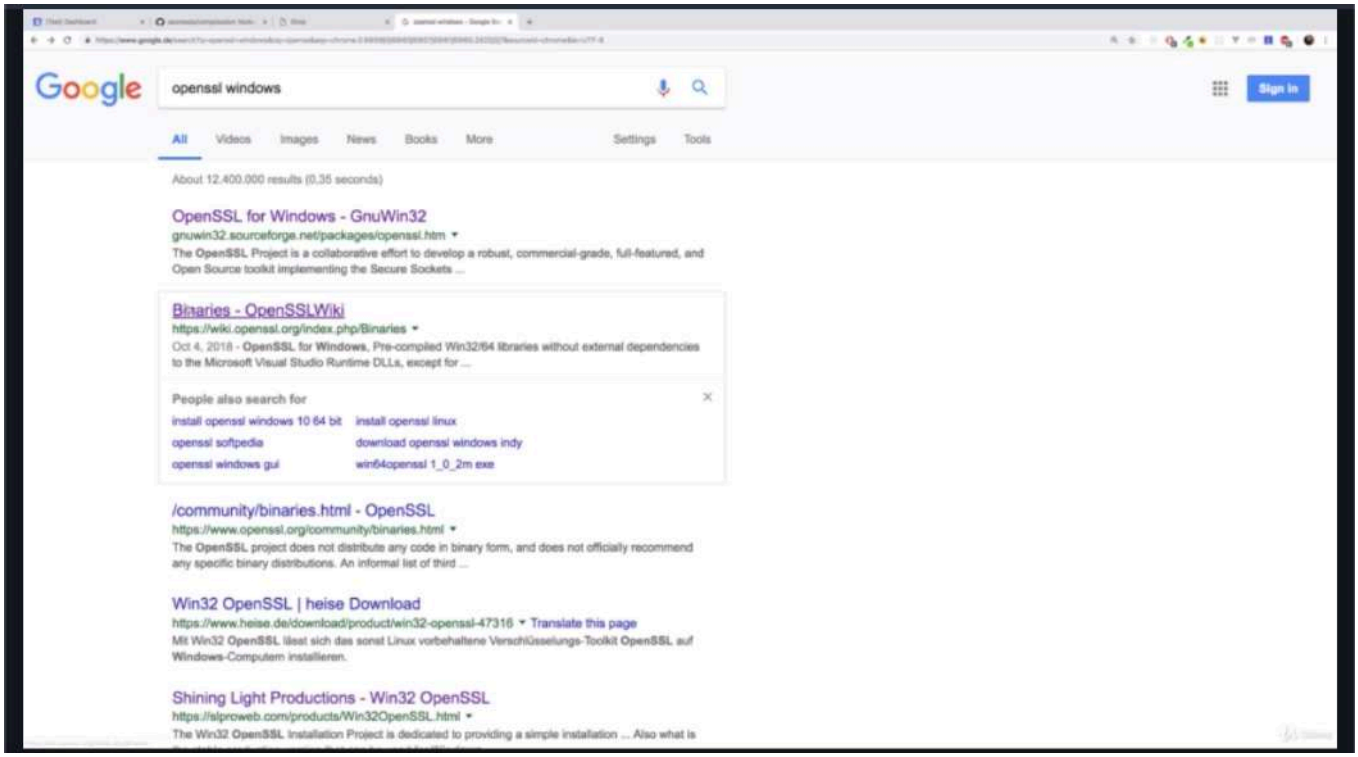
* Chapter 452: Setting Up A SSL Server

1. update
- app.js

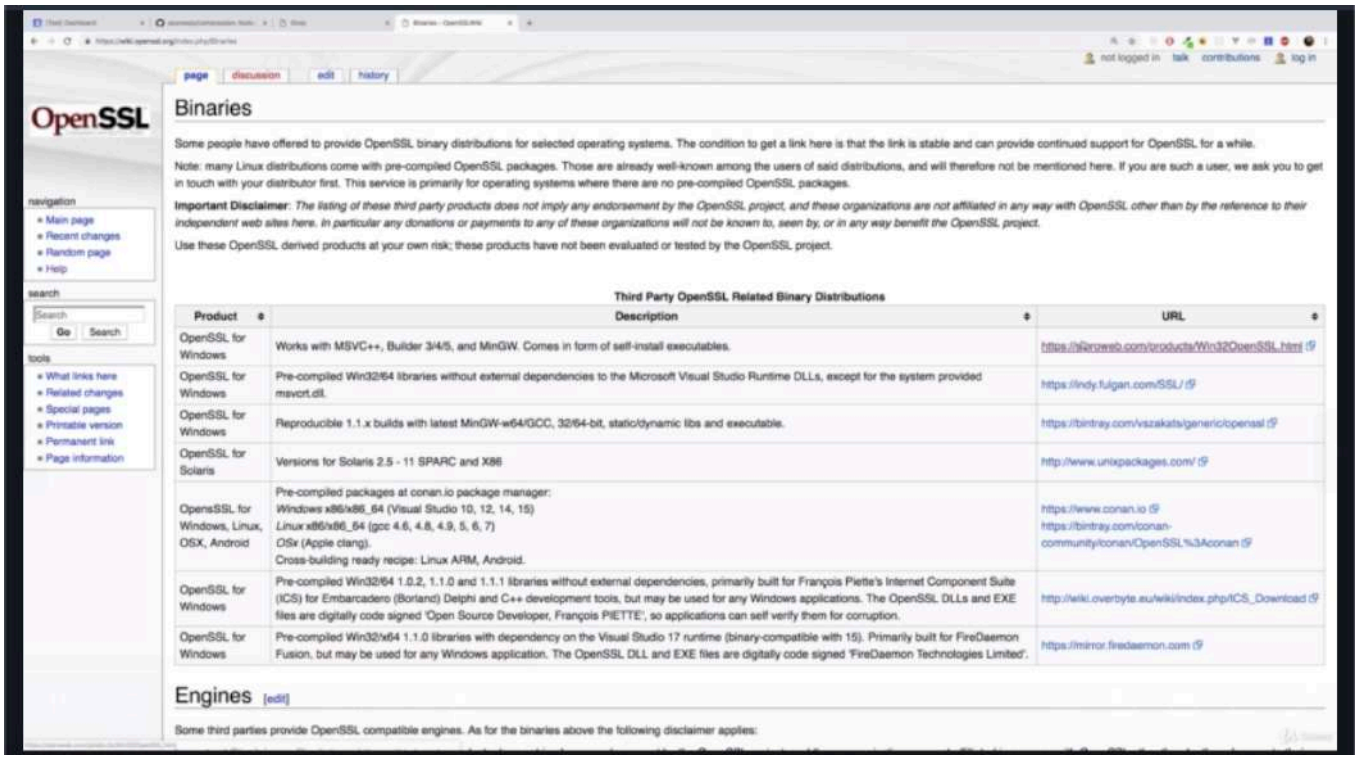


- TLS is newer version of SSL. people know more about SSL however both is about securing your data that is sent from a client to server.
- one such encryption is in place, eavesdropping is not possible anymore because while the data is unreadable as long as it is in transit and it will be decrypted on the server now. to enable that encryption and to be able to decrypt it, we work with a public private key pair. both is known to the server. public key is not something we

have to protect. private key will ever only be known by the server because the private key will later be important for decrypting the data.
- identity is something like the domain, the admin, email address you set to get data when you create a certificate. SSL connects a public key and a server and send that to the client browser. so that the client also is aware of the public key and knows it belongs to that server.
- when you create your own keys, then the browser doesn't trust you that information in there is correct and that is when you get informations or warnings like hey does page uses SSL but doesn't seem to be secure. do you really wanna visit it? hence in production you would use a SSL certificate provided by a known certificate authority which browser trusts and therefore you have a real secure and trusted protection.
- nontheless the way it works always is the same, we have that public key part of that certificate ideally is not created by you but by a trusted authority. we will create it here on our own because that will be free.
- public key is then received by the client through SSL certificate and now the client can encrypt the data which it sends to the server and the server can decrypt the data with that private key and only that private key can decrypt that data.



- we need to create a certificate and we do it with a command named 'openssl' on Mac and Linux. you have that available by default.
- 'openssl req -nodes - new -x509 -keyout server.key -out server.cert' will give you that private key and the public key packaged up in a certificate.
- once you hit enter, you will be asked a couple of questions and there make sure to choose valid values. though that doesn't really matter too much but the idea here is that you connect your identity of your application to your public key. though again your own self-censored defecate will not be accepted by browsers anyways. for production you should not use that option still.
- one important value is just common name. you must set this to 'localhost' otherwiase the certificate will not work because this has to be set to your domain. so if you were to use your self-signed certificate on the server, you deploy your app to and you host this app on example.com then you would have to set this to example.com. again typically you request a certificate for your domain by some authority and then they will do this for you. but if you create your own one use the domain your app is running on and locally that is localhost and this certificate will be denied and he will not be accepted if you set another value.



OpenSSL

navigation
- Main page
- Recent changes
- Random page
- Help

search
Search
Go Search

tools
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link
- Page information

page discussion edit history

not logged in talk contributions log in

Binaries

Some people have offered to provide OpenSSL binary distributions for selected operating systems. The condition to get a link here is that the link is stable and can provide continued support for OpenSSL for a while.

Note: many Linux distributions come with pre-compiled OpenSSL packages. Those are already well-known among the users of said distributions, and will therefore not be mentioned here. If you are such a user, we ask you to get in touch with your distributor first. This service is primarily for operating systems where there are no pre-compiled OpenSSL packages.

Important Disclaimer: *The listing of these third party products does not imply any endorsement by the OpenSSL project, and these organizations are not affiliated in any way with OpenSSL other than by the reference to their independent web sites here. In particular any donations or payments to any of these organizations will not be known to, seen by, or in any way benefit the OpenSSL project.*

Use these OpenSSL derived products at your own risk; these products have not been evaluated or tested by the OpenSSL project.

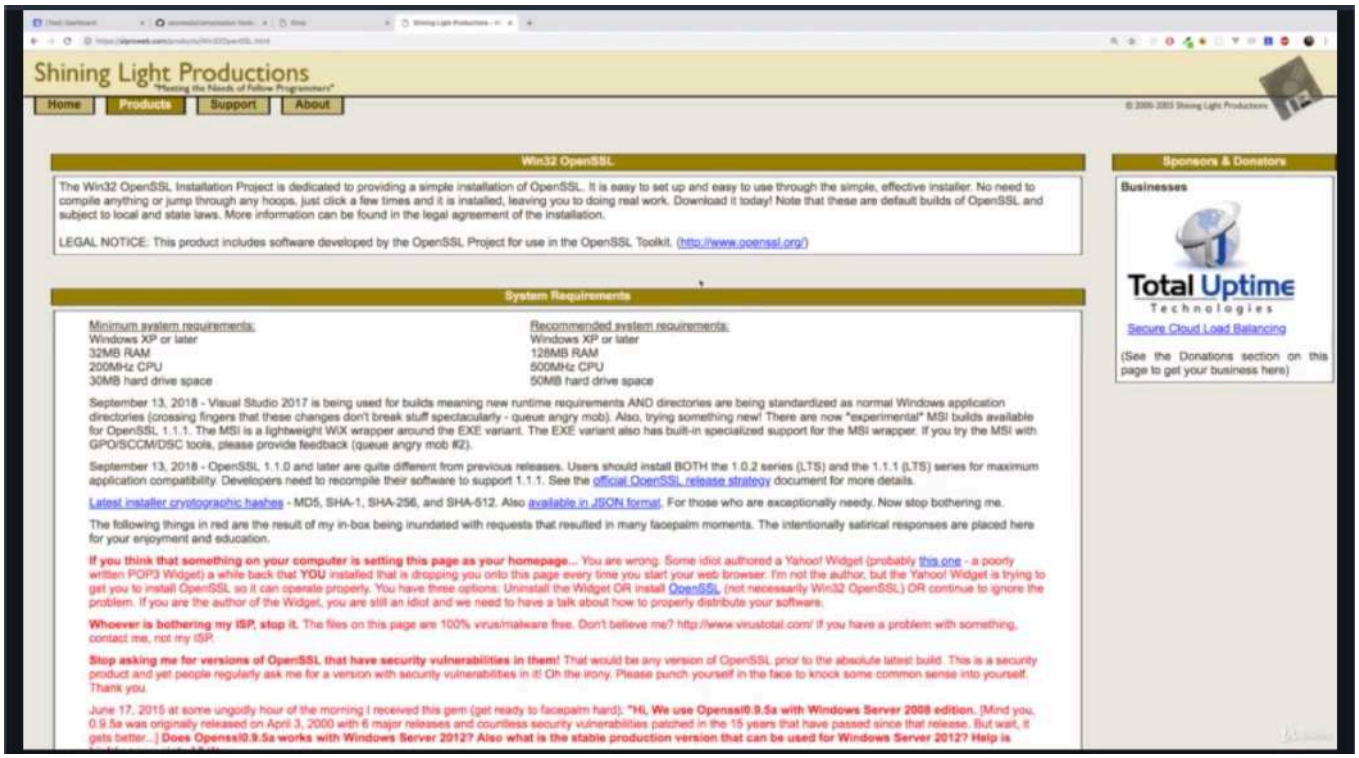
Third Party OpenSSL Related Binary Distributions

Product	Description	URL
OpenSSL for Windows	Works with MSVC++, Builder 3/4/5, and MinGW. Comes in form of self-install executables.	https://slproweb.com/products/Win32OpenSSL.html
OpenSSL for Windows	Pre-compiled Win32/64 libraries without external dependencies to the Microsoft Visual Studio Runtime DLLs, except for the system provided msvcrt.dll.	https://indy.fulgan.com/SSL/
OpenSSL for Windows	Reproducible 1.1.x builds with latest MinGW-w64/GCC, 32/64-bit, static/dynamic libs and executable.	https://bintray.com/vszakats/generic/openssl
OpenSSL for Solaris	Versions for Solaris 2.5 - 11 SPARC and X86	http://www.unixpackages.com/
OpensSSL for Windows, Linux, OSX, Android	Pre-compiled packages at conan.io package manager: *Windows* x86/x86_64 (Visual Studio 10, 12, 14, 15) *Linux* x86/x86_64 (gcc 4.6, 4.8, 4.9, 5, 6, 7) *OSx* (Apple clang). Cross-building ready recipe: Linux ARM, Android.	https://www.conan.io https://bintray.com/conan-community/conan/OpenSSL%3Aconan
OpenSSL for Windows	Pre-compiled Win32/x64 1.0.2, 1.1.0 and 1.1.1 libraries without external dependencies, primarily built for François Piette's Internet Component Suite (ICS) for Embarcadero (Borland) Delphi and C++ development tools, but may be used for any Windows applications. The OpenSSL DLLs and EXE files are digitally code signed 'Open Source Developer, François PIETTE', so applications can self verify them for corruption.	http://wiki.overbyte.eu/wiki/index.php/ICS_Download
OpenSSL for Windows	Pre-compiled Win32/x64 1.1.0 libraries with dependency on the Visual Studio 17 runtime (binary-compatible with 15). Primarily built for FireDaemon Fusion, but may be used for any Windows application. The OpenSSL DLL and EXE files are digitally code signed 'FireDaemon Technologies Limited'.	https://mirror.firedaemon.com

Engines [edit]

Some third parties provide OpenSSL compatible engines. As for the binaries above the following disclaimer applies:

- after you did all that, you will find 2 new files 'server.cert' which is the certificate and 'server.key' which is the private key. now a private key will always on your server. the certificate is what we send to the client in the end.



Shining Light Productions
"Meeting the Needs of Fellow Programmers"

Home Products Support About

© 2000-2003 Shining Light Productions

Win32 OpenSSL

The Win32 OpenSSL Installation Project is dedicated to providing a simple installation of OpenSSL. It is easy to set up and easy to use through the simple, effective installer. No need to compile anything or jump through any hoops, just click a few times and it is installed, leaving you to doing real work. Download it today! Note that these are default builds of OpenSSL and subject to local and state laws. More information can be found in the legal agreement of the installation.

LEGAL NOTICE: This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)

System Requirements

Minimum system requirements:
Windows XP or later
32MB RAM
200MHz CPU
30MB hard drive space

Recommended system requirements:
Windows XP or later
128MB RAM
500MHz CPU
50MB hard drive space

September 13, 2018 - Visual Studio 2017 is being used for builds meaning new runtime requirements AND directories are being standardized as normal Windows application directories (crossing fingers that these changes don't break stuff spectacularly - queue angry mob). Also, trying something new! There are now "experimental" MSI builds available for OpenSSL 1.1.1. The MSI is a lightweight WiX wrapper around the EXE variant. The EXE variant also has built-in specialized support for the MSI wrapper. If you try the MSI with GPO/SCCM/DSC tools, please provide feedback (queue angry mob #2).

September 13, 2018 - OpenSSL 1.1.0 and later are quite different from previous releases. Users should install BOTH the 1.0.2 series (LTS) and the 1.1.1 (LTS) series for maximum application compatibility. Developers need to recompile their software to support 1.1.1. See the official OpenSSL release strategy document for more details.

Latest installer cryptographic hashes - MD5, SHA-1, SHA-256, and SHA-512. Also available in JSON format. For those who are exceptionally needy. Now stop bothering me.

The following things in red are the result of my in-box being inundated with requests that resulted in many facepalm moments. The intentionally satirical responses are placed here for your enjoyment and education.

If you think that something on your computer is setting this page as your homepage... You are wrong. Some idiot authored a Yahoo! Widget (probably this one - a poorly written POP3 Widget) a while back that **YOU** installed that is dropping you onto this page every time you start your web browser. I'm not the author, but the Yahoo! Widget is trying to get you to install OpenSSL so it can operate properly. You have three options: Uninstall the Widget OR install OpenSSL (not necessarily Win32 OpenSSL) OR continue to ignore the problem. If you are the author of the Widget, you are still an idiot and we need to have a talk about how to properly distribute your software.

Whoever is bothering my ISP, stop it. The files on this page are 100% virus/malware free. Don't believe me? http://www.virustotal.com/ If you have a problem with something, contact me, not my ISP.

Stop asking me for versions of OpenSSL that have security vulnerabilities in them! That would be any version of OpenSSL prior to the absolute latest build. This is a security product and yet people regularly ask me for a version with security vulnerabilities in it! Oh the irony. Please punch yourself in the face to knock some common sense into yourself. Thank you.

June 17, 2015 at some ungodly hour of the morning I received this gem (get ready to facepalm hard): **"Hi, We use Openssl0.9.5a with Windows Server 2008 edition.** [Mind you, 0.9.5a was originally released on April 3, 2000 with 6 major releases and countless security vulnerabilities patched in the 15 years that have passed since that release. But wait, it gets better...] **Does Openssl0.9.5a works with Windows Server 2012? Also what is the stable production version that can be used for Windows Server 2012? Help is**

Sponsors & Donators

Businesses

Total Uptime Technologies
Secure Cloud Load Balancing

(See the Donations section on this page to get your business here)

Download Win32 OpenSSL

Download Win32 OpenSSL today using the links below!

File	Type	Description
Win64 OpenSSL v1.1.1 Light EXE | MSI (experimental)	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.1.1 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.1.1 EXE | MSI (experimental)	43MB Installer	Installs Win64 OpenSSL v1.1.1 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1 Light EXE | MSI (experimental)	3MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.1.1 (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1 EXE | MSI (experimental)	30MB Installer	Installs Win32 OpenSSL v1.1.1 (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.1.0i Light	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.1.0i (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.1.0i	37MB Installer	Installs Win64 OpenSSL v1.1.0i (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.0i Light	3MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.1.0i (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.0i	30MB Installer	Installs Win32 OpenSSL v1.1.0i (Only install this if you are a software developer needing 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.0.2p Light	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.0.2p (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.0.2p	23MB Installer	Installs Win64 OpenSSL v1.0.2p (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.0.2p Light	2MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.0.2p (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.0.2p	20MB Installer	Installs Win32 OpenSSL v1.0.2p (Only install this if you are a software developer needing 64-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

EXPLORER

NODEJS-COMPLETE-GUIDE

- .vscode
- controllers
- data
- images
- middleware
- models
- node_modules
- public
- routes
- util
- views
- .gitignore
- access.log
- app.js
- message.txt
- nodemon.json
- package-lock.json
- package.json
- server.key U

OUTLINE

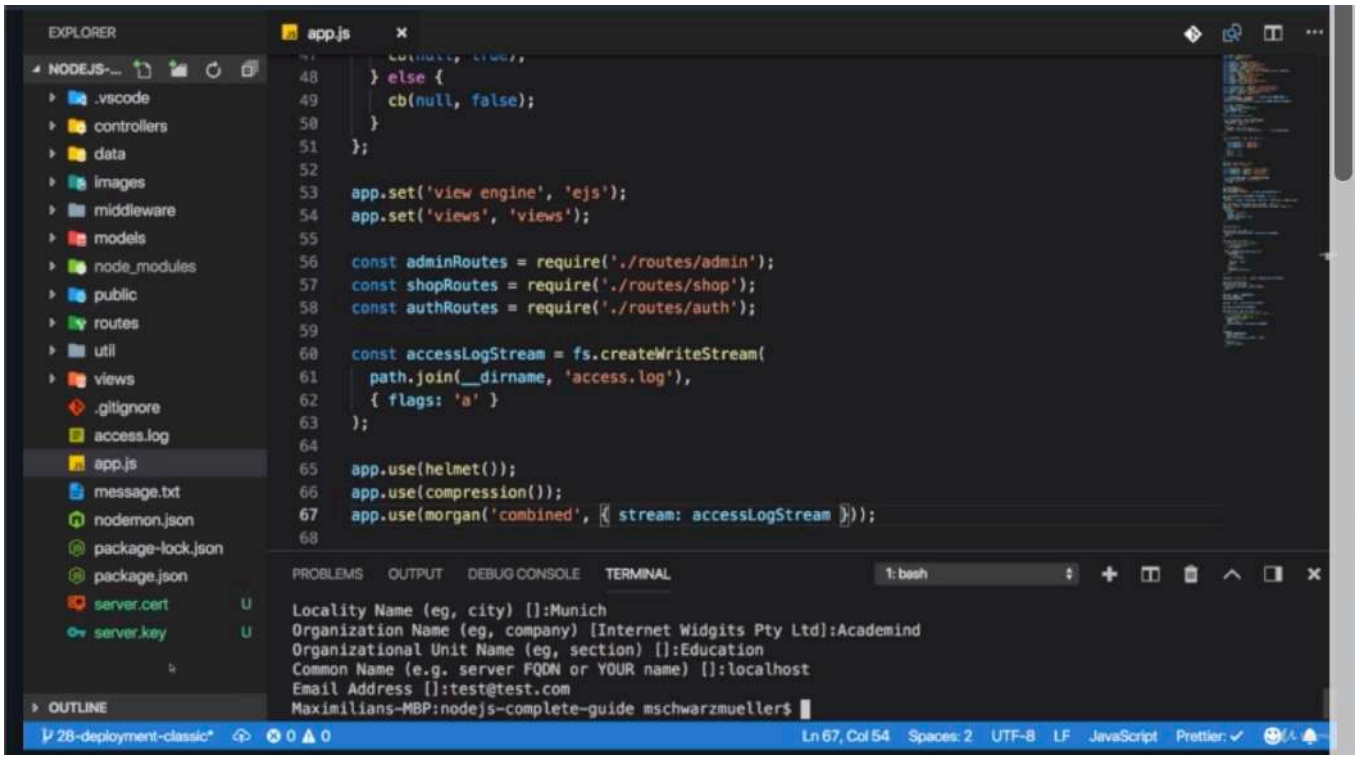
app.js

```
  } else {
    cb(null, false);
  }
};

app.set('view engine', 'ejs');
app.set('views', 'views');

const adminRoutes = require('./routes/admin');
const shopRoutes = require('./routes/shop');
const authRoutes = require('./routes/auth');

const accessLogStream = fs.createWriteStream(
  path.join(__dirname, 'access.log'),
  { flags: 'a' }
);

app.use(helmet());
app.use(compression());
app.use(morgan('combined', { stream: accessLogStream }));
```

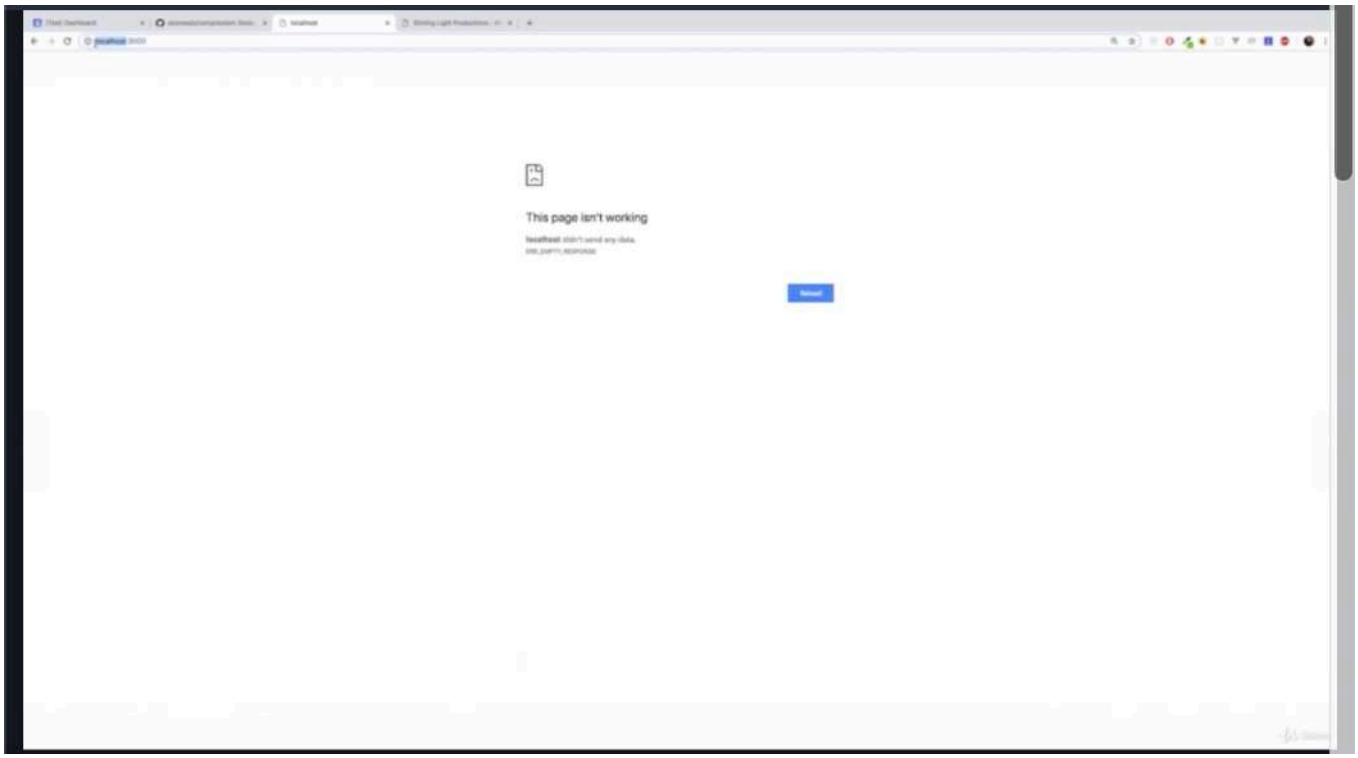
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: openssl

```
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Bavaria
Locality Name (eg, city) []:Munich
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Academind
Organizational Unit Name (eg, section) []:Education
Common Name (e.g. server FQDN or YOUR name) []:localhost
```

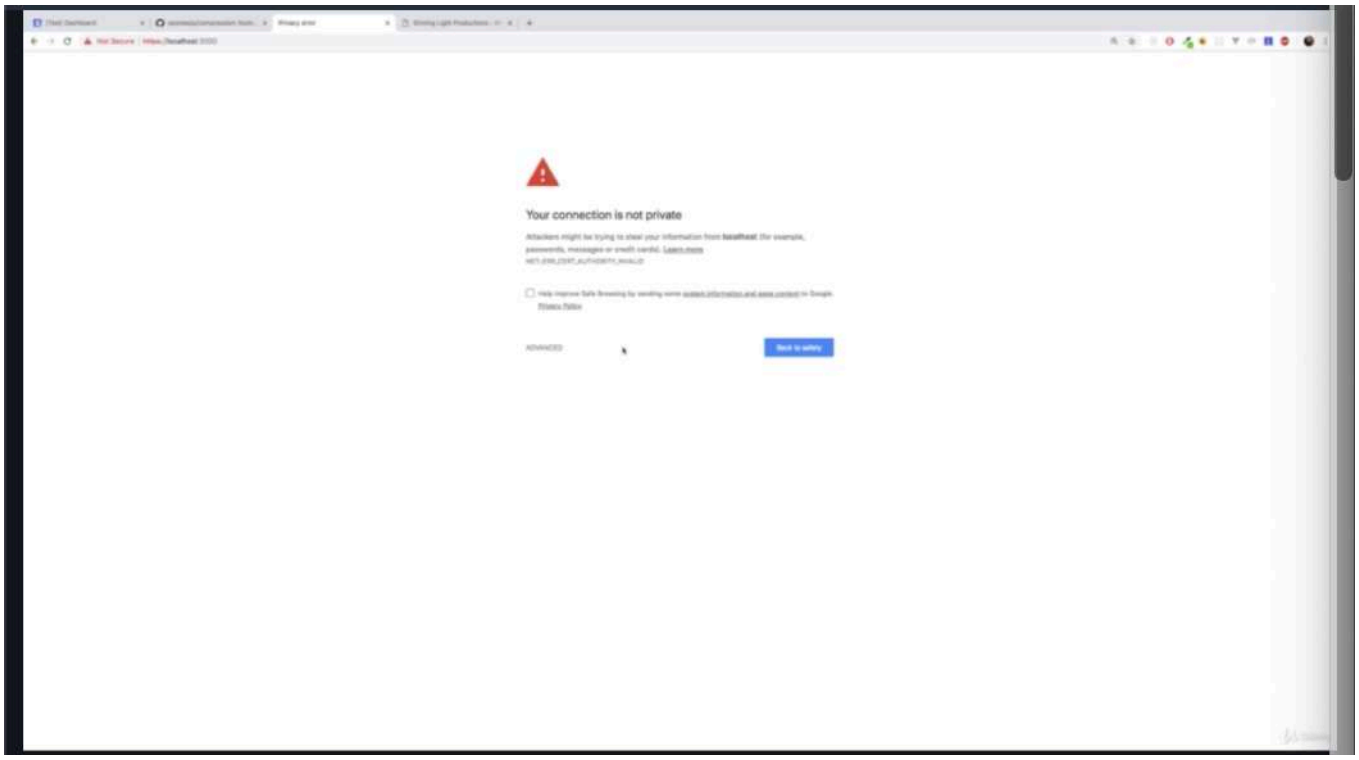
28-deployment-classic* 0 0 Ln 67, Col 54 Spaces: 2 UTF-8 LF JavaScript Prettier:



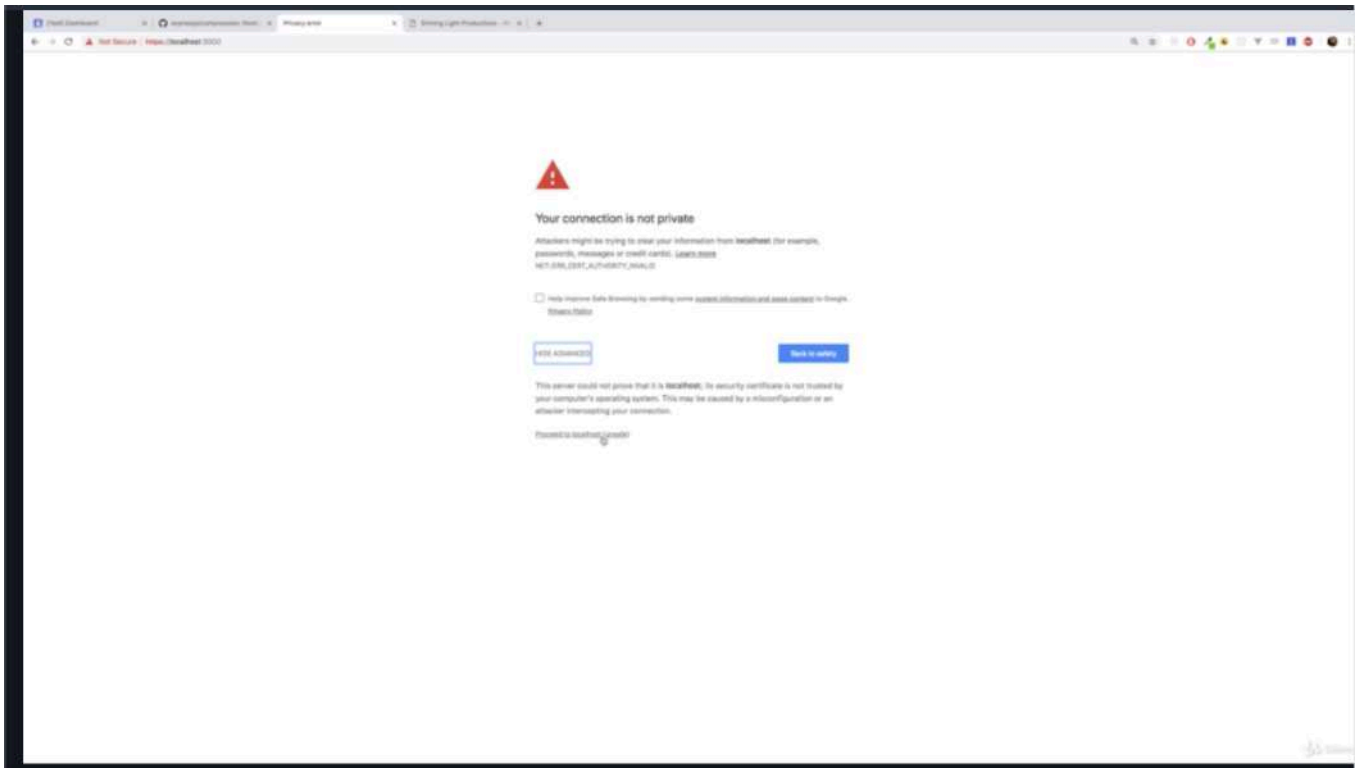
- but on windows you don't get by default. so you need a extra process.

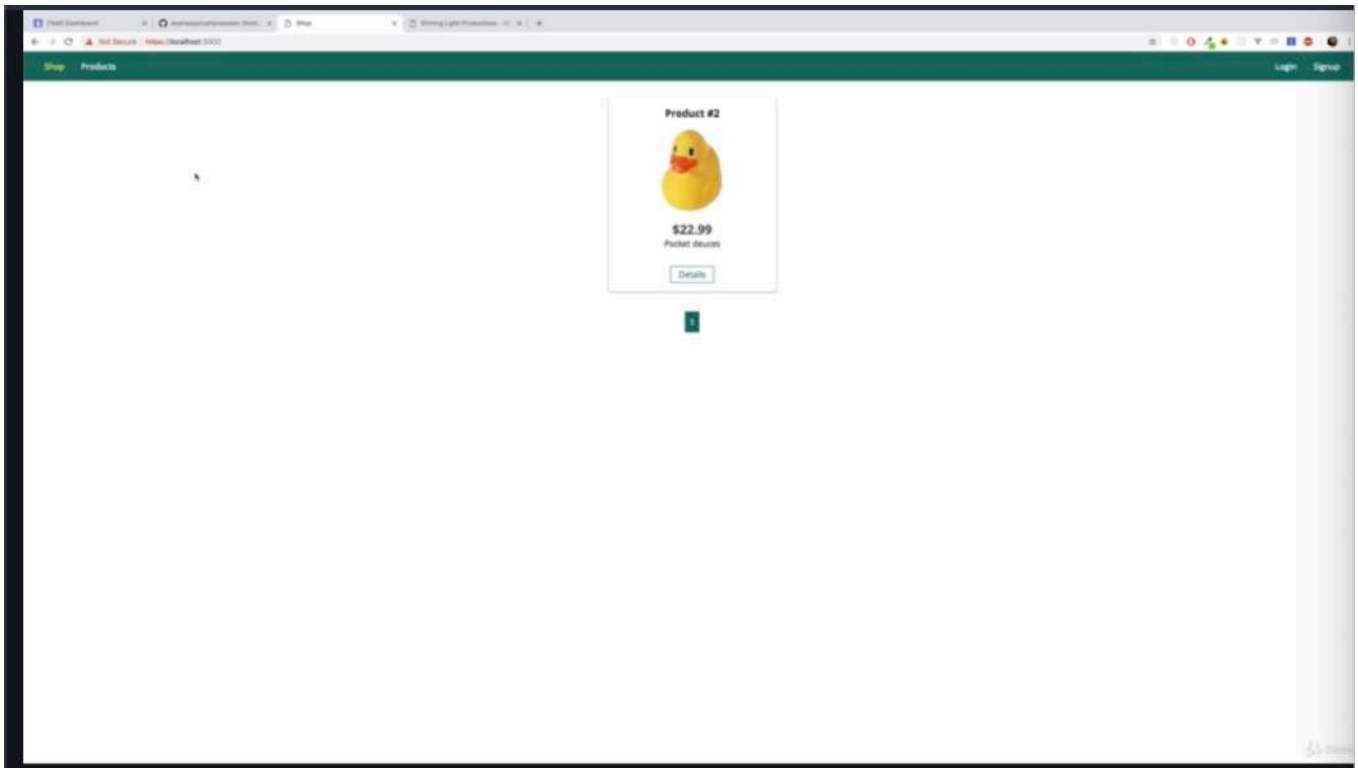


- now using SSL encryption and if we now go back to our application and really reload localhost:3000, this will fail because by default it uses to HTTP.

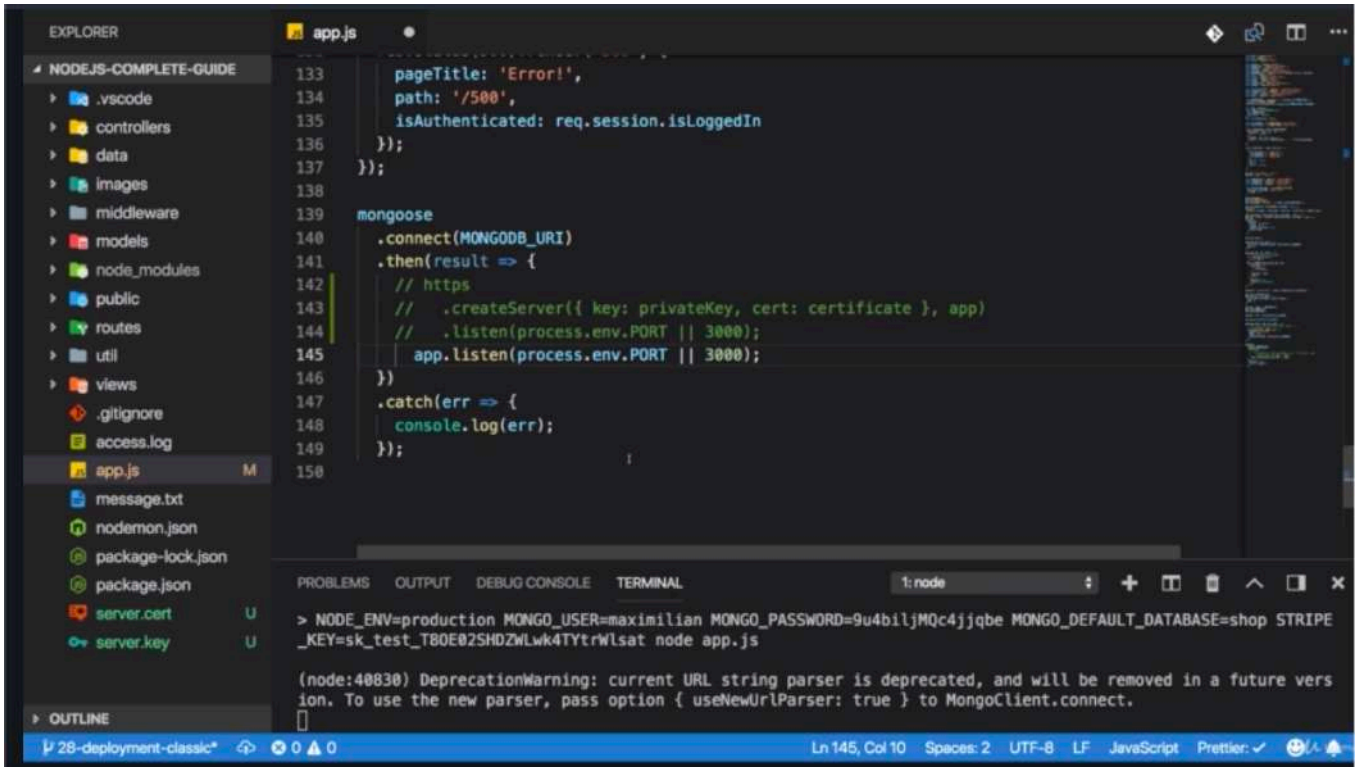


- if i use https://localhost:3000, it will fail because the browser doesn't accept that custom or that self-signed certificate as you learned.





- but if you click on ADVANCED, you can proceed to localhost and again the browser does warn us because it doens't like our self-signed certificate. but technically we are now using SSL protection and this is how you enable it. but just as with logging and compression, typically you would set this up differently. you would let your hosting provider set this up because technically the hosting provider often also has its own service in front of yours and the servers of the hosting provider then use SSL and to traffic between your app and the in-between servers does use HTTP because it's blocked or it's not available to the public and the hosting providers fronts would implement this logic. so you wouldn't write that code on your own.



- and indeed i will fall back to my old code where i had app.listen because we will need that later when we deployed because we will let our hosting provider manage SSL. but if you ever need to do it manually, this is how you start a node server in HTTPS mode.

```
//app.js
```

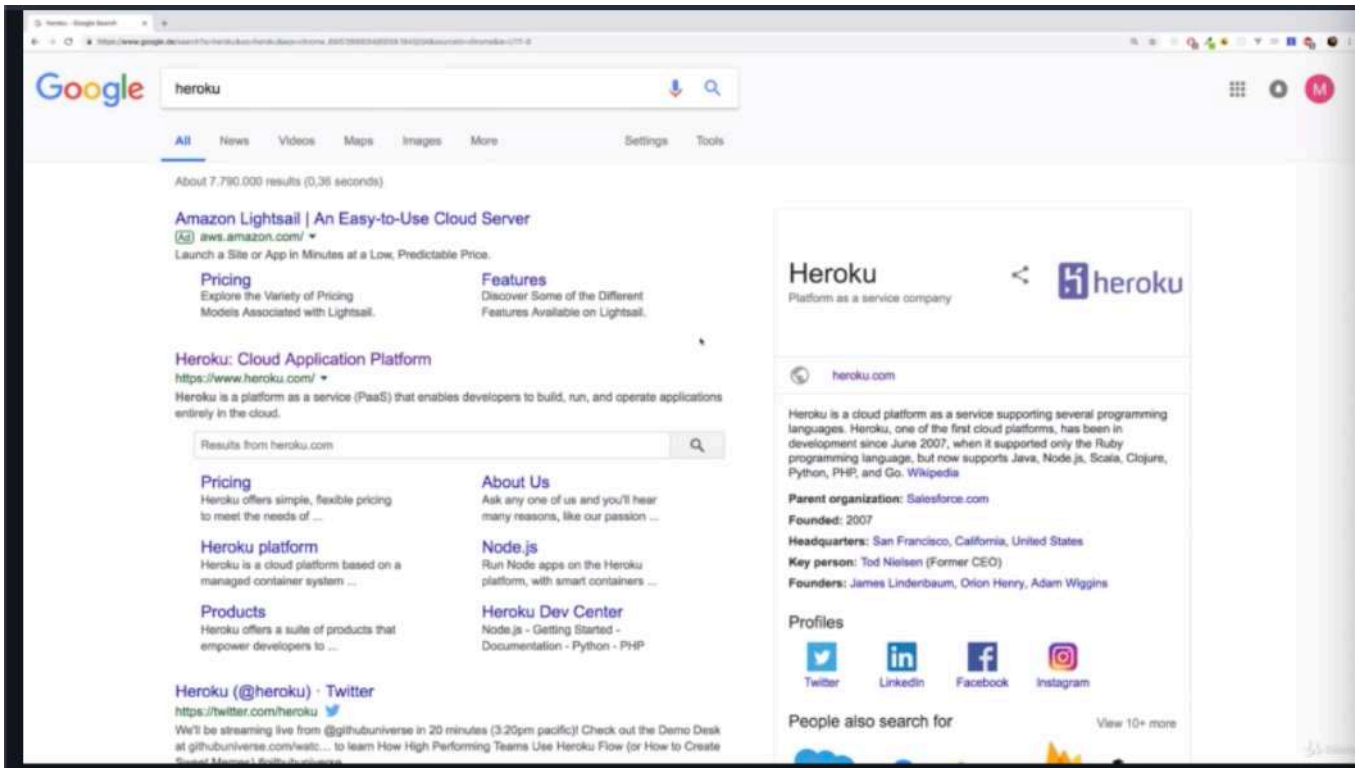
```
2
3 const path = require('path');
4 const fs = require('fs');
5 /**'https' allows us to spin up and HTTPS server
6  * we directly or indirectly through app.listen() use the HTTP
7  * now we will use HTTPS
8  */
9 const https = require('https');
10
11 const express = require('express');
12 const bodyParser = require('body-parser');
13 const mongoose = require('mongoose');
14 const session = require('express-session');
15 const MongoDBStore = require('connect-mongodb-session')(session);
16 const csrf = require('csurf');
17 const flash = require('connect-flash');
18 const multer = require('multer');
19 const helmet = require('helmet');
20 const compression = require('compression');
21 const morgan = require('morgan');
22
23 const errorController = require('./controllers/error');
24 const shopController = require('./controllers/shop');
25 const isAuth = require('./middleware/is-auth');
26 const User = require('./models/user');
27
28 const MONGODB_URI =
29   `mongodb+srv://${process.env.MONGO_USER}:${process.env.MONGO_PASSWORD}@cluster0-
z3vlk.mongodb.net/${process.env.MONGO_DEFAULT_DATABASE}`;
30
31 const app = express();
32 const store = new MongoDBStore({
33   uri: MONGODB_URI,
34   collection: 'sessions'
35 });
36 const csrfProtection = csrf();
37
38 /**'readFileSync' will block code execution
39  * until the is read
40  * and you will learn this is what you wanna do
41  * but i don't wanna continue with starting the server
42  * unless i have read that file.
43  *
44  *  */
45 const privateKey = fs.readFileSync('server.key');
46 const certificate = fs.readFileSync('server.cert');
47
48 const fileStorage = multer.diskStorage({
49   destination: (req, file, cb) => {
50     cb(null, 'images');
51   },
52   filename: (req, file, cb) => {
53     cb(null, new Date().toISOString() + '-' + file.originalname);
54   }
55 });
56
```

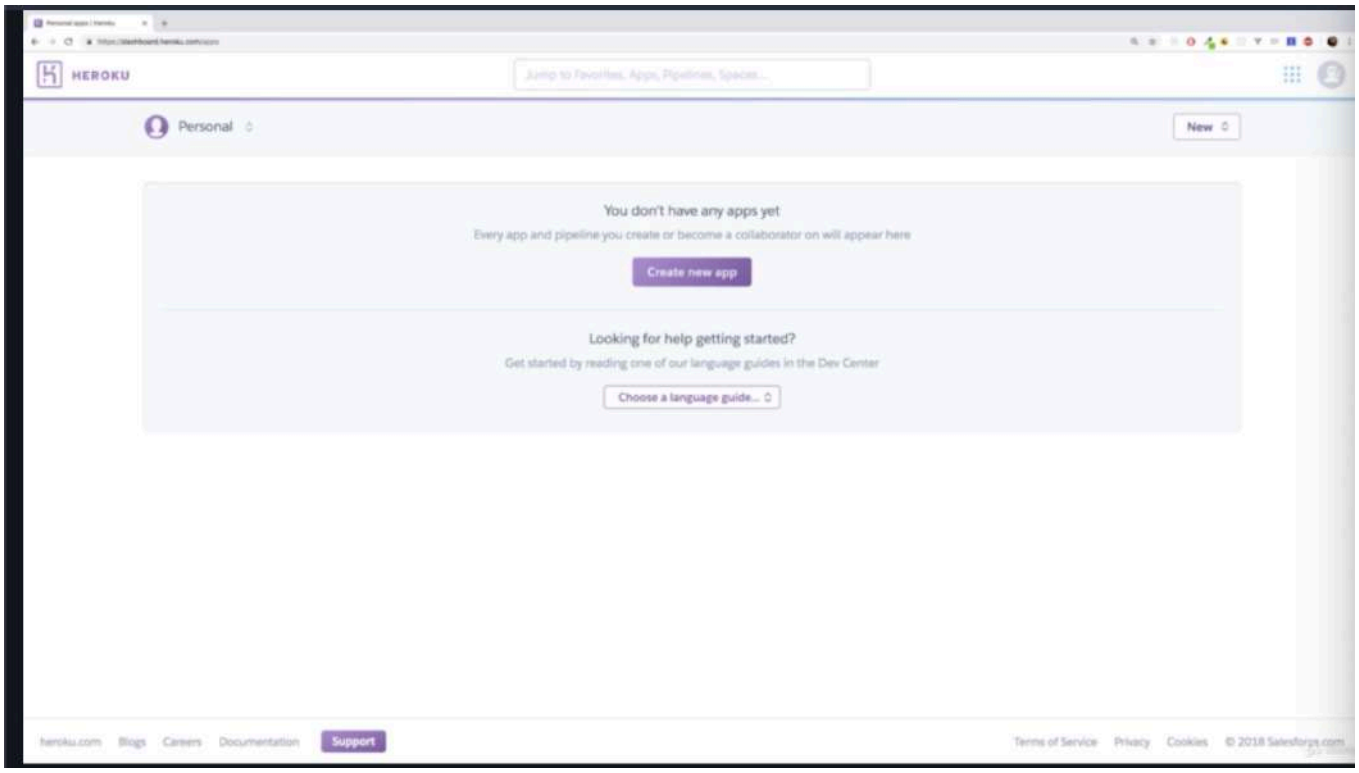
```
57 const fileFilter = (req, file, cb) => {
58   if (
59     file.mimetype === 'image/png' ||
60     file.mimetype === 'image/jpg' ||
61     file.mimetype === 'image/jpeg'
62   ) {
63     cb(null, true);
64   } else {
65     cb(null, false);
66   }
67 };
68
69 app.set('view engine', 'ejs');
70 app.set('views', 'views');
71
72 const adminRoutes = require('./routes/admin');
73 const shopRoutes = require('./routes/shop');
74 const authRoutes = require('./routes/auth');
75
76 const accessLogStream = fs.createWriteStream(
77   path.join(
78     __dirname,
79     'access.log'),
80     { flags: 'a' }
81   );
82
83 app.use(helmet());
84 app.use(compression());
85 app.use(morgan('combined', { stream:accessLogStream }));
86
87 app.use(bodyParser.urlencoded({ extended: false }));
88 app.use(
89   multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
90 );
91 app.use(express.static(path.join(__dirname, 'public')));
92 app.use('/images', express.static(path.join(__dirname, 'images')));
93 app.use(
94   session({
95     secret: 'my secret',
96     resave: false,
97     saveUninitialized: false,
98     store: store
99   })
100 );
101
102 app.use(flash());
103
104 app.use((req, res, next) => {
105   res.locals.isAuthenticated = req.session.isLoggedIn;
106   next();
107 });
108
109 app.use((req, res, next) => {
110   // throw new Error('Sync Dummy');
111   if (!req.session.user) {
112     return next();
```

```
  }
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
    .catch(err => {
      next(new Error(err));
    });
});

app.post('/create-order', isAuth, shopController.postOrder);

app.use(csrfProtection);
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});

app.use('/admin', adminRoutes);
app.use(shopRoutes);
app.use(authRoutes);

app.get('/500', errorController.get500);

app.use(errorController.get404);

app.use((error, req, res, next) => {
  // res.status(error.httpStatusCode).render(...);
  // res.redirect('/500');
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
});

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    /**1st argument is the server.
     * here we have to point it at our private key and certificate
     * and the 2nd argument will be our request handler.
     * in our case, our express application.
     * so the 2nd argument will be our 'app'
     *
     * 1st argument will be a javascript object
     * where you need to set 2 things
     * you need to set the 'key: privateKey'
     * and you also need to set the cert key
     * which you set to certificate constant we created.
     */
    //https.createServer({ key: privateKey, cert: certificate },
```

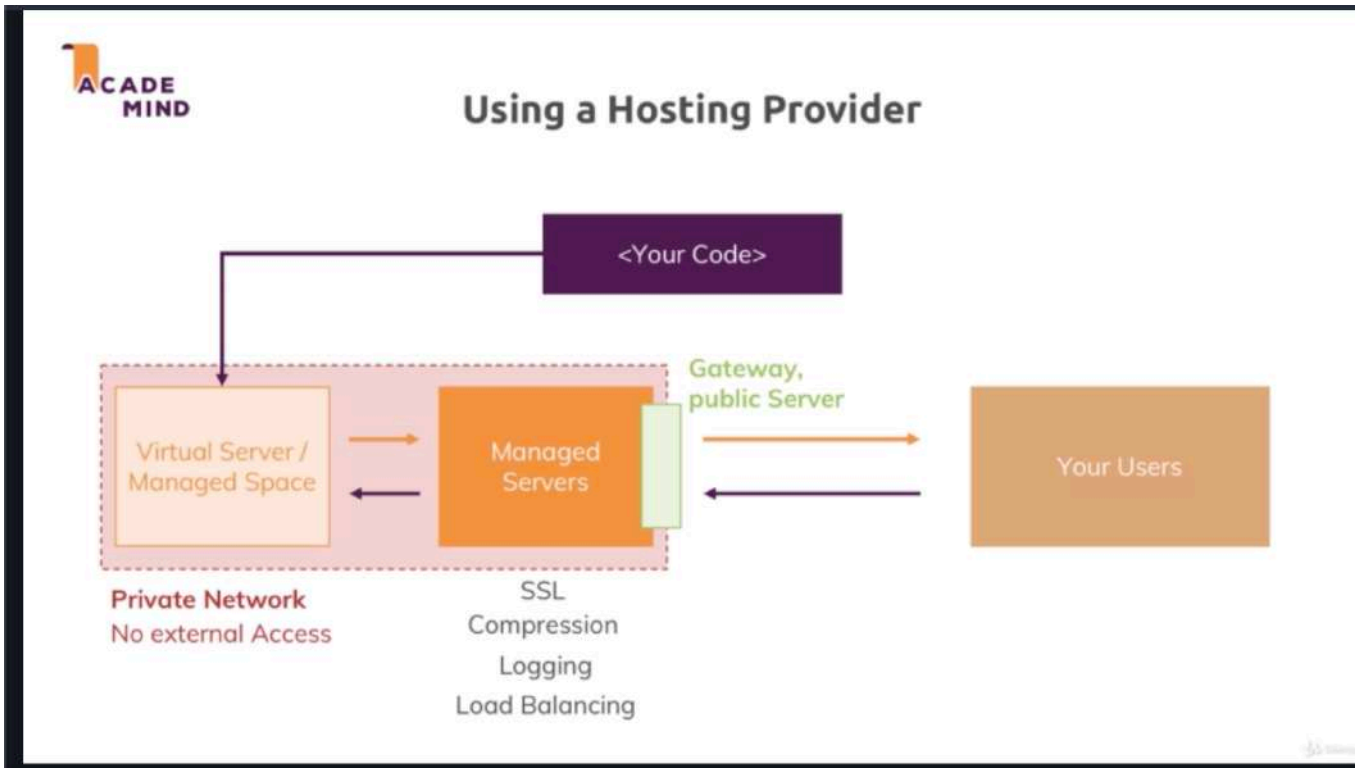
```
app).listen(process.env.PORT || 3000);
    app.listen(process.env.PORT || 3000);
  })
  .catch(err => {
    console.log(err);
  });

```

* Chapter 453: Using A Hosting Provider



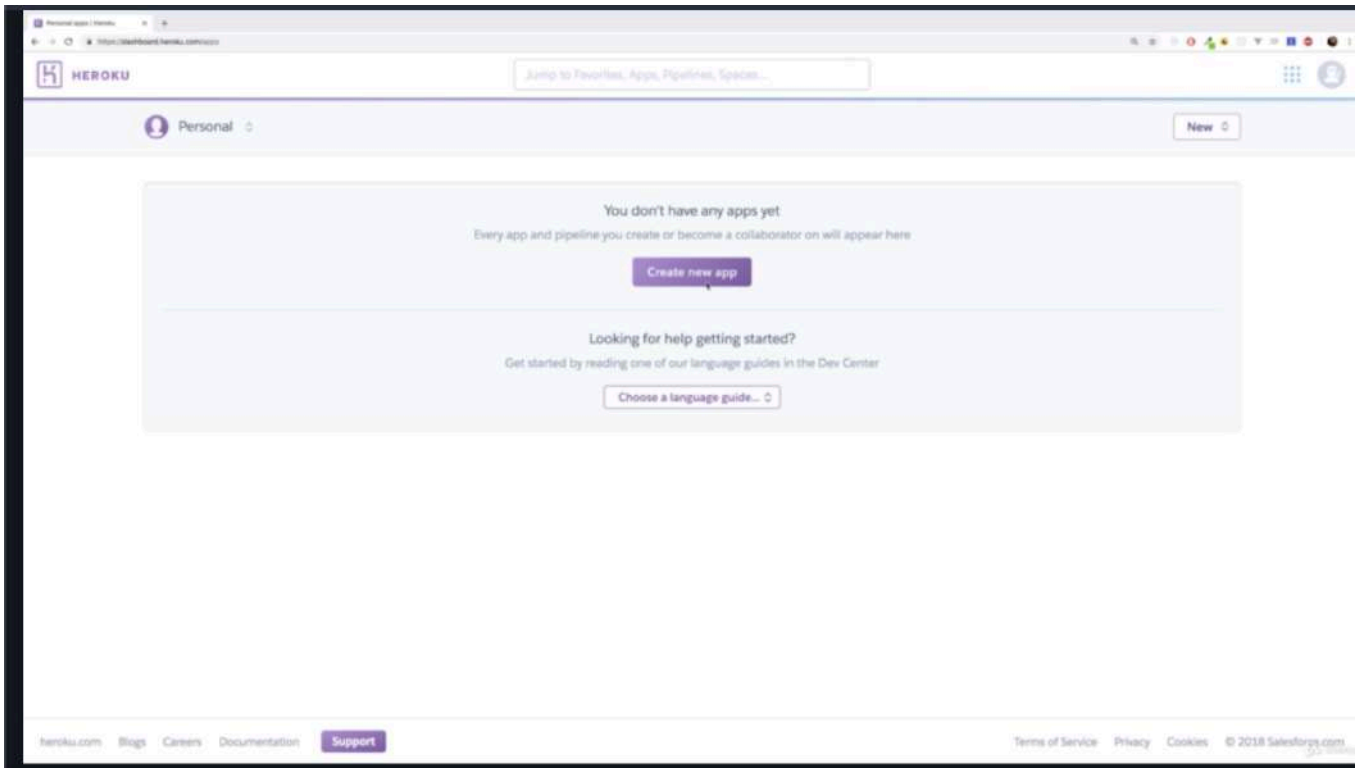


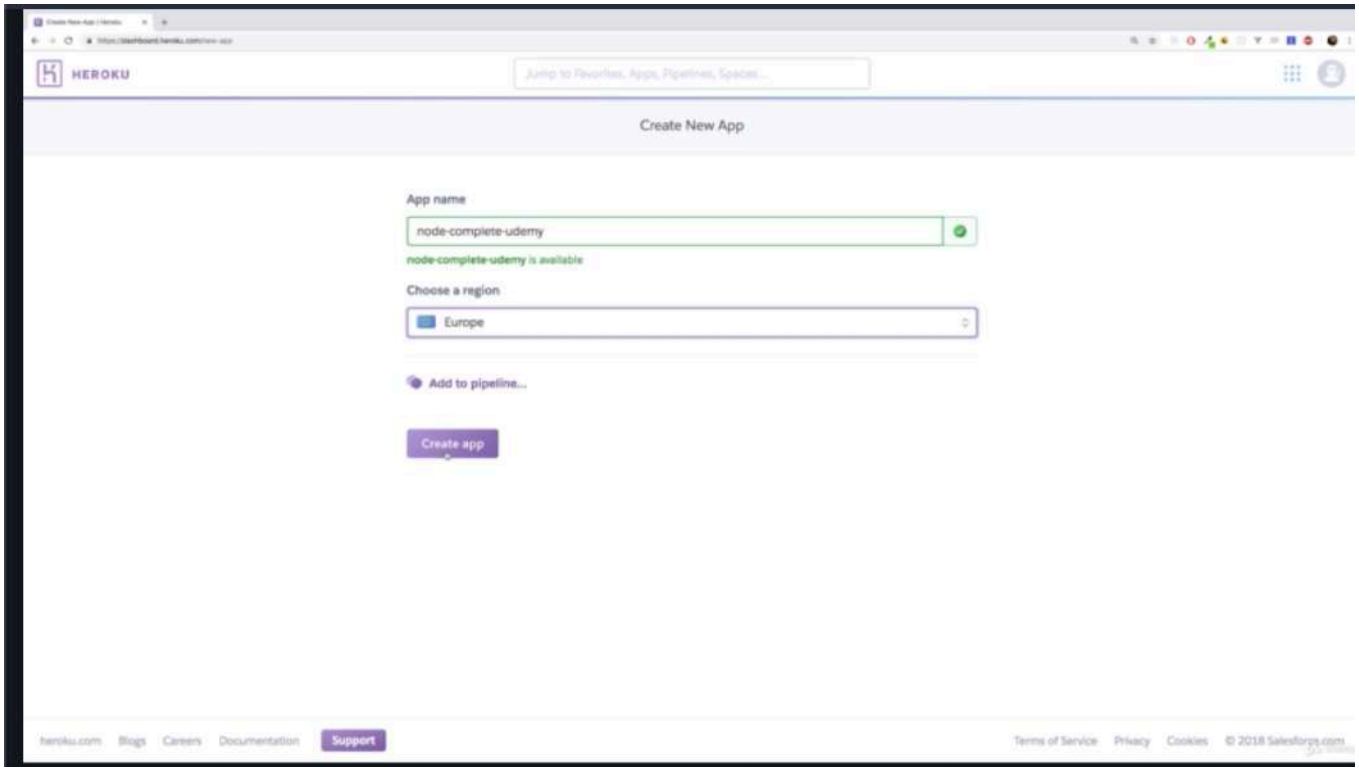
- i will use a hosting provider called ‘Heroku’.



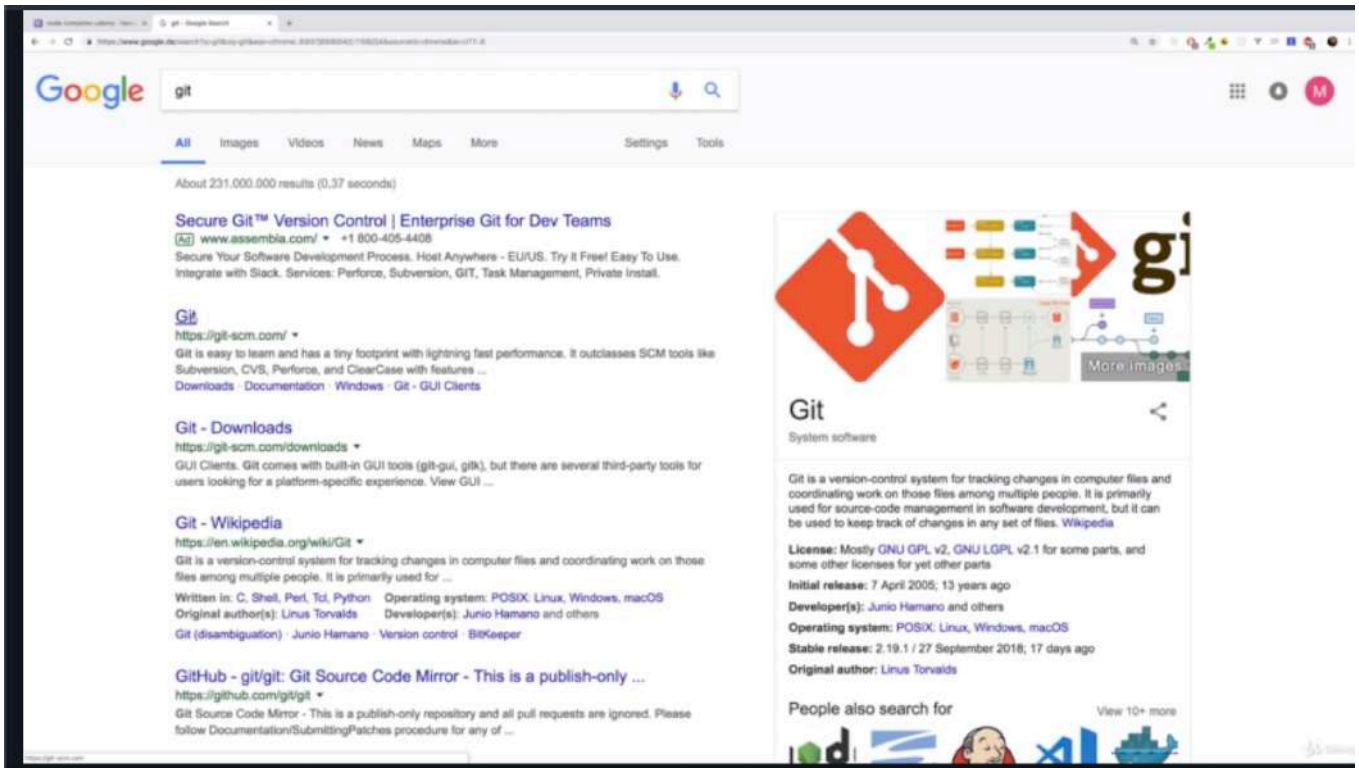
- virtual server means that these providers have very large and powerful machines in their data center and you don’t rent an entire machine. your code runs totally separated from the averages apps which might be running on the same computer on the same server.
- ‘Managed Servers’ is invisible to you which you don’t configure.
- this all runs in a privagte network which means that your own server and your code is not directly exposed to the web but it’s exposed to that managed server which then in turn talks to the web and they offer to your users througha public server gateway. and that is like a door where requests can come in there and then forward it to your server to your virtual server.

* Chapter 454: Understanding The Project & The Git Setup

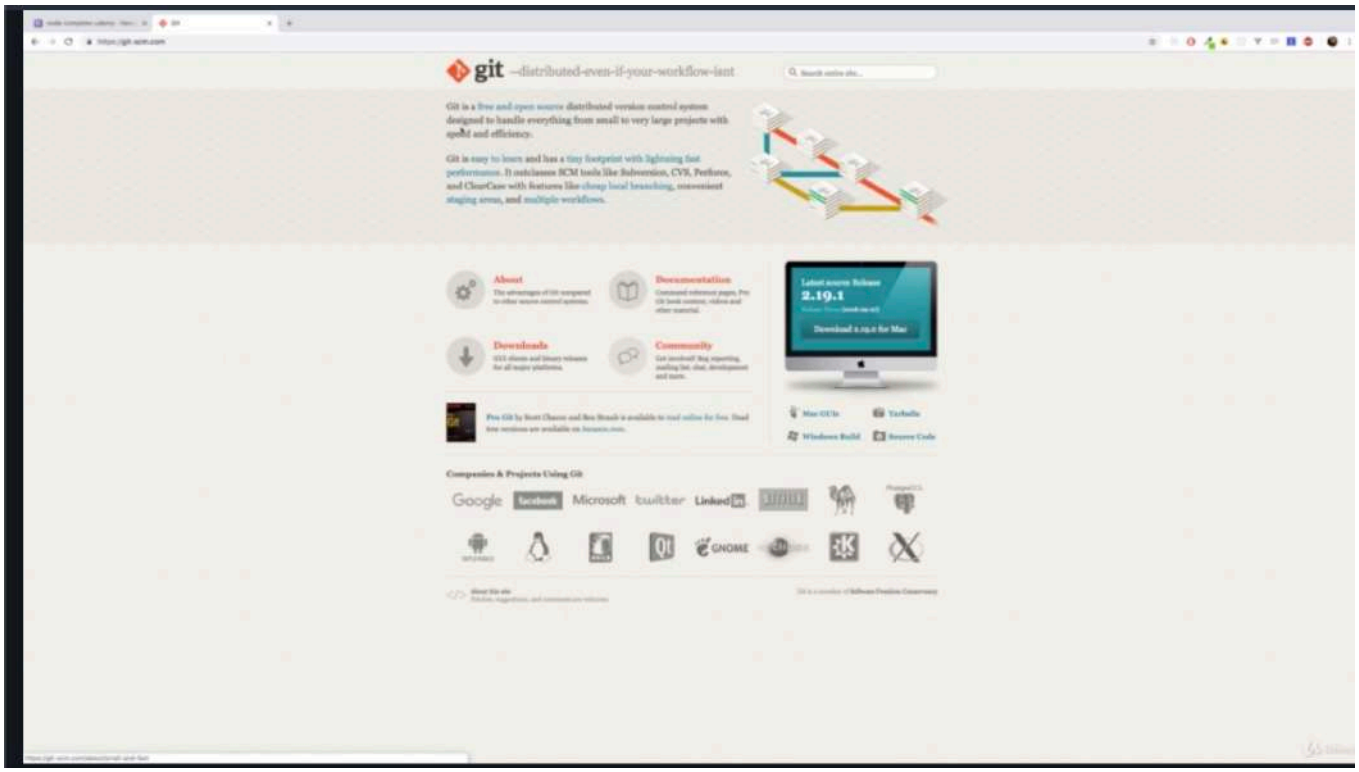


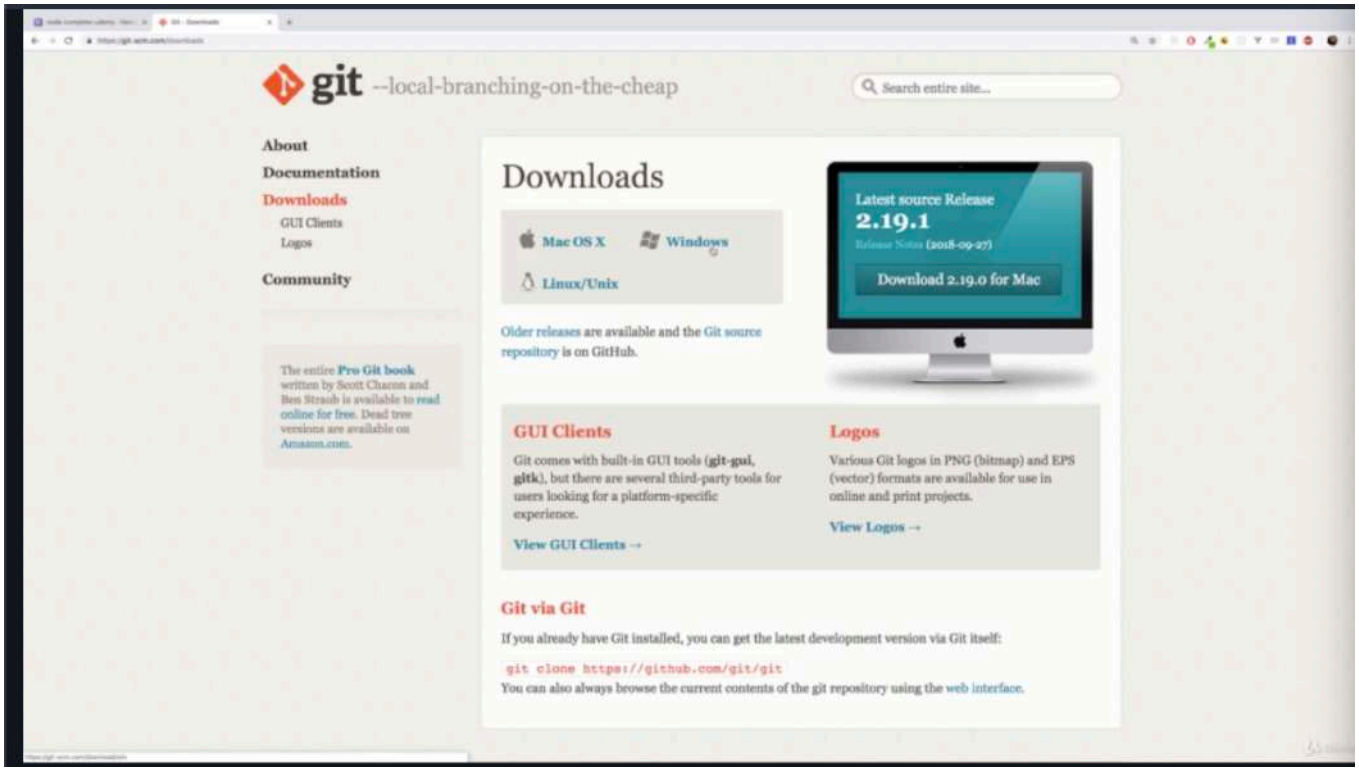


- first of all, we can ignore the pipeline feature here. the deployment method will use 'Heroku Git'. Git is a tool which is not part of Heroku but used by Heroku.



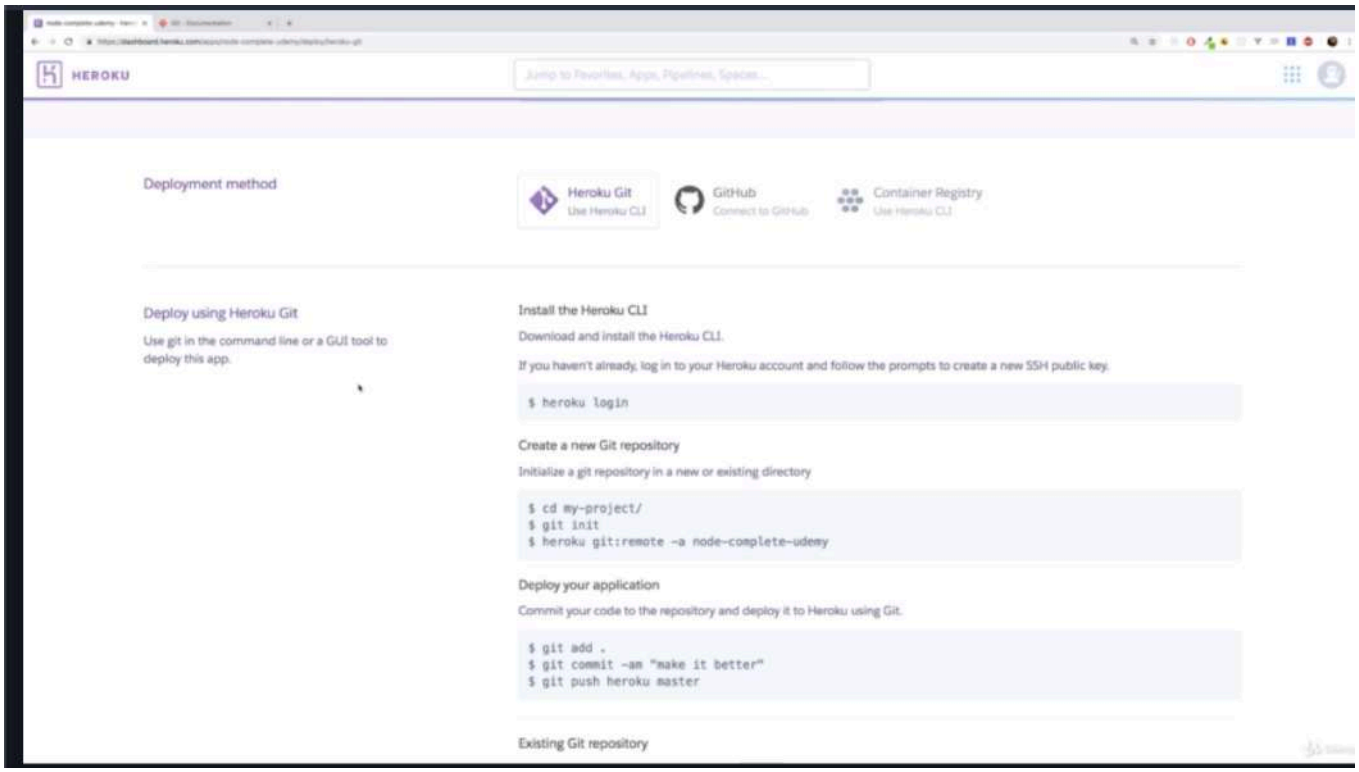
- Git is a version control system and it's totally optional but it helps a lot with saving and managing your source code.
- it allows you to work with so-called commits branches and remote repositories to name some of its most important features
- commits are basically snapshots of your code which you can take. but when you can always switch so you can go back to the older version of your code and have a look at it and then go back to your most recent one or rollback to an older commit. this allows you to revert to older snapshots easily or safely added to code because you can always go back. you can create commit after bug fixes, new features and so on.
- branches also allow you to not just have one history of snapshots but multiple histories for different version of your app. so you could have the master project where your production ready cde is in and then you wanna fix bugs or add new features in other branches so that your main code is untouched. but when a new feature done you can do something which is called 'merging' and merge the new feature branch with your main bransh so that you have one branch which you can put back into production again. but it allows you to work on different features in different branches without affecting your main finished code for now.
- remote repositories means that your code is not only stored locally as it is by default but that you can store it and it's commit and braches in the cloud and that means you can protect against loss of local data nad you can also access your source code from different machines and share it with average developers and you can use that feature to deploy your code automatically. that is what Heroku does because you will use Heroku as a remote repository which means when you push your code to that remote repository to Heroku, it will then be takenn by Heroku, and it will be put into production and a server will be spun up based on it automatically.



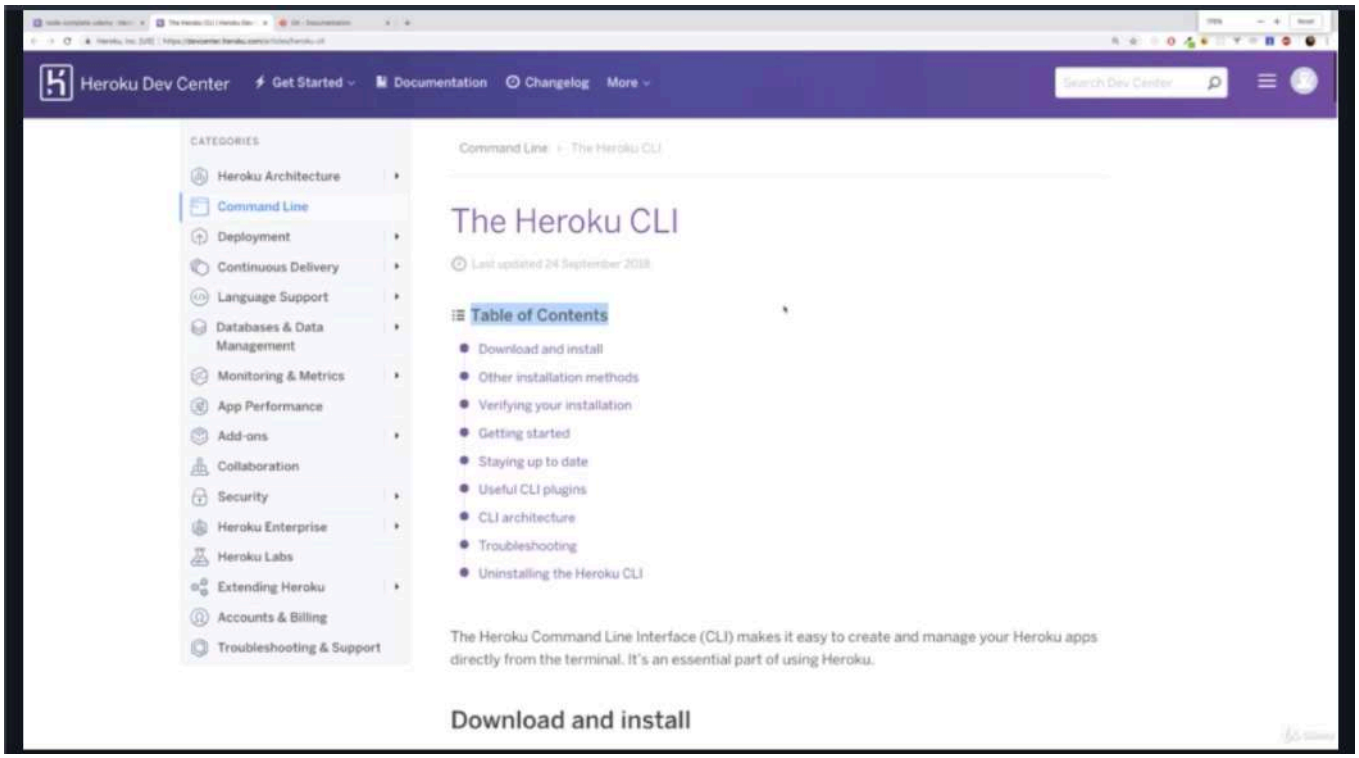


* Chapter 455: A Deployment Example With Heroku

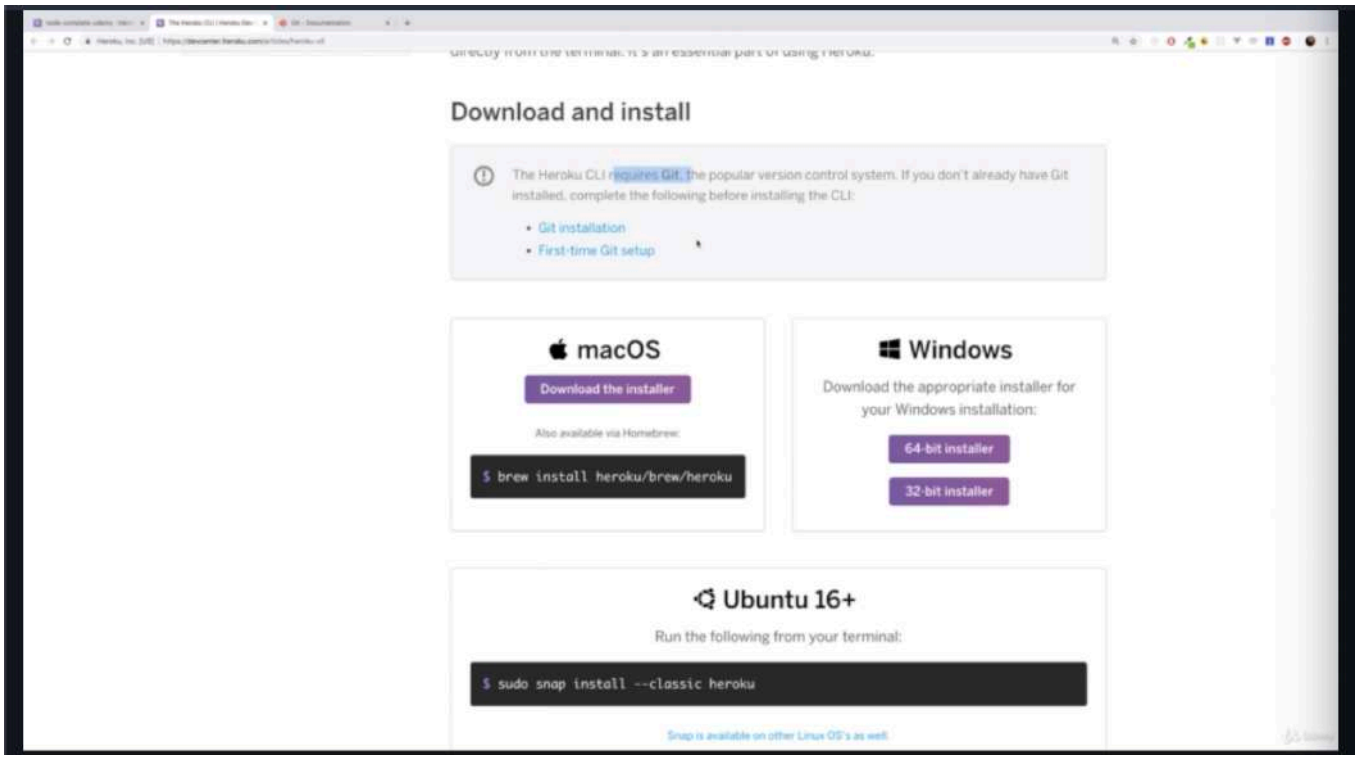
1. update
- package.json
- app.js
- Procfile
- .gitignore

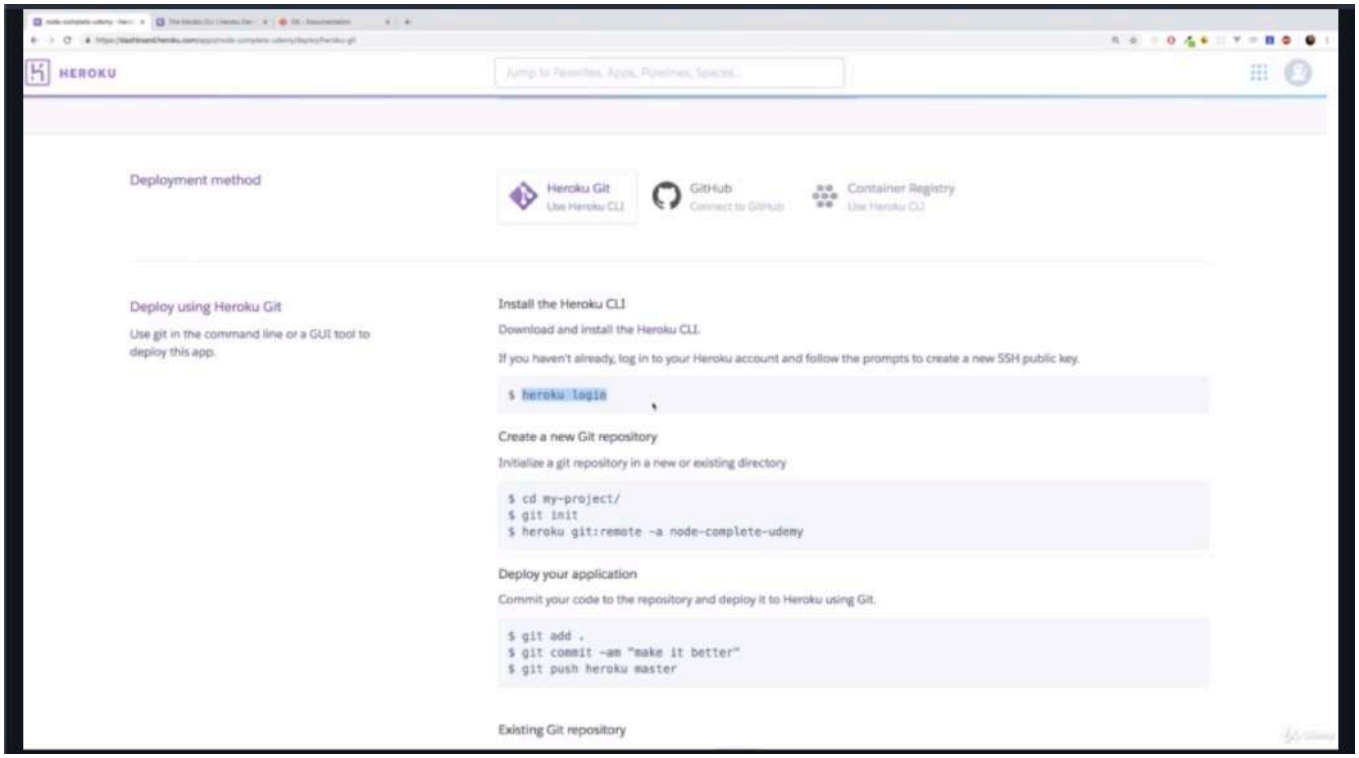


- the next steps on Heroku to use its CLI it's command line interface. and that is Heroku specific on our hosting providers. you might simply be able to drag and drop your code. but Heroku doesn't use such a drag and drop alternative instead Heroku uses that command line interface which allows you to run your code or to deploy your code. the command line by typing on commands.



- make sure to follow the instructions on the page.now i will go with my Mac OS version and i will quickly walk through the installer there and install Heroku on my system. click 'Download the Installer'





- once you installed it, the next step is to run this command and you run this in your normal terminal or command prompt.

EXPLORER

app.js

NODEJS-COMPLETE-GUIDE

- .vscode
- controllers
- data
- images
- middleware
- models
- node_modules
- public
- routes
- util
- views
- .gitignore
- access.log
- app.js
- message.txt
- nodemon.json
- package-lock.json
- package.json

```
124
125  app.get('/500', errorController.get500);
126
127  app.use(errorController.get404);
128
129  app.use((error, req, res, next) => {
130    // res.status(error.httpStatusCode).render(...);
131    // res.redirect('/500');
132    res.status(500).render('500', {
133      pageTitle: 'Error!',
134      path: '/500',
135      isAuthenticated: req.session.isLoggedIn
136    });
137  });
138
139  mongoose
140    .connect(MONGODB_URI)
141    .then(result => {
142      // https
143      //   .createServer({ key: privateKey, cert: certificate }, app)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ heroku login
heroku: Enter your login credentials
Email: maximilian
```

OUTLINE

28-deployment-classic* 0 0 Ln 41, Col 3 Spaces: 2 UTF-8 LF JavaScript Prettier

HEROKU

Deploy using Heroku Git

Use git in the command line or a GUI tool to deploy this app.

Install the Heroku CLI

Download and install the Heroku CLI.

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Create a new Git repository

Initialize a git repository in a new or existing directory

```
$ cd my-project/
$ git init
$ heroku git:remote -a node-complete-udemy
```

Deploy your application

Commit your code to the repository and deploy it to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

Existing Git repository

For existing repositories, simply add the `heroku` remote

```
$ heroku git:remote -a node-complete-udemy
```

heroku.com Blogs Careers Documentation Support Terms of Service Privacy Cookies © 2018 Salesforce.com

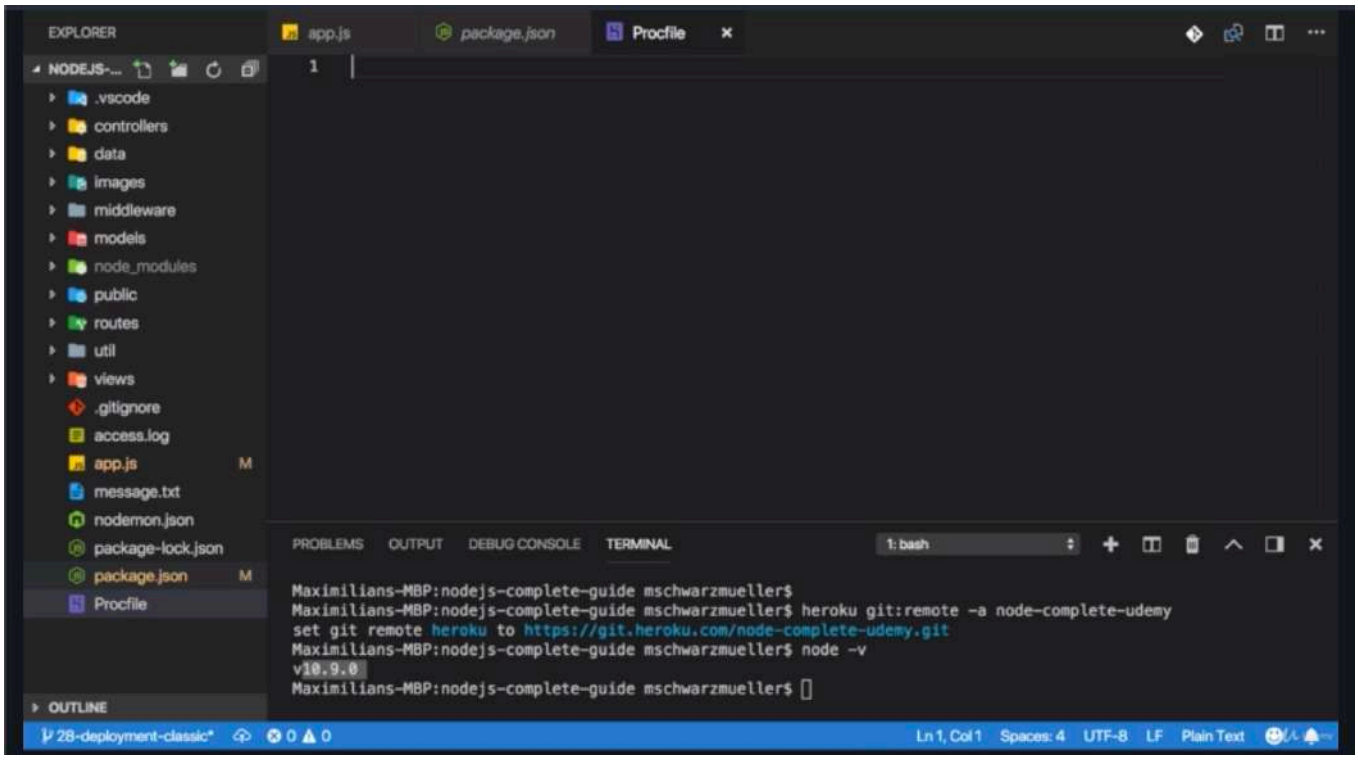
```
app.get('/500', errorController.get500);

app.use(errorController.get404);

app.use((error, req, res, next) => {
  // res.status(error.httpStatusCode).render(...);
  // res.redirect('/500');
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
});

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    // https
    //   .createServer({ key: privateKey, cert: certificate }, app)
```

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ heroku git:remote -a node-complete-udemy
set git remote heroku to https://git.heroku.com/node-complete-udemy.git
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```

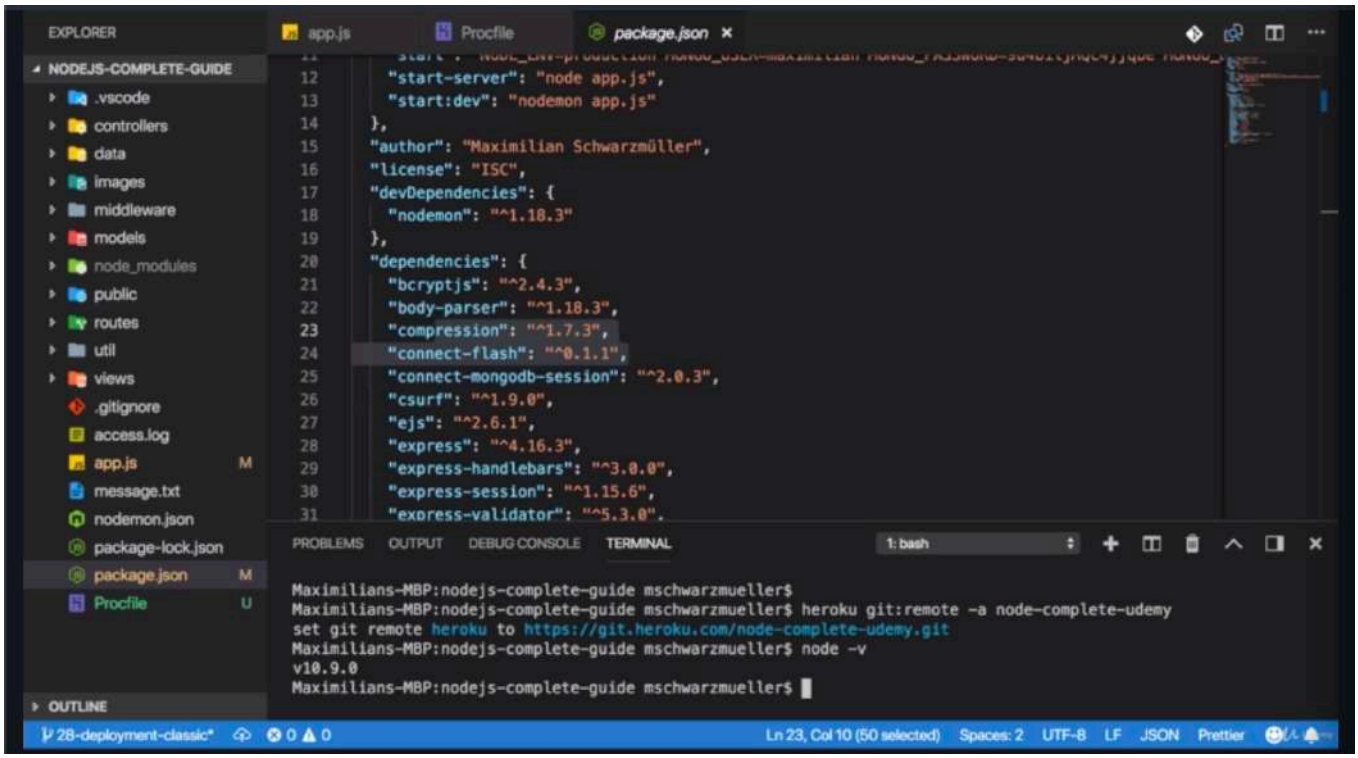
- next in your package.json and follow of your project, you wanna add a new entry ‘engines’ entry anywhere maybe above the scripts and there add “node” and now set the version of node you are using. you can detect that by running ’node -v’ in your project folder or anywhere on your computer.

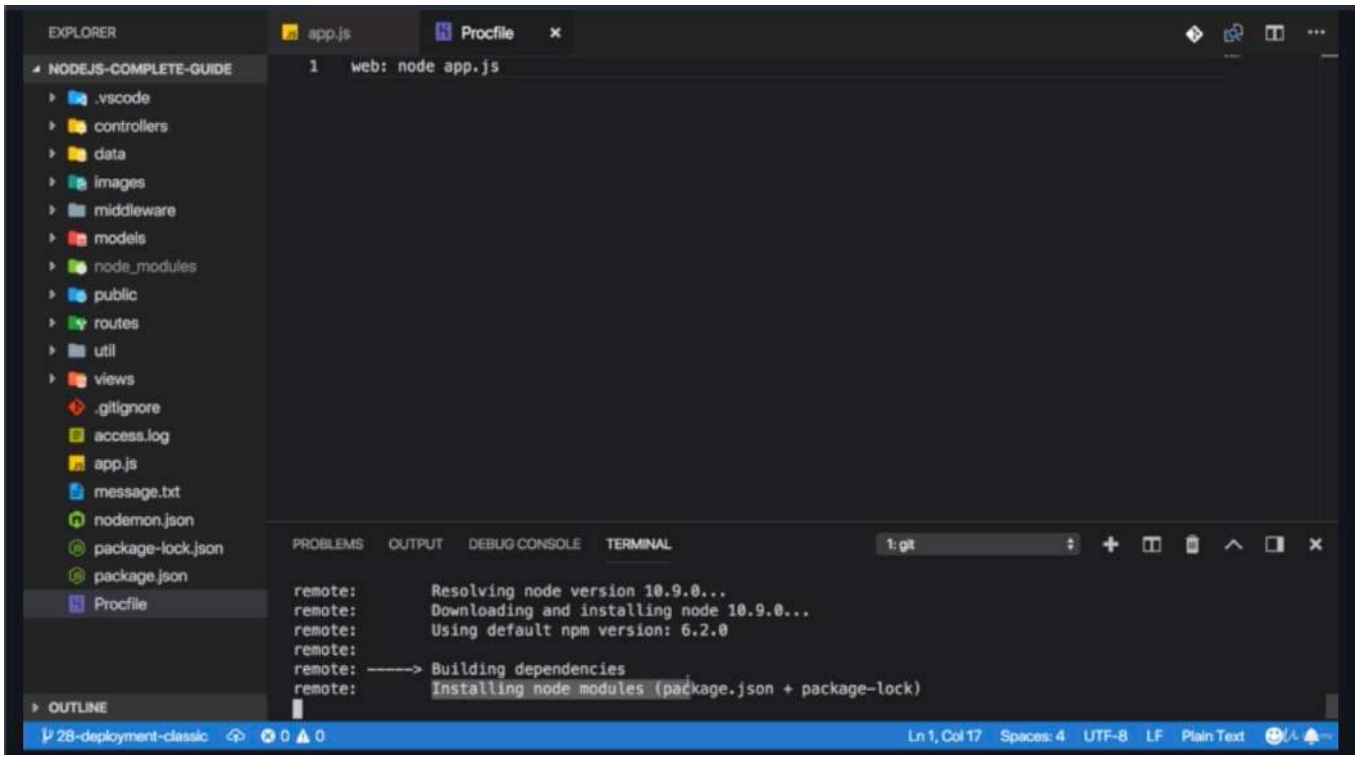

```
{
  "name": "nodejs-complete-guide",
  "version": "1.0.0",
  "description": "Complete Node.js Guide",
  "main": "app.js",
  "engines": {
    "node": "10.9.0"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "NODE_ENV=production MONGO_USER=maximilian MONGO_PASSWORD=9u4biljMQc4jjqbe MONGO_D
    "start-server": "node app.js",
    "start:dev": "nodemon app.js"
  },
  "author": "Maximilian Schwarzmüller",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^1.18.3"
  },
  "dependencies": {
```

```
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ heroku git:remote -a node-complete-udemy
set git remote heroku to https://git.heroku.com/node-complete-udemy.git
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$ node -v
v10.9.0
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```

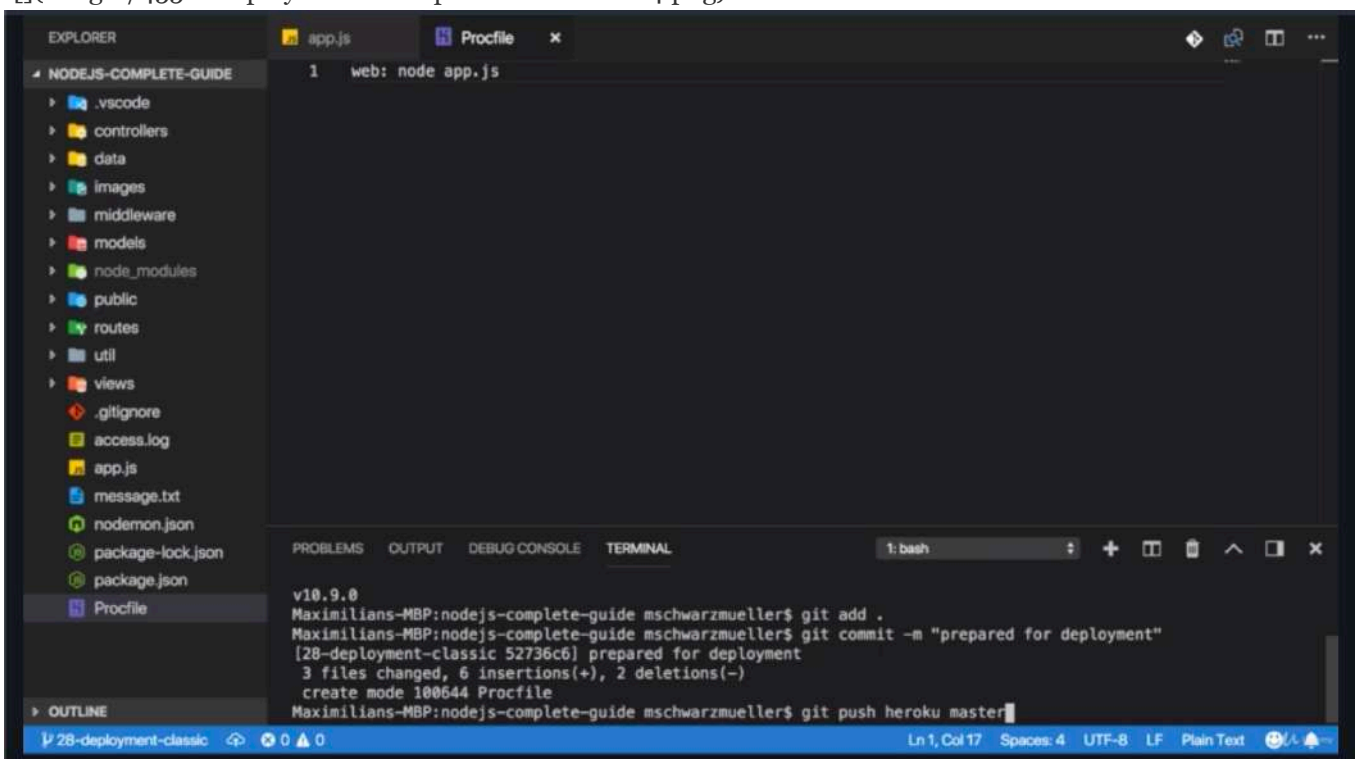


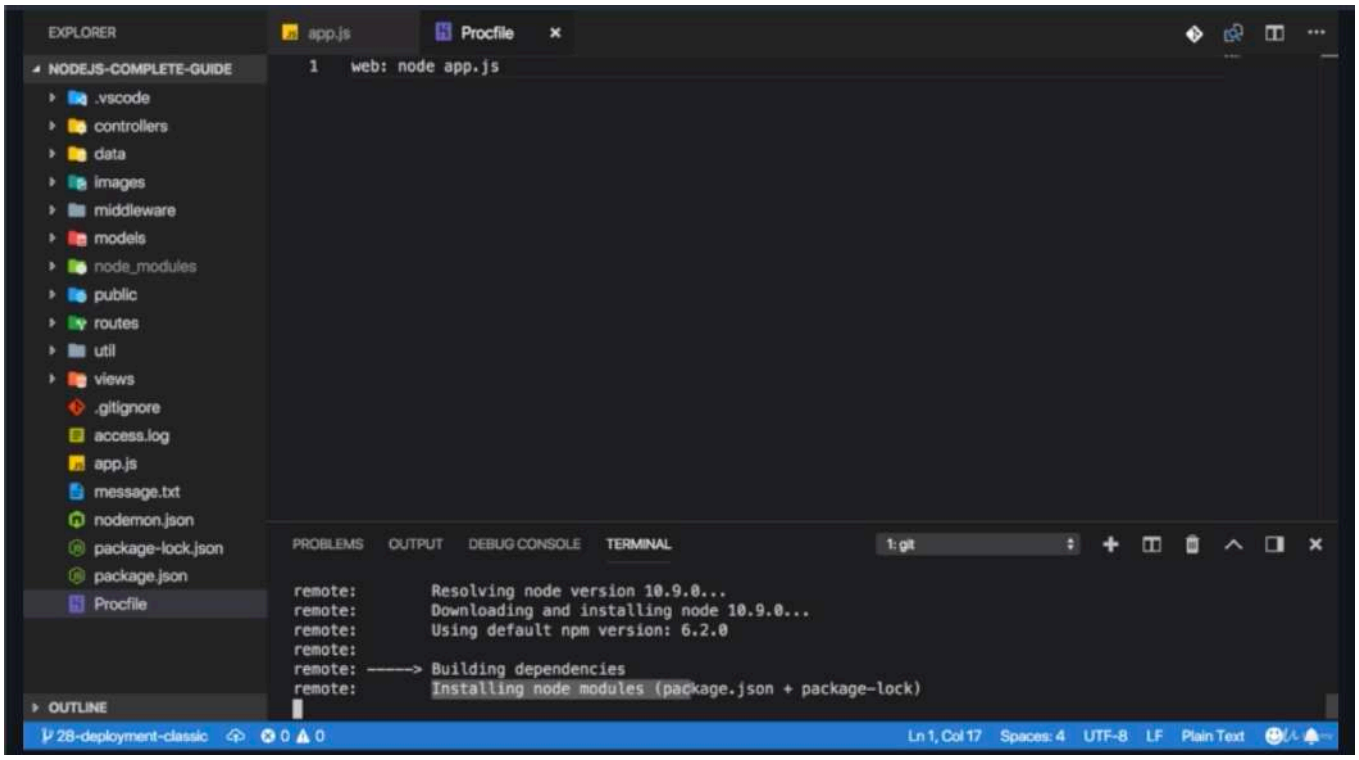
- 'Procfile' is Heroku specific. Procfile without a file extension and you add 'web: node app.js' which will instruct Heroku to execute your app.js file when it tries to run your application.

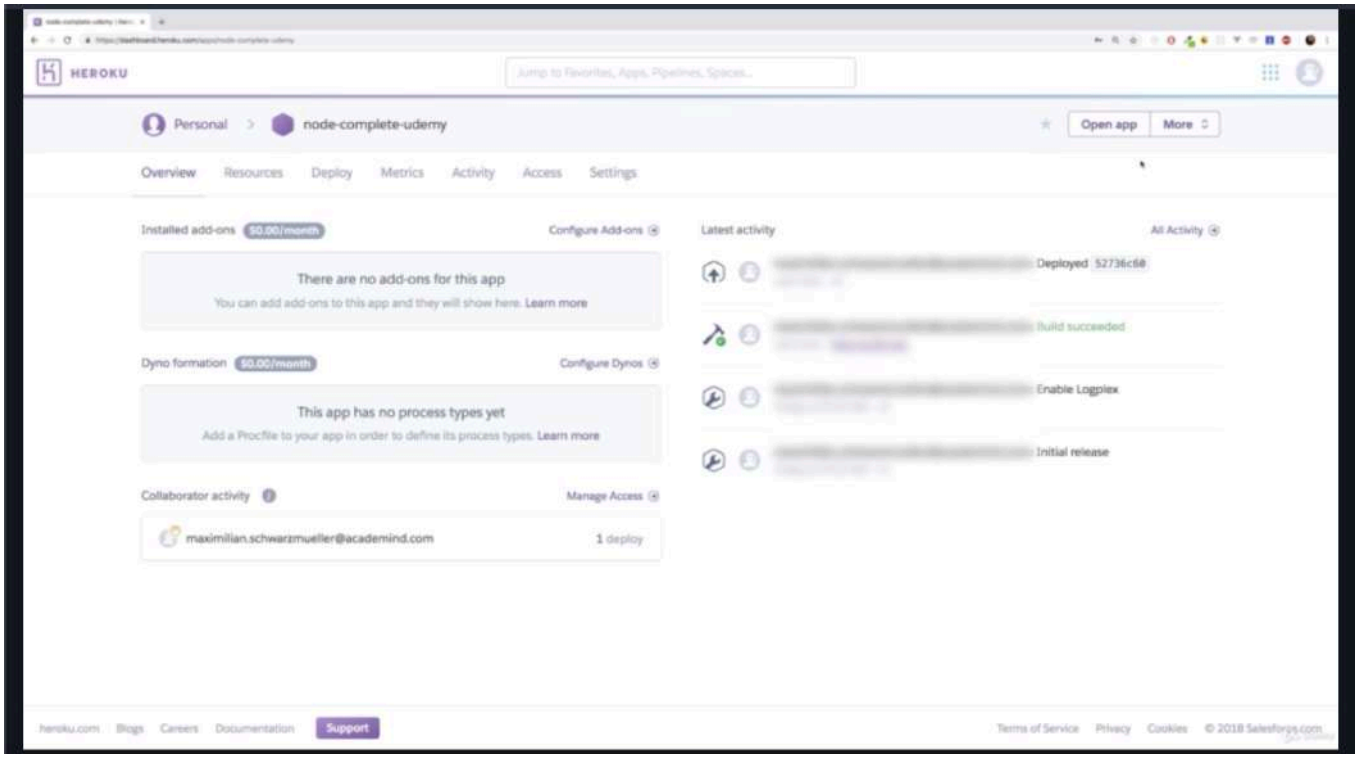




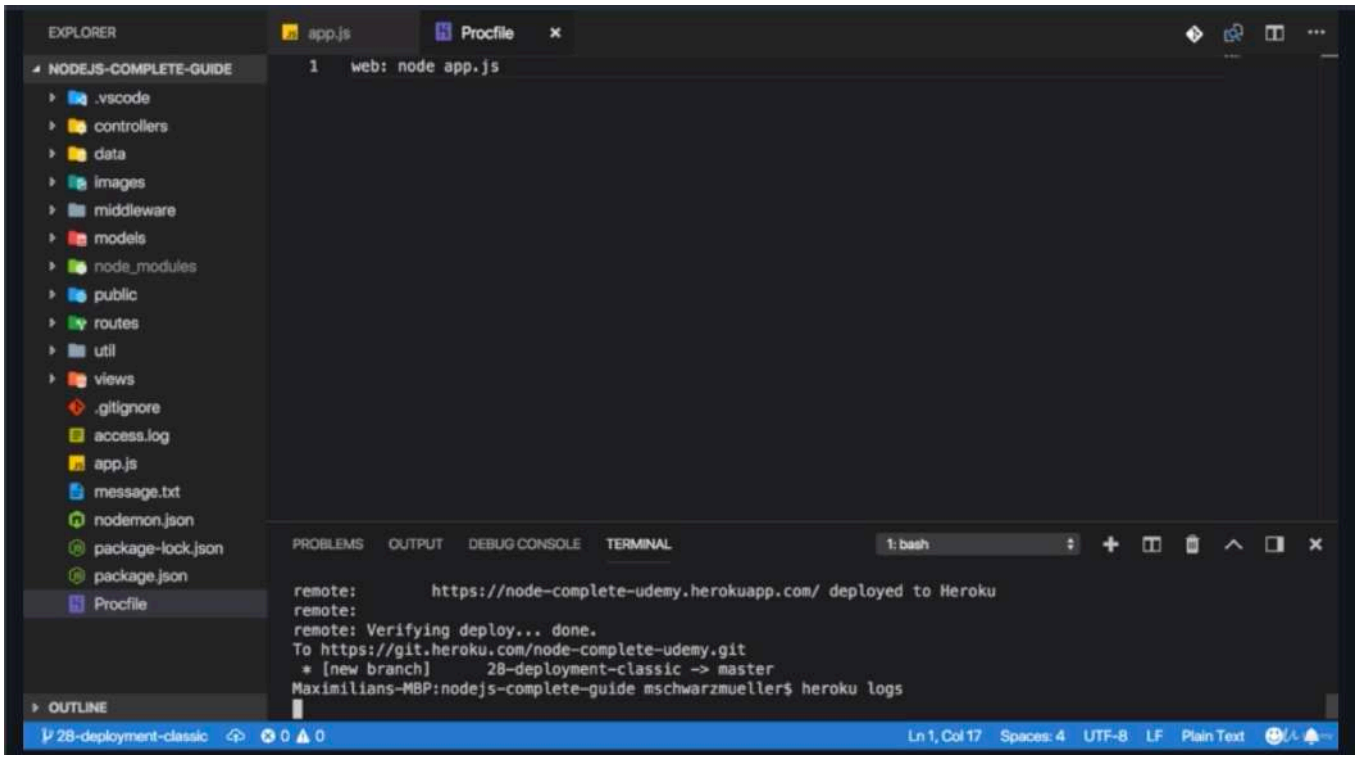
- .gitignore will tell git which folders it should not include in its snapshots and here the node_modules folder is important. all your 3rd party package are stored there. and we won't deploy that. because that will just increase the size of data we have to transmit over the wire. instead this will automatically be recreated on the Heroku because Heroku and that is the case for the hosting providers do install your dependencies on the server after you deploy your code. remember package.json. we have a list of all the 3rd party packages we are using and the words we need. so package.json will be taken by the hosting provider and it will install all these packages on the server. that is why we always use 'npm install —save' because that added such entries to the packages.json file which can be used during the deployment.

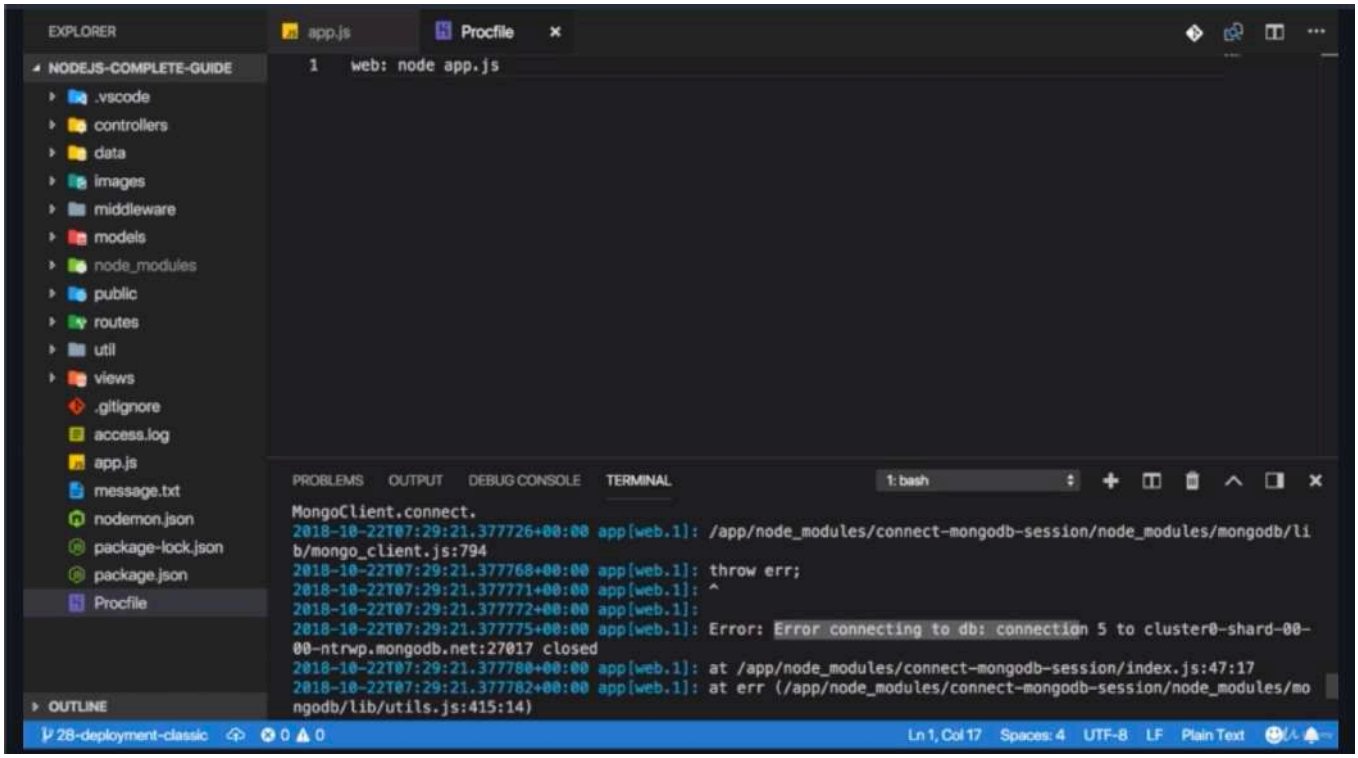






- after all installed, with that you can go back to your dashboard and click on overview and you should see that 'build succeed' and you can now click on 'Open app'. this will open your app in a new tab and most likely this will not really succeed.





- and the reason for that is that we deployed our application but now if you type 'heroku logs' in your terminal, you will see what went wrong. and to be percise and the error message you will see that it failed to connect to the database and that make sense

EXPLORER

NODEJS-COMPLETE-GUIDE

- .vscode
- controllers
- data
- images
- middleware
- models
- node_modules
- public
- routes
- util
- views
- .gitignore
- access.log
- app.js
- message.txt
- nodemon.json
- package-lock.json
- package.json
- Procfile

app.js × Procfile

```
const multer = require('multer');
const helmet = require('helmet');
const compression = require('compression');
const morgan = require('morgan');

const errorController = require('./controllers/error');
const shopController = require('./controllers/shop');
const isAuth = require('./middleware/is-auth');
const User = require('./models/user');

const MONGODB_URI = `mongodb+srv://${process.env.MONGO_USER}:${
  process.env.MONGO_PASSWORD                    string
}@cluster0-ntrwp.mongodb.net/${process.env.MONGO_DEFAULT_DATABASE}`;

const app = express();
const store = new MongoDBStore({
  uri: MONGODB_URI,
  collection: 'sessions'
});
const csrfProtection = csrf();

// const privateKey = fs.readFileSync('server.key');
// const certificate = fs.readFileSync('server.cert');
```

OUTLINE

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

```
2018-10-22T07:29:23.320361+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/favico
n.ico" host=node-complete-udemy.herokuapp.com request_id=d528d525-7138-411a-9c7e-7d0acf515f9f fwd="188.174.117
.155" dyno= connect= service= status=503 bytes= protocol=https
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```

28-deployment-classic ⊗ 0 ⚠ 0 Ln 24, Col 67 (23 selected) Spaces: 2 UTF-8 LF JavaScript Prettier

EXPLORER

NODEJS-COMPLETE-GUIDE

- .vscode
- controllers
- data
- images
- middleware
- models
- node_modules
- public
- routes
- util
- views
- .gitignore
- access.log
- app.js
- message.txt
- nodemon.json
- package-lock.json
- package.json
- Procfile

app.js Procfile ×

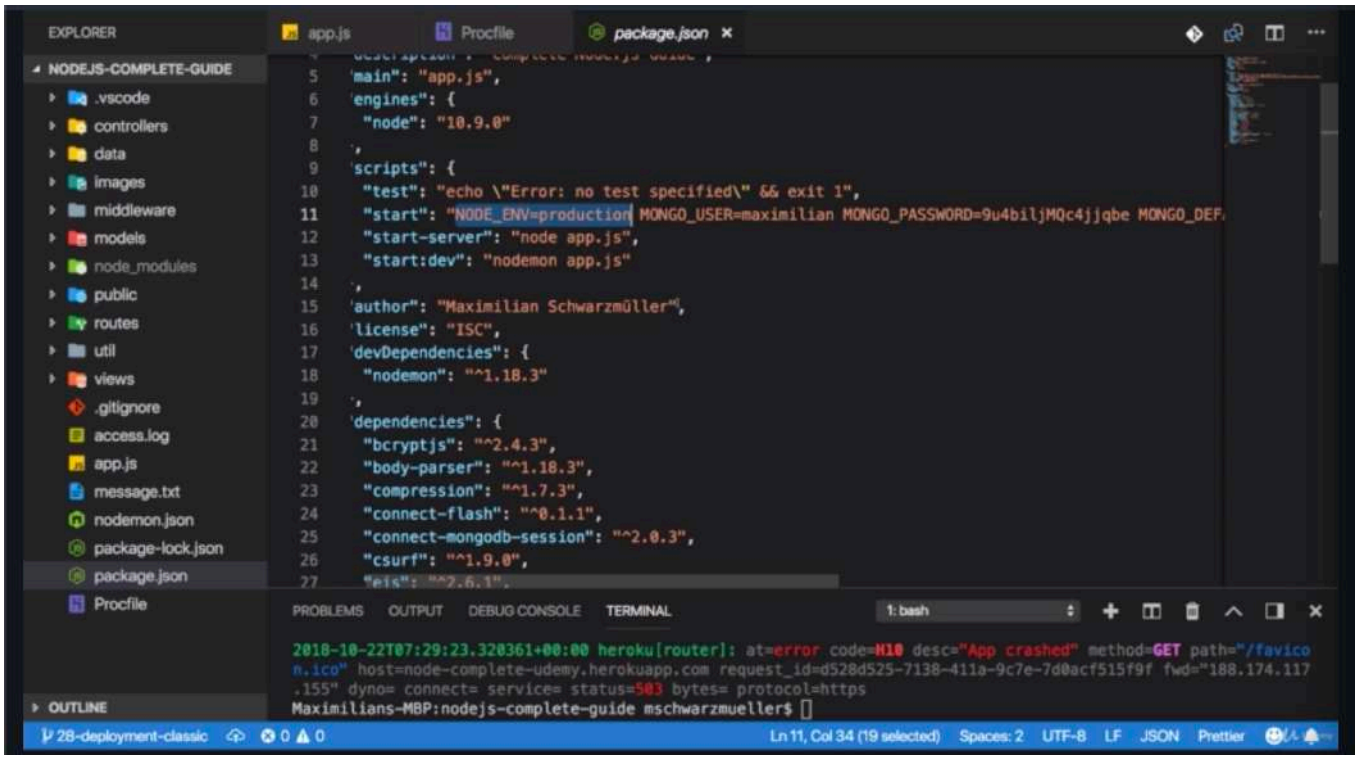
```
web: node app.js
```

OUTLINE

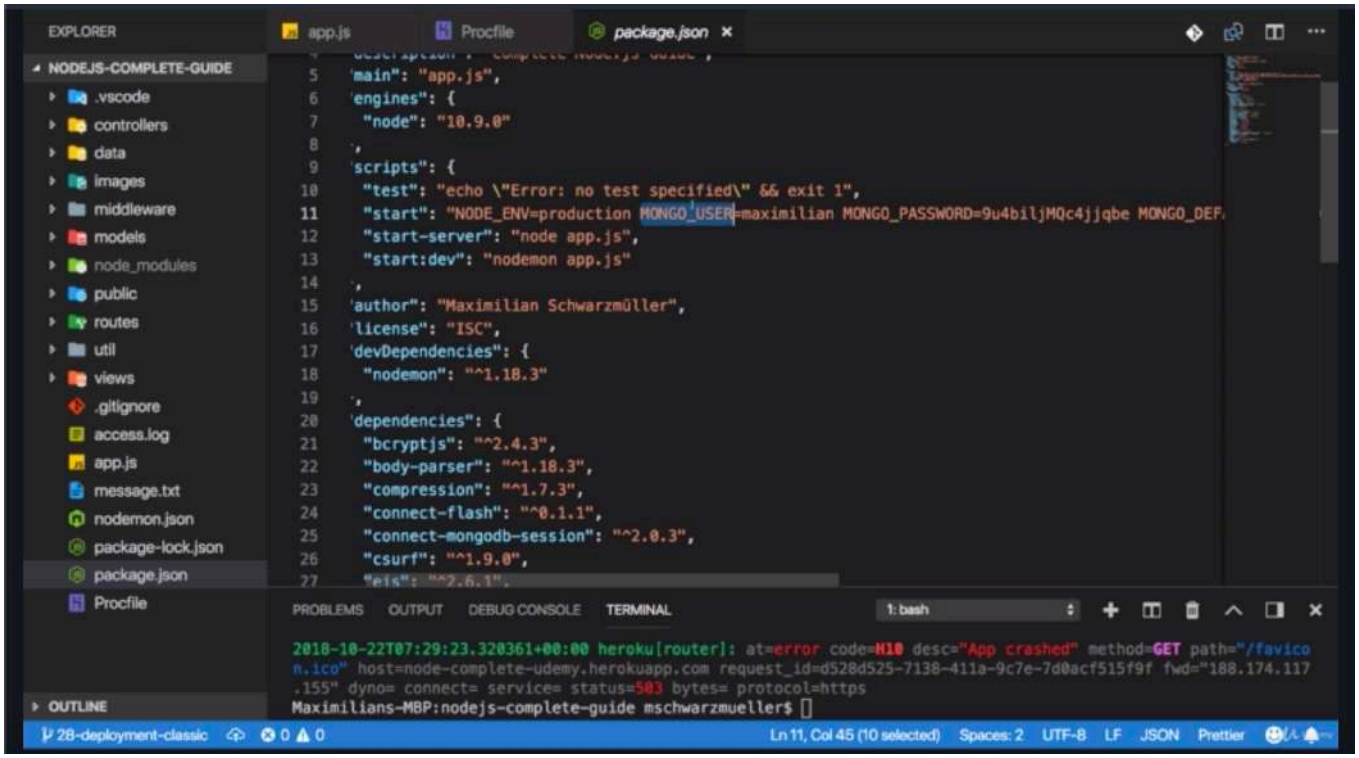
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

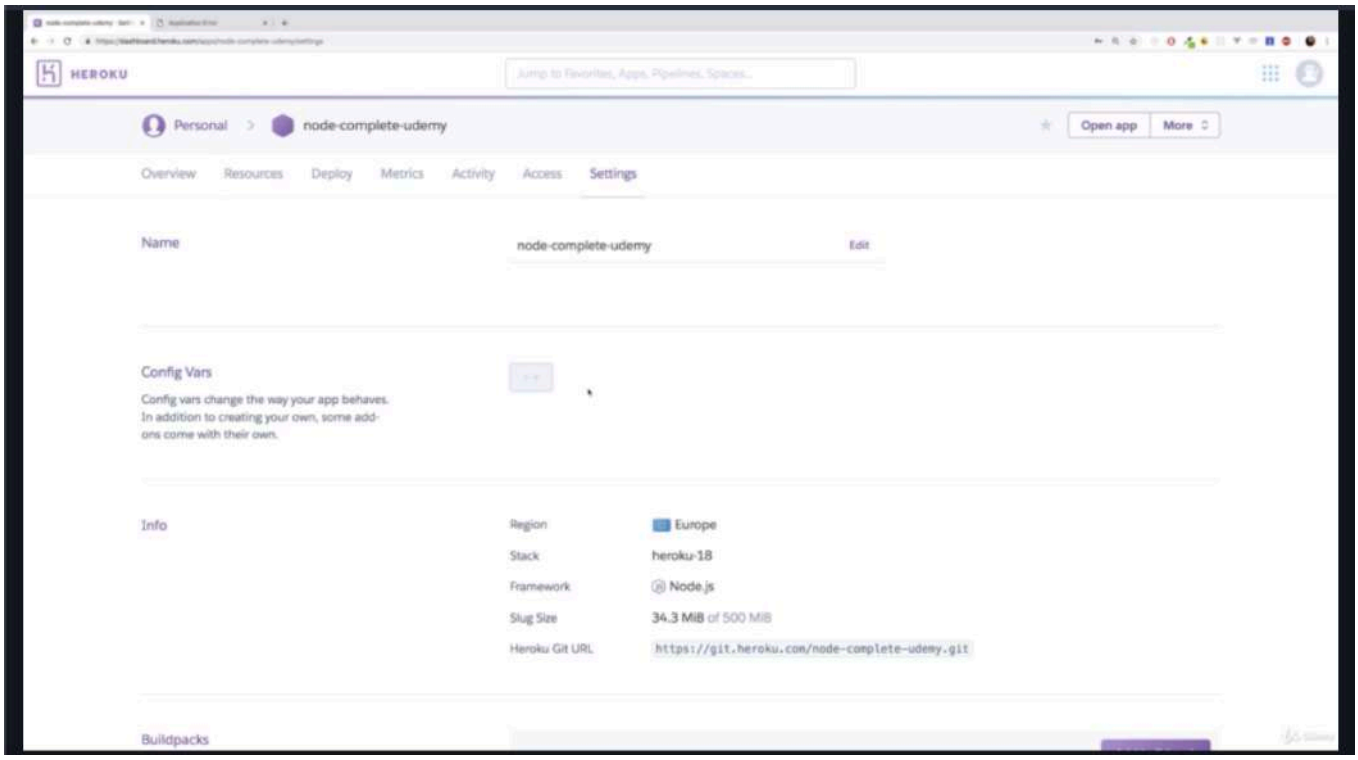
```
2018-10-22T07:29:23.320361+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/favico
n.ico" host=node-complete-udemy.herokuapp.com request_id=d528d525-7138-411a-9c7e-7d0acf515f9f fwd="188.174.117
.155" dyno= connect= service= status=503 bytes= protocol=https
Maximilians-MBP:nodejs-complete-guide mschwarzmueller$
```

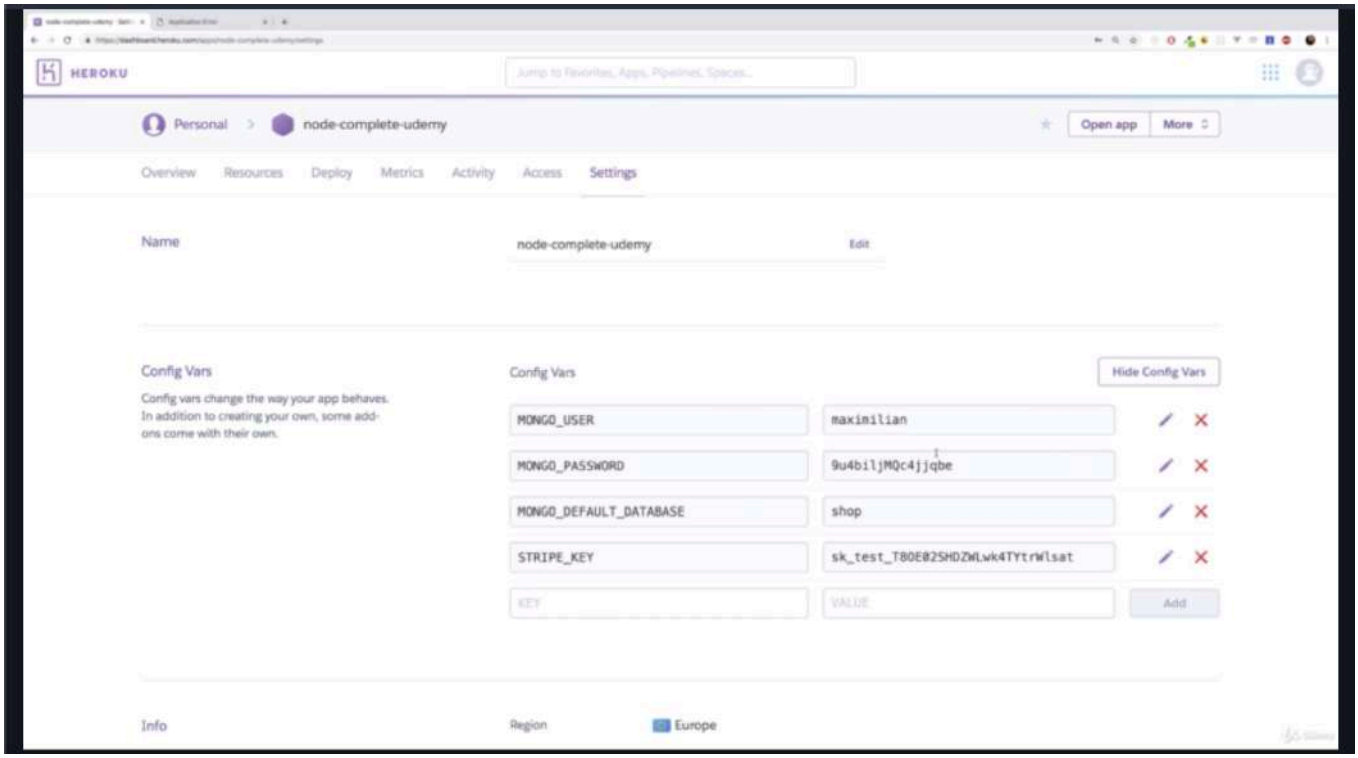
28-deployment-classic ⊗ 0 ⚠ 0 Ln 1, Col 6 (11 selected) Spaces: 4 UTF-8 LF Plain Text



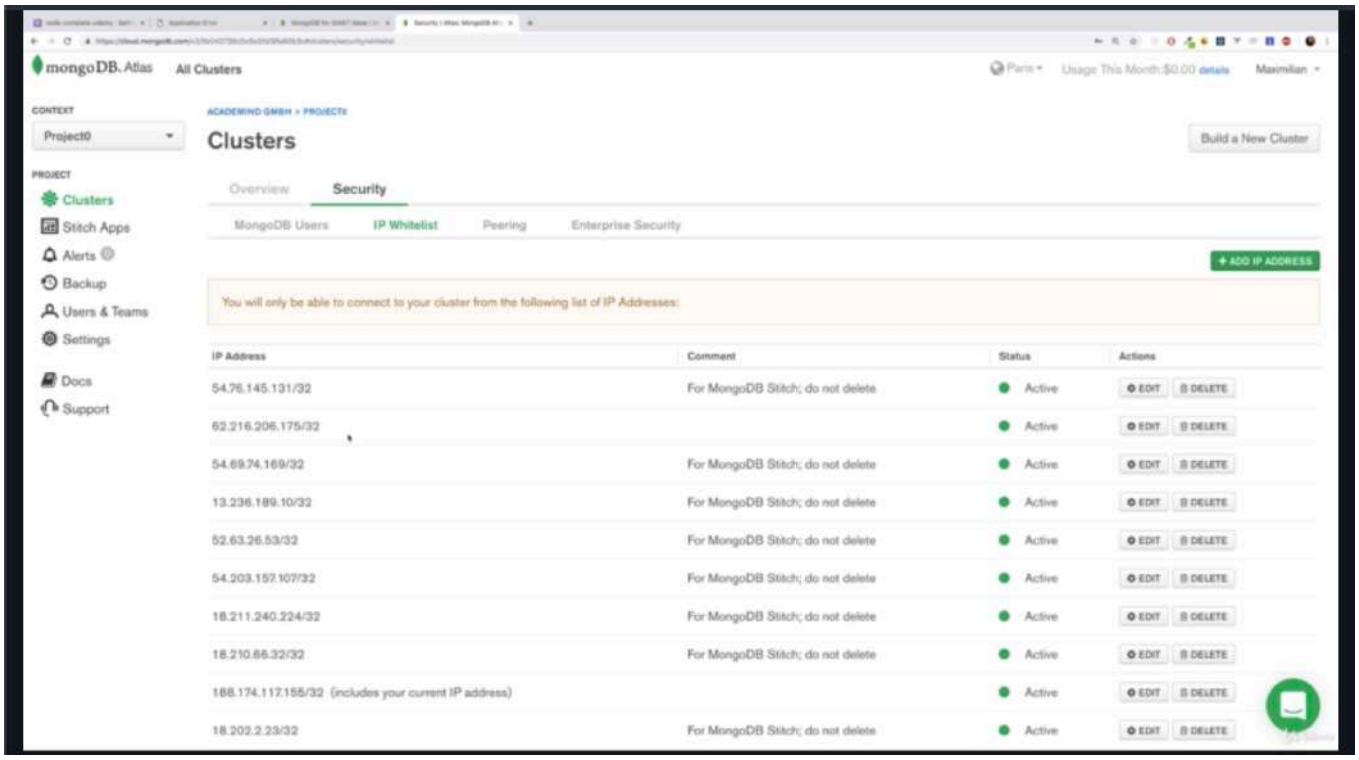
- because all our node environment variables which we rely on for example to connect to the database are now not set anymore because in Procfile, we instruct heroku to just execute the app.js file. this will not pass the environment variables only one environment variable is passed by default by default. by Heroku and that is node. and this is set to production.

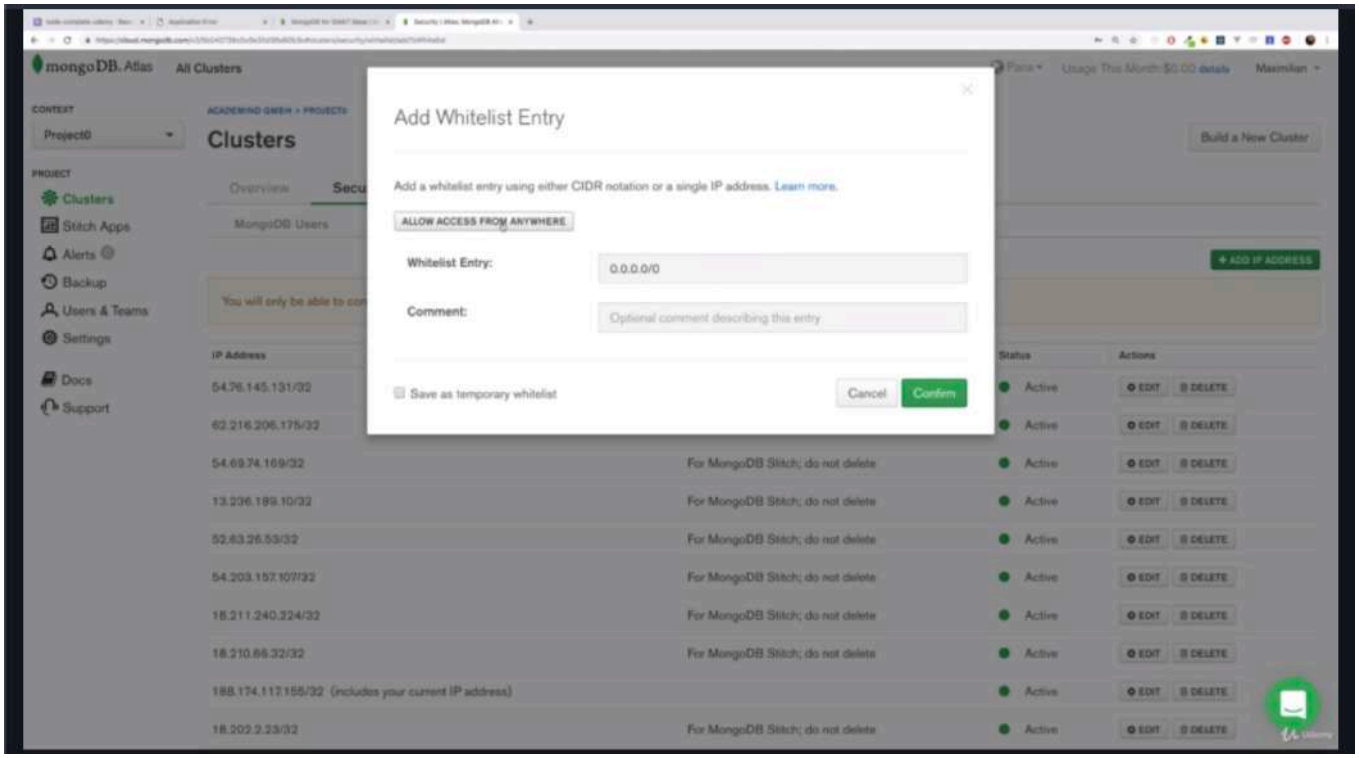




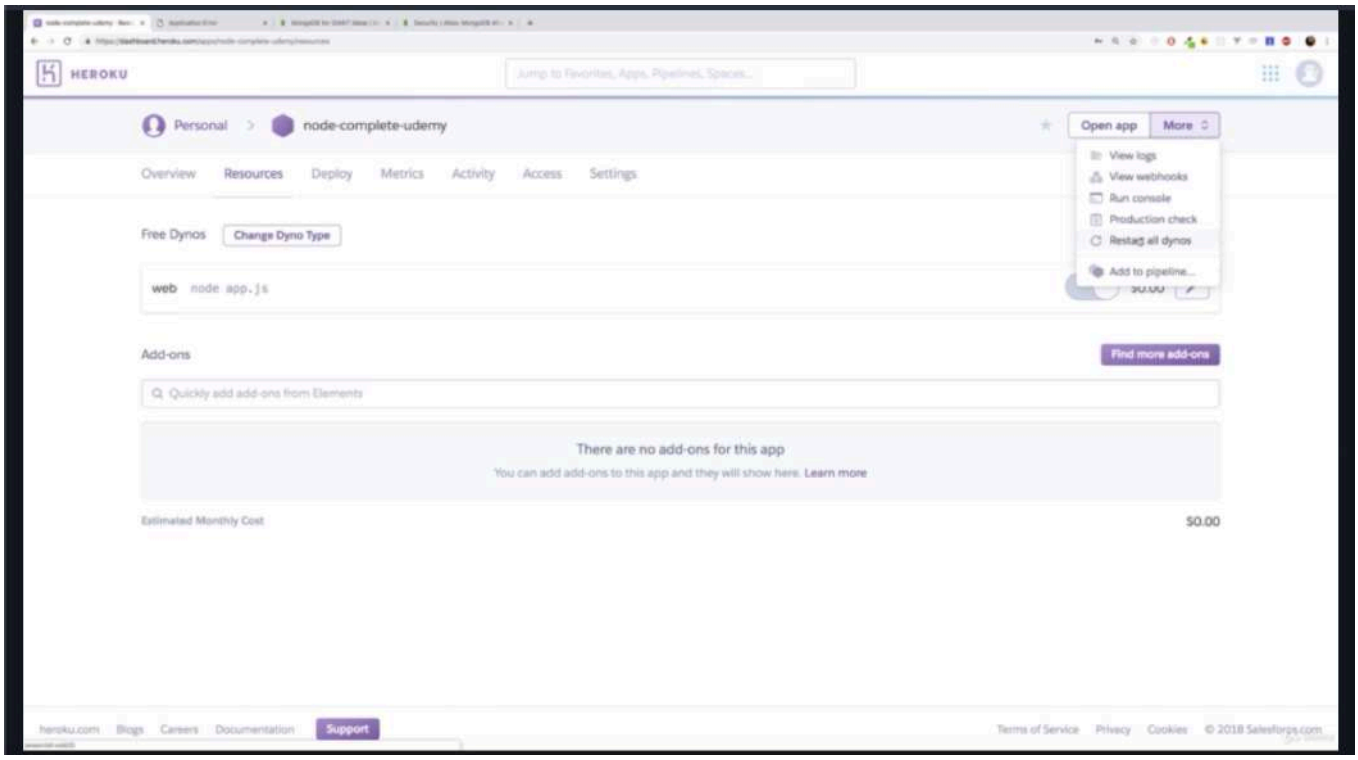


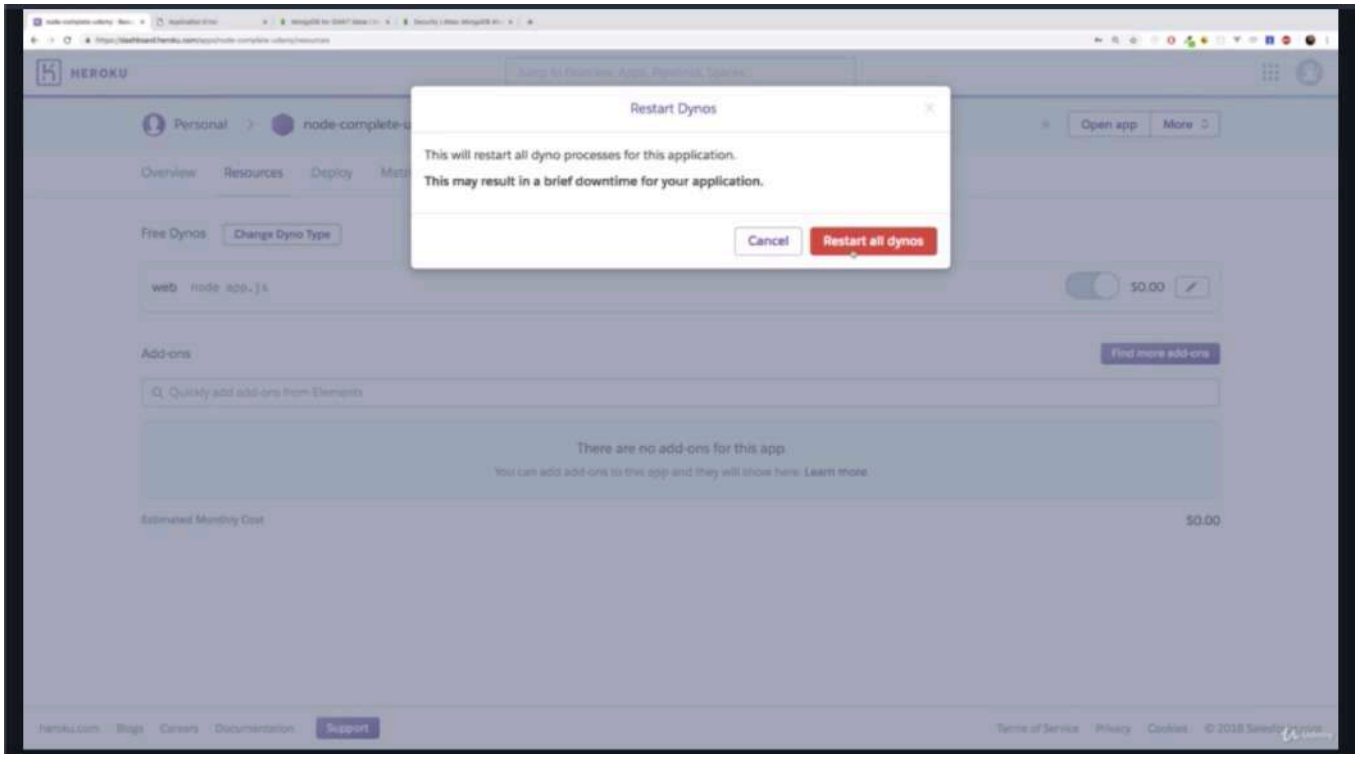
- so now we have to do it on our ownn by simply grabbing 'MONGO_USER' and going back toward dashboard and there on the dashboard, you wanna go to settings for your app and then go to config Vars. and now we have all these environment variables edits here.





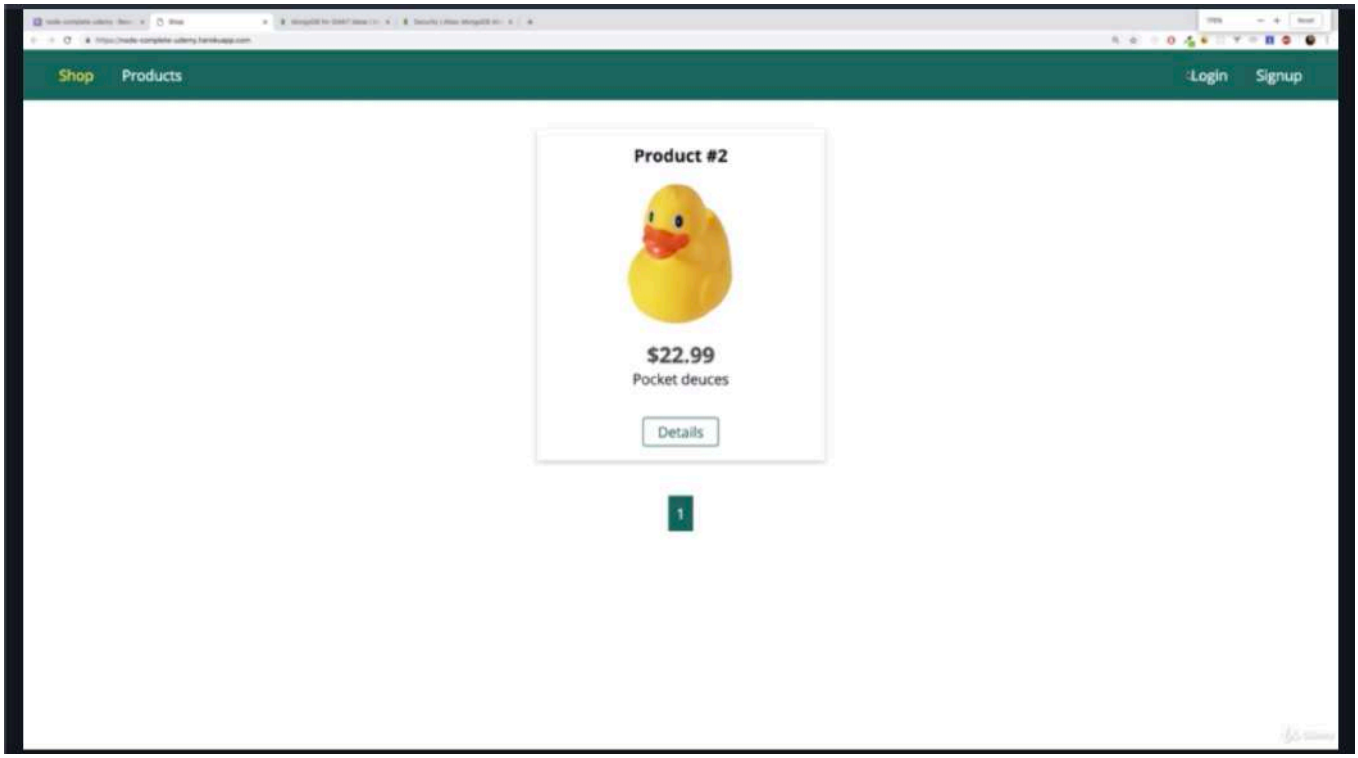
- we also need to change something on our mongoDB set up in our case becasue we are using that hosted mongoDB atlas solution. in atlas and in IP Whitelist, you need to wide list of IP of your running application.
- and the thing about Heroku and its basic version is that we don't have a static IP assigned to our project. instead it's a dynamic range.
- running MongoDB on the same server as your web project is not really an alternative because a database server is all not very trivial to set up especially if it should be able to scale and so on.

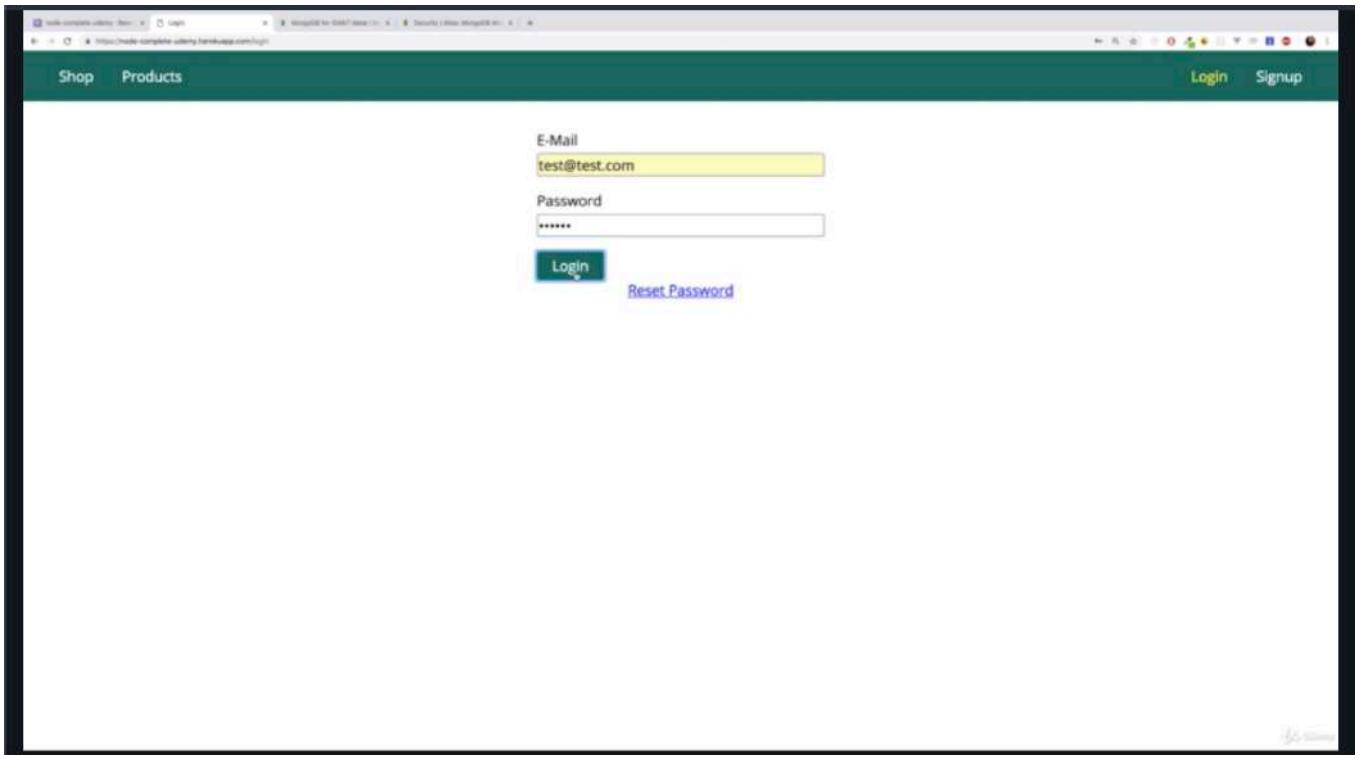


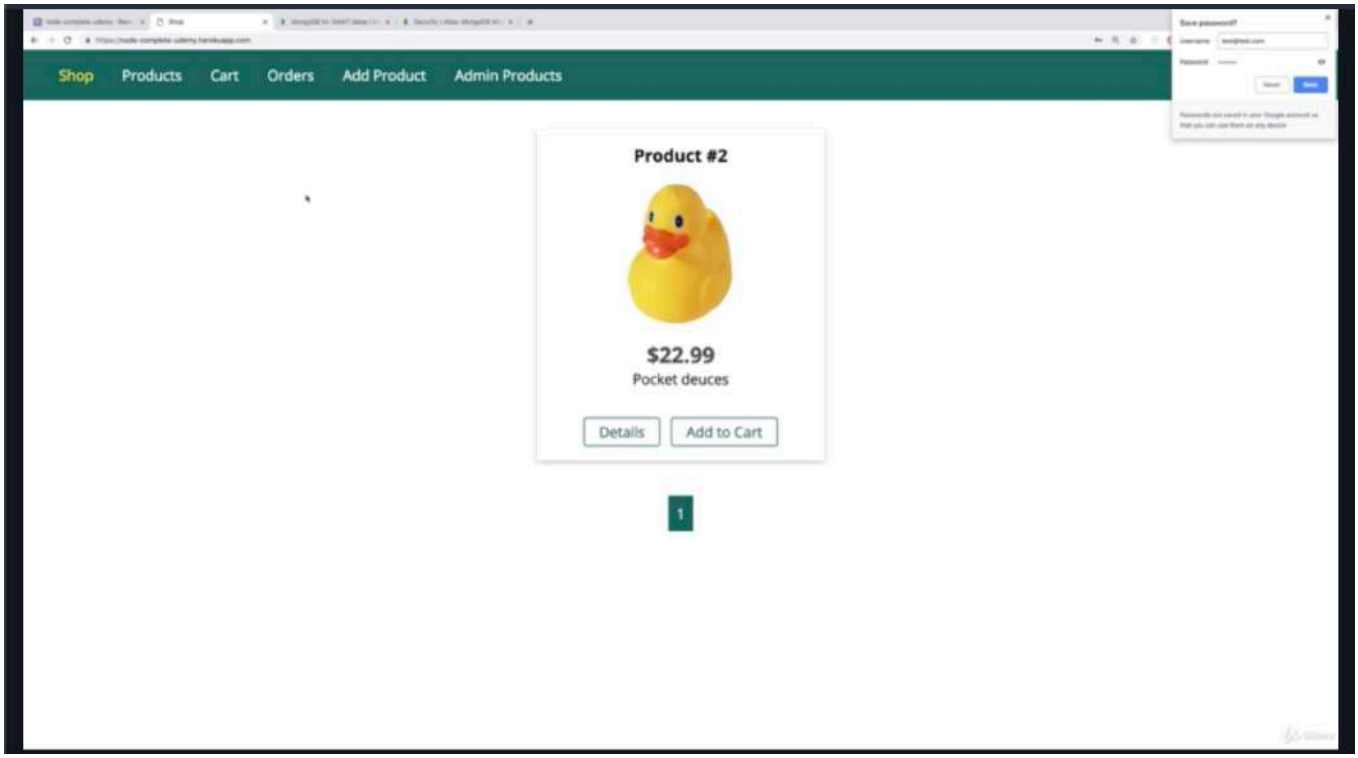


- click ‘Restart all dynos’ and let’s dive into our application. let’s try reloading our application. now there you should see a running app.


Application error


Shop
Products
Login
Signup
Product #2
$22.99
Pocket deuces
Details
1






```
//package.json

{
  "name": "nodejs-complete-guide",
  "version": "1.0.0",
  "description": "Complete Node.js Guide",
  "main": "app.js",
  "engines": {
    "node": "10.15.0"
  }
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "NODE_ENV=production MONGO_USER=maximilian MONGO_PASSWORD=rldnjs12
MONGO_DEFAULT_DATABASE=shop STRIPE_KEY=sk_test_dAPYh6CKcipsTX13kbjOFklT00NICMWhhg node
```

```
app.js",
    "start-server": "node app.js",
    "start:dev": "nodemon app.js"
  },
  "author": "Maximilian Schwarzmüller",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^1.18.3"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.18.3",
    "compression": "^1.7.4",
    "connect-flash": "^0.1.1",
    "connect-mongodb-session": "^2.0.3",
    "csurf": "^1.9.0",
    "ejs": "^2.6.1",
    "express": "^4.16.3",
    "express-handlebars": "^3.0.0",
    "express-session": "^1.15.6",
    "express-validator": "^5.3.0",
    "helmet": "^3.18.0",
    "lodash": "^4.17.11",
    "mongodb": "^3.1.6",
    "mongoose": "^5.2.17",
    "morgan": "^1.9.1",
    "multer": "^1.4.0",
    "mysql2": "^1.6.1",
    "nodemailer": "^4.6.8",
    "nodemailer-sendgrid-transport": "^0.2.0",
    "pdfkit": "^0.8.3",
    "pug": "^2.0.3",
    "sequelize": "^5.0.0-beta.11",
    "stripe": "^6.12.1"
  }
}

```

```
//app.js

const path = require('path');
const fs = require('fs');
const https = require('https');

const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const session = require('express-session');
const MongoDBStore = require('connect-mongodb-session')(session);
const csrf = require('csurf');
const flash = require('connect-flash');
const multer = require('multer');
const helmet = require('helmet');
const compression = require('compression');
const morgan = require('morgan');

const errorController = require('./controllers/error');
```

```
20 const shopController = require('./controllers/shop');
21 const isAuth = require('./middleware/is-auth');
22 const User = require('./models/user');
23
24 const MONGODB_URI =
25   `mongodb+srv://${process.env.MONGO_USER}:${process.env.MONGO_PASSWORD}@cluster0-
z3vlk.mongodb.net/${process.env.MONGO_DEFAULT_DATABASE}`;
26
27 const app = express();
28 const store = new MongoDBStore({
29   uri: MONGODB_URI,
30   collection: 'sessions'
31 });
32 const csrfProtection = csrf();
33
34 /**also make sure that you are not trying to read in our certificate and private key for SSL
35  * so comment out
36  * because these files will not be deployed
37  */
38 //const privateKey = fs.readFileSync('server.key');
39 //const certificate = fs.readFileSync('server.cert');
40
41 const fileStorage = multer.diskStorage({
42   destination: (req, file, cb) => {
43     cb(null, 'images');
44   },
45   filename: (req, file, cb) => {
46     cb(null, new Date().toISOString() + '-' + file.originalname);
47   }
48 });
49
50 const fileFilter = (req, file, cb) => {
51   if (
52     file.mimetype === 'image/png' ||
53     file.mimetype === 'image/jpg' ||
54     file.mimetype === 'image/jpeg'
55   ) {
56     cb(null, true);
57   } else {
58     cb(null, false);
59   }
60 };
61
62 app.set('view engine', 'ejs');
63 app.set('views', 'views');
64
65 const adminRoutes = require('./routes/admin');
66 const shopRoutes = require('./routes/shop');
67 const authRoutes = require('./routes/auth');
68
69 const accessLogStream = fs.createWriteStream(
70   path.join(
71     __dirname,
72     'access.log'),
73     { flags: 'a' }
74   );
```

```

app.use(helmet());
/**for Heroku, you may wanna make sure you are using compression
 * because Heroku doesn't offer you to set up compression on the fly over hosting providers do that.
 */
app.use(compression());
app.use(morgan('combined', { stream:accessLogStream }));

app.use(bodyParser.urlencoded({ extended: false }));
app.use(
  multer({ storage: fileStorage, fileFilter: fileFilter }).single('image')
);
app.use(express.static(path.join(__dirname, 'public')));
app.use('/images', express.static(path.join(__dirname, 'images')));
app.use(
  session({
    secret: 'my secret',
    resave: false,
    saveUninitialized: false,
    store: store
  })
);

app.use(flash());

app.use((req, res, next) => {
  res.locals.isAuthenticated = req.session.isLoggedIn;
  next();
});

app.use((req, res, next) => {
  // throw new Error('Sync Dummy');
  if (!req.session.user) {
    return next();
  }
  User.findById(req.session.user._id)
    .then(user => {
      if (!user) {
        return next();
      }
      req.user = user;
      next();
    })
    .catch(err => {
      next(new Error(err));
    });
});

app.post('/create-order', isAuth, shopController.postOrder);

app.use(csrfProtection);
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});
```

```

app.use('/admin', adminRoutes);
app.use(shopRoutes);
app.use(authRoutes);

app.get('/500', errorController.get500);

app.use(errorController.get404);

app.use((error, req, res, next) => {
  // res.status(error.httpStatusCode).render(...);
  // res.redirect('/500');
  res.status(500).render('500', {
    pageTitle: 'Error!',
    path: '/500',
    isAuthenticated: req.session.isLoggedIn
  });
});

mongoose
  .connect(MONGODB_URI)
  .then(result => {
    /**because will not use SSL,
     * here, you should not be spinning up your own HTTPS server
     * because we will do taht through Heroku managed server
     * or you would do it for Heroku's managed server
     * so spin up a normal HTTP server instead.
     */
    //https.createServer({ key: privateKey, cert: certificate }, app).listen(process.env.PORT || 3000);
    app.listen(process.env.PORT || 3000);
  })
  .catch(err => {
    console.log(err);
  });

```

```
//Procfile

web: node app.js
```

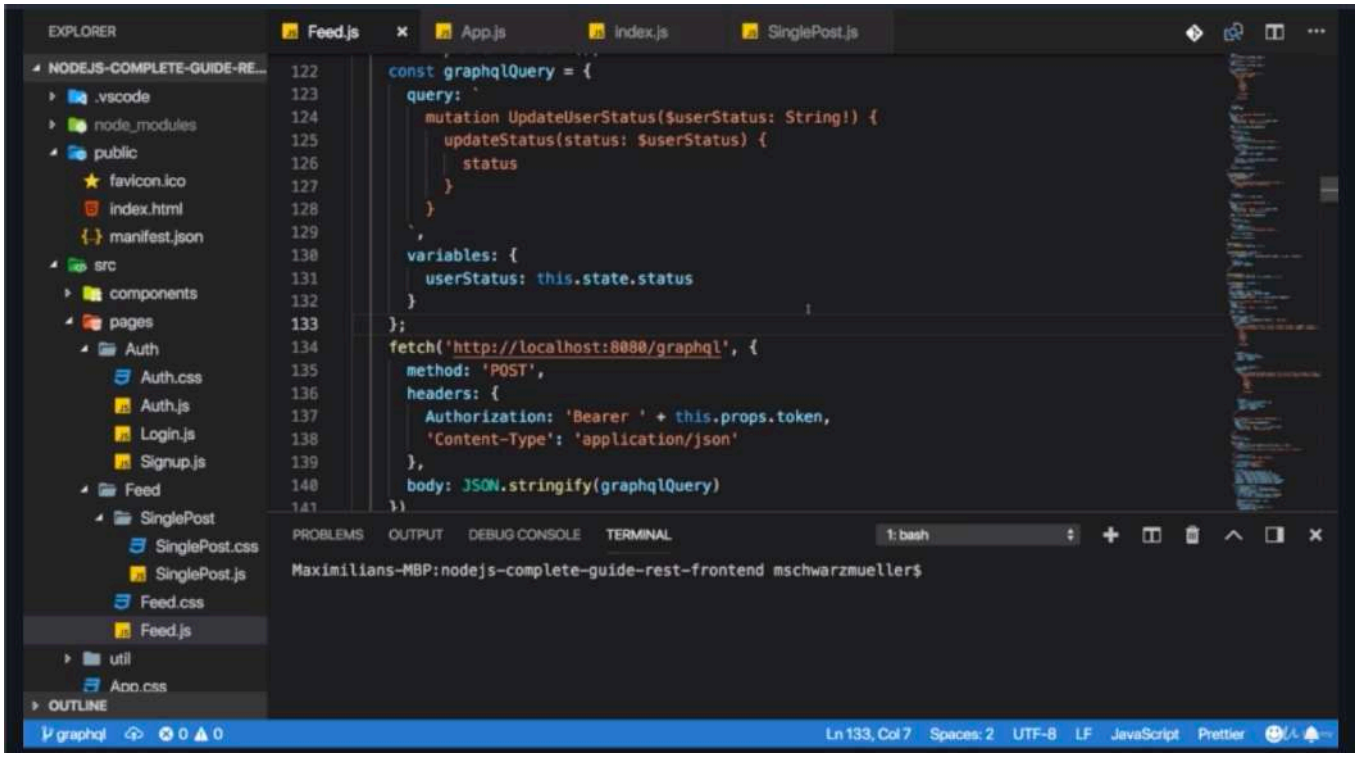
```
//.gitignore

node_modules
server.cert
server.key
```

* Chapter 456: Storing User-Generated Files On Heroku

- Storing User-generated Files on Heroku: The user-generated/ uploaded images are saved and served as intended. but like all hosting providers that offer virtual servers, your file storage is not persistent.
- your source code is saved and re-deployed when you shut down the server(or when it goes to sleep, as it does automatically after some time in the Heroku free tier)

- but you generated and uploaded files are not stored and re-created. they would be lost after a server restart
- therefore it's recommended that you use a different storage place when using such a hosting provider. in case where you run your own server, which you fully own/ manage, that does of course not apply. what would be alternatives?
- a popular and very efficient + affordable alternative is AWS S3(Simple Storage Service): https://aws.amazon.com/s3/
- you can esily configure multer to store your files there with the help of another package: https://www.npmjs.com/package/multer-s3
- to also serve your files, you can use packages like s3-proxy: https://www.npmjs.com/package/s3-proxy
- for deleting the files (or interacting with them on your own in general), you would use the AWS SDK: https://aws.amazon.com/sdk-for-node-js/

* Chapter 457: Deploying APIs



- we have the API running now where we can send requests to and will then be our front-end application or our mobile application where we have to adjust. you are able to send the request to our now running hosted application and not to localhost anymore.
- so there in your application, you will have to exchange your hosted domain, not localhost.
- and then the front-end app or the mobile app is deployed differently anyways.