Flask App - 06/15/2021

Tuesday, 15 June 2021 20.04

Portfolio Building

Flask Basics

Table of Contents

- What is Framework
- What is Flask
- Create Basic Flask App

What is Framework

Kullanici internete giris yaptiginda karsina sayfalar gelmektedir. Ama girmis oldugum sayfanin arka kisminda ne oldugunu tam olarak bilmiyoruz. Yani mutfak kismindan haberimiz yoktur.

Internet sitelerinin altinda data base baglantilari, kullanici ag arayuzleri ve serverlar vardir. En onemlisi bunlarin kendi aralarinda koordineli calismalari gerekmektedir. Sistem ancak bu sekilde saglam bir sekilde ayakta kalmaktadir. Sistemin calismasi icin webframework olmasi gerekmektedir.

Features of Frameworks

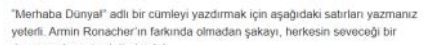
- Skeleton — Bir iskelet olusturulursa sistem hizli bir sekilde ayaga kalkar ==> HIZ
- Templates — Cogu Framework bizlere template sunar ve bu template sayesinde sistem ayaga kalkar. Index html gibi template frameworkun olsumasinda yardimci olacak
- Web Caching — Back end denilen mutfakta bir sey olurken cashleme sayesinde istekler on bellege aliniyor arka mutfak rahatlatiliyor.
- URL Mapping — Var olmayan ya da uzun ve karışık bir adres üzerinden başka bir sayfayı gösterme işlemine URL eşleştirme denir. Bu sayede asıl web sayfası gizlenmiş olmakla beraber, güvenlik açısından daha verimli sayfalar elde edilmiş ve bağlantının karışıklığı giderilmiş olur.
- Security — Guvenlik kirilganligina dikkat etmek gerekmektedir.

Advantages of Frameworks

- Quick
- Open source
- Customisable — Istenildigi zaman modifiye edilebiliyor
- Support — Inanilmaz bir destek agi var bloglardan, stack over flowdan yardim istenebilir.

What and Why is Flask

Flask tamamıyla Python programlama dilinde yazılmış bir web geliştirme çatısı, framework'üdür. Öğrenmesi ve kullanması oldukça kolaydır. Basite indirgenmiş yapısıyla yeni başlayanlar için doğru bir tercihtir. Flask'i yaratıcısı Armin Ronacher


"Merhaba Dünya!" adlı bir cümleyi yazdırmak için aşağıdaki satırları yazmanız yeterli. Armin Ronacher'ın farkında olmadan şakayı, herkesin seveceği bir

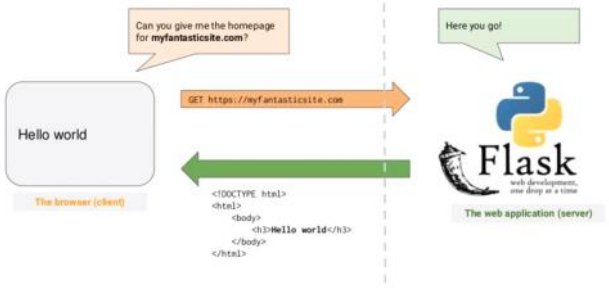
Türkce'ye catı olarak

What and Why is Flask

Micro Framework

Micro kavramını basit ama genişletilebilir olarak kısaca açıklayabiliriz. Burada basit ve genişletilebilir nedir sırasıyla ele alalım. Örneğin; kendinize bir araba yaptığınızı varsayalım. Motor, kapılar, tampon, bagaj gibi birçok parçaya ihtiyacınız var. Bu motorun parçalarını almak için iki yere gidiyorsunuz. İlk yer size saydığımız ve daha fazlasını paket halinde vermek istiyor. İkinci yer ise sadece motoru veriyor ve dilerseniz eklemeler yaparsınız diyor. İşte gittiğimiz ikinci yer Micro Framework'ün ta kendisi. İstediğiniz parçayı kullanma veya kullanmama özgürlüğü tanıyan özünde basit bir yapı.
Bu parçalara bilgisayar dünyasından örnek vermek gerekirse; veri tabanı, kimlik doğrulama, form doğrulama gibi sistemlerdir. Bu sistemlere olan ihtiyacınıza göre ekleme yapabilme özgürlüğüne sahipsiniz.

What and Why is Flask



What and Why is Flask

- Flexible
- Easy to learn
- easy to redirect URLs
- Do not need to compile

framework'üdür. Öğrenmesi ve kullanması oldukça kolaydır. Basite indirgenmiş yapısıyla yeni başlayanlar için doğru bir tercihtir. Flask'i yaratıcısı Armin Ronacher

Türkçe'ye çatı olarak çevirebileceğimiz framework'ü insan iskelet sistemi gibi düşünebiliriz. Ya bu iskeleti sıfırdan kendimiz yazacağız ya da herkesin ortak olarak kullandığı iskelet yapılarından birini seçeceğiz. İlki büyük bir zaman kaybı olacağından, framework'leri tanıyıp onlarla çalışmanın büyük bir katkısı var.

Flask Kullanan Global Şirketler

- Netflix
- Reddit
- Airbnb
- Lyft
- Mozilla
- MIT
- Uber
- Red Hat
- Patreon
- MailGun

Django Python'da yazılmış bir başka framework'tür. İki framework de Python'da yazılmış olmasına rağmen birbirlerinden oldukça zıt yapılara sahiplerdir. Yazımızda Flask'ın micro framework yapısından bahsettik. Django ise bunun aksine içinde tüm gereksinimleri barındırır. Araba almak için gittiğimiz, bize tüm parçaları sunan yer gibi düşünün.

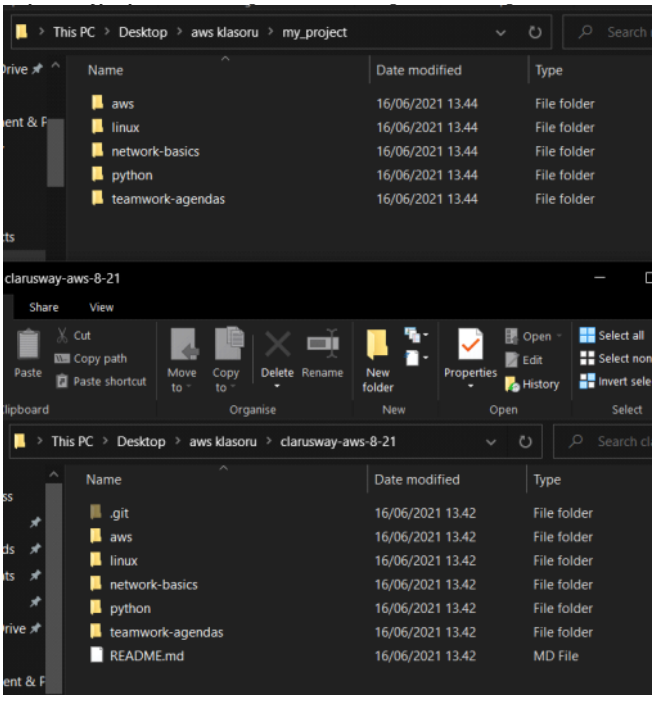
İlk farkı Flask'ın micro framework yapısı olması. Bu sayede geliştirici istediği özelliği ekleyerek projesini geliştirebiliyor. Öte yandan büyük projelerde Django'nun tercih edildiğini görüyoruz. Bundan bir çıkarım yaparsak büyük projelerde hazır olarak sunulmuş ve tüm paket içeren bir framework yapısı tercih edilmekte.

Yeni başlayanlar için Flask daha iyi bir tercih. Flask için bir kod örneği vermiştik. Sadece o kodu çalıştırarak bir sonuç almak mümkün. Öte yandan Django ile bunu gerçekleştirmek için birkaç kod ve paket gibi ayrıntılar içeriyor. Yeni başlayan birinin ikincisini tercih etmesi zorlu bir öğrenme yolu olabilir.

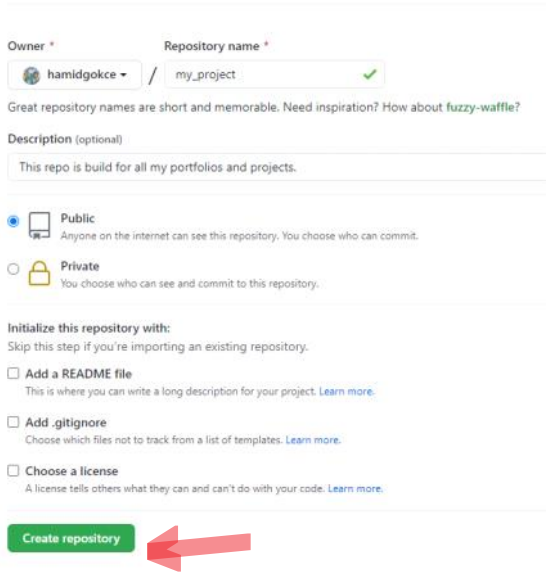
1- https://github.com/clarusway/clarusway-aws-8-21 adresine gidelim. Uygun gorulen bir yere Repodaki verileri klonlayalim.

git clone https://github.com/clarusway/clarusway-aws-8-21.git

2- Daha sonra istenilen bir yere MY_PROJECTS adi altinda bir klasor olusturalim ve klonladigimiz repodaki .git ve README.md haricindeki dosyalari kopyalayalim. Olusturdugumuz klasorler asagidaki sekildeki gibi olacaktir.

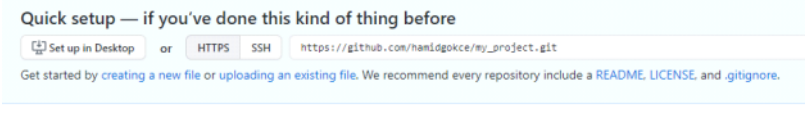


3- Localde '**MY_PROJECTS**' adli bir klasor olusturmustuk ve icerisinde yukaridaki 5 ayri klasoru kopyalamistik. Simdi Github da ayni isimde yeni bir repository olusturuyoruz. ***(Repository name = my_procet, Description = 'This repo is build for all my portfolios and projects.' , Public olacak ve README file isaretlenmeyecek.)***



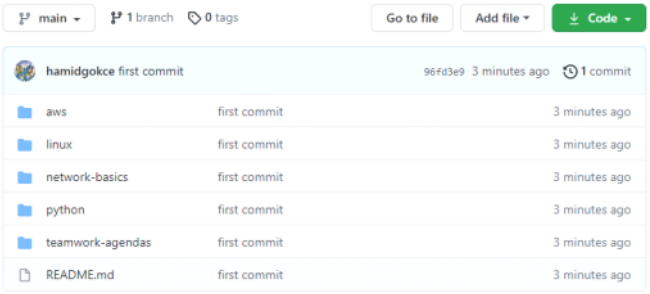
4- COMMANDS

```
echo "# my_project" >> README.md
git init
git add .
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/hamidgokce/my_project.git
git push -u origin main
```



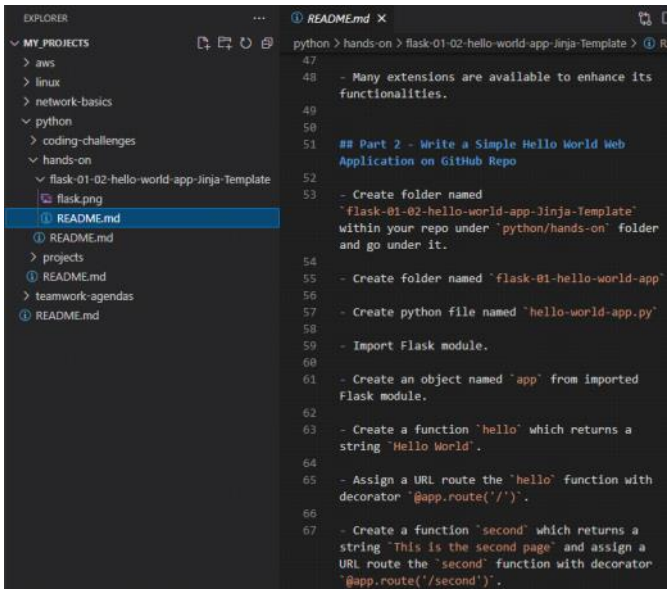
Git add . Komutunu girebiliriz. Cunku bizim olusturdugumuz dosya icerisinde Farkli klasorlerde mevcut. Hepsini eklememiz gerekmektedir.

5- Push ettigimiz butun dosyalarin Github repomuza eklendigini goreceksiniz

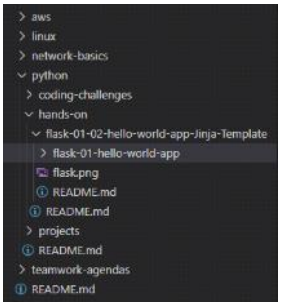


6- Olusturmus oldugumuz **MY_PROJECTS** klasorunu VS Code programinda acalim

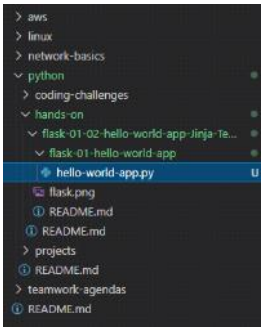
7-



8- Create folder named `flask-01-hello-world-app` (hands-on /flask-01-02-hello-world-app-Jinja-Template ==> klasorunun icerisine)



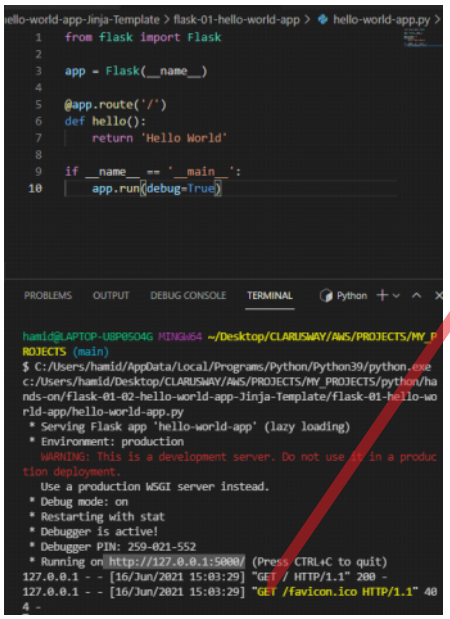
9- Create python file named `hello-world-app.py` (8. maddedeki klasorun icerisinde olusturuyoruz) (Amacimiz Hello World yazisi cikaran bir internet sitesi olusturmak)

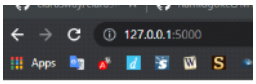


10- Olusturdugumuz python dosyasi icerisine asagidaki komutlari yaziyoruz.

Flask i yazmanin 3 kurali
1- Flask import etmek
2- Object olusturmak
3- Olusturulan object kontrol ediliyor, calistiriyor. Geriye dogru giderek debug etmemize kontrol etmemize yariyor.

11- Oncelikle Hello World yazmasi icin bir fonksiyon tanimlamak lazim. Bu fonksiyonu da internet ortaminda calistirmak icin ==> 'Flask te URL assign etmek gerekmektedir. Flask te bu isleme decorater deniyor ve @app.route('/') yaziliyor'



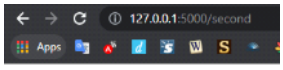


Hello World

Yazilan kod run edildikten sonra terminalde
http://127.0.0.1:5000/ adresi cikacak ve tikladigimizda
Hello World yazisini yukaridaki gibi gorecegiz

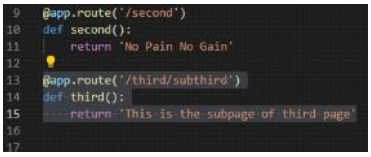
Terminal ekranini
- CTRL + C ==> islem sonlandirilir
- CTRL + L ==> sayfa temizlenir

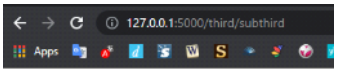
12- @app.route('/second')
def second():
return 'No Pain No Gain' ==> yaziyoruz ve yukaridaki islemi tekrarliyoruz



No Pain No Gain

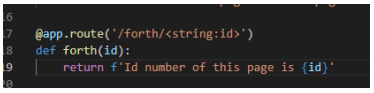
13- @app.route('/third/subthird')
def third():
return 'This is the subpage of third page'==> yaziyoruz ve yukaridaki islemi tekrarliyoruz

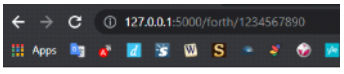




This is the subpage of third page

14- @app.route('/forth/<string:id>')
def forth(id):
return f'Id number of this page is {id}''==> yaziyoruz ve yukaridaki islemi tekrarliyoruz



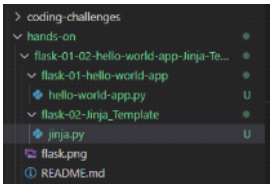


Id number of this page is 1234567890

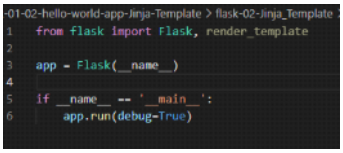
Yazilan tum komutlar

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
    return 'Hello World'
@app.route('/second')
def second():
    return 'No Pain No Gain'
@app.route('/third/subthird')
def third():
    return 'This is the subpage of third page'
@app.route('/forth/<string:id>')
def forth(id):
    return f'Id number of this page is {id}'
if __name__ == '__main__':
    app.run(debug=True)
```

15- Actigimiz sayfalari kaydedip kapatiyoruz ve **README.md** dosyasindaki **88**. asamaya geciyoruz.
flask-01-02-hello-world-app-Jinja-Template klasorunun icerisine **flask-02-Jinja_Template** klasorunu ve daha sonra **jinja.py** dosyasini olusturuyoruz.

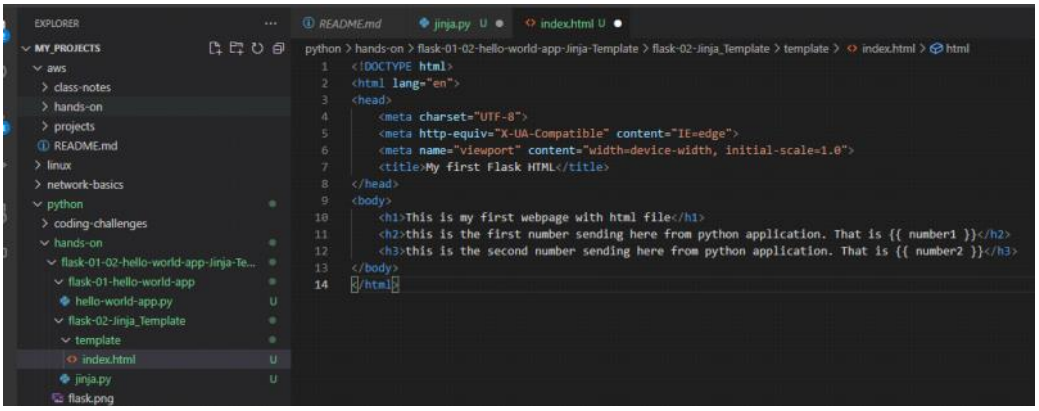


16-



- Modul vasitasiyla template kiralayacagiz
- Ikinci sart olarak obje olusturuyoruz
- En son islem olarak da kontrol ediyoruz.

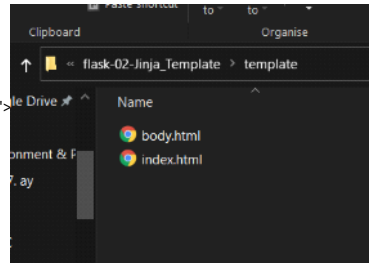
17- **flask-02-Jinja_Template** klasorunun altinda **template** adli bir folder olusturacagiz. **Template** klasorunun altina da **index.html** adli dosya olusturuyoruz. Daha sonra asagidaki kodu **index.html** dosyasinin icerisine yapistiriyoruz.



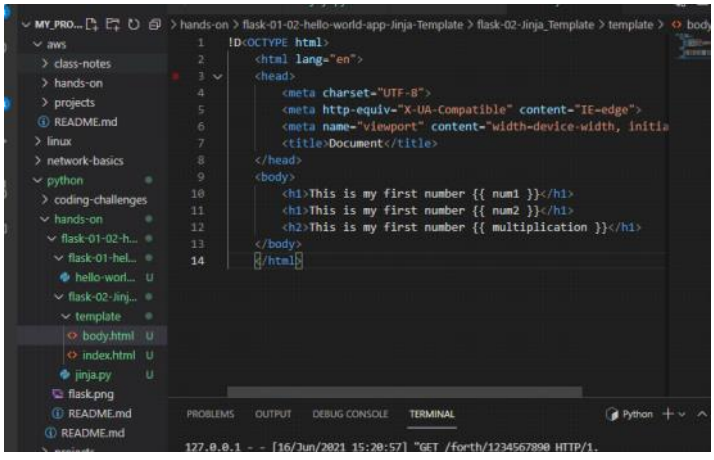

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My first Flask HTML</title>
</head>
<body>
  <h1>This is my first webpage with html file</h1>
  <h2>this is the first number sending here from python application. That is {{ number1 }}</h2>
  <h3>this is the second number sending here from python application. That is {{ number2 }}</h3>
</body>
</html>
```

18- Template klasorunun altina da **index.html** adli dosya olusturmustuk. Bu dosyanin bulundugu yere **body**.**html** isimli bir dosya daha olusturuyoruz ve asagidaki kodu yapistiriyoruz

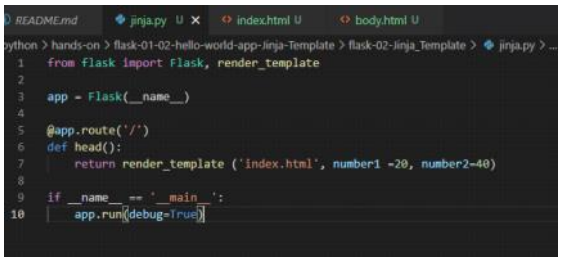
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>This is my first number {{ num1 }}</h1>
  <h1>This is my first number {{ num2 }}</h1>
  <h2>This is my first number {{ multiplication }}</h1>
</body>
</html>
```



Python dosyasi ile Template dosyasinin Ayni yerde olmasi onemli



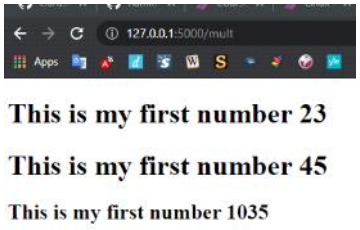
19-



Komutlarini py uzantili dosya icerisine yaziyoruz.
Index.html i kiraladik ve icerisine
Template atadik

20- Asagidaki komutlari tekrar yaziyoruz, kaydediyoruz ve calistirinca asagidaki goruntuyu aliyoruz.

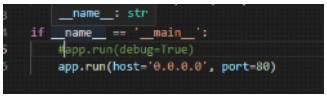
```
@app.route('/mult')
def number():
    var1, var2 = 23, 45
    return render_template('body.html', num1=var1 , num2=var2, multiplication= var1*var2)
```



- Yani body.html yi kiraladik
- Oradaki degiskenleri kullandik
- Flask ile framework' u kurduk
- Python ile calistirdik
- Ve sonunda internet sayfasi ayaga kalkti.

Python dosyasi ile template klasoru ayni dizinde olmali

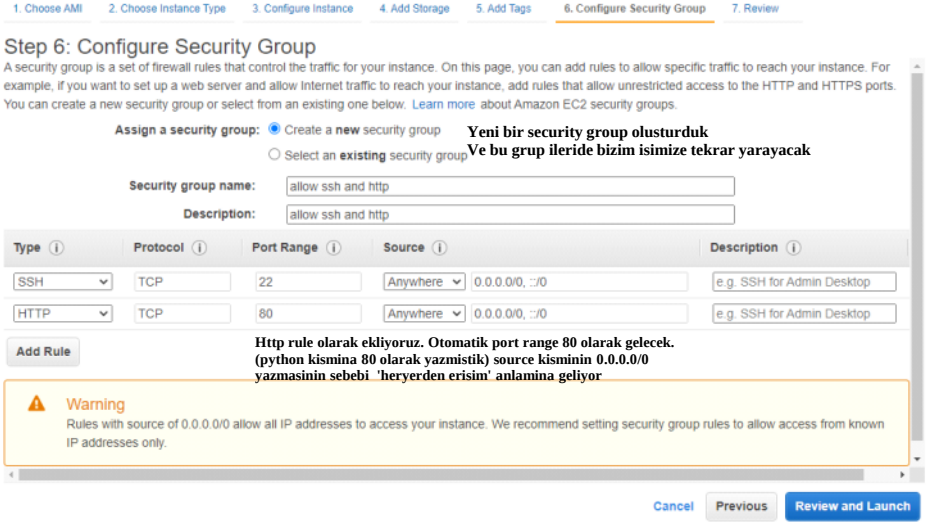
21- **Ec2 ile cloud** uzerinden calistirmak istiyorsak
`app.run(host='0.0.0.0', port=80)` ==> kodunu kullanmaliyiz ve bir ustekini comment haline getirmeliyiz



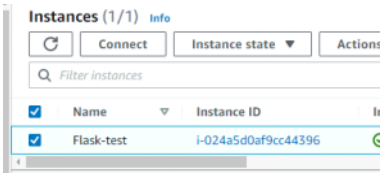
Ayni islemi daha once olusturdugumuz hello-world dosyasina da uygulamaliyiz

22- Yapmis oldugumuz islemlerin hepsini GITHUB' a atmak icin asagidaki komutlari sirayla gitbash e yaziyoruz;
- **Git status**
- **Git add .**
- **Git commit - m 'flask 01-02 added'**
- **Git push**

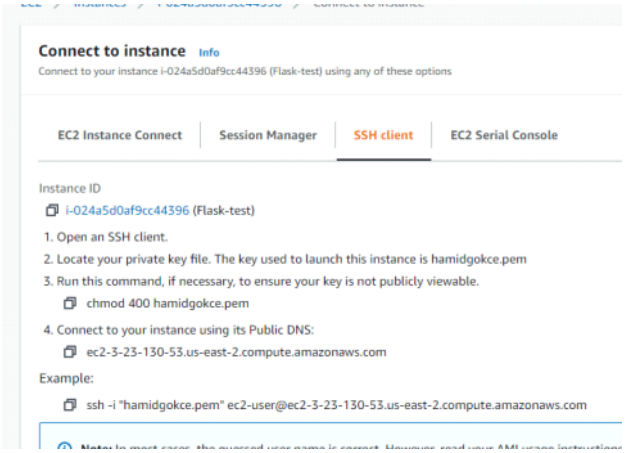
23- AWS hesabimizi aciyoruz. Bir EC2 makina calistirmaya calisacagiz.
Step 6 ya kadar butun secenekler default olarak kalacak. Step 6 da asagidaki gorseldeki degisiklikleri yapacagiz. Gerekirse tekrardan key.pair olusturabiliriz.



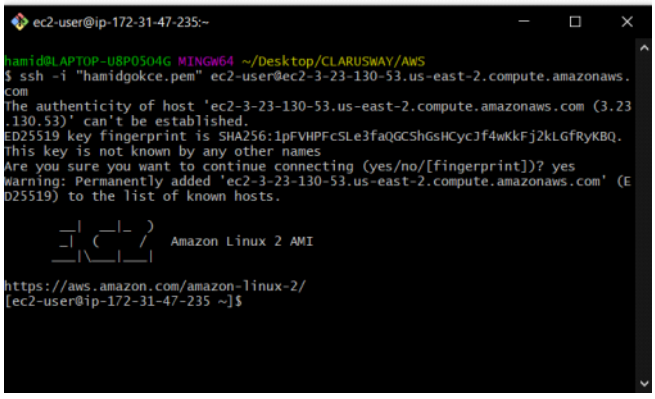
24- Key.pair lar neredeyse git bash i orada calistirmamiz gerekiyor veya uzantisini cd komutu ile girmemiz gerekiyor.



Connect secenegini tikla

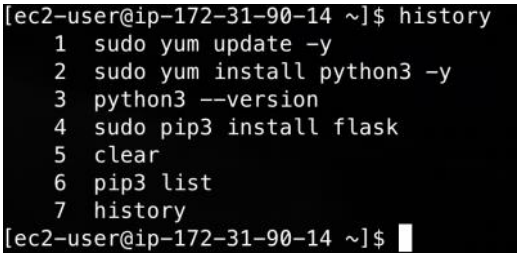


SSH client alanina git ve asagida bulunan Example daki ssh i kopyala actigimiz git bash e yapistir

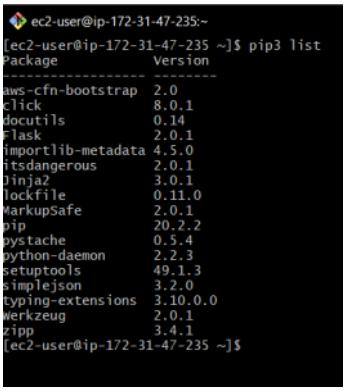


Kopyaladigimiz komutu buraya yapistirdik EC2 nun icerisine girdik

25-

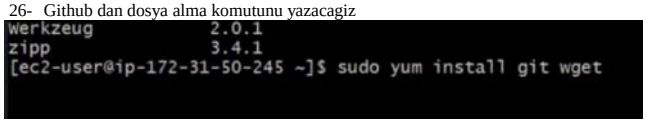


Komutlarini sirayla giriyoruz. Ilk komut herseyi update ediyor(1)

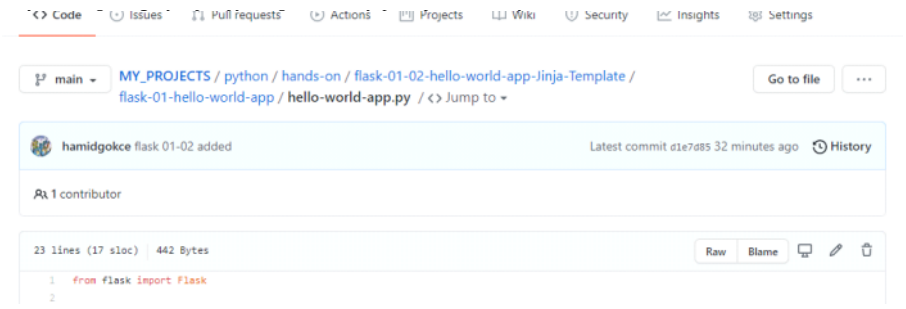


Flask in oldugunu gorebiliyoruz.

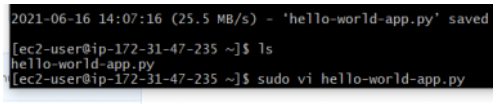
26- Github dan dosya alma komutunu yazacagiz



27- GIT HUB da asagidaki dizine giriyoruz ve **Raw** i tikliyoruz ve cikan pencereden adres cubugunu kopyaliyoruz. Git bash yardimi ile github dan dosya cekecegiz. **wget** yazip kopyaladigimiz kismi yapistirip enter a basiyoruz

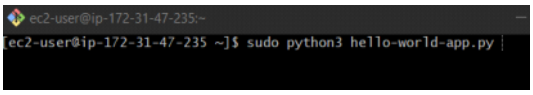


28-

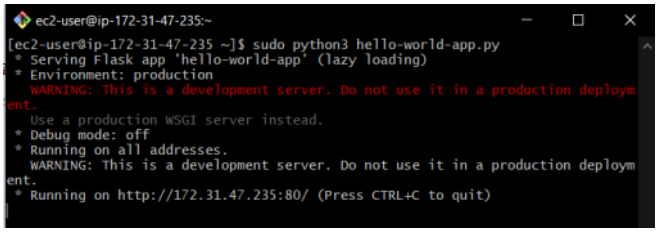


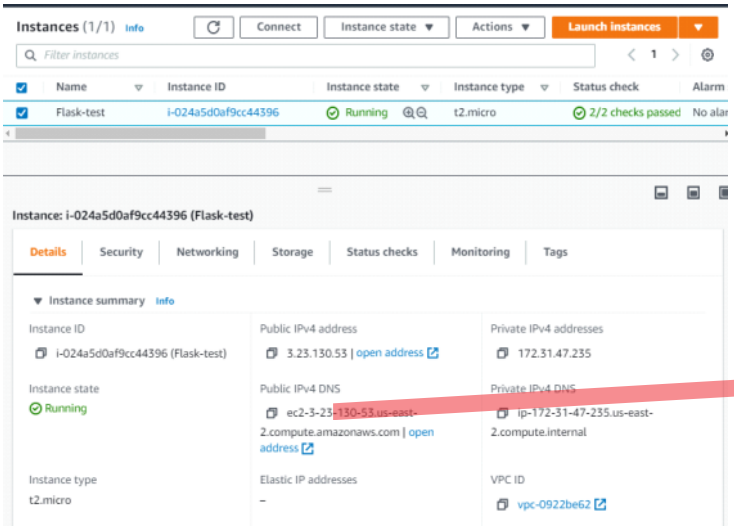
Komutlarini yazinca github daki dosyamizin EC2 makinasina geldigini goruyoruz. **Esc : wq** ==> cikilabilir

29-

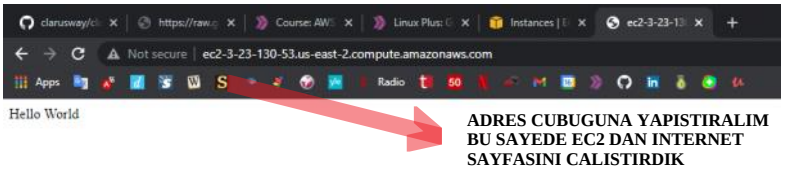


ENTER





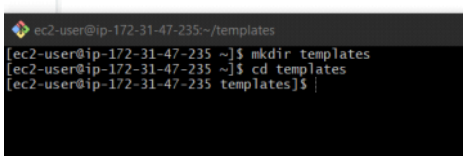
ADRESINI KOPYALAYALIM



ADRES CUBUGUNA YAPISTIRALIM BU SAYEDE EC2 DAN INTERNET SAYFASINI CALISTIRDIK

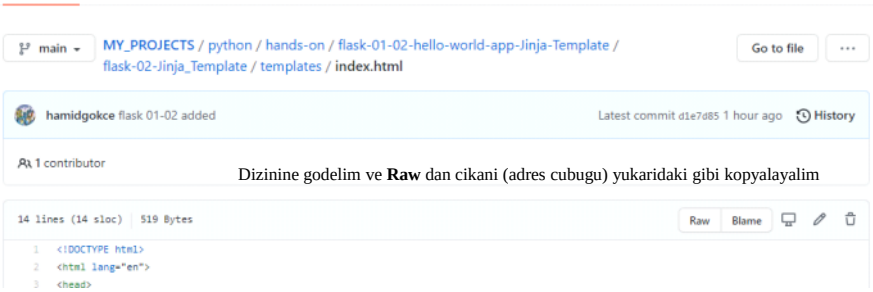
- **GITHUB REPOMUZDAN KOD CEKTIK**
- **CLOUD DA PC ACTIK (ISLETIM SISTEMIZIN NE OLDUGU ONEMLI DEGIL)**

30-



Komutlarini girelim.

31-

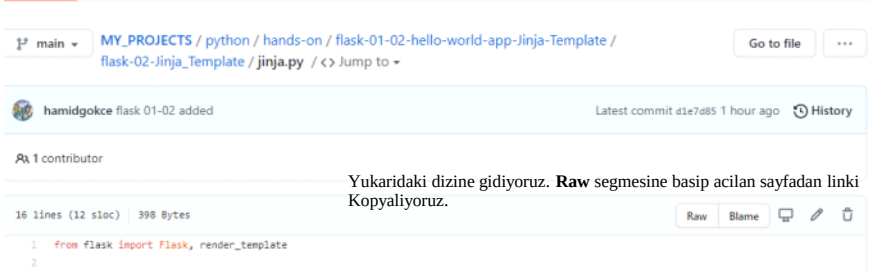


Dizinine godelim ve **Raw** dan cikani (adres cubugu) yukaridaki gibi kopyalayalim

32-

Wget yazip kopyaladigimizi yapistiriyoruz. Yukari tusuna basip yazdigimiz son koddaki index kismini silip body yaziyoruz

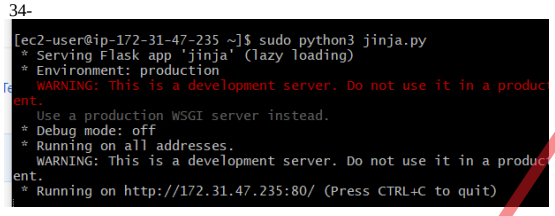
33-



Yukaridaki dizine gidiyoruz. **Raw** segmesine basip acilan sayfadan linki Kopyaliyoruz.

wget + link

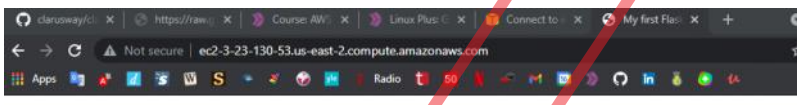
34-



Sudo python3 jinja.py ==> komutunu girelim

Asagidaki sayfamizi tekrar yenileyelim

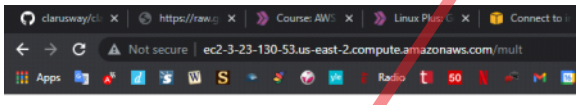
Daha sonra adres cubuguna iki sayinin carpimini gormek icin **mult** yazalim



This is my first webpage with html file

this is the first number sending here from python application. That is 20

this is the second number sending here from python application. That is 40

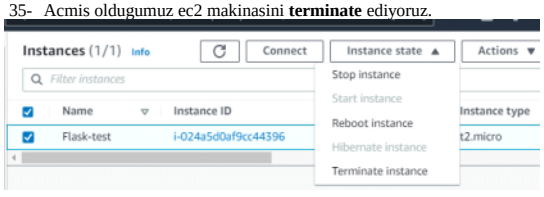


This is my first number 23

This is my first number 45

This is my first number 1035

35- Acmis oldugumuz ec2 makinasini **terminate** ediyoruz.



36- Son olarak biz ne yaptik :)

- Sahsi pc mizden flask frame worku kullanarak bir internet sayfasi olusturduk
- Buna decoreta dedigimiz kodlar ile mudahale ettik
- Ana sayfada ne cikmasini istediysek onu yazdik
- /second ile No pain no gain yazdirdik
- /third ile ile subthird yazdirdik
- Index.html, body.html den render_template kiraladik
- Template kiralarken degiskenlere deger verdik
- Butun bu isleri sahsi pc de yaptik
- Ec2 ayaga kaldirdik
- Gerekli programlari ec2 ya yukledik
- Daha sonra git hub repomuzdan o dosyalari ec2 ya cektik
- Ec2 uzerinden 80 portundan flask frameworkunu calistirdik
- Ec2 public adresinde sayfamizi gorebildik