# Spring - Mar 14

Messaging system does not important than Spring, in the interview.

- Cuz it is related to design, not **particular** technique.
- talk very brief in our training
    - it will take very in depth where specific so that will take a long time in the interview

Messaging System

- JMS (java EE)
    - implementation or vendors to implement JMS specifications
        - active MQ(Apache)
        - IBM MQ series
- AMQP
    - rabbitMQ

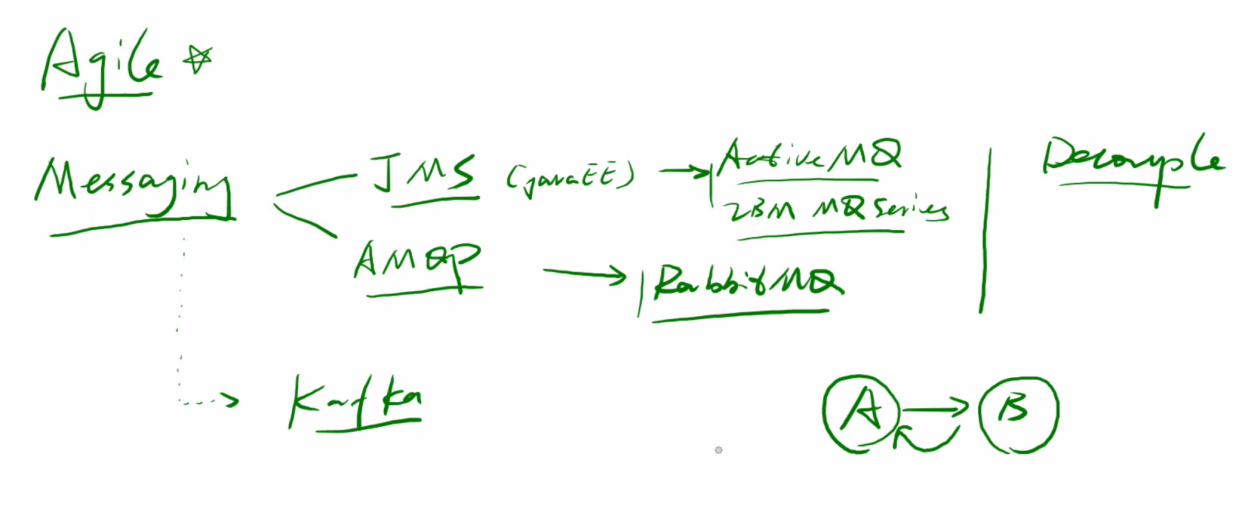
Kafka is separate messaging System, which is very similar with messaging system but with different use case.

- call it streaming system instead of messaging system
    - messaging system is used to decouple project/system
        - **have only one separate server**
    - Kafka, we also can used to decouple system, but it mainly uses for streaming process of data, like data analysis
        - have a very large volume of data, you want to process them in a very short period of time.

- Kafka is designed for distribution

Why we need to use message system?

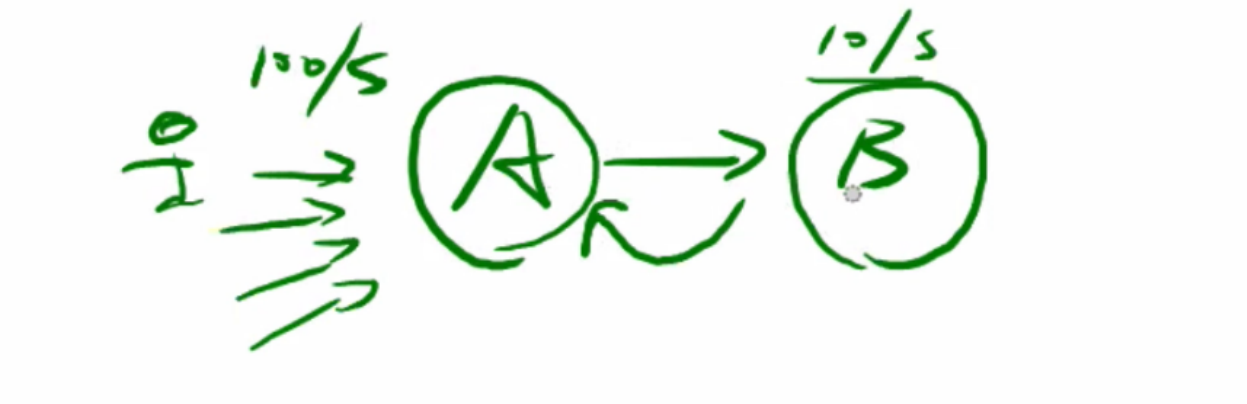
The main idea is to decouple two systems




for example

A→B request, A need to wait for B response(sync system)

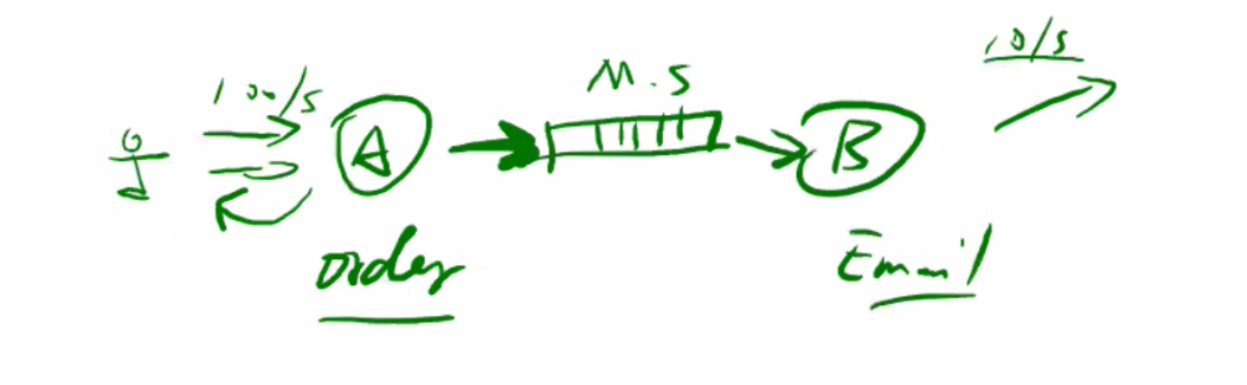
using messaging system(async system), A don't need to wait for B each other



in this case

whatever how powerful(CPU, Memory) A is, it still need to wait for B

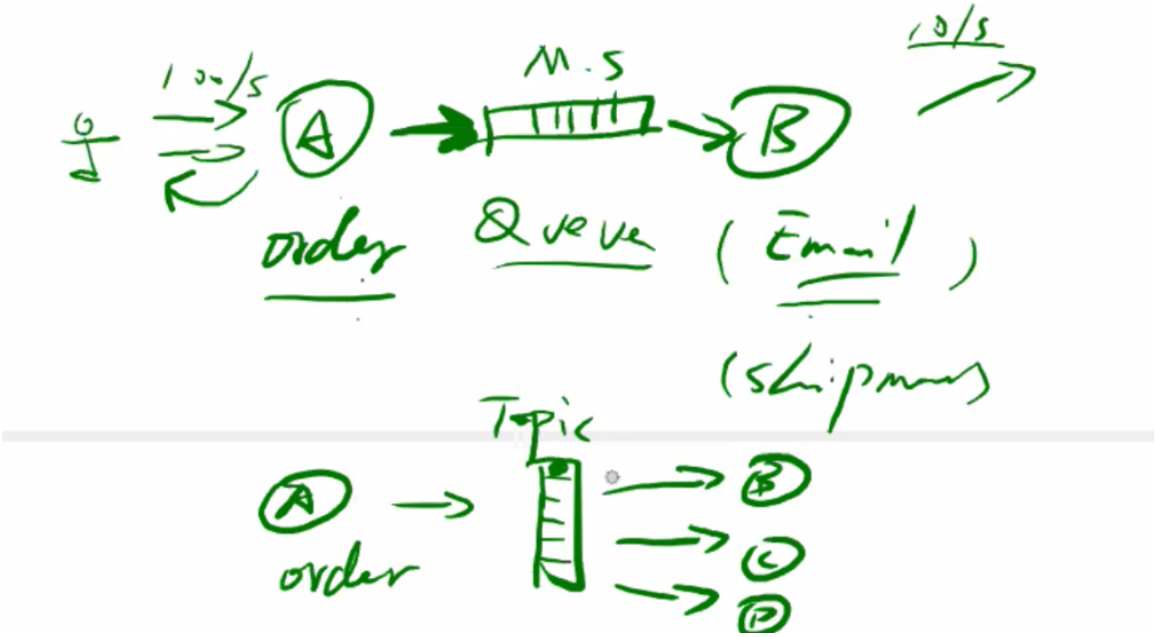
- messing sys uses to decouple this situation




instead of sending request to B, A send messages to M.S directly. B receives request from M.S

Publish subscriber pattern

-



- if it achieve one to one like A → B, we can use queue(FIFO)
- multiple services (email just one of them), if use queue, the message will be consumed if just B(email) get the request. So we need use Topic
    - one topic can be subscribed by lots of services, achieve one to many communication
    - the order info will message to be a topic, whoever subscribe the topic will receive/get the message
    - still loosely couple

lots of concerns going on:

- how to get response
    - result/solution: create other queue/M.S to be responsible for response, and A listen to the queue(called response queue) → overhead
- lots of overhead
    - M.S could be down(error, issue)

so whether use M.S or just RESTful api is only considered if it’s suitable

**IQ: What kinds of messaging sys you are familiar with, what the usage of M.S.?**

**Why we need a messaging system? So REST API is good, right? It's convenient. Why do we still need a messaging system?**

Kafka

- topic, not has Queue
- but even for the same topic, it has different partitions of different servers.
- So the same topic would be divided into different servers. So lots of servers can just starting getting getting the message and then send out a message to

the consumers.

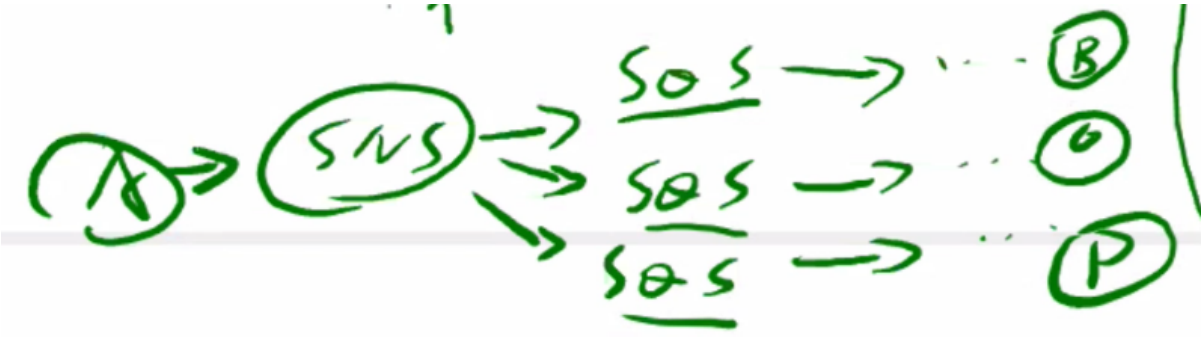
- **So that's why it can achieve a higher throughput.**

Attention : research

AWS:

- SQS → queue
  - Queue service
- SNS
  - notification service(push service) → push messages to subscribers

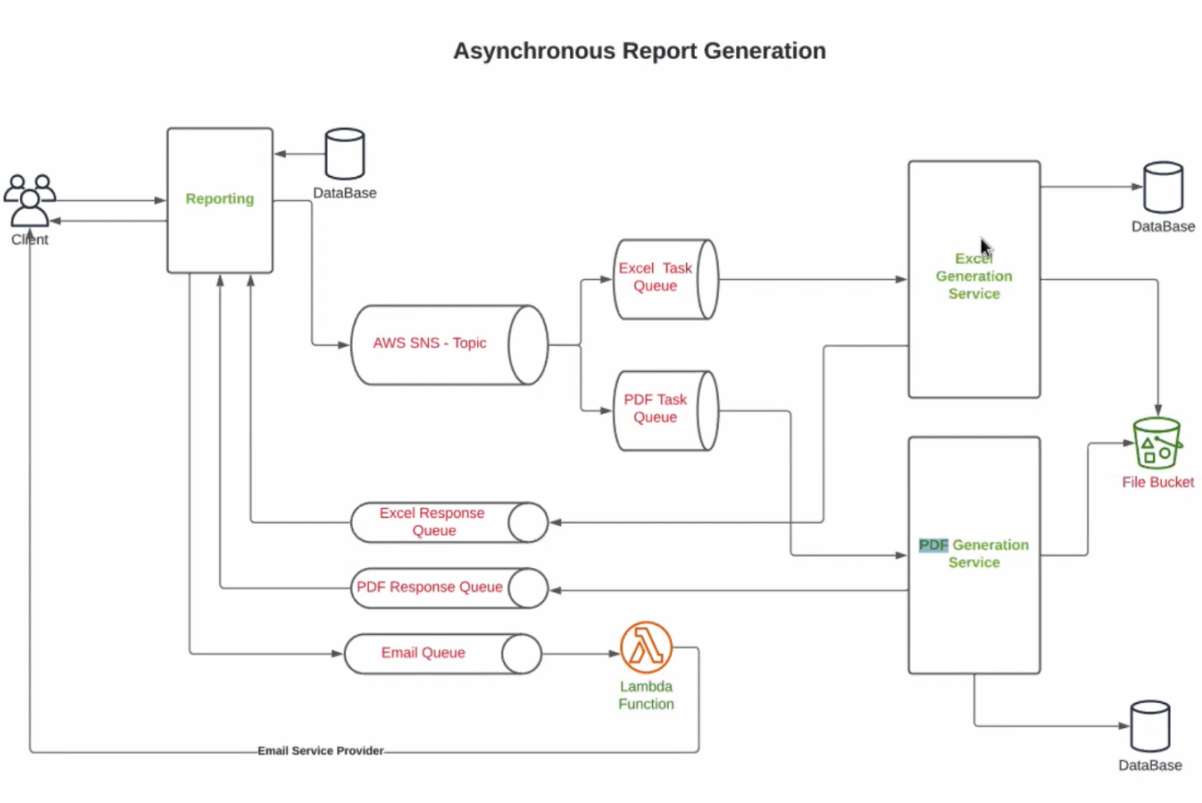
**AWS does not have topics, so we can use SQS+SNS to achieve topics**




messaging system: one system wants to send messages to another system

middleware of two systems to achieve decouple of these two systems

Project: Reporting System




File Bucket → S3

- S3 like a dropbox
  - create bucket
    - bucket just like a folder/category
    - like GCS(google cloud storage) → storage files get a link
    - ~~Problem: if users get link directly, they connect to AWS directly and not need the permission from our server.~~
    - key ← → object storage (not like windows files system)
    - EC2 → Elastic Block Store is like windows files

LMS → video to set up SNS/SQS/S3

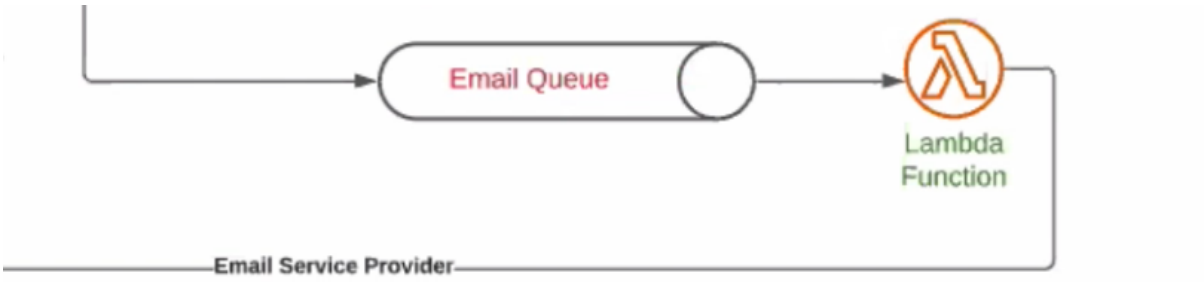
ASW EMR ↔ Hadoop

**The Queue supposes to be fast and message supposes to be small**

- Also save the file locations to database
    - just in case, someone want to query, just query database. Service do Not touch the file itself.
    - That’s why have the separate database everywhere

SQL and no-SQL suppose to store structure data

- files system is un-structure, which can not be query and create index. So it does not suppose to use database




Lambda Function

- Our application just send messages to Queue and let queue to deal with how to send email. And lambda Function will handle sending things
- who gonna listen to the Queue is the thing that lambda deals with

Use python codes to send emails

/sendEmailCode.py

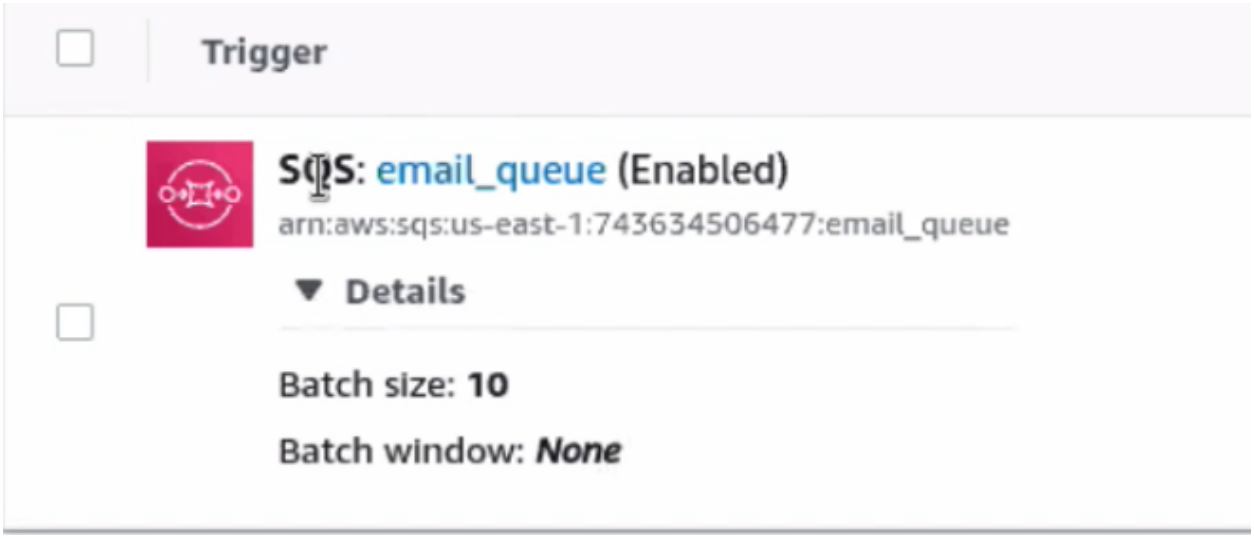
AWS Lambda(The Lambda Function/The Lambda Service)

SQS

You just have a small piece of codes. You want it to run on the server environment but not want to maintain the server

Commonly use Node.js and Python

PS: Java can be run, but slow to start, lots of overhead. Just not suitable




SQS event

mean once a message arrive on this SQS, then the lambda will be triggered

e.g use case

> When user uploads a file to the bucket, it will trigger Lambda to scan the file, to call third-party libraries

why we need to use lambda

What lambda can do?

use trigger to

e.g.

user uploads a file to bucket → trigger lambda to convert or scan the files(antivirus)

upload a picture, if too large → resize it and create different size versions(resolutions(分辨率)) for different use case

Project: Reporting System

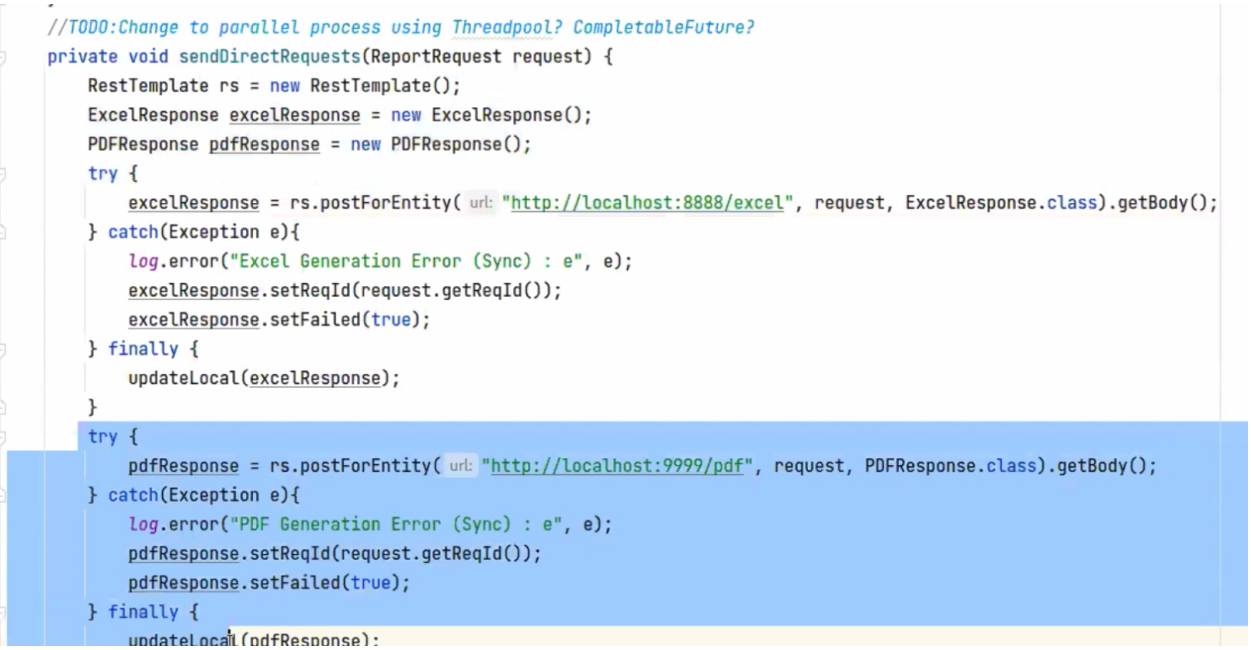
/ResportServiceImpl.java

Where can we use multi-threading?

change to parallel process using thread pool? CompletableFuture?

Two request send to two different services, they are **irrelevant**

we do not wait first request finished and then send second one



```
//TODO:Change to parallel process using Threadpool? CompletableFuture?
private void sendDirectRequests(ReportRequest request) {
    RestTemplate rs = new RestTemplate();
    ExcelResponse excelResponse = new ExcelResponse();
    PDFResponse pdfResponse = new PDFResponse();
    try {
        excelResponse = rs.postForEntity( url: "http://localhost:8888/excel", request, ExcelResponse.class).getBody();
    } catch(Exception e){
        log.error("Excel Generation Error (Sync) : e", e);
        excelResponse.setReqId(request.getReqId());
        excelResponse.setFailed(true);
    } finally {
        updateLocal(excelResponse);
    }
    try {
        pdfResponse = rs.postForEntity( url: "http://localhost:9999/pdf", request, PDFResponse.class).getBody();
    } catch(Exception e){
        log.error("PDF Generation Error (Sync) : e", e);
        pdfResponse.setReqId(request.getReqId());
        pdfResponse.setFailed(true);
    } finally {
        updateLocal(pdfResponse);
```

Route 53(DNS service)

e.g different geolocations have different IP address

Network configuration inside of our AWS Cloud

Regions

Availability Zones

subnet: public Network, Private Network

- we can have multiple subnet

AWS VPCs: virtual network

Images: Clones of your server

Elastic Block Store: removable Hard Drive

- unplug it to one server and plug it to another server(instances)

Security Group

- who can visit to what server
- inbound and out bound rules

Load Balancing/Load Balancer

CDN

distribution

AWS Cloud Front is CDN solution for AWS

We put the content to S3 buckets, and then we tell cloudFront that we want the buckets can distribute out. It CDN(CloudFront) will distribute all the files to different location

PS: command line tool → “dig xxx.com” (domain) → ip address

API Gateway

It’s the exact same API Gateway concept with MicroServices in Spring Cloud

Spring Cloud Gateway is **software Gateway**

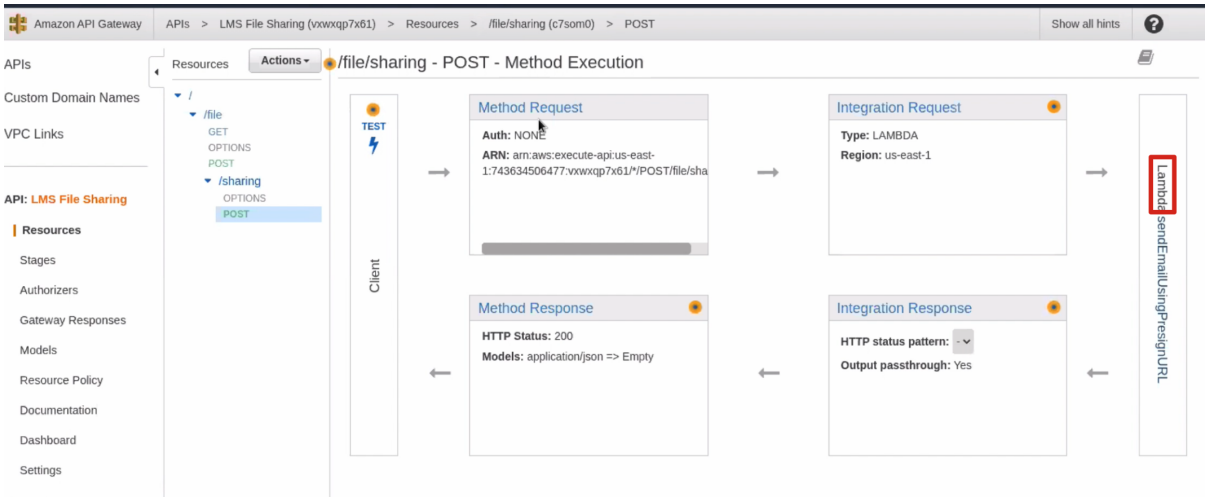
- coding for which URL to which URL
-

```
@SpringBootApplication
public class DemogatewayApplication {
	@Bean
	public RouteLocator customRouteLocator(RouteLocatorBuilder builder) {
		return builder.routes()
			.route("path_route", r -> r.path("/get")
				.uri("http://httpbin.org"))
			.route("host_route", r -> r.host("*.myhost.org")
				.uri("http://httpbin.org"))
			.route("rewrite_route", r -> r.host("*.rewrite.org")
				.filters(f -> f.rewritePath("/foo/(?<segment>.*)",
"/${segment}"))
				.uri("http://httpbin.org"))
			.route("hystrix_route", r -> r.host("*.hystrix.org")
				.filters(f -> f.hystrix(c -> c.setName("slowcmd")))
				.uri("http://httpbin.org"))
			.route("hystrix_fallback_route", r -> r.host("*.hystrixfallback.org")
				.filters(f -> f.hystrix(c ->
c.setName("slowcmd").setFallbackUri("forward:/hystrixfallback")))
				.uri("http://httpbin.org"))
			.route("limit_route", r -> r
				.host("*.limited.org").and().path("/anything/**")
				.filters(f -> f.requestRateLimiter(c ->
c.setRateLimiter(redisRateLimiter())))
				.uri("http://httpbin.org"))
			.build();
	}
}
```

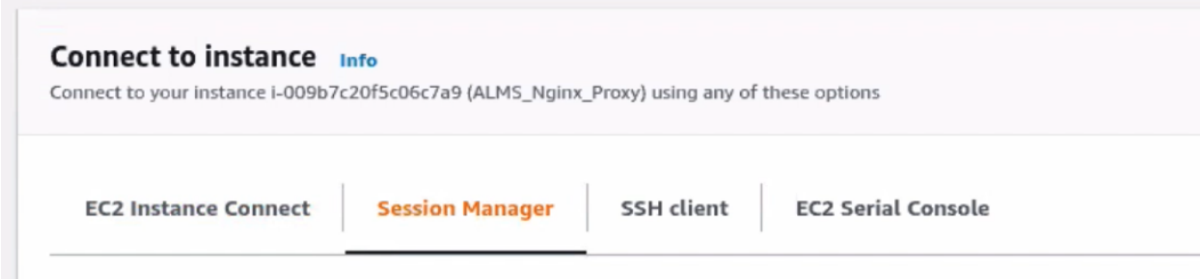
COPY

Why we also use AWS API Gateway?

- use same security rule and so on
-




Connect to Instances(like EC2)




**The main points are not what they are. The points are how to use them. Under what situation we need to use them**