AWS Lambda & API Gateway

Wednesday, 11 August 2021 19.27

AWS Lambda &
API Gateway

Table of Contents

- Introduction to Serverless
- AWS Lambda
- API Gateway

Introduction to Serverless

What is Serverless?

- Lambda **(FaaS-Function as a Service)**
- DynamoDB
- Aurora Serverless
- API Gateway
- SQS

Serverless : Serverin olmadigi bir yapi ==> Cunku server kismi bizi ilgilendirmiyor. Amac serverin isini server imkanlarini saglayan provider in yapmasi

Lamda; farkli dillerde run time enviromentleri bize sunar biz kodu yukleriz ve lambda herseyi ile ilgilenmektedir.

Introduction to Serverless Infrastructure

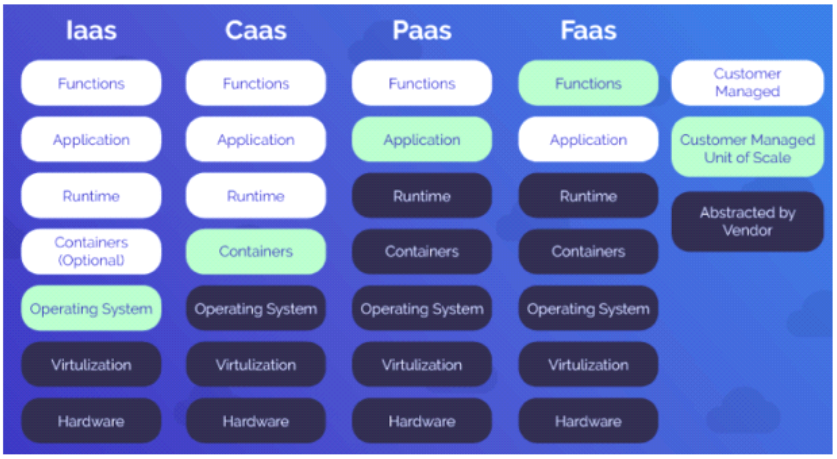
What is Serverless Infrastructure and Why?

- Run applications **without servers.**
- **Eliminate infrastructure management**
- Cost effective

Amac : Application lar server a ihtiyac duymadan handle

edilmesi

✓ Serverless-Function as a Service (FaaS)



FaaS : Sadece fonksiyonu calistiracak servisin satin alinmasi
==> LAMBDA
DynamoDB ==> Serverless (Bize hazir bir sistem servis gelmektedir)

As you remember from the Cloud Computing Basics, there are **3** types of Cloud Service model; **IaaS**(Infrastructure as a Service), **PaaS**(Platform as a Service) and **SaaS**(Software as a Service).

Recently, cloud providers have improved the **PaaS** service and developed the **FaaS** service, which allows only customers' code to run as a function. **FaaS** is also called **Serverless**.

It is a category of cloud computing services that provides a platform allowing
customers to develop, run, and manage application functionalities.

AWS Lambda is the first **FaaS** offering by a global public cloud provider.

AWS Lambda

Cloud providerlar arasinda ilk FaaS i kullanima sunan ==> AWS

s

✓What is Lambda?

AWS Lambda

AWS Lambda is a **serverless compute service** that runs your code in response to events and **automatically** manages the underlying computing resources for you.

Instead of launching an EC2 instance to run your code on it, you can just deploy your code in Lambda services and you can get the same result. AWS Lambda lets you run code without provisioning or managing servers.

But, what makes lambda valuable is the **trigger function**. Thanks to the trigger function, Lambda **automatically** operates the code you deploy in it. After you upload your code to AWS Lambda, you can associate your function with specific AWS resources (e.g. a particular **Amazon S3 bucket** or **Amazon SNS notification**). Then, when the resource changes, Lambda will execute your function and

manage the computing resources as needed in order to keep up with incoming requests.

In the Lambda service, you will be charged only the functions that you deployed are run. So, you pay only for the compute time you consume. But when you prefer to run your code in EC2 instance you will be charged as long as your instance is running even if your code runs or not.

Lambda also natively supports Java, Go, PowerShell, Node.js, C#, Python, and Ruby code, and provides a Runtime API which allows you to use any additional programming languages to author your functions.

✓ How does Lambda work?

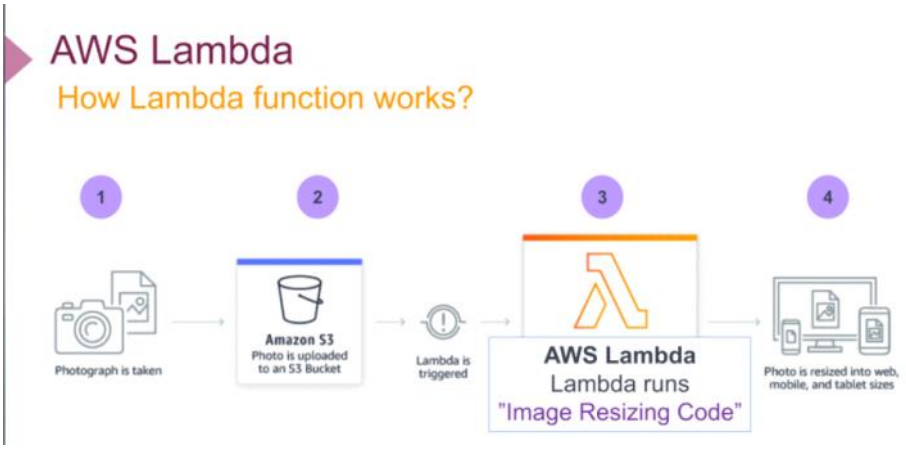
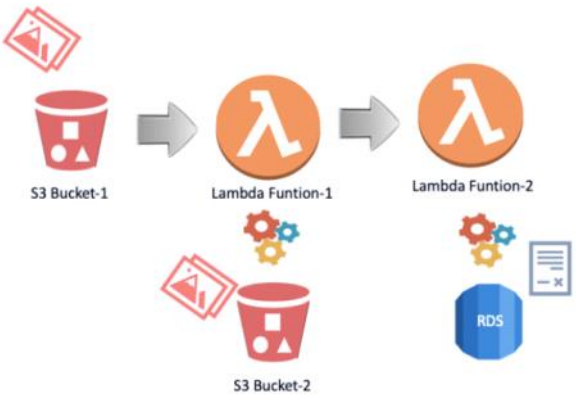


Foto uzerinden ornekleme



Lambda Workflow

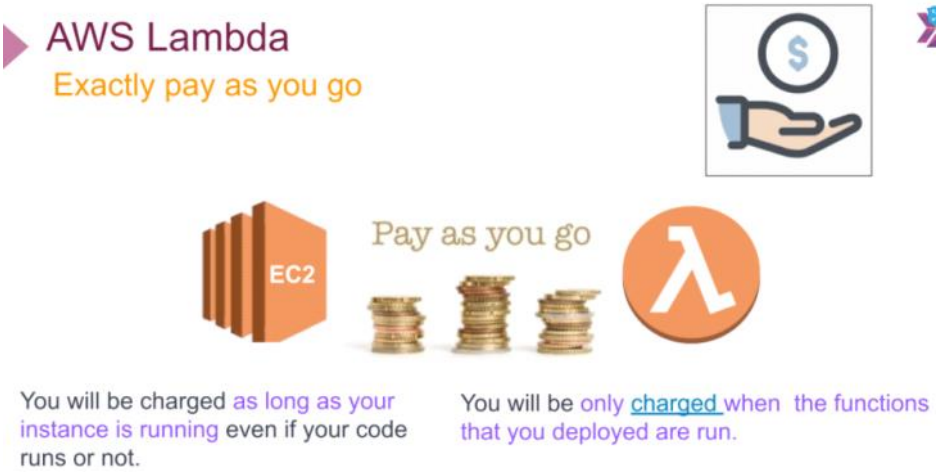
Let's see how Lambda function works with the example as you see in the picture above,

- First, the user uploads a file to S3 Bucket-1,
- This event causes a trigger for Lambda Function-1.
- Then the Lambda-1 function starts to run. This function provides to send a copy of the uploaded file to the S3 Bucket-2
- When the copied file sends to S3 Bucket-2 it triggers Lambda Function-2.
- Lambda Function-2 starts to run. Lambda Function-2 provides to write the log record to the RDS Database.

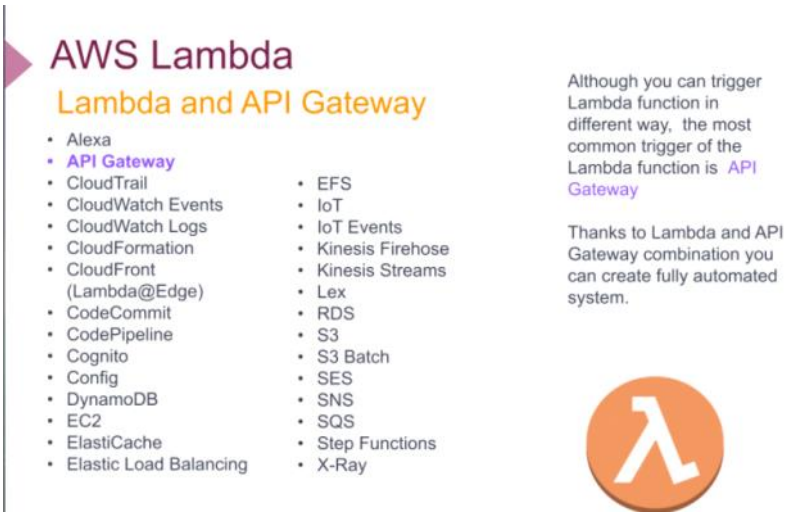
As you see, the Lambda function can be triggered by custom events generated by your applications/ devices or another Lambda

Function.

You are charged based on the number of requests for your functions and the duration, the time it takes for your code to execute.



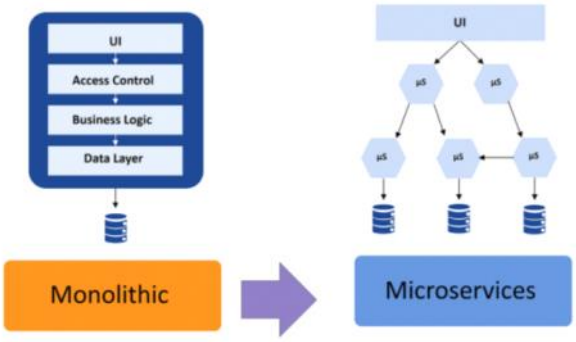
KAC KERE CAGIRDIK
CAGIRDIGIMIZI NE KADAR SURELIGINE KULLANDIK
(COK KISA SURELI KULLANIMLAR ICIN UYGUN / 15 DK FAZLASINI CALISTIRMAMAKTADIR)
https://aws.amazon.com/lambda/pricing/



API Gateway

Amazon **API Gateway**, yazılım geliştiriciler tarafından istenen ölçekte **API** yayımlanmasını, **API**'lerin izlenmesini, bakımının yapılmasını, güvenliğinin sağlanmasını ve çalıştırılmasını mümkün kılan, tam olarak yönetilen bir hizmettir.

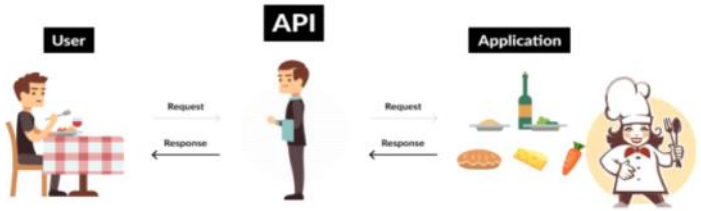
API Gateway



Monolitic : tek dilde yazilmaktadir
Microservices : bir kac dil kullanilmaktadir

API Gateway

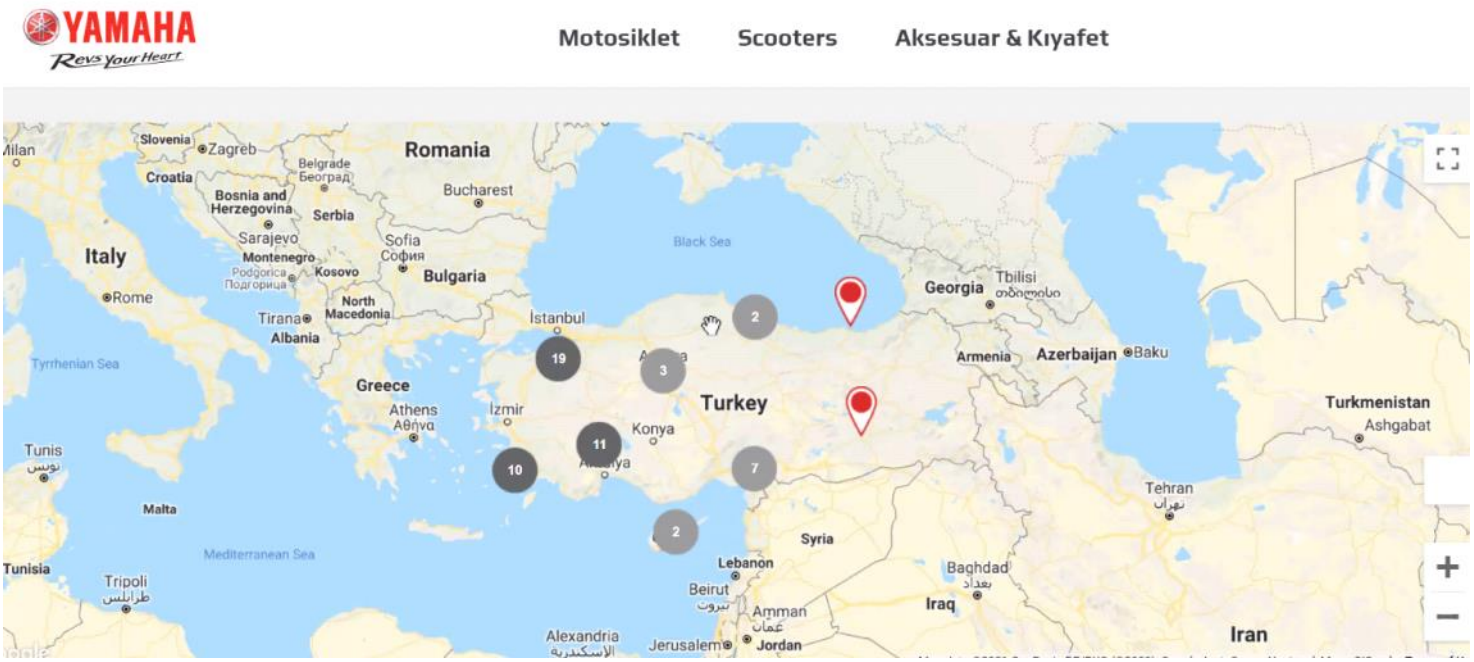
What is API?



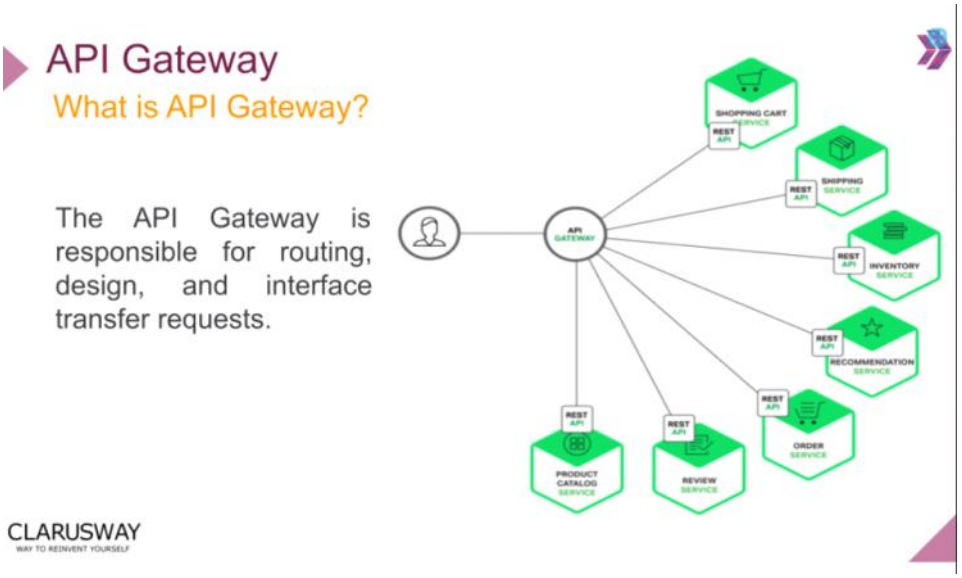
API stands for Application Program Interface. Basically, the API defines how device components can communicate with each other.

API lar birbirleriyle json dilinde konusur
'uygulamalarin birbirleriyle konusmasi icin bir arayuz diyebiliriz'

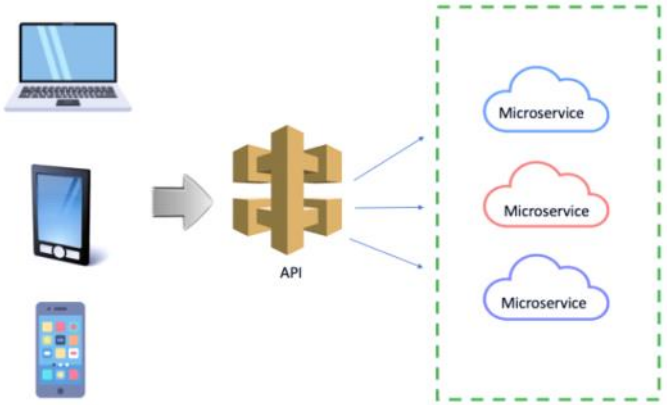
Her web servisi bir Web Api dir, ancak Web Api ler web servisi değildir. Yani Web Api web servislerini kapsar. ... Web servisler sadece üç hizmeti kullanabilirler SOAP, REST ve XML-RPC. Web api ler için herhangi bir kısıt yoktur aslında.



Yamaha google map sin API sini kullanmaktadir



✓[What is API Gateway?]

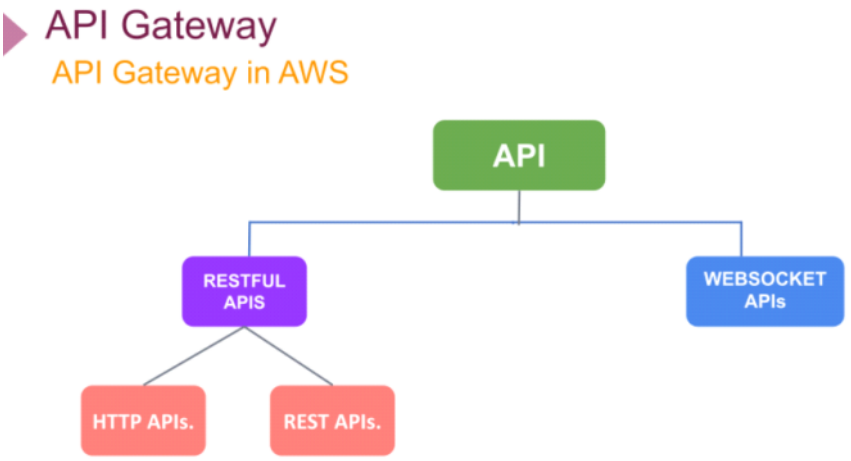


API Gateway

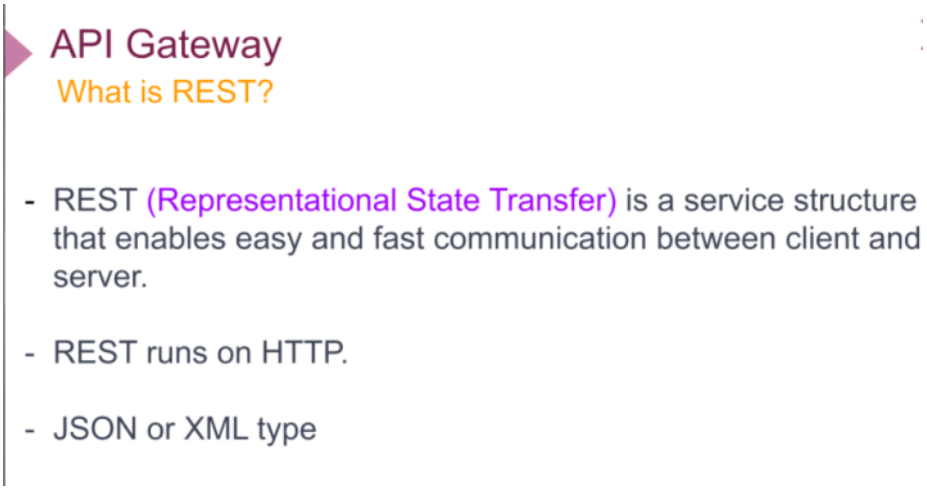
API stands for **Application Program Interface**. Basically, the API defines how device components can communicate with each other.

The API Gateway is responsible for routing, design, and interface transfer requests. All application requests first go through the

API gateway. It then sends a message to the correct microservice. The API Gateway can also process a request by invoking(cagirmak, yardim istemek) several microservices and aggregating the output.



Websocket APIs : iki server arasinda full dublex communication kuruyor.



✓ API Gateway in AWS

Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing **REST**, **HTTP**, and **WebSocket APIs** at any scale.

APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications.

Amazon API Gateway offers 3 options to create RESTful APIs, HTTP APIs, REST APIs, and WebSocket APIs.

- **HTTP API:** HTTP APIs are optimized for building APIs that proxy to AWS Lambda functions or HTTP backends, making them ideal for serverless workloads. They do not currently offer API management functionality.
- **REST API:** REST APIs offer API proxy functionality and API management features in a single solution. REST APIs offer API management features such as usage plans, API keys, publishing, and monetizing APIs.
- **WebSocket API:** WebSocket APIs maintain a persistent connection between connected clients to enable real-time message communication such as chat apps and streaming dashboards.

API Gateway

What is RESTful ?

- Web services that use REST architecture are called RESTful services.

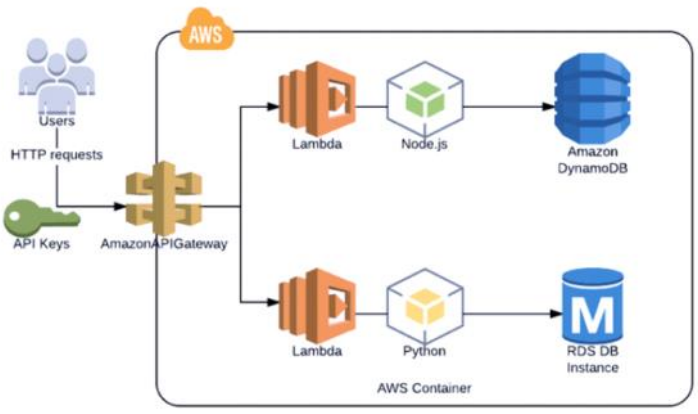
- GET,
- POST,
- PUT,
- DELETE

API Gateway

Pricing

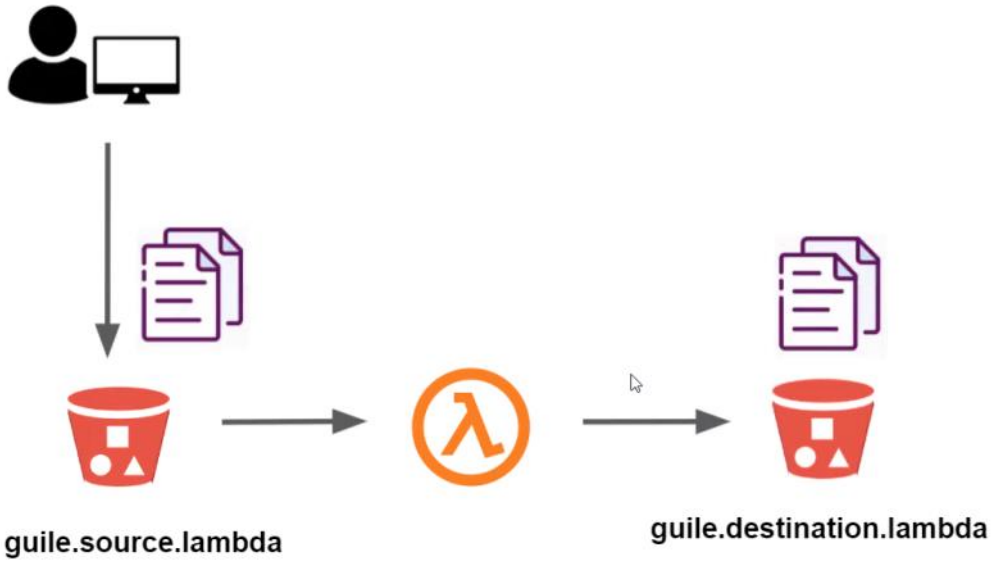
- With Amazon API Gateway, you only pay when your APIs are in use. There are no minimum fees or upfront commitments.

✓ Lambda and API Gateway



As we mentioned in previous lessons, you can trigger Lambda function in different ways. But the most common usage is to set the API Gateway as a Lambda function trigger. And usually, the Lambda function triggers the other AWS resources as you see into the

picture above. So, you are able to create a fully automated environment with Lambda and API Gateway combination.



Part 1 - Prep - Creating a S3 Bucket

```
- Create a bucket of `clarusway.source.lambda` with following properties,

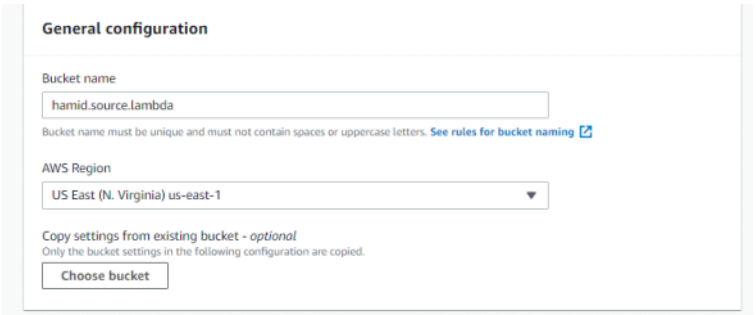
```text
Bucket name                 : clarusway.source.lambda
Region                      : N.Virginia
Block all public access     : Checked (KEEP BlOCKED)
Versioning                  : Disabled
Tagging                     : 0 Tags
Default encryption          : None
Object-level logging        : Disabled
```
PS: Please, do not forget to select "US East (N.Virginia)" as Region

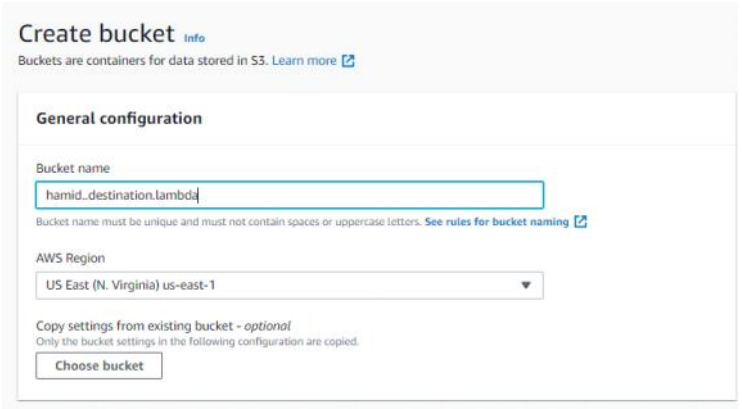
- Create another bucket of `clarusway.destination.lambda` with following properties,

```text
Bucket name                 : clarusway.destination.lambda
Region                      : N.Virginia
Block all public access     : Checked (KEEP BlOCKED)
Versioning                  : Disabled
Tagging                     : 0 Tags
Default encryption          : None
Object-level logging        : Disabled
```

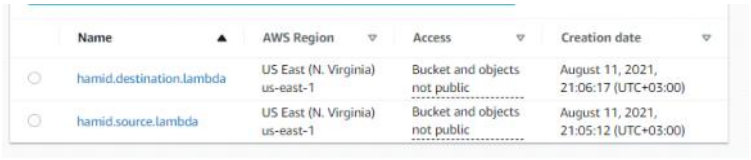
PS: Please, do not forget to select "US East (N.Virginia)" as Region
```

2 tane bucket olusturalim

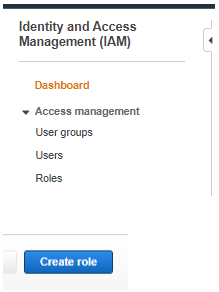


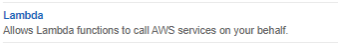


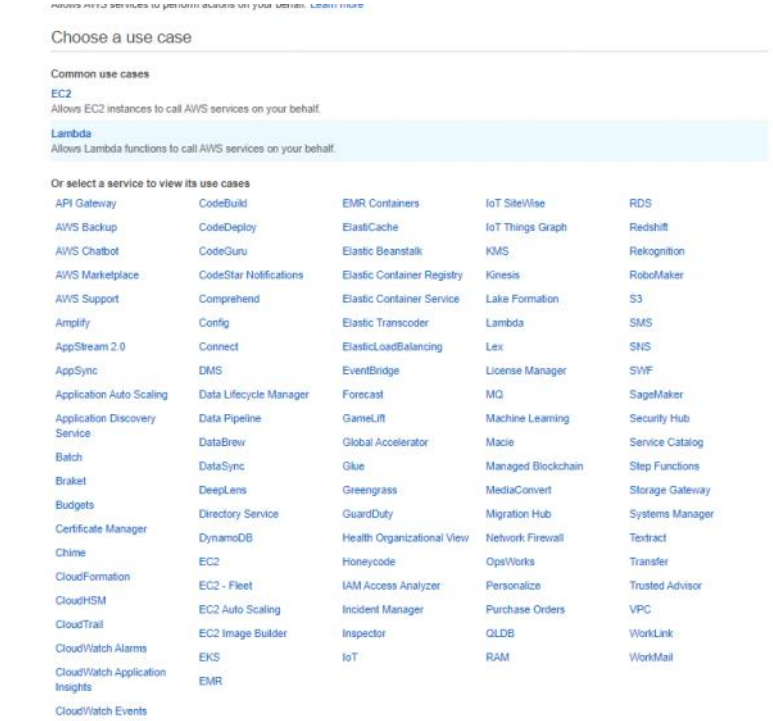
Olusturulan bucketlar

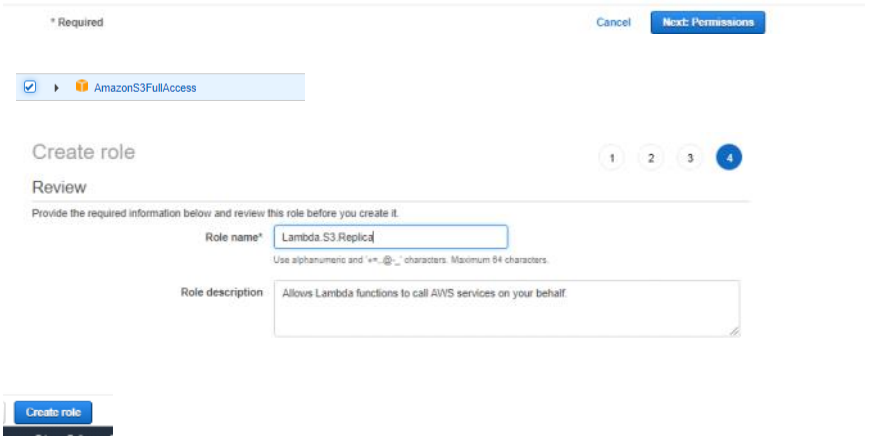


Lambda nin s3 ye ulasmasi icin rol olusturacagiz

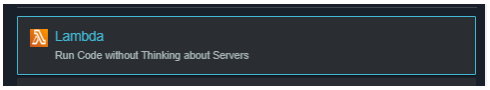


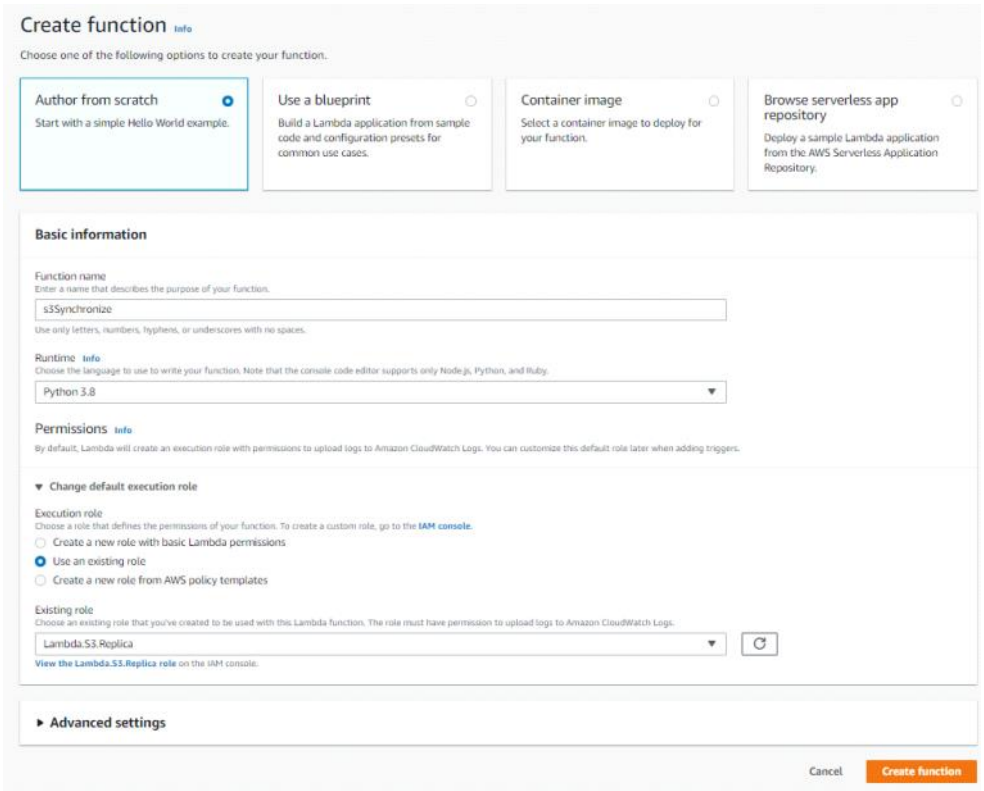






Lambda servisine gecelim





1- kodunuzu kendiniz yaziyorsunuz
2- Hazir kodlar
3- Lambda runtime API olan container lari calistiriyor

Lambda ya rol verecegiz : bizim adimiza s3 ye gidip ve bir seyler yazacak

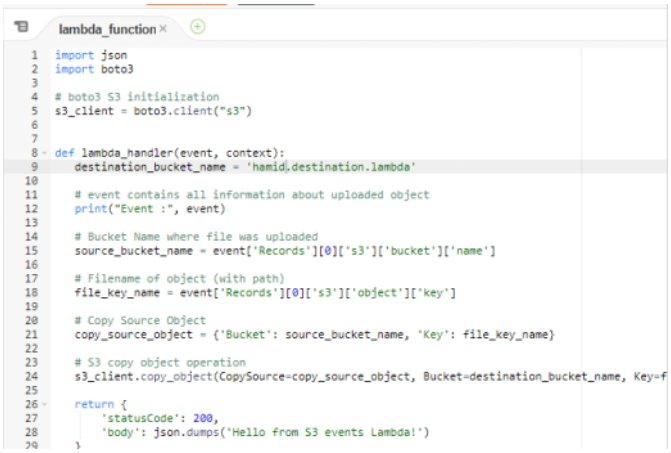
Default bir lambda function ile birlikte geliyor. Kodu yazacagimiz kisim asagida

Python kodunu yapistiralim (destination bucket ismine dikkat)

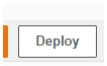
```
import json # degisiklikler icin json import ediyor
import boto3 # CLI benzer ama cok fazlasini python koduyla yapabiliriz
# boto3 S3 initialization
s3_client = boto3.client("s3")

def lambda_handler(event, context): # s3 buckete bir sey konulmasi == event
   destination_bucket_name = 'clarusway.destination.lambda' # bucket name giri
lecek
   # event contains all information about uploaded object
   print("Event :", event)
   # Bucket Name where file was uploaded
   source_bucket_name = event['Records'][0]['s3']['bucket']['name']
   # Filename of object (with path)
   file_key_name = event['Records'][0]['s3']['object']['key']
   # Copy Source Object
   copy_source_object = {'Bucket': source_bucket_name, 'Key': file_key_name}
   # S3 copy object operation
   s3
_client.copy_object(CopySource=copy_source_object, Bucket=destination_bucket_n
```

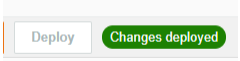
```
ame, Key=file_key_name)
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from S3 events Lambda!')
    }
```



Deploy etmemiz gerekir



Deploy edilmis hali



Tetikleme islemi : lambda fonksiyonu ne yaparsak calisacak 's3 buckete bir seyler yuklenirse otomasyon yapilsin gibi'

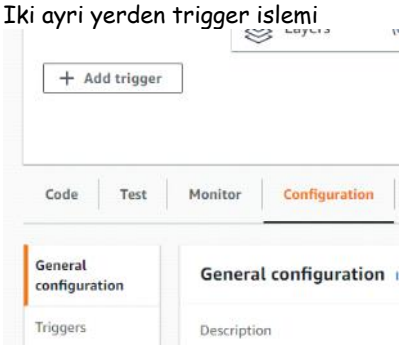
```
Trigger Configuration : S3

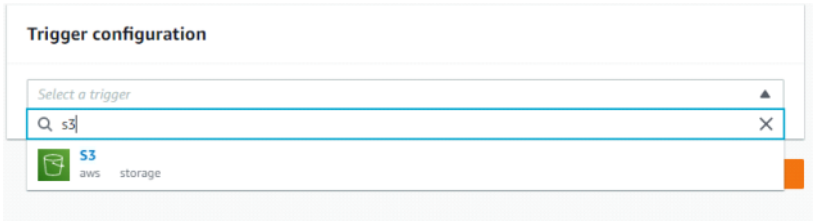
- Bucket              : clarusway.source.lambda

- Event Type          : All object create events

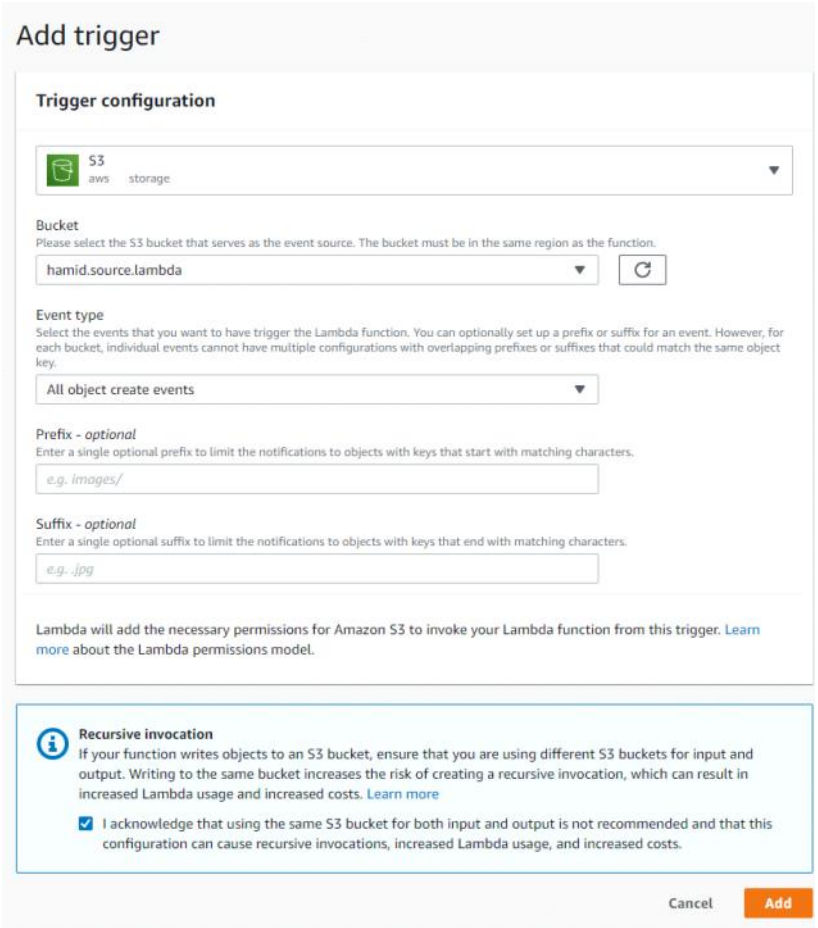
- Acknowledge         : checked
```

Iki ayri yerden trigger islemi





Event i tetikleyecek bucket source

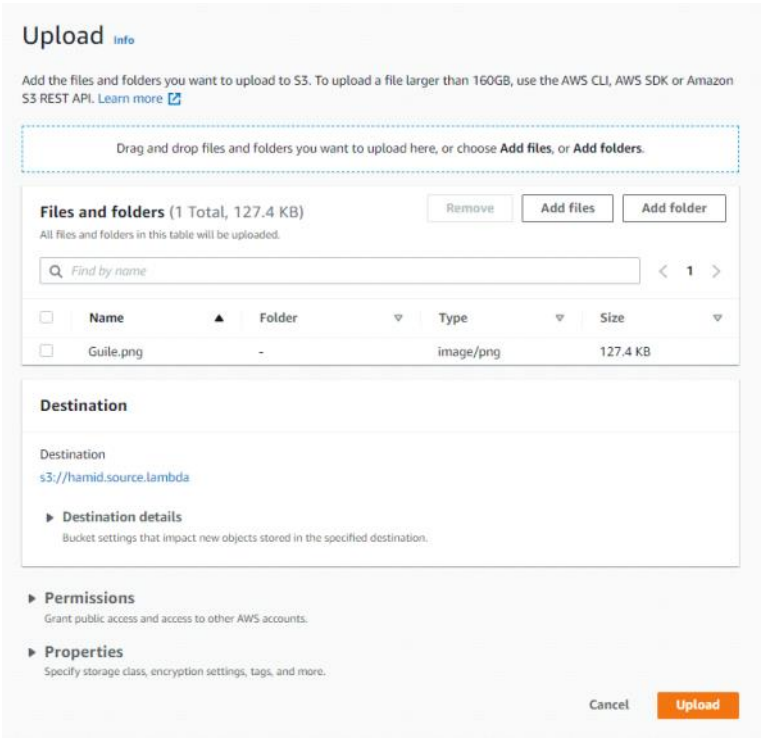


Recursive invocation : Eger kopylama gibi bir islem yapacaksan kesinlikle input ile output yapacagin bucketlari secme(karistirma) demektedir.

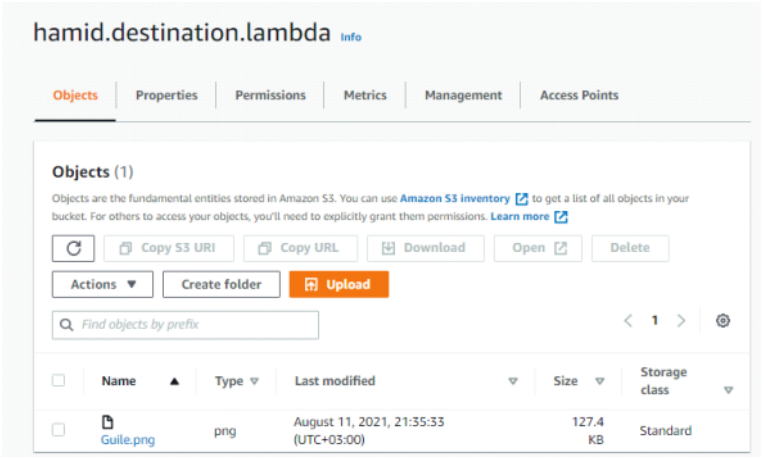
BlackHole

Butun sistemi kurduk ve deneme**(test)** kismi kalmis oldu

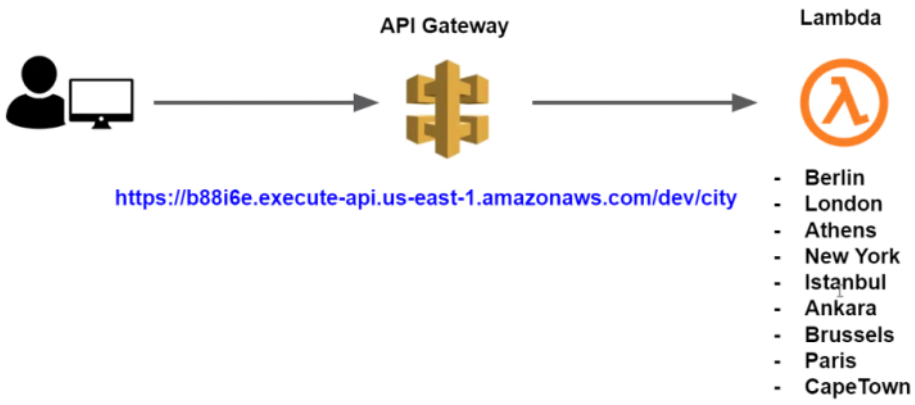
Source ye bir seyler atinda destinationa da gitmesi gerektigini test edelim ve rastgele bir dosya yukleyelim



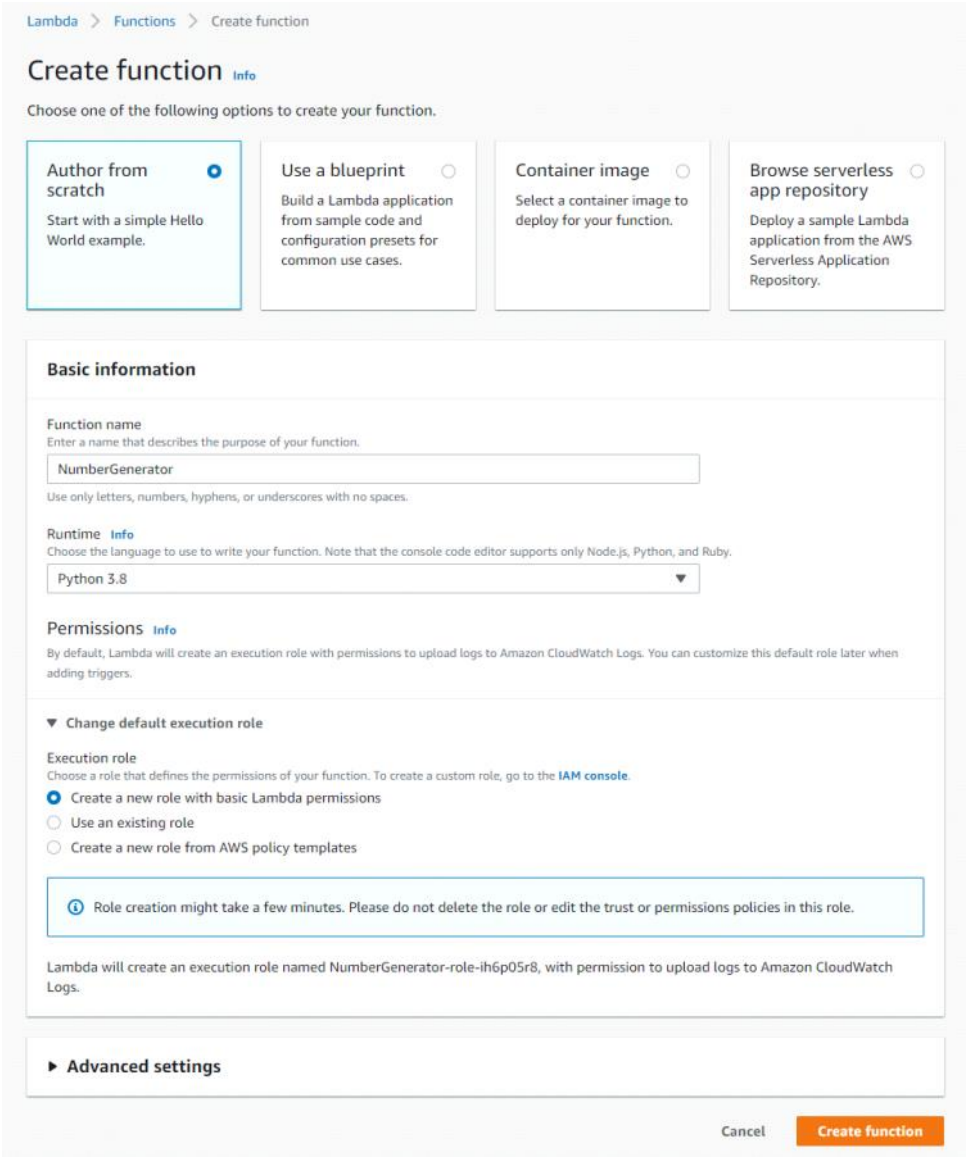
Asagida destinationa geldigini gorebiliriz



****



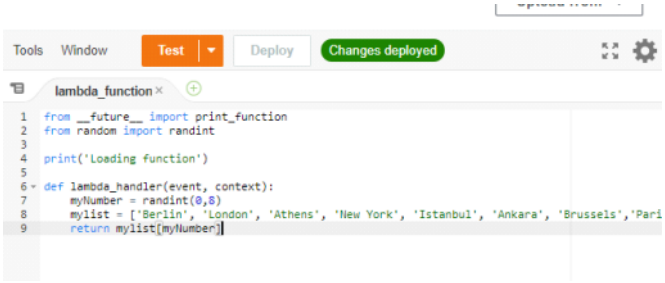
API lambda yi trigger edecek



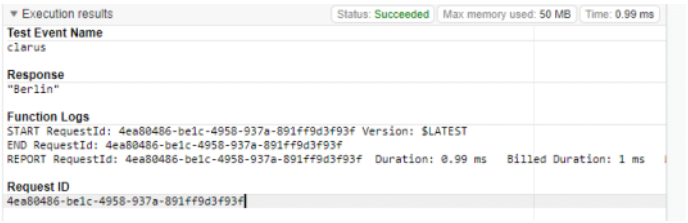
Az onceki gibi bir rol atamasi yapmayacak

```
from __future__ import print_function
from random import randint
print('Loading function')
def lambda_handler(event, context):
    myNumber = randint(0,8)
    mylist = ['Berlin', 'London', 'Athens', 'New York', 'Istanbul', 'Ankara',
'Brussels','Paris','CapeTown']
    return mylist[myNumber]
```

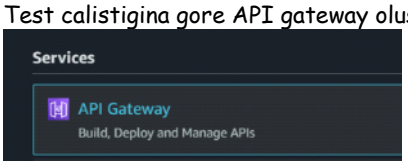
Deploy edelim

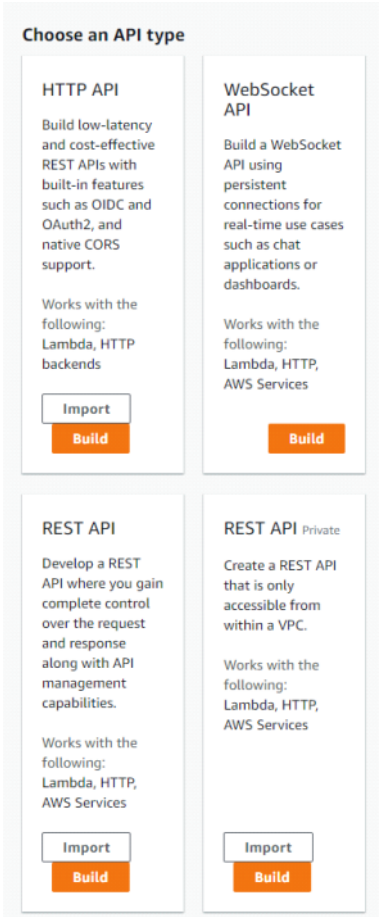


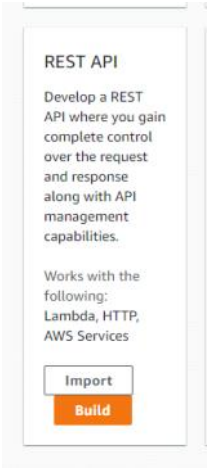
Test edelim



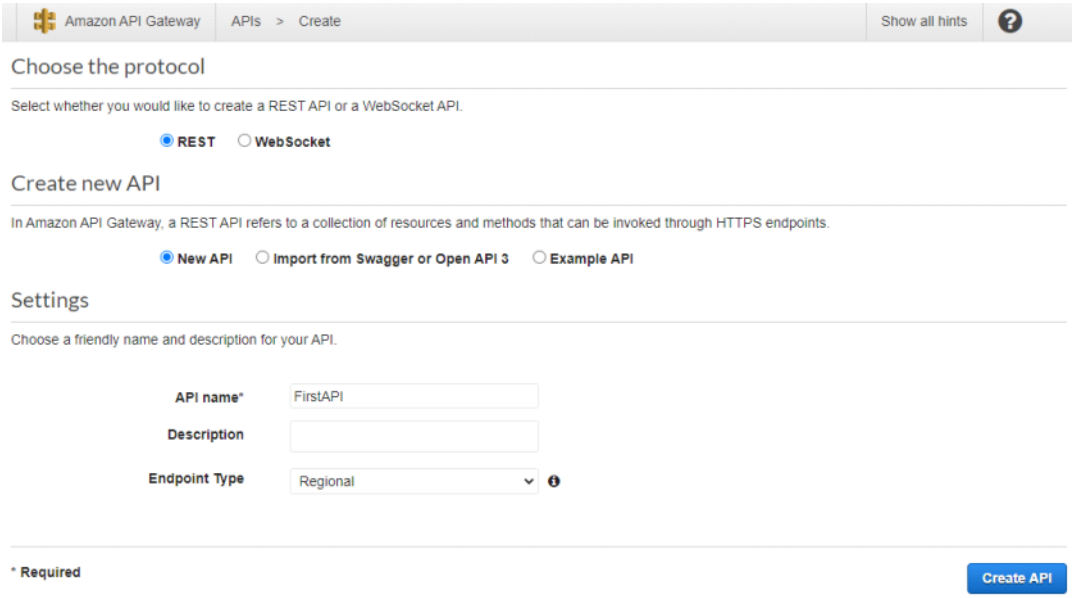
Test calistigina gore API gateway olusturalim





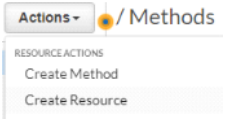


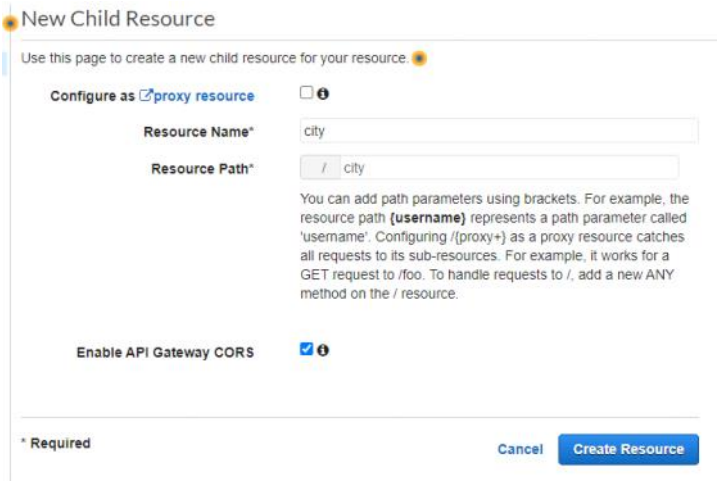
Build diyelim



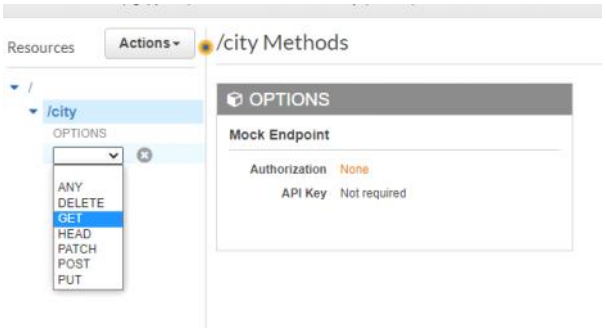
Bazi onemli componentler olusturup api yi deploy edecegiz

- Resource olusturacagiz





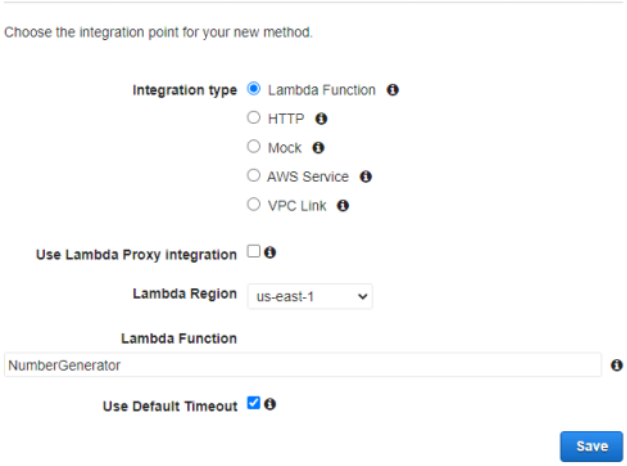
- Method olusturalim

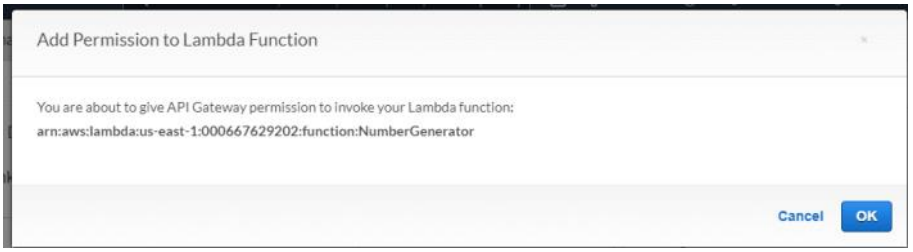


Get sectikten sonra tik i isaretleyelim

Mock : backend servisimiz yoksa sanal bir backend servisi olusturulur

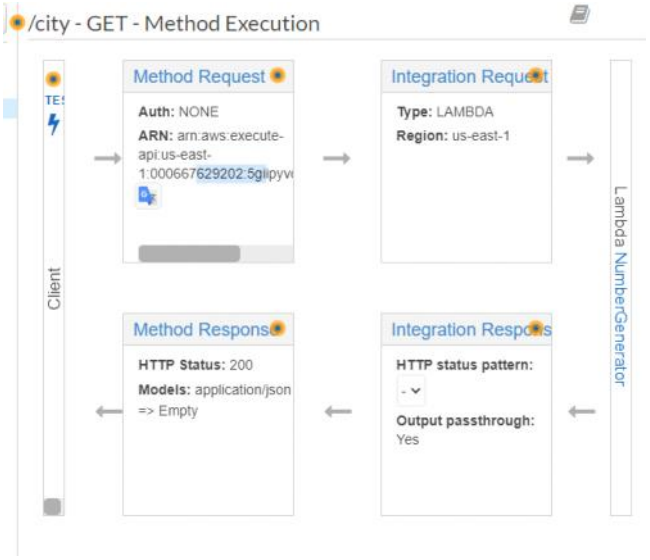
Daha once olusturdugumuz lambda functionu secip resource baglama islemi yapiyoruz



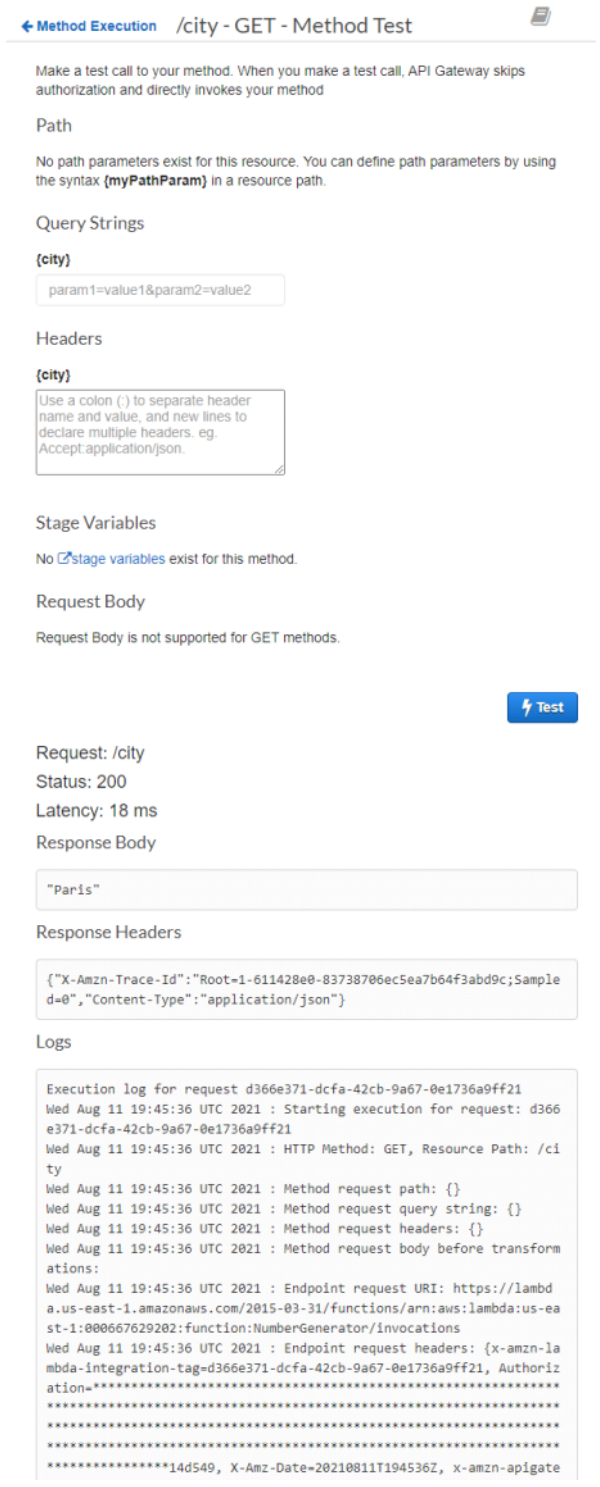


OK

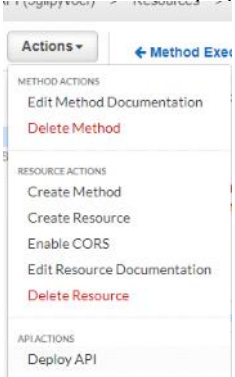
Test ekranimiz

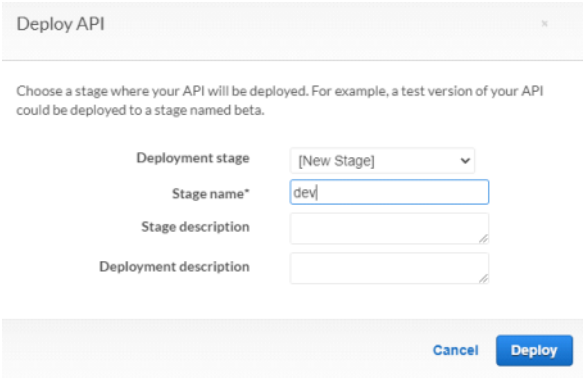


Testi tiklayalim
Her seferinde farkli bir sehir verdigini gorebiliriz



- Son olarak deploy islemimiz kaldi





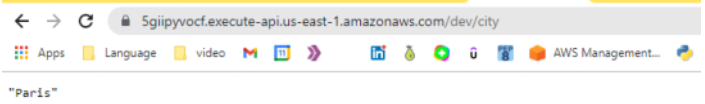
Bize bir url verdi

Ama asagidaki sonucu aliyoruz

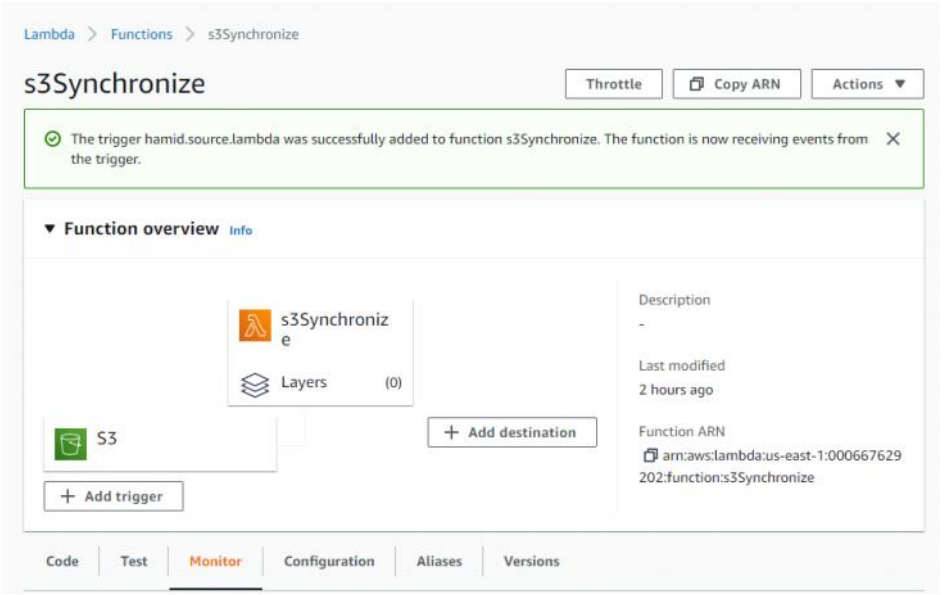
```
{"message":"Missing Authentication Token"}
```

Browser da

Dev den sonra city yazarsak her seferinde farkli bir sehir aliriz

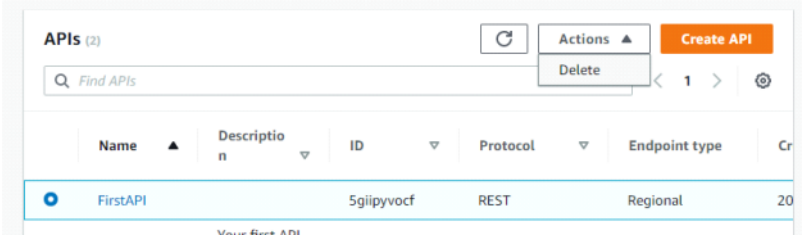


API gateway su anda public (url)
Lambda functionun ne kadar calistigini gorebiliyoruz. ==> **monitor**

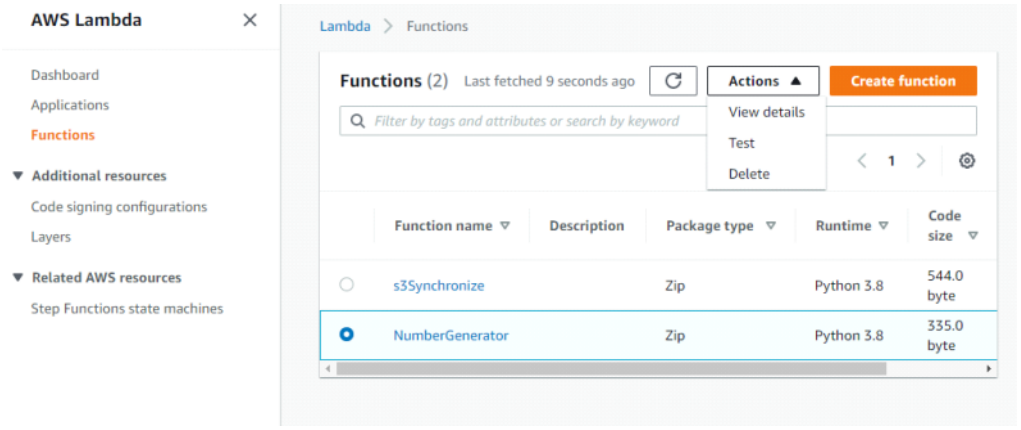


Silme kismi

- API Gateway



- Lambda function



- Buckets