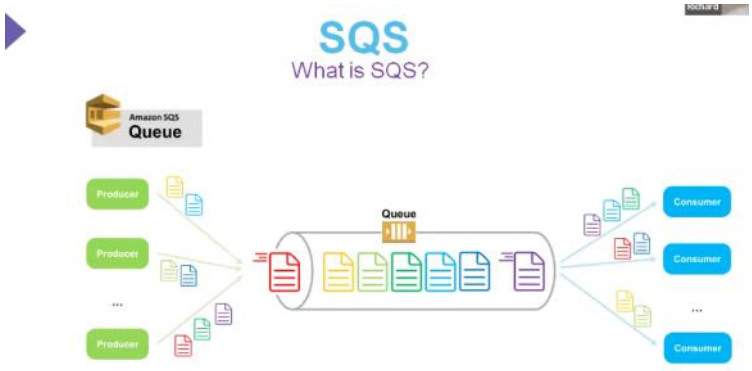
SQS-SNS

Thursday, September 12, 2024 9:48 AM

Hem bildirim hem tetikleme amaciyla kullandigimiz iki tane servisimiz var.
Ortak noktalari mevcut

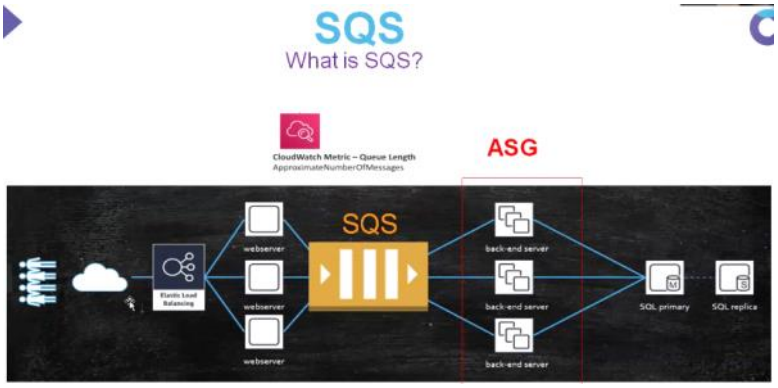
Simle Queue Service yani basit sirlama servisi.
Gelen serverlardaki istekleri belirli bir siraya koyarak karsi islemci serverda bunlarin manipule edilmesi, caistirilmasi amaciyla kullanilir.



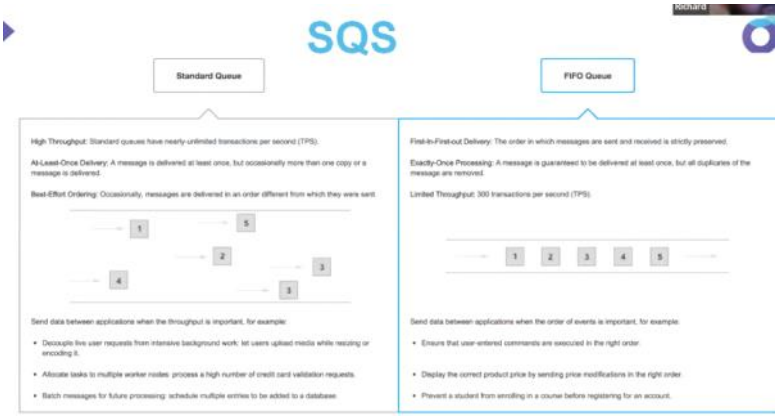
Bu sistemde client ve istemci dedigimiz iki yapi var.
Client yapilarinda istekler cluck yardimiyla istemcilere gonderiliyor.
Burada 3 farkli bilgisayar ya da host var. onlarin her biri istekte bulunuyor.
Ornegin bir websitemiz var ve bu websitemiz farkli farkli arayuzleri var. mesela muhasebe icin ayri bir yone, alisveri iin ayri bir yone ya da kayit icin ayri bir yone yonlendiriyor olsun. Bunlarin herbirisi ayri bir APIlarla yonlendiriliyor.
Burasi icin de ayni seyi dusunebiliriz. 3 serverimiz var ayri istek gonderdigimiz. Her produserin istegi Queue servisinde toplaniyor. Tum istekler bu ara yapi dedigimiz yerde toplaniyor. Daha sonra bu istekler yapilan politika cercevesinde kime iletilmesi gerekiyorsa karmasik olarak cekilerek islem tabi tutulyor.
Burada ya sirayla tek tek consumera cekiliyor istekler ya da karma bir sekilde islem gorerek cevaplamdiriliyor.
Yani producerden bir cevap isteniyor. Bu cevaplar bir istek sirasina sokuluyor. Cevap sirasinda da sirayla ya da karmasik bir sekilde cevaplandiriliyor. Bunlarin hepsi belirli bir politika cercevesinde merkeze response olarak geri donut sagliyor.
iste tum bu yapiya siralamaya SQS yani basit siralama servisi diyoruz.

Bizim kendi kurdugumuz load balancer yapilarini dusunun. Istemciler, databaseler, serverlar, cevap vericiler vs bunlarin herbirisini bir birim olarak dusundugumuzde buradaki client olan yapilar, bulut yapisiyla birlikte load balancerla fronende baglaniyor. Frontendi yapinin hemen arkasinda bir backend yapisi bulunuyor ve o da hemen bizim databsemize bagli bir yapida.
Bu kismi guvenli bir mimari yapi olarak dusundugumuzde burada backednlerimiz, frontendlerimiz ce databaseimiz bulunmakta. Coklu sayida bir musteri kitlesine hizmet veriyorsak onlerine load balancer yapisini kurdugumuzu varsayarsak, musteriler tarafindan gelen istek load balancerla frontende yonlendiriliyor. Daha sonra frontendten yapilan cagrilar dogrultusunda eger backende ugramasi gerekiyorsa tekr bir load balancerla istekler backende yonlendiriliyor. Backendte database ile entegre bir sekilde sorguyu cevaplandirip tekrar musteriye response olarak donduruyor.
Bu mimari aslinda bizim genel olrak AWS te handsonlarimizda uyguladigimiz mimari.
Peki biz bunu bir Queue servisi olarak kullanmak istedigimizde karsimiza nasil bir yapi cikacak? istemci tarafindan gelen istemler frontend tarafina gidecek. Bu sefer frontendteki isteklerin basckend tarafinda cagri yapilmasi isteniyor. Fakat bizim her bir instanceimizin bir kapasitesi var. dolayisiyla her bir instance her bir birim dahilinde response verebilecegi bir degeri var. iste biz SQSi bu mimaride kullaniyoruz.
Normalde daha fazla istek aldigimizi varsayalim 5000-10000 request oldugunu dusunelim. Bu durumda bu yapida bu calisma yetersiz kalacak. Her bir instancea 1000 er tane request yonlendirebilidigimizi dusunelim bu durumda geriye kalan 2000-7000 request cevapsz kalacak. Yeni server da isleme dahil olmuyor. Dolayisiyla bu buyuk bir sikinti.

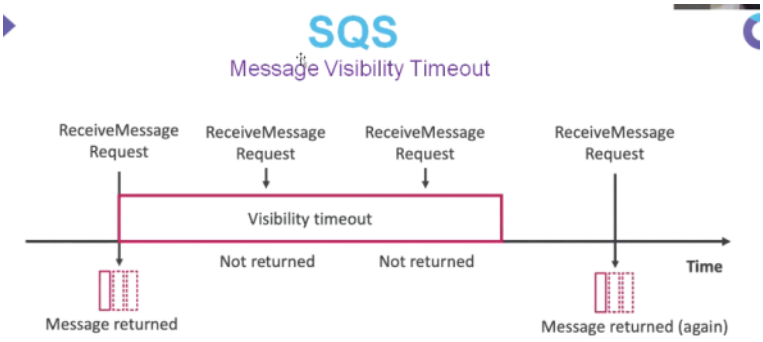
Burada yine 5000 kiilik bir cagri oldugunu varsayalim. 1000er tane instancelara dagittiktan sonra kalan kisim SQS servisimizde toplaniyor. Once instancelara giden cagrilar cevaplandiriliyor adindan bu kalan 2000lik cagri karmasik olarak backend serverlara yonlendiriliyor ve cozumlendikten sonra cevaplari yine clientlara gidiyor.
Cevap donulene kadar musterilere 404 veya buna benzer hata sayfasi gonderilmiyor. Boylece musteri hata sayfasiyla karsilasmamis normal sistemine devam ediyor gibi oluyor. Site calisiyormus gibi ama gecikme olmus gibi gorunecek.
SQS kurulmasaydi bu kalan cevaplar 404 hatasi verecekti direkt olarak.
SQS te kac dakika beklesin, kuyruktan ne zaman ayrilsin vs konfigurrasyon ayarlarini biz yapiyoruz.



Buradaki sistemde az once dedigimiz gibi bir SQS ile kuyruk olusturduk serverlarda ve error hatasi vermesini engellemi olduk musteriye.
Burada olusturdugumuz kuyrugun da bir doyum noktasi olacak. Bunun configurasyonu bizim elimzide. Yani mesela soyle dioruz: eger su sayiyi asan bir cagri gelirse su su servisleri trigger et, calistir. Bu sekilde farkli cozum mimarileri sunuyor. Aslinda burada bizim amaacimiz da tam olarak bu. Bir servisle baska servislerin tetiklenmesi.
Buradaki ornekte ise autoscaling tetiklenmis belirli bir sayiyi gecme ihtimalinde. Yani ayni mimarideki yapilar sayiya gore azaltilacak ya da cogaltilacak.
Burada aslinda olay su:nerede ihtiyac varsa SQS oray gider ve servisleri trigger eder. Illa backend onune konulmak zorunda degil. Buradaki senaryosa yogunluk backende farzedilmis.

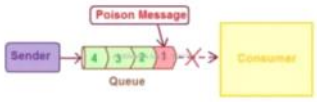


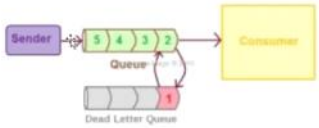
Iki tur SQS kullaniyoruz. Birisi karmasik yapida sira gozetmeksizin responde veren Standart Queue digeride sirayla cevap veren FIFO Queue.
Burada standart FIFO dan daha hizli.
FIFOda belirlenmis bir thtoughput var (TPS) yani 300 kislik limiti var ama standartta high throughput var yani bir limit belirlenmemis.
Standartta clientin requesti birden fazla cevaplandirilabilir yani mesela bir sayfa birden fazla acilabilir ama FIFOda yalnizca bir kez acilir sayfa. Ama mesela bankacilik gibi herseyin kaydinin tutuldugu bir sirketse tek kez acilmasi lazim sayfanin tek bir cevap alinmasi lazim. Dolayisiyla FIFO tercih edilir. Ama mesela bir alisveris sitesinde alisveris kisminda standart odeme kisminda FIFO kullanilmis olabilir.
Bu her iki policyde SQS policysi.
BU KISIMLA ILGILI SORU KESIN GELECEKTIR!!!!!
SIRAYLA GITMISSE HANGI SERVIS KULANILIR YA DA DAGINIK GITMISSE HANGISI KULLANILIR GIBI SORULABILIR YA DA BUNA BENZER ORNEKLENDIRME SENARYOLAR KURUP SORABILIRLER!!!!



Bir istek gonderildi bir client tarafindan. Bu cevaplanana kadar ikinci bir islem yapilamiyor.
Mesela bir carpim tablosu yapilcak. Sayilar girildi ve sonuc icin tiklandi. Cevp gelene kadar basla bir islem yapilamaz. O request cevapalninca ise aetik SQSte o request gorulmez siradakiler gorulur.
Bunun aslinda adi message visibility timeout yani mesajiniz suan cevaplanmakta ve bu sebepten isleminiz gorulmemekte demek.
Kuyruga gelen her islem tamamlandiktan sonra silinmek zorunda. Aksi halde kuyruk bir kisir dongu halinde ayni islemleri dondurur durur. Dolayisiyla olumlu islenen her istem silinmek sorunda bitis sonrasi.

SQS
Dead Letter Queue (DLQ)





Burada goruldugu gibi olumlu sonuclananlar kuyruktan silinir.
Olumsuz olanlar ise baska bir kuyruga yazilir Dead Letter Queue dedigimiz bir yapida. Burada 4 gun bekleme zamani var. bu sure tamamlanmis ve hala bir islem yapilmamissa otomatik olarak kendisi siliniyor.

SQS
Pricing

- Pay only for what you use
- AWS **Free Tier** includes **1 million requests** with Amazon Simple Queue Service (SQS).

1 milyon requeste kdar servisten faydalanmak bedava.

SNS

Simple Notificition Service. Bildirim gondermek amacli kullanilir. Ayni zamanda gelen bildirimlerle farkli servisleri tetiklemesi amaciyla da kullanabiliyor.
Mesela benim ec2larimin herhangi birinde bir statu degisikligi oldugunda bana mail olarak bildirimde bulun diyoruz. Bunun icin erekli email ayarlarini yapiyoruz ve bu servisi aktive ediyoruz. O ec2muzla birlikte calisan baska bir servisimiz de varsa mesela senaryoya gore o servisleride aktive edip calistirabiliyoruz.

SNS
What is SNS?

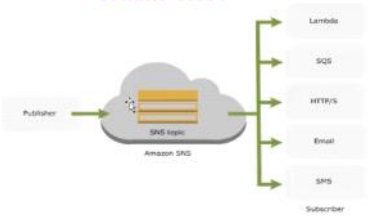
Amazon SNS

- Amazon **Simple Notification Service** (Amazon SNS) is a managed service that provides message delivery from **publishers** to **subscribers** (also known as **producers** and **consumers**).

Burada bu yapiyi olusturacak servisleri belirlememiz lazim. Ayni zamanda kimler dahil olacak bunu belirlememiz lazim. Mesela ec2 ayaga kaldirdik ve onun statusuyle alakli bilgi amak istiyrouz. Bununla iligli konfigurasyonu yapiyoruz ve bunu kimlerin ilgilendirigini de belirlememeiz gerekir. Kimler haber alsin yani bu degisiklikten. Bununla iligli subscriber olusturulmasi gerekiyor.

SNS

What is SNS?



- Clients can subscribe to the **SNS topic** and receive published messages using a supported protocol, such as Amazon SQS, AWS Lambda, HTTP, email, mobile push notifications, and mobile text messages (SMS).

Lambda, SQS, HTTP/S, Emsil ve SMS ile kullanilabiliyor ve ihtiyac durumuna gore hangisiyle kombinlemek isterseniz ona gore confgure ediyorsunuz.

SNS

What is SNS?

Application-to-Application (A2A)

Buradaki tum servislerle integre edebiliriz.

SNS

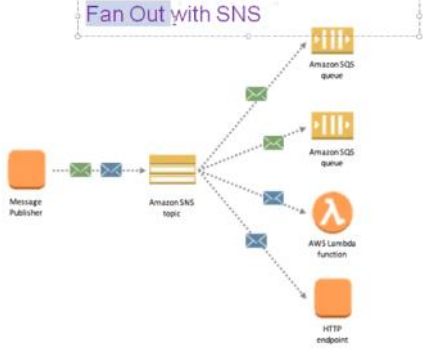
What is SNS?

SNS is integrated with many AWS Services

- CloudWatch (Alarms/Events)
- S3 (Bucket Events)
- CloudFormation (State changes etc.)
- Auto Scaling Groups
- And many others **can invoke SNS.**

SNS

Fan Out with SNS



Burada bir senaryo hazirliyoruz ve bu esik asilip asilmamasina gore bu senaryoya ilave olacak servisleri isliyoruz.
SQS te tek bir servise entegre ediyorduk. Burada birden fazla entegrasyon saglayabiliyoruz.
BU KISIM DA SINAVDA MUHTEMEL GELEBILECEK SORULARDAN BIR TANESI!!!!!
FAN OUT GENELDE CIKMA OLASILIGI YUKSEK BIR KONU.

SNS Pricing

- Amazon SNS has **no upfront costs** and you can **pay as you go**. You pay based on the number of notifications you publish, the number of notifications you deliver, and any additional API calls for managing topics and subscriptions. Delivery **pricing varies by endpoint type**. You can get started for free with the SNS free tier.

Servisi kurmak ucretsiz ama sonrasinda kullandiginiz kadar oduyrosunuz. Tetiklettirdiginiz servisin ucret kistasina tabi olur cost kismi. Mesela lambdayi tetiklettin o zaman lamba pricingi kadarini odemis olursun.