

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 olusmasinda yordimci 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 karisik bir adres üzerinden baska bir sayfayi gosterme islemine URL eslestirme denir. Bu sayede asil 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 yordim istenebilir.

What and Why is Flask

Flask tamamiyla [Python](#) programlama dilinde yazilmiş bir web gelistirme 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

Türkçe'ye catı olarak

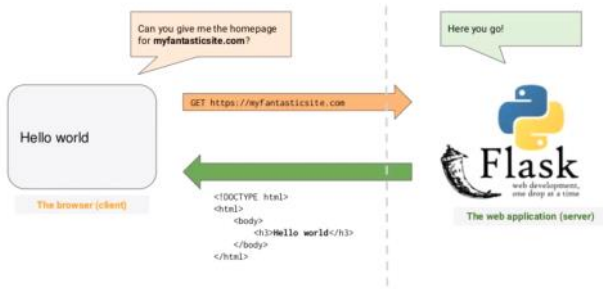
What and Why is Flask



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. İstedığınız 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 yapabilmek 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](#)

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

hello.py

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def index():
    return 'Hello Flask!'
if __name__ == '__main__':
    app.run()
```



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. İlk 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 yapılarla sahiplerdir. Yazımızda Flask'in micro framework yapısından bahsettik. Django ise bunun aksine içinde tüm gereksinimleri barındırır. Araba almak için gittiğimiz, bize tüm parçaları sunan yer gibi düşünün.

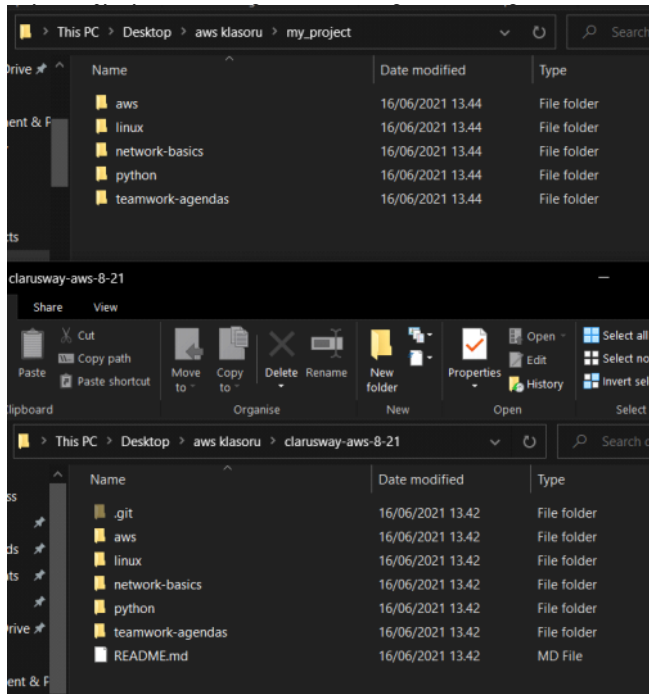
İlk farkı Flask'in micro framework yapısı olması. Bu sayede geliştirici istediği özelliği ekleyerek projesini geliştirebiliyor. Öte yandan büyük projelerde Django'nun tercih edildiğini görüyoruz. Bundan bir çıkarım yaparsak büyük projelerde hazır olarak sunulmuş ve tüm paket içeren bir framework yapısı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

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

2- Daha sonra istenilen bir yere MY_PROJECTS adi altında bir klasör oluşturalım ve klonladığımız repodaki .git ve README.md haricindeki dosyaları kopyalayalım. Oluşturduğumuz klasörler aşağıdaki şekildeki gibi olacaktır.



- 3- Localde 'MY_PROJECTS' adlı bir klasör oluşturmştuk ve içerisinde yukarıdaki 5 ayrı klasörü kopyalamıştık. Şimdi Github da aynı isimde yeni bir repository oluşturunuz. (*Repository name = my_projet, Description = 'This repo is build for all my portfolios and projects.', Public olacak ve README file işaretlenmeyecek.*)

Owner * Repository name *

hamidgokce / my_project

Great repository names are short and memorable. Need inspiration? How about [fuzzy-waffle?](#)

Description (optional)

This repo is build for all my portfolios and projects.

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

[Create repository](#)

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
```

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) https://github.com/hamidgokce/my_project.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

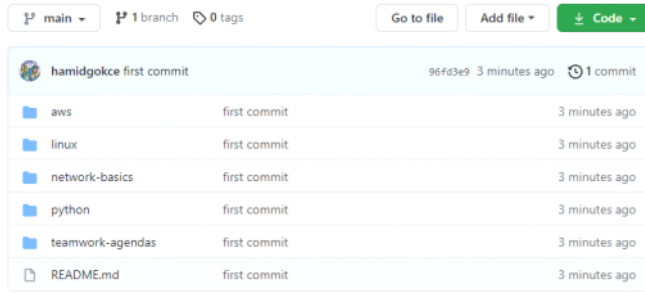
```
echo "# my_project" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/hamidgokce/my_project.git
git push -u origin main
```

Git add . Komutunu girebiliriz. Cunku bizim oluşturdugumuz dosya içerisinde Farklı klasörlerde mevcut. Hepsini eklememiz gerekmektedir.

...or push an existing repository from the command line

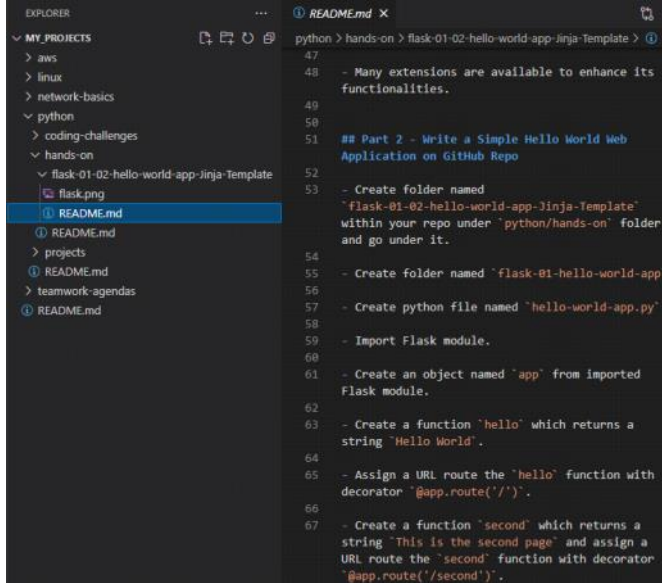
```
git remote add origin https://github.com/hamidgokce/my_project.git
git branch -M main
git push -u origin main
```

- 5- Push ettiğimiz bütün dosyaların Github repomuza eklendiğini göreceksiniz

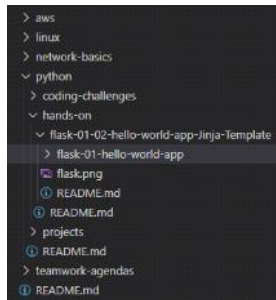


6- Olusturmus oldugumuz MY_PROJECTS klasorunu VS Code programinda acalim

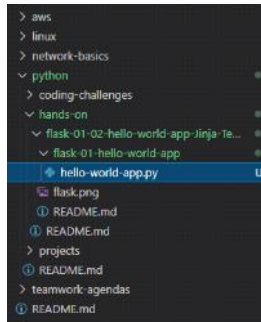
7-



8- Create folder named `flask-01-hello-world-app` (hands-on /flask-01-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

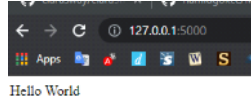
- 1- Flask import etmek
 - 2- Object olusturmak
 - 3- Olusturulan object kontrol ediliyor, calistiriyor.
- Geriye dogru giderek debug etmemize kontrol etmemize yariyor.

- 11- Öncelikle Hello World yazması için bir fonksiyon tanımlamak lazım. Bu fonksiyonu da internet ortamında çalıştırmak için ==> Flask te URL assign etmek gerekmektedir. Flask te bu işleme decorator deniyor ve `@app.route('/') yazılıyor`

```
hello-world-app-jinja-template > flask-01-hello-world-app > hello-world-app.py >
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def hello():
7     return 'Hello World'
8
9 if __name__ == '__main__':
10     app.run(debug=True)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + v ^ X

```
hamid@LAPTOP-USP9504G MINGW64 ~/Desktop/CLARUSWAY/AWS/PROJECTS/MY_P
ROJECTS (main)
$ C:/Users/hamid/AppData/Local/Programs/Python/Python39/python.exe
c:/Users/hamid/Desktop/CLARUSWAY/AWS/PROJECTS/MY_PROJECTS/python/ha
nds-on/flask-01-02-hello-world-app-jinja-template/flask-01-hello-wo
rld-app/hello-world-app.py
* Serving Flask app 'hello-world-app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a produc
tion deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 259-021-552
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [16/Jun/2021 15:03:29] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Jun/2021 15:03:29] "GET /favicon.ico HTTP/1.1" 404 -
```



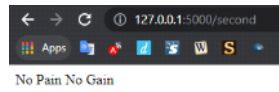
Yazılan kod run edildikten sonra terminalde <http://127.0.0.1:5000/> adresi çıkacak ve tıkladığımızda Hello World yazısını yukarıdaki gibi göreceğiz

Terminal ekranını

- CTRL + C ==> işlem sonlandırılır
- CTRL + L ==> sayfa temizlenir

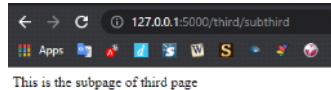
- 12- `@app.route('/second')`
`def second():`
`return 'No Pain No Gain'` ==> yazıyoruz ve yukarıdaki işlemi tekrarlıyoruz

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def hello():
7     return 'Hello World'
8
9 @app.route('/second')
10 def second():
11     return 'No Pain No Gain'
12
13
14 if __name__ == '__main__':
15     app.run(debug=True)
```



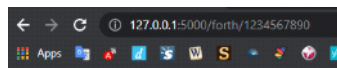
- 13- `@app.route('/third/subthird')`
`def third():`
`return 'This is the subpage of third page'` ==> yazıyoruz ve yukarıdaki işlemi tekrarlıyoruz

```
9 @app.route('/second')
10 def second():
11     return 'No Pain No Gain'
12
13 @app.route('/third/subthird')
14 def third():
15     return 'This is the subpage of third page'
16
17
```



- 14- `@app.route('/forth/<string:id>')`
`def forth(id):`
`return f'Id number of this page is {id}'` ==> yazıyoruz ve yukarıdaki işlemi tekrarlıyoruz

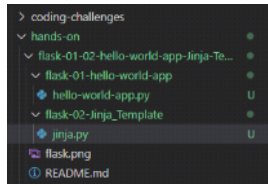
```
6
7 @app.route('/forth/<string:id>')
8 def forth(id):
9     return f'Id number of this page is {id}'
10
```



```
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)
```

Yazılan tüm komutlar

- 15- Aktığımız sayfaları kaydedip kapatıyoruz ve README.md dosyasındaki 88. aşamaya geçiyoruz. **flask-01-02-hello-world-app-Jinja-Template** klasörünün içerisine **flask-02-Jinja_Template** klasörünü ve daha sonra **jinja.py** dosyasını oluşturuyoruz.

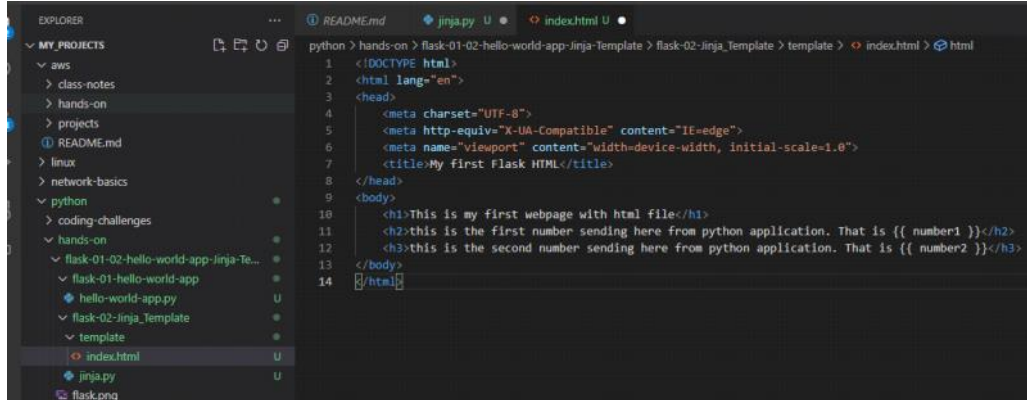


16-

```
01-02-hello-world-app-jinja-Template > flask-02-jinja_Template
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4
5 if __name__ == '__main__':
6     app.run(debug=True)
```

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

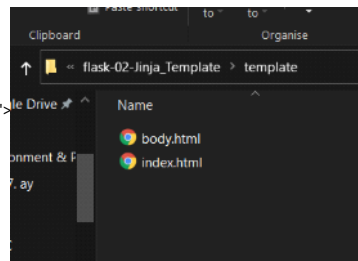
17- flask-02-Jinja_Template klasorunun altinda **template** adli bir folder olusturacagiz. **Template** klasorunun altina da **index.html** adli dosya olusturuyoruz. Daha sonra asagidaki kodu **index.html** dosyasinin icerisine yapistiriyoruz.



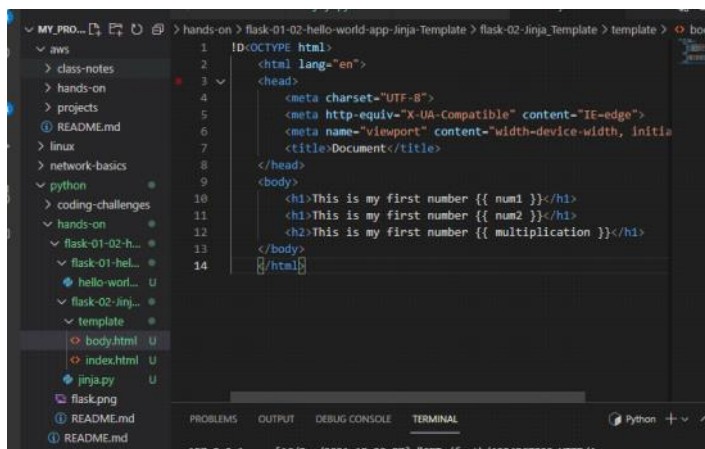
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My first Flask HTML</title>
</head>
<body>
  <h1>This is my first webpage with html file</h1>
  <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 bulunduгу 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-

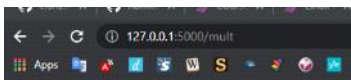
```
python > hands-on > flask-01-02-hello-world-app-jinja-template > flask-02-jinja-template > jinja.py > ...
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def head():
7     return render_template('index.html', number1=20, number2=40)
8
9 if __name__ == '__main__':
10     app.run(debug=True)
```

Komutlarını py uzantılı dosya içerisine yazıyoruz.
Index.html i kiraladık ve içerisine
Template atadık

20- Aşağıdaki komutları tekrar yazıyoruz, kaydediyoruz ve çalıştırınca aşağıdaki görüntüyü alıyoruz.

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

```
python > hands-on > flask-01-02-hello-world-app-jinja-template > flask-02-jinja-template > jinja.py > ...
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def head():
7     return render_template('index.html', number1=20, number2=40)
8
9 @app.route('/mult')
10 def number():
11     var1, var2 = 23, 45
12     return render_template('body.html', num1=var1, num2=var2, multiplication= var1*var2)
13
14 if __name__ == '__main__':
15     app.run(debug=True)
```



- Yani body.html yi kiraladık
- Oradaki değişkenleri kullandık
- Flask ile framework' u kurduk
- Python ile çalıştırdık
- Ve sonunda internet sayfası ayağa kalktı.

Python dosyası ile template klasoru aynı dizinde olmalı

21- Ec2 ile cloud üzerinden çalıştırmak istiyorsak

`app.run(host='0.0.0.0', port=80) ==>` kodunu kullanmalıyız ve bir üstekini comment haline getirmeliyiz

```
if __name__ == '__main__':
    app.run(debug=True)
    app.run(host='0.0.0.0', port=80)
```

Aynı işlemi daha önce oluşturduğumuz hello-world dosyasına da uygulamalıyız

22- Yapmış olduğumuz işlemlerin hepsini GITHUB' a atmak için aşağıdaki komutları sırayla gitbash e yazıyoruz;

- Git status
- Git add .
- Git commit -m 'flask 01-02 added'
- Git push

23- AWS hesabımızı açıyoruz. Bir EC2 makina çalıştırmaya çalışacağız.

Step 6 ya kadar bütün seçenekler default olarak kalacak. Step 6 da aşağıdaki görseldeki değişiklikleri yapacağız. Gerekirse tekrardan key.pair oluşturabiliriz.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group **Yeni bir security group oluşturduk**
☐ Select an existing security group **Ve bu grup ileride bizim ismimize tekrar yarayacak**

Security group name:

Description:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere 0.0.0.0/0 ::0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere 0.0.0.0/0 ::0	e.g. SSH for Admin Desktop

Add Rule

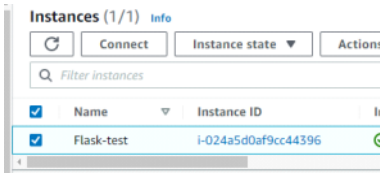
Http rule olarak ekliyoruz. Otomatik port range 80 olarak gelecek. (python kısmına 80 olarak yazmistik) source kısmının 0.0.0.0/0 yazmasının sebebi 'her yerden erisin' anlamına geliyor

Warning

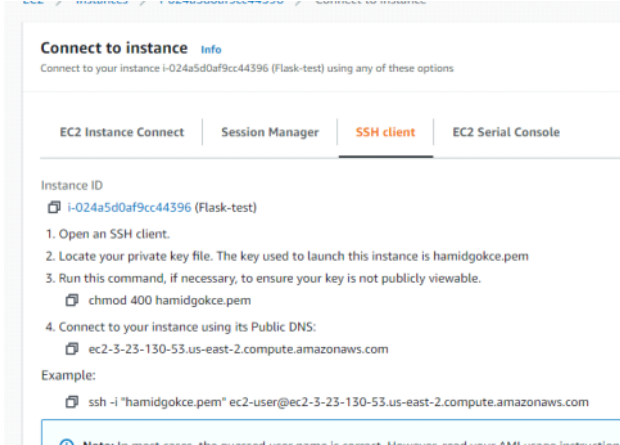
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Cancel](#) [Previous](#) [Review and Launch](#)

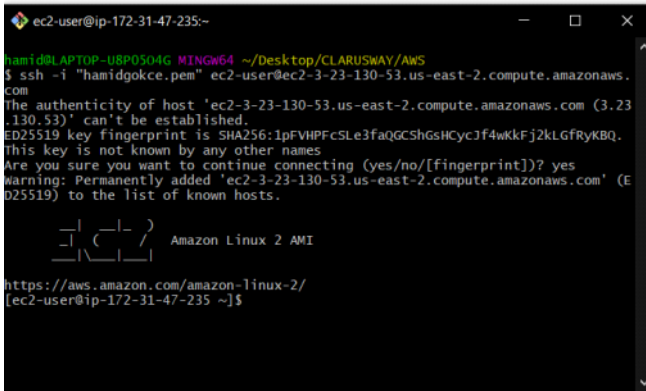
24- Key.pair lar neredeyse git bash i orada çalıştırmamız gerekiyor veya uzantısını cd komutu ile girmemiz gerekiyor.



Connect secenegini tikla

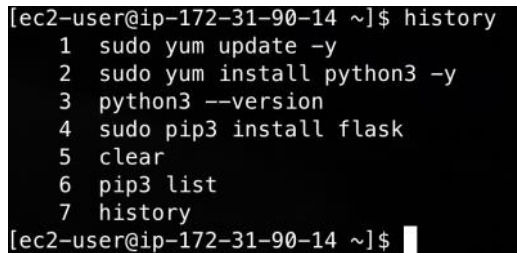


SSH client alanina git ve asagida bulunan Example daki ssh i kopyala actigimiz git bash e yapistir

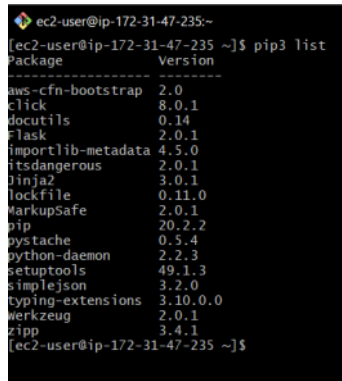


Kopyaladigimiz komutu buraya yapistirdik EC2 nun icerisine girdik

25-

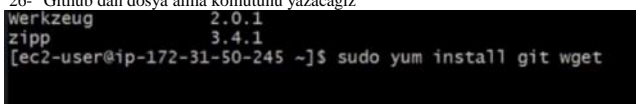


Komutlarini sirayla giriyoruz. Ilk komut herseyi update ediyor(1)

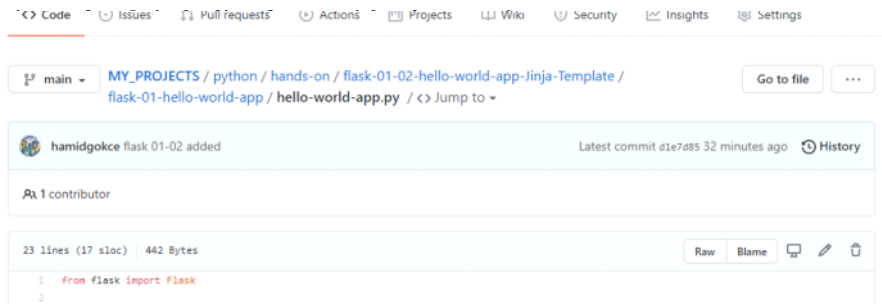


Flask in oldugunu gorebiliyoruz.

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 kısmi yapistirip enter a basiyoruz



28-

```
2021-06-16 14:07:16 (25.5 MB/s) - 'hello-world-app.py' saved
[ec2-user@ip-172-31-47-235 ~]$ ls
hello-world-app.py
[ec2-user@ip-172-31-47-235 ~]$ sudo vi hello-world-app.py
```

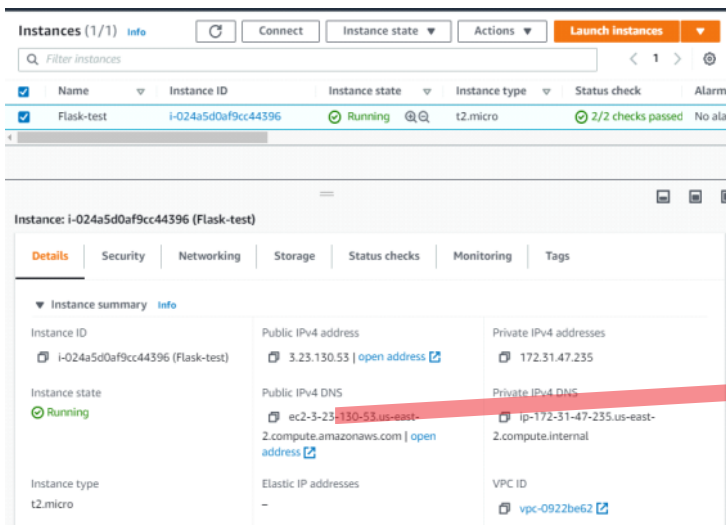
Komutlarini yazinca github daki dosyamizin EC2 makinasina geldigini goruyoruz. **Esc : wq** ==> cikilabilir

29-

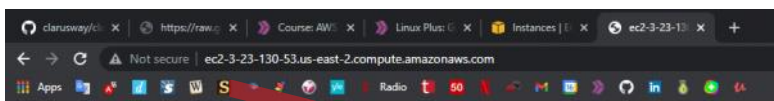
```
ec2-user@ip-172-31-47-235:~
[ec2-user@ip-172-31-47-235 ~]$ sudo python3 hello-world-app.py
```

ENTER

```
ec2-user@ip-172-31-47-235:~
[ec2-user@ip-172-31-47-235 ~]$ sudo python3 hello-world-app.py
* Serving Flask app 'hello-world-app' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.31.47.235:80/ (Press CTRL+C to quit)
```



ADRESINI KOPYALAYALIM



Hello World

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-

```
ec2-user@ip-172-31-47-235:~/templates
[ec2-user@ip-172-31-47-235 ~]$ mkdir templates
[ec2-user@ip-172-31-47-235 ~]$ cd templates
[ec2-user@ip-172-31-47-235 templates]$
```

Komutlarini girelim.

31-



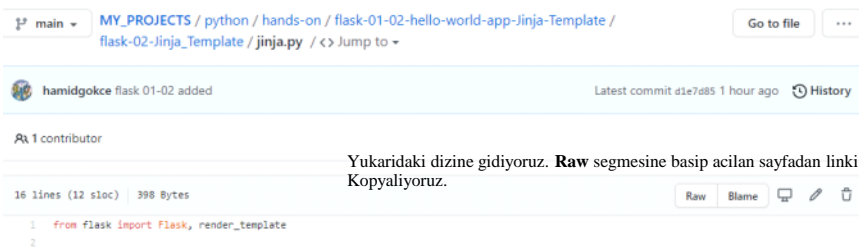
32-

```
ec2-user@ip-172-31-47-235:~/templates
[ec2-user@ip-172-31-47-235 ~]$ mkdir templates
[ec2-user@ip-172-31-47-235 ~]$ cd templates
[ec2-user@ip-172-31-47-235 templates]$ wget https://raw.githubusercontent.com/hamidgokce/MY_PROJECTS/main/python/hands-on/flask-01-02-hello-world-app-Jinja-Template/flask-02-Jinja_Template/templates/body.html
```

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

```
[ec2-user@ip-172-31-47-235 templates]$ wget https://raw.githubusercontent.com/hamidgokce/MY_PROJECTS/main/python/hands-on/flask-01-02-hello-world-app-Jinja-Template/flask-02-Jinja_Template/templates/body.html
```

33-



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

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-47-235 ~]$ wget https://raw.githubusercontent.com/hamidgokce/MY_PROJECTS/main/python/hands-on/flask-01-02-hello-world-app-Jinja-Template/flask-02-Jinja_Template/jinja.py
```

wget + link

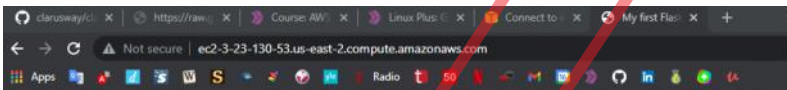
34-

```
[ec2-user@ip-172-31-47-235 ~]$ sudo python3 jinja.py
* Serving Flask app 'jinja' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production
environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production
environment.
* Running on http://172.31.47.235:80/ (Press CTRL+C to quit)
```

Sudo python3 jinja.py ==> komutunu girelim

Asagidaki sayfamizi tekrar yenileyelim

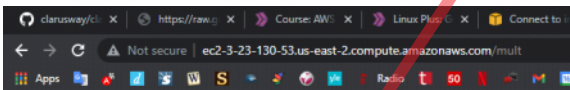
Daha sonra adres cubuguna iki sayinin carpimini gormek icin **mult** yazalim



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

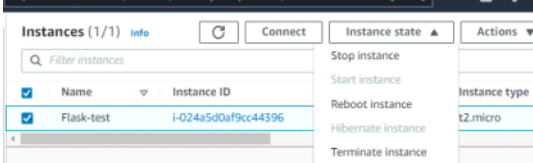


This is my first number 23

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 decorate dedigimiz kodlar ile mudahale ettik
- Ana sayfada ne cikmasini istediysek onu yazdik
- /second ile No pain no gain yazdirdik
- /third 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