1. NODEJS
   1. KOMUTLAR

* Node -v veya npm -v gıbı metodlarla versıyonları kontrol edebiliriz.
* \_\_filename : O anki calisilan dosyanin adiyla birlikte dosya yolunu
* \_\_dirname : O anki calisilan dosyanin adiyla olmadan dosya yolunu
  + 1. NPM Komutlar
* Npm i --save moduleAdi = ile modulleri yukleyebilirsiniz.

--save = package.json dosyamizin icine kaydeder

İ = install demek. Direk install da yazabilirsiniz.

Npm i [jquery@2.2.7](mailto:jquery@2.2.7) gibi yazarakda istedigimiz versiyona gore packagelerimizi ekleyebiliriz.

--save-dev = ile developmentDependenciese kaydedebiliriz. Ornegin bu paketi kurup dosyalarimizi cevirdikten sonra production asamasinda musterilerin kullanmak icin bu pakete ihtiyaci olmadigi icin daha sonra bunu silebiliriz. Bu yuzden bu yol kullanilir.

-g = ile package modullerimizi global olarak kurabiliriz.

* Npm uninstall moduleAdi = yazarakda modullerimizi silebiliriz.
* Npm init = package.json dosyasi olusturabilirsiniz.

Npm init --yes = ilede direk olarak package dosyanizi olusturabilirsiniz.

* Npm list -g = ile daha once global olarak yukledigimiz tum paketleri goruntuluyebiliriz.

Npm list -g --depth 0 = ile sadece ana globale yuklenmis paketler gozukur. Bagimliliklari gozukmez.

* 4.45.1 = Versiyonlardaki bu sayilar sirasiyla. Major.Minor.Patch dir.

"underscore": "^1.9.1"

Yukaridaki ^ isareti Minor ve Patch de her hangi bir degisiklik yenilik oldugu zaman direk en son surumu al kullan komutu anlamina gelmektedir.

* + 1. Node Komutlar
* Res.send = res.write ve res.end in express js ile birlestirilmis halidir.
* 404 Page Not Found

app.use((req, res)=>{

    //res.status(404).send('Page not Found'); Tek basina asagidaki islemler yerine buda tercih edilebilir,

    res.status(404);

    res.send('Page not Found');

})

* Farkli bir kullanim res.render dan.
* Path join ile birlestiriyoruz hepsini sonra sendFile ile gonderiyoruz.
* Ilk once dirname ile kodun yazili oldugu dosyanin adresini buluyoruz. Daha sonra bir ust dizine cikdiyoruz. Daha sonra views klasorune gir diyoruz. Daha sonra index.html i sec diyoruz.

app.get('/', (req,res)=>{

    res.sendFile(path.join(\_\_dirname, '../', 'views,', 'index.html'));

});

**REST API**

**Represesentational State Transfer – Application Programming Interface (REST API or RESTful API)**

Bir web uygulamasinin clientta bir dizi islemlerden gecerek clientlarin bu kaynaklara eriserek faydalanmasini ve yonetmesini saglayan uygulamadir.

REST API ile CRUP operatorumuzu kullanabiliriz.

Create POST /tasks

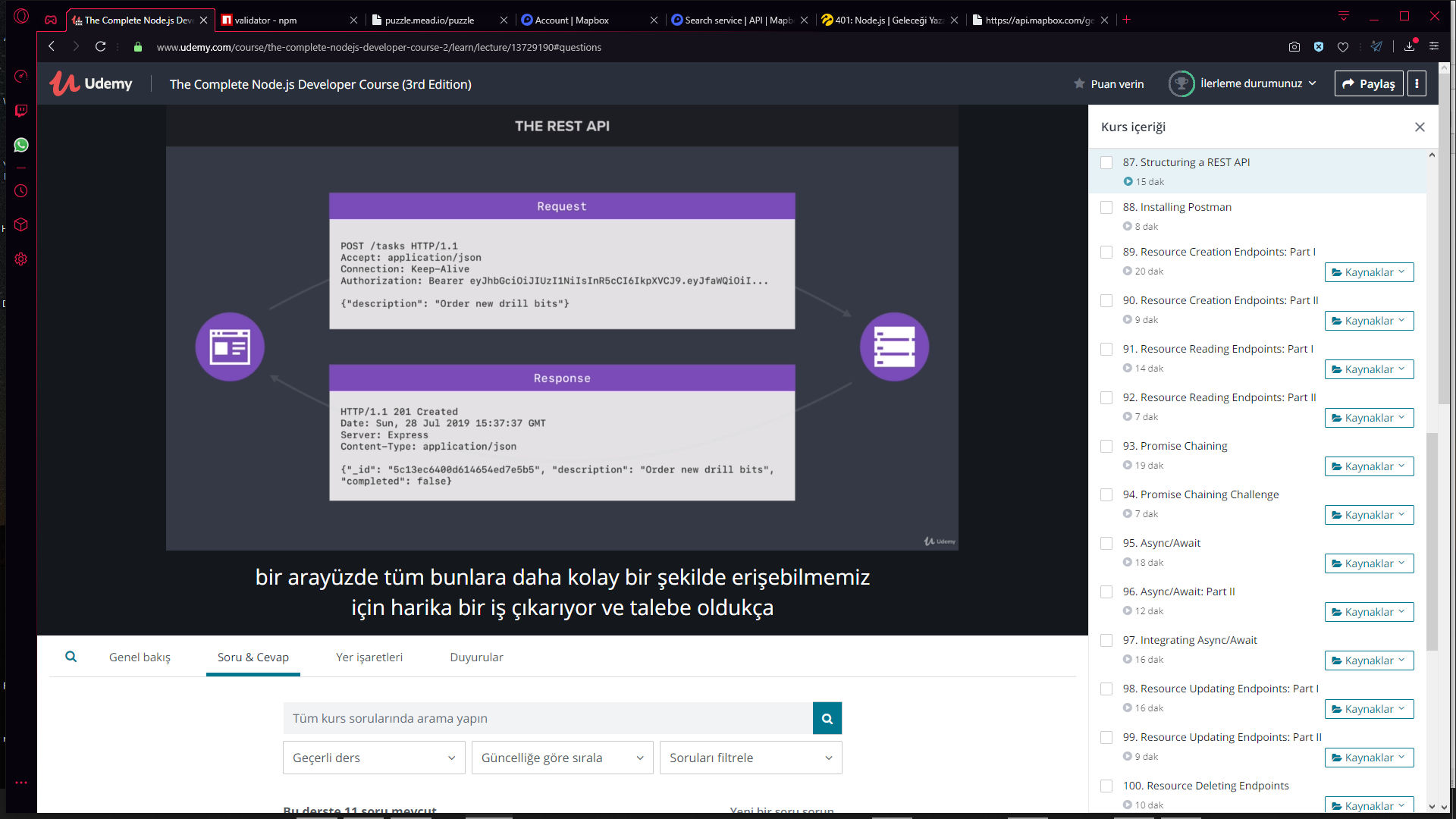
Read GET /tasks

Read GET /tasks/:id

Update PATCH /tasks/:id

Delete DELETE /tasks/:id

For Example for an basic http Request and Response



* 1. MODULLER
* 3 cesit modul vardir. Custom Module, Node Module, 3rd Party Module
* Examples for Node Modules : FileSystem-Path-Events-Os-http vs…
* Examples for 3rd Party Modules : Npm ile yuklenen moduller. En buyuk ve en onemlisi Express diyebiliriz.
* Examples for Custom Modules : Kendimiz export edip require ile yakaladigimiz modullerdir.
  + 1. Path Module
* Dosyanin uzantisini bulmak tam yolunu bulmak gibi islemler yapilir.

const path = require('path');

//Dosyanin tam yolunu bulmak

let result = path.resolve('app.js');

//Dosyanin uzantisini bulmak

result = path.extname('app.js');

console.log(result);

//Dosya ile ilgili bilgileri veren OBje dondedir.

//Ve bunlari asagidaki gibi yakalayabiliriz.

result = path.parse(\_\_filename);

console.log(result.root);

console.log(result.dir);

console.log(result.base);

console.log(result.ext);

console.log(result.name);

* + 1. URL Module

const url = require('url');

// Uygulamaya gelen adres icerisindeki bolumlere ulasabiliyoruz.

const adress = 'http://www.sadikturan.com/kurslar?year=2019&month=nisan';

// true dememizin nedeni linki bolumler olarak obje

//seklinde cevrilmesini istedigimiz icin.

let result = url.parse(adress, true);

console.log(result);

* Bize verecegi cikti

Url {

    protocol: 'http:',

    slashes: true,

    auth: null,

    host: 'www.sadikturan.com',

    port: null,

    hostname: 'www.sadikturan.com',

    hash: null,

    search: '?year=2019&month=nisan',

    query: [Object: null prototype] { year: '2019', month: 'nisan' },

    pathname: '/kurslar',

    path: '/kurslar?year=2019&month=nisan',

    href: 'http://www.sadikturan.com/kurslar?year=2019&month=nisan' }

* + 1. FileSystem FS Modules

const fs = require('fs');

//Uygulama icersinde bulunan dosya ve klasorlerimizi alabiliriz.

//Herhangi bir dosya icerigini okuyabiliriz.

//Dosya olusturabilir, silebilir, isim degistirebiliriz.

const files = fs.readdir('./', (err, data)=>{

    if(err)

        throw err;

    console.log(data);

});

// Verecegi Cikti

// [ 'app.js', 'deneme.txt' ]

const data = fs.readFile('index.html', 'utf8', (err,data)=>{

    if(err)

        throw err;

    console.log(data);

});

//HTML sayfasinin ici komple kodlarla birlikte yazdirilir.

//Dosya Olusturmak (Dosya ismi - Icerik - CallbackFunction)

fs.writeFile('deneme.txt', 'Hello world', (err)=>{

    if(err)

        throw err;

    console.log('Dosya Olsuturuldu'); //Ne istersen.

});

//Dosya Olusturmak (Dosya ismi - Icerik - CallbackFunction)

//appendFile eger verdiginiz dosya isminde dosya yoksa yeni olusturur.

//dosya varsa eger uzerine ikinci parametre oldugunuz mesaji ekler.

fs.appendFile('deneme.txt', 'Hello world', (err)=>{

    if(err)

        throw err;

    console.log('Dosya Olsuturuldu'); //Ne istersen.

});

//Dosya Silmek

fs.unlink('deneme.txt', (err)=>{

    if(err)

        throw err;

    console.log('Dosya silindi.');

});

//Yeniden adlandirmak

fs.rename('deneme.txt', 'yeniisimlideneme.txt', (err)=>{

    if(err)

        throw err;

    console.log('Dosya ismi basariyla degistirildi.');

});

* + 1. Events Module

const EventEmitter = require('events');

const emitter = new EventEmitter();

//Yukaridaki gibi bir emitter olusturabiliriz.

//Ve olusturdugumuz .on ile asagidaki gibi istedigimiz kadar Listener olusturabilir

//ve onlari .emit ile tetikleyebiliriz.

//Listener Kayit Et

emitter.on('connection', ()=>{

    console.log('Baglanti kuruldu');

});

emitter.on('logout', ()=>{

    console.log('Baglanti koptu');

});

//Eventi Tetikle

emitter.emit('connection');

//Asagidaki params1 ve params2 degerlerini emit ederken

//2. deger ile baslayarak sira ile degerleri atayabiliriz.

emitter.on('connection', (params1, params2)=>{

    console.log('Baglanti kuruldu');

});

emitter.emit('connection', 'Adana', 01);

//VEYA

emitter.on('connection', (args)=>{

    console.log('Baglanti kuruldu' + args.name);

});

emitter.emit('connection', {name: 'Mucahid', age:30});

* + 1. Http Module

const http = require('http');

//req = istektir

//res = sonucdur

//CreateServer ile serverimizi kurariz.

const server = http.createServer((req,res)=>{

    res.write('Hello World');

    res.end();

    if(req.url === '/api/products'){

        //Tarayiciya yazdirma islemi yapar.

        res.write('product list');

        //Yazdirma isleminin bittigini soyler.

        res.end();

    }

});

//Http Module de EventEmitter oldugu icin bu sekildede mesaj gonderebiliriz.

server.on('connection', ()=>{

    console.log('New connection');

});

//Serverimizi bu portta dinleriz.

//Boylelikle localhost:3000 yazdigimizda tarayicimizda acabiliriz.

server.listen(3000);

console.log('Listening port ' + server.address().port);

* + 1. Underscore
* En buyuk yada en kucuk sayisi bulmak, buyukten kucuge siralamak gibi yuzlerce islemi basitce yapabiliriz. Detaylar icin npm sayfasina bak.

ORNEK

var \_ = require('underscore');

var numbers = [10, 5, 100, 2, 1000];

console.log(\_.min(numbers));

* + 1. Nodemon
* Serverı sureklı guncel tutar. Bizim yazip kaydettighimiz degisiklikleri direk olarak servera yansitir.
* Nodemon yazarak server bilgilerini tanimladigimiz js dosyasini yazariz. Ornek nodemon app.js
  + 1. Validator

<https://www.npmjs.com/package/validator>

**Request ve Response Kavramlari**

Requestler ile gelen isteklerin postmu getmi gibi oldugu gibi bilgileri alabiliriz.

Responsela ise isteklerin basarili basarisiz oldugunu, sever hatasi veya arananin bulunup bulunmadigini, json gibi degerleri alip yakalayabiliriz.

Requests : GET POST

GET ile gelen istekler Header kisminda gelir.

POST ile gelen istekler HEADER++BODY kisminda gelir.

Response ile gelen isteklerin http Sattus Code leri doner.

HEADER ve BODY kisimlari doner. Bunlarlada formlarimizda name ile isimlendirdigimiz degerleri yakalayabiliriz.

Sadece html degil json xml text gibi icerikleride yakalayabiliriz.

.setHeader(‘Content-Type’, ‘text/plain’); Text Dosyasi dondurmek istersek

.setHeader(‘Content-Type’, ‘application/json’); JSON dosyasi dondurmek istersek. JSON.stringfy ile yakalamayi unutma.

.setHeader(‘Content-Type’, ‘text/html); HTML dosyasi dondurmek istersek. Kodlarida uygulayip yazar kod seklinde gostermez.

* 1. GLOBAL OBJECTS
* Window tarayıcı için bir Global Scope dur. Node.js de tanınmaz.
* Farklı JS dosyalarında tanımladıgımız tüm değişkenler window objesinin içinde tanımlanıyor. Bu sayede her bırını ıstedıgımız js dosyalarında kullanabılırız. Ornegin setTimeout setInterval gibi metodlar window icerisinde tanimlanmis global degiskenlerdir.
* NodeJS de ise farkli dosyalarda calismak ve degerleri kullanmak icin module ile require etmek gerekiyor. Fakat NodeJS dede setTimeout setInterval gibi global metodlar tanimlanmistir.

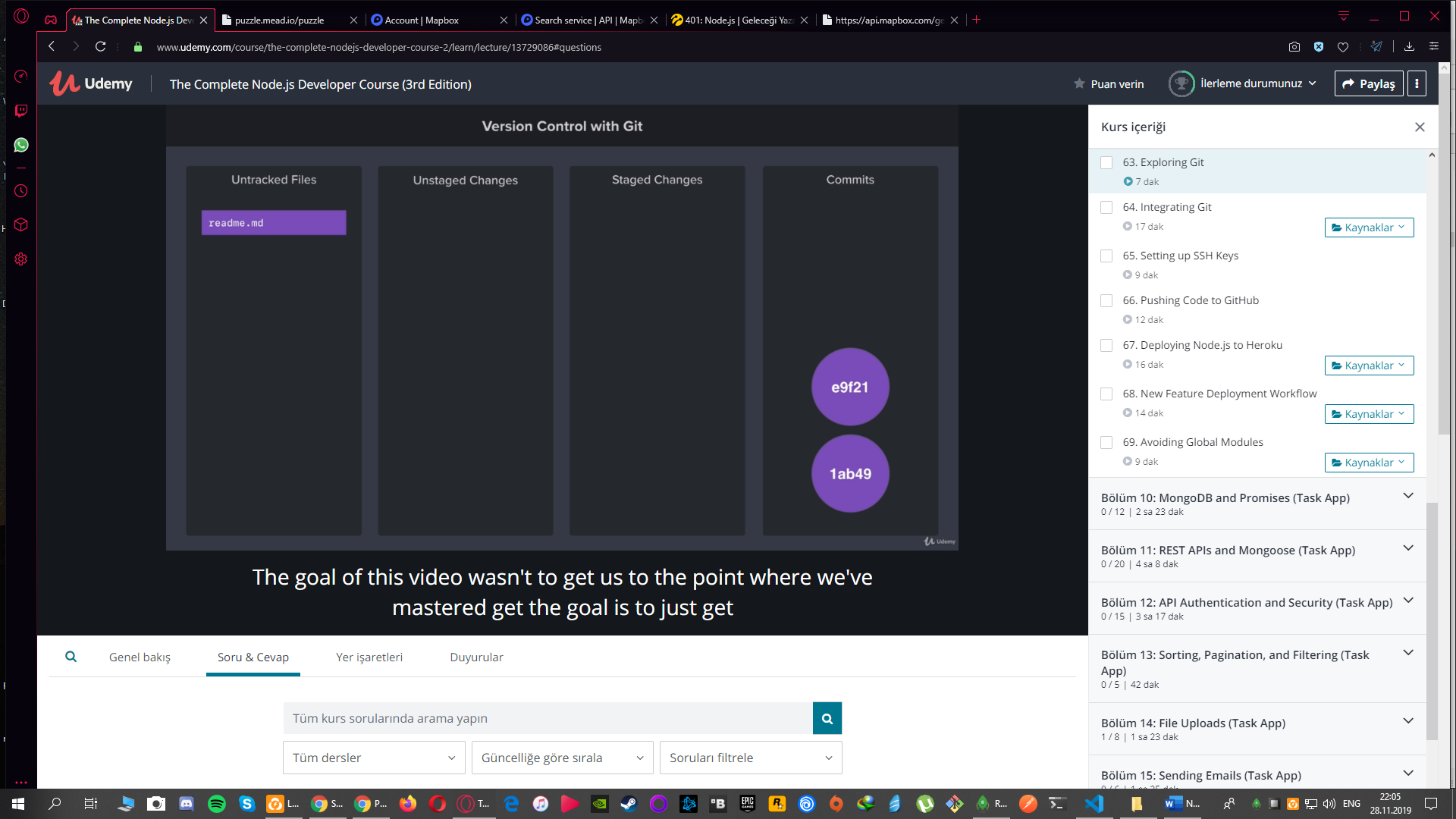
GITHUB AND HEROKU KULLANIMI

Heroku login = Herokuya terminalde giris yapip baglanti kurariz

Heroku -v = heroku versiyonunu kontrol ederiz.

Git --version = Git versiyonunu kontrol ederiz.

**GIT MANTIGI**



Ornek olarak app.js ve readme.md yi ele alalim.

Proje Dosyalarimiz ilk basta Untracked Files’da oluyor.

Git add app.js yaparsak app.js dosyasi untracked files dan Staged changes’a gider. Readme.md halen untracked filesda kalir.

Commit app.js yaparsakda app.js dosyamiz commitsde en altta rasgele bir adda sirada olur. Bu bizim ilk versiyon app.js imizdir.

Daha sorna app.js de degisiklik yaptigimizda dosyamiz Staged Changese rastgele adli yerden bir kopya gonderir. Ve daha sonra ister git add app.js yaparak app.js i staged changes e aliriz yada git add . yaparak unstaged changesdeki ve untracked files daki tum dosyalari staged changes a atariz. Daha sonrada commit ile yukleriz.

Komutlarimizi yazarken bunlari adim adim git statusle kontrol edip durumlarinin Untrackedmi trackedmi yoksa staged mi oldugunu gorebiliriz.

Web dosyalarimizin oldugu ana dizine .gitignore adinda bir dosya olusturup, gitin islem yapmasini istemedigimiz dosya ve klasorlerin adini buraya yazariz. Ornegin node\_modules/ app.js/ vs…

Git add yazarak dosyalarimizi staged area ya tasiyabiliriz.

Git add src/ yazarak src ve icindeki tum dosyalari commit etmeye hazir olmak uzere staged areaya gondeririz. Yada . yazarak hepsini.

Git commit -m “first commit” (buradaki m message’nin m si anlaminda ve sonrasinda gelen first commit onun devami olan mesaj demek oluyor) ve bu kodu yazarak ilk commitimizi yapmis oluyoruz.

Tum dosyalari commit ettikten sonra herhangi bir dosyayi degisirsek ne olur.

Diyelim ki app.js i degistirdik o zaman app.js o zaman dosyamiz untracked files olur.

Git add . yapariz tek o degistigi icin sadece onu staged areaya eklleriz commit etmek icin

Daha sonra git commit -m “ve bu araya yapilan degisikligi mesaj olarak yazabilir” ve commit edebilirsiniz.

Daha sorna ustteki kodu enterladiktan sonra terminal bize yapilan degisiklikleri ozetleyecektir. Ornek 1 file changed 2 deletions gibi.

Yani kisaca

1. Eger Init yapilmadiysa ilk kez ise **git init** yaz
2. git add . (hepsi icin nokta veya istedigin dosya klasoru yaz)
3. git commit -m “first commit” (commit mesajini kendin sec)

Bunlari yaparak git initialize ve commit islemimiz tamamdir.

Daha sonra GitBash ile SSH Key olusturuyoruz veya varsa kullaniyoruz.

Var olan keylerimizi gormek icin

ls -a -l ~/.ssh

Burada anlami

Ls bulundugumuz dizindeki dosyalari siralar

-a = gizli dosyalarin gorundugunden emin oluruz.

-l = Listeleme yapar tum dosyalari sira halinde

~/.ssh = ~ = oldugumuz konumdur. /.ssh = ve .ssh bizim baktigimiz dosyadir yani aradigimiz.

Yoksa olusturmak icin

ssh-keygen -t rsa -b 4096 -C "mucahidyazar@gmail.com"

ssh-keygen izin verir olsuturmamiz icin

-t type secimidir.

Rsa = guvenli bir sifreleme platformudur

-b = byte demektir 4096 = kac byte oldugudur.

-C = Buyuk harf oldugundan emin olur

Daha sonra email adresimizi yazariz..

Daha sorna bize asagidaki 3 soruyu soracak ve default olarak kalmasi icin hicbirsey yazmayacagiz

**Mkdir ~/.ssh** ile cannot access hatasini cozumleriz eger alirsak

Daha sonra ls -a -l ~/.ssh tekrar bu kodu yazarak dosya dizinimizdeki keylerimizi kontrol ederiz.

Burada id\_rsa ve id\_rsa.pub dosyalarini gorecegiz

İd\_rsa = Makinemizde gizli tutacagimiz ve asla paylasmayacagimiz gizli dosyamiz

İd\_rsa.pub =Bu dosyamiz ise heroku ve git gibbi platforlarla paylasacagimiz bizim kendi makinemiz ve heroku arasinda baglantimizi saglayacak guvenligi saglayacak dosyamizdir.

Artik yeni key dosyalarimizi kaydetmek kaldi

**eval $(ssh-agent -s)**

Ve yukaridaki kodu yazarak kaydederiz

ssh-add ~/.ssh/id\_rsa

Daha sonra bu kodlada ekleme islemini tamamliyoruz. Ve ssh islemini bitiriyoruz.

Kisa bir egitim videosu bu konuda

<https://www.youtube.com/watch?v=Vi-WqFKYpnw>

GIT’e Proje Yukleme

Artik projemizi gite yukleyebiliriz.

1. Gitin web sayfasina gidip new yada + isaretine tikliyoruz.
2. Projemiz tamam degilse Private yaparak yada Public olarak ayarliyoruz yeni repository mizi
3. Daha sonra olustur yapiyoruz.
4. Sonra asagidaki kodla remote olarak ekliyoruz. Ve buradaki link repositorymizi olusturduktan sonra bizim projemiz icin gitin bize verdigi link olacak. Origin bizim remote adimizdir. Bu istedigimiz birsey olabilir ama genel olarak bu kullanilir. O yuzden degistirmiyoruzz. Ornek asagida:

**git remote add origin https://github.com/mucahidyazar/nod-weather-website.git**

Bu kod ile dosyalarimiz sadece gite yuklenmez. Bu kod ile Dosyalarimiz ile git arasindaki baglantiyi saglariz. Cep telefonuna bir kisi eklemek gibi dusunebilirsin.

1. Daha sonra SSH ile github baglantisini saglamamiz lazim. Bunun icin SSH leri kurdugumuz yere yani default olan C:\Users\Mucahid Yazar\.ssh adrese gidip id\_rsa.pub dosyasinin icindeki keyi kopyalayip veya terminale **cat ~/.ssh/id\_rsa.pub** yazip gelen kodu kopyalayip (ikiside ayni) .Git / Settings / SSH and GPG keys / Add new SSH key yapip basligi belirledikten sonra alt kisma bu keyimizi girmemiz gerekmektedir.

Daha sonra ssh -T [git@github.com](mailto:git@github.com) yazarak terminal uzerinden bu ssh baglantisini test edip git ssh sayfasinda bu hareketi gorebiliriz.

Bu SSH keyi tum projelerimizde kullanabiliriz.

1. Daha sonra asagidaki kodu yazarak gite pushlama islemi yapiyoruz. Nereye pushluyoruz. Origin remotuna. Neyi pushluyoruz master branchimizi pushluyoruz.

**Git push -u origin master**

HEROKU YUKLEME

1. SSH keyimizi herokuyada ekliyoruz. Bu git ve heroku arasinda bir bag kurmamizi saglayacak guvenlik duvari olacak. Bunun icin asagidaki kodu yaziyoruz.

**heroku keys:add**

Daha sorna bize buldugu keyleri getirecek ve yes yada no diyecegiz.

1. Simdi herokuda olusturulacak proje ismini kullanacagiz eger biz yazmazsak birsey heroku bize karisik bir isim otomatik olarak uretecektir. Olusturacagimiz isim ozel oolmalidir. Onlineda diger kullanicilar almamis olmalidir.

Heroku create weather-application

1. Artik son islemimiz kaldi package.json dosyamizi kontrol ediyoruz. Burada scriptslerde “start”: “node src/app.js” oldugundan emin oluyoruz yoksa olusturuyoruz.
2. App.js dosyamizin icinde local olarak say i ile tanimladigimiz portu asagidaki gibi degistiriyoruz.

const port = process.env.PORT || 3000

Burada eger sunucu calistiriyorsa otomatik port atar biz calistirirsak localde 3000i kullanir port degiskeni.

Ve web dosyalarimiz icinde localhost lu http baglantilari kullanmadigimizdan emin olalim.

1. Ve artik dosyalarimizi degistigimiz icin yeniden commit ediyoruz.
2. Ve simdide dosyalarimizi asagidaki gibi herokuya yukluyoruz.
3. Git remote ile remote baglantilari gorebiliyoruz. Bizim remote baglantimiz heroku oldugu icin

**Git push heroku master** diyoruz. Master bizim branchimizdi unutmayin.

**Herşeyi Deoploy Ettikten Sonraki Çalışmalar**

Diyelim ki About.ejs de bir degisiklik yaptik yapmamiz gerekenler sirasiyla

1. Git add .

Ilk once hepsini git status changed a tasiyoruz bu sekilde.

1. Git commit -m “Changes in about.ejs”

Ve changeddan da commit ediyoruz. Ve artik puishlanmaya hazirlar.

1. Git push (git push origin master)

Kisaca git e push etmek icin git push yazmak yeterli veya uzun haldede yazabilirsiniz. Master yerine farkli branchiniz varsa onu yazin.

1. Git push heroku master

Ve son olarakda herokuya master branchini bu sekilde push luyoruz.

Npm i nodemon --save-dev = Bu sekilde devDpendencies e modullerimizi yukleyebiliriz.

Heroku devDpendencies i kullanmaz. Bu projemizi indirip daha sonra calisanlar icin yardim saglayacaktir. Diyelimki package.json icinde scriptlerde “dev”: “nodemon src/app.js” tanimalrsak ve nodemon bu projede calisacak arkadasa npm install yaptiginda yuklenmezse o sorun yasayabilir. Bu yuzden devDependenciese ekleyip unutup yuklememe ihtimalinin onune geciyoruz.

Eger nodemon devDependencies de kayitliysa terminalden nodemon src/app.js yazarak calistiramayiz. Scripte kaydederiz “dev”: “nodemon src/app.js” bu sekilde ve **npm run dev** dememiz gerekiyor.

**MONGODB**

Db.version() = Mobgodb versionunu sorgulariz