به نام خدا

اون بخش behave part بماند.

از الان میریم سراغ اون ابر و کلودی که میخایم بسازیم !

اول از همه یه سرچ کلی رو سخت افزار و ابر و کلود اینا داشتم (و متاسفانه خلاصه ننوشتم) سعی کردم بفهمم اگر بخوام این داستان رو پیاده سازی کنم باید چی کار کنم خلاصه که دریافتم اینه که

اول از همه پروژه ای که باهاش طرف هستم بیگ دیتا عه ! یعنی باید مسائل مربوط به اون رو بفهمم سعی کردم با رزیری پای زیرو یکم ور برم و مثلا کل زیرساختی که میخوام به عنوان کلود بسازم رو اونجا بیارم بالا ، خب اولین نکته ای که متوجه میشی اینه که با یه دونش کارت راه نمیوفته و باید کلاسترشون کنی . پس باید برم سمت ابزار های کلاستر کردن ، مثل kubernetis (هنوز استفاده نکردم ولی کلیتش اینه که کلی کامیپوتر یا رزبری پای مثلا یکی میشن و همشون با هم یک کار رو انجام میدن و بار پردازش بینشون پخش میشه) خب حالا برنامه ای که دارم رو چه جوری بریزم روش و بخش های مختلفی که دارم چه طوری میخان کنار هم کار کنن و مثلا این جریان اطلاعاتی که هست چه جوری میخواد کار کنه ، اینجاست که docker و container ها میان وسط (خیلی سریع یه توتریال رد کردم و تقریبا میدونم میدونم چیه ، جزئیات هیچی ولی از طریق این و docker-compose و فایل YAML که میسازیم امیج ها رو لایه بندی میکنیم و جزئیات نرم افزاری رو بهم وصل میکنیم)

حالا میخایم بریم سراغ ابزار هایی که تو فضای ابری میخوام استفاده کنم اولین بخشی که داریم همون بخشی هست که قراره با پروتکل mqtt (چون میخایم نود ها مون با low power , low bandwith کار کنن ) دریافت کنه و تاپیک ها رو برای ما بسازه با چنتا ابزار مختلف مواجه شدم مثل (EMQ,RabbitMQ,VernmMQ) از بین اینا RabbitMQ داکیمنتیش بهتری داشت و امکانات دقیق تری برای کنترل صف داشت از همه مهم ترش این بود برای کلاستر کردن کلی داکیومنت داشت که احساس کردم میتونه مناسب باشه و یه چیز جالب که RPC داشت چیزی که قدیم تو mongooses دیده بودم و برام جالب بود که کجا میسازه همچین چیزی یو همین یه آیتم بهت اجازه میده که بتونی رول بنویسی ، پس همین رو انتخاب میکنم.

در ادامه تو سایت hivemq داشتم میخوندم که کافکا ها رو میریزه تو تاپیک mqtt broker بعد پایپ لاین و ستریم دیتا رو از اونجا می سازه میره جلو. از طرف دیگه تو سایت خود آپاچی کافکا دیدم که یه چیزی هست تحت عنوان mqtt proxy که مستقیم از سنسور میریزی تو این ! منتها از اون جایی که هنوز به کافکا ور نرفتم و نمیدونم چیه ، با همون rabbitMq شروع میکنم ،

هدف از طرح 2 پیاده سازی بخش های نرم افزاری که ازش تو معماری اینترنت اشیا به عنوان ابر و کلود یاد میشه است . به این شکل که اول یه mqtt بروکر میارم بالا ، دیتا میگیرم ، یه پایپ لاینی درست میکنم (ترجیحا با کافکا ) این استریم میاد ، رو یه وب سرور نمایش میدم یه استریم هم میریزم رو یه دیتا بیسی خروجیش رو میدم یه مدل که هر 1 دقیقه مثلا امتیاز راننده رو اعلام میکنه و بر میگردونه رو وب سرورعه نمایش میده .

فعلن با rabbitMq شروع میکنم ببینیم چی هست چی کار میکنیم چی میشه

یه توتوریال سایت خودش داره اونو بخوام توضیح بدم ؛

اول از طریق داکر نصبش میکنیم :

docker run -it --rm --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3-management

بعد یه برنامه گیرنده وفرستنده با پایتون مینویسه و تست میکنه و حالا قابلیت هایی که میتونیم داشته باشیم رو ادامه میده

چیزی که موقع بررسی داشتم میرفتم جلو به عنوان طراح سیستم ، این بود که یه فایل yaml و docker-compos اینا درست میکنی و rabitmq.config و plugin هایی که میخای رو بهش میدی (mqtt) حالا داکر رو ران میکنی و همه چیز خودش میره جلو ! حالا اینجا میخای کافکا اضافه کنی اون باید یه داکر جدا باشه که از طریق همین docker-compose بیاد به این اضافه بشه کل پروژه میشه همون فایل docker-compse تهش ، احساس کردم جالب شد 😊

حالا خیلی نرم و راحت بریم سراغ اون توتوراله ، پیش بریم که محیط و اینا آشنا شیم ، بعدش میریم سراغ فایل rabbitmq.conf بسم الله

خب الان بعد از یک هفته میخام گزارش بنویسم

توتوریال اول رو شروع کردم ودیدم به فایل کانفیگ اشاره کرد رفتم ، مجبور شدم که برم سراغ همین فایل کانفیگ تا با خیال راحت پیش برم (خیلی ممنون از جادی کمک کرد) این فایل داکر کامپوز اینا رو هندل کردم

یه توضیح مختصر بدم که چی به چی یه که فراموش نکنم تو آینده

یه فایل Dockerfile داریم که موقعی که بیلد میکنیم این اجرا میشه ما میایم فایل definitions.json و rabbitmq.conf رو میایم کپی میکنیم توی ایمجی که قراره ساخته شه و بیاد بالا حالا موقعی که داریم با داکر کامپوز میاریم بالا ، با وریبل اینوایرمنت جای فایل کانفیک رو میدیم بهش و فایل کانفیگ دلخواهمون ران میشه

برای دیباگ یه سری کامند که موقع اجرا میتونیم ران کنیم رو توی فایل readme.md گزاشتم که جلو چشم باشه

بعد از این داستان شروع کردم به توتوریال هایی که خود سایت داره ، جزئیاتش توی پوشه Clients-AMQP هست ، مسئله اینجاس که این امکاناتی که ارائه میده و مفاهیمی که ارائه میکنه برای پروتکل AMQP هست

ولی توضیح مختصر بدم که یادم نره

یه سری producer اطلاعات داریم یه سری consumer ، این تولید کننده های اطلاعات ، وقتی اطلاعات رو میفرستن به بروکر یا همون rabbitmq ، اطالاعات رو مستقیم میفرستن به exchange ها ، در کنار exchange ها ما یه سری صف queue تعریف میکنیم و exchange ها رو به queue ها BIND میکنیم

و اینکه ما 4 مدل exchange داریم ،

1-forsage )یا همچین چیزی( که مسیج رو به صف هایی که بهش BIND کردی برودکست میکنه

2-direct ، یه چیزی میاد تعریف میکنه اینجا به اسم routing key که موقع فرستادن مشخصش میکنی و بعد از این که exchange پیام رو دریافت کرد ، موقع Bind کردن میگی به کدوم صف بفرسته

3-topic هست که از همون routing key استفاده میکنه ولی مفهوم متفاوتی رو میسازه

اینجوری پیاده سازی شده که با . بخش ها رو جدا میکنه و مشخص میکنه که چه اطلاعاتی باید بره کجا

حالا از دو تا المان داره یکی \* یکی هم # حالا الان توضیحش سخته با مثال باید گفت ، باشه برای آینده

4- header هست

که توتوریال هاش توضیح نداد منم ، از اونجایی که فهمیدم اینا مال پروتکل mqtt که من رو سخت افزار هام دارم نیست ، فعلن صرف نظر میکنم اگر نیاز شد میام

حالا میریم سراغ داکیومنتیشن پروتکل مطلوبمون که ببنیم چه قابلیت هایی در اختیارمون میزاره و از اون جایی که قراره پایان نامه بنویسیم سعی میکنم با ترجمه و دقیق برم جلو

که تو آینده یکمی به درد بخوره !

ویژگی هایی که ساپورت میکند:

* QoS0 and QoS1 publish & consume
* QoS2 publish (downgraded to QoS1)
* Last Will and Testament (LWT)
* TLS
* Session stickiness
* Retained messages with pluggable storage backends

کلاینت ها میتوانند با پروتکل های دیگری نیز تعامل داشته باشند. همه ی ویژگی های [management UI](https://www.rabbitmq.com/management.html) میتوان استفاده نمود و همچنین میتوان از بعضی از پلاگین های دیگر استفاده کرد. ولی محدودیت هایی وجود دارد که برای استفاده باید مورد نظر قرار داد.

بعد از فعال کردن پلاگین(از طریق داکر فایل میتونیم کامند رو تنظیم کنیم ران شه اونجا فعال میکنیم) برای یک اتصال موفق ، باید کاربران به طور کامل از طریق virtual host مورد تصدیق و شناسایی قرار گیرند و اجازه ارسال پیام دریافت کنند. کلاینت ها در اکثر مواقع باید برای اتصال یک credentials ارائه کنند.

برای تنظیم نحوه مدل اتصال چند روش موجود است. 1) مجموعه درستورات  [rabbitmqctl](https://www.rabbitmq.com/cli.html)که میتوان از طریق dockerfile پیش از شروع تنظیم کرد نوشتن (راه بهتر با امکانات بیشتر هست. )2 ) استفاده از پلاگین  [management UI](https://www.rabbitmq.com/management.html)  که یک HTTP API

در اختیار قرار میدهد که میتوان به سادگی تنظیمات را انجام داد ، دسترسی هایی که در اختیار ما قرار میدهد شامل :

تعریف کاربر جدید ، تعیین نوع کاربر (که اطاعاتی که تنظیم میکنیم میتوان تحت عنوان definition.json خروجی گرفت ، همچنین میتوان این اطلاعات را در هنگام اجرای داکر مجدد بارگزاری کرد تا نیازی به تنظیم مجدد تنظیمات نباشد ، برای این کار باید در فایل config محل فایل definition را تنظیم نمود و ان را فعال کرد ، همچنین در داکر میتوان فایل را با . volume را روی همان فایل ارسالیmount کرد تا تغییراتی که میدهیم اعمال شود)

{در ادامه پلاگین mannegment بیشتر بررسی شود}

پلاگین mqtt بر اساس بالاترین سطح امکانات Rabbitmq است و از exchange ها و queue ها پشتیبانی میکند.

پیام های ارسالی به تاپیک به exchange عه ، amq.topic ارسال می شوند(به صورت پیش فرض) و مصرف کنندگان پیام ، پیام ها را از صف هایی که به این اکسچنج بایند شده اند دریافت میکنند. هر دوی اینها این امکان را به ما میدهند که ، که با کمک پلاگین mannegment بتوانیم حجم و تعداد صف ، تعداد اکسچنج و ... را کنترل و مدریت کنیم. {قبل گفتیم که تو اکسچنج هایی از جنس تاپیک با . مفاهیم رو از هم جدا میکنن ، اینجا با / میکنن}

تا اینجا مدل داکیومنتی رفتم یه چیزایی هم پیاده سازی کردم ، ولی اصولی نبود ، صرفا تمرینی بود که چه قابلیت هایی هست و فضای مسئله چه شکلی یه

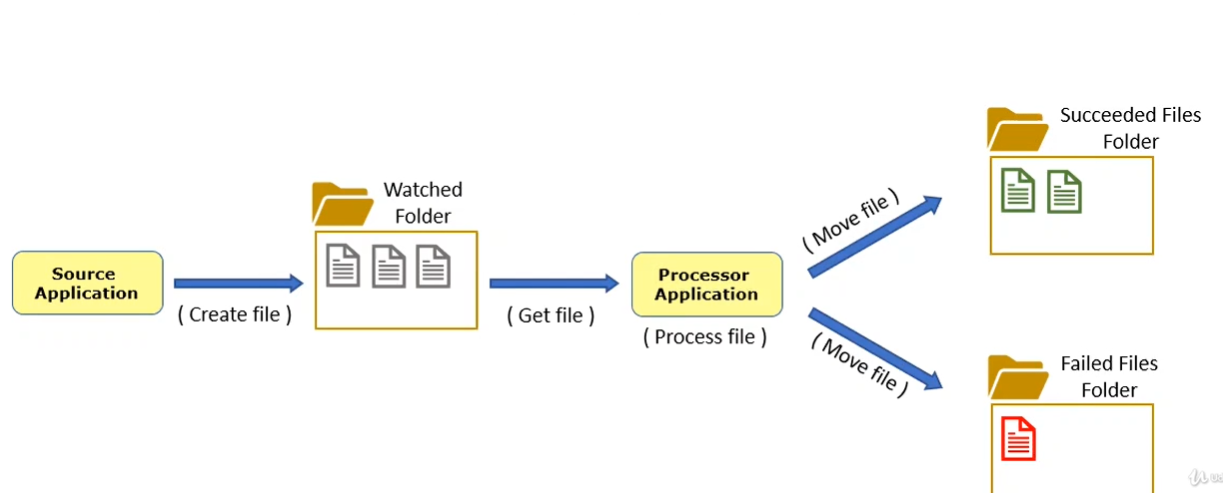
از الان به بعد به کورس مناسب تو udemy پیدا کردم ، ومیخام اون رو شروع کنم و خیلی مناسب گزارش نویسی کنم چون حس میکنم روندش برای پایان نامه نویسی عالیه

مجدد بسم الله الرحمن رحیم

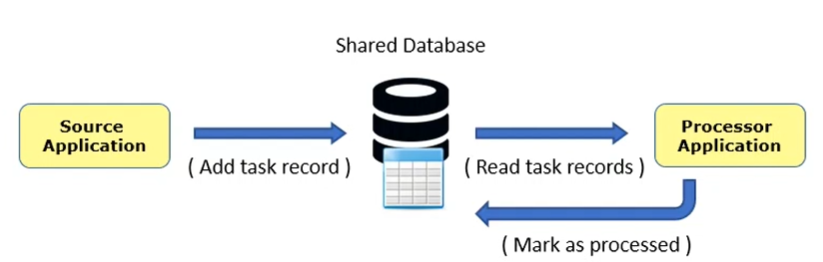
مقدمه

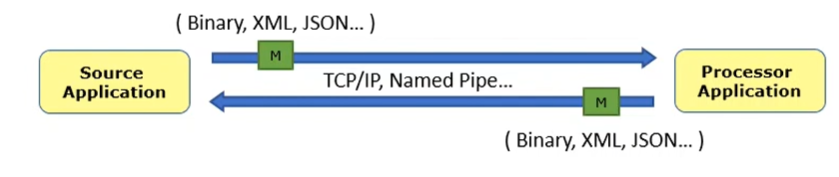
یک سیستم کامل از چند اپلیکیشن تشکیل شده که وظایف متفاوتی را انجام میدهند ، در اغلب مواقع این اپلیکیشن ها نیاز دارند با یکدیگر تعامل داشته باشند و باهم ارتباط برقرار کنند. به ایجاد کردن ارتباط میان اپلیکیشن ها integration میگوینند. شیوه های متفاوتی از اینترگیشن وجود دارد :

مدل اول ؛ اینتگریشن های مبتنی بر فایل هستند ، در این مدل، اپلیکشین اولیه فایلی که باید پردازش شود را ایجاد میکند در پوشه ای مشخص قرار میدهد سپس برنامه ای دیگر فایل هایی که در این پوشه قرار گرفته اند را به محض ایجاد بررسی می کند و فایل های خراب را از سالم، جدا کرده و در پوشه هایی متفاوت قرار می­هد که فایل های بعدا مجدد مورد بررسی قرار بگیرند، در اغلب مواقع این دو برنامه از یک دیگر مستقل هستند تا سیستم بهتر کار کند.

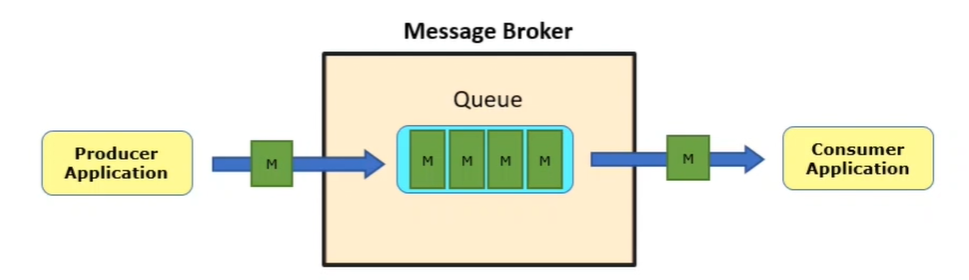


مدل دوم ؛ اینتگرشین های مبتنی بر دیتابیس مشترک یا توزیع شده است ، در این مدل ابتدا یکی از اپلیکیشن ها اطلاعاتی را در دیتابیس تغییر میدهد ، سپس اپلیکیشن دیگری آن را برداشته پردازش میکند و مجددا به دیتابیس برمیگرداند و به آن تگ پردازش شده میزند.



مدل سوم ؛ اینتگریشن های مبتنی بر ارتباط مسقیم است ، به طوری که اپلیکیشن ها به صورت مستقیم ارتباط tcp/ip برقرار کرده (این نوع ارتباط به named pipe نیز معروف اند ) بعد از برقرار ارتباط شروع به ارسال پیام به یکدیگر میکنند ، فرمت پیام ها میتواند به گونه های متفاوتی از جمله باینری یا تکس بیس مانند , xml, json ... باشد . 

مدل چهارم ؛ اینتگریشن بر پایه پیام دادن نامتقارن با کمک یک بروکر پیام است ، اپلیکشین اولیه یا تولید کننده اطلاعات در یک سو با هر فرمتی که بخواهد میتواند پیام ارسال کند ولی این بار یک اپلیکیشن میانی پیام ها را دریافت میکند (که اغلب به آن message broker یا message bus میگویند ) پیام ها را در یک لیست قرار داده که به آن Queue یا صف می­گوید و آن ها را به هدف ، مقصد یا مصرف کننده منتقل میکنند.



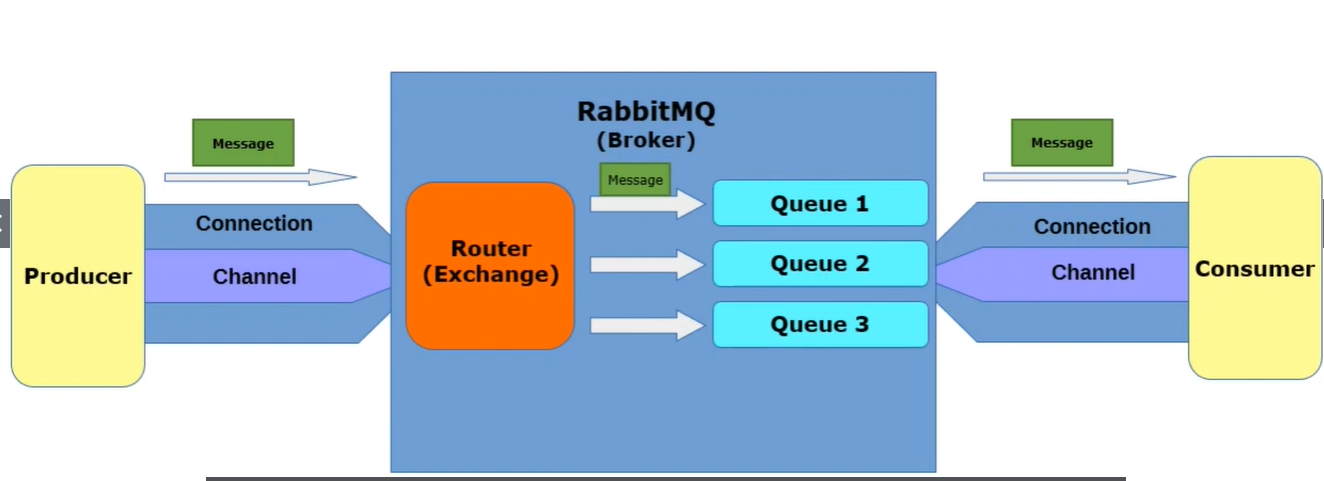
{این روش یه روش محبوب است که در اپلیکیشن های متفاوتی میتواند مورد استفاده قرار بگیرد ، ما از این روش استفاده میکنیم در زیر به ویژگی های این مدل اشاره میکنیم} ویژگی های این مدل اینتگریشن

* جدا بودن اپلیکیشن تولید کننده و اپلیکیشن مصرف کننده
  + اپلیکیشن تولید کننده و مصرف کننده نیازی ندارد یک دیگر را بشناسند
  + ادرس ip و .. یکدیگر و تکنولوژی های مورد استفاده یکدیگر را نمیدانند
  + تنها چیزی که نیاز است نسبت به هم آگاه باشند ، فرمت اطلاعات ارسالی و دریافتی است.
* ایجاد محیطی مطمئن ، قابل اعتماد برای انجام پردازش ها و ارتباط میان اپلیکیشن ها
  + برنامه های تولیدکننده میتواند به ارسال اطلاعات بپردازند درحالی هیچ برنامه در آن لحظه اطلاعات را مورد استفاده قرار نمیدهد ، به این معنا که اطلاعات در برنامه میانی به صورت موقت ذخیره میشوند ، در ادامه هر گاه که برنامه مصرف کننده اطلاعات در موجود شد ، میتواند از اطلاعات استفاده نماید در حالی که برنامه ارسال کننده به سیسیتم متصل نباشد
  + در صورتی که برنامه مصرف کننده ، در پردازش شکست بخورد ، میتواند ، در خواست ارسال مجدد داشته باشد ، یا اینکه به اطلاعات به مصرف کننده دیگری ارسال شود.
* فراهم کردن راهی برای پیاده سازی معماری های افقی (به این معنا که در کنار هم کار کنند) و تعداد بالا
  + اگر برنامه مصرف کننده اطلاعات به تنهایی نتواند همه پیام ها را پردازش کند ، متوان چندین مصرف کننده اطلاعات در کنار هم قرار داد تا فرایند کوتاه تر شود.
* بسیار پر بازده تر از مدل اینتگریشن مبتنی بر دیتابیس ، چرا که کار کردن با حافظه ها سرعت به مراتب کمتری نسبت به این مدل دارند

RabbitMQ چیست؟

RabbitMQ یک message broker است که پیام را از تولیدکنندگان اطلاعات دریافت میکند و به یک یا چند مصرف کننده اطلاعات منتقل میکند ، یک نرم افزار متن باز است که توسط زبان Erlang نوشته شده ، ویژگی هایی میتوان به این بیس اولیه این نرم افزار اضافه کرد که به صورت پلاگین میتوان روی آن نصب نمود ، از چندین پروتکل برای دریافت اطلاعات پشتیبانی میکند ، که بر روی ویندوز ، لینوکس و مک توسعه یافته و همچنین داکر کانتینرآماده به استفاده آن نیز بر روی داکرهاب موجود است

در تصویر زیر معماری کلی RabbitMQ(rbmq) قابل مشاهده است ، Producer (pr) و counsumer(cr) در دو سمت این معماری و rbmq در مرکز قرار گرفته ، pr,cr به rbmq متصل شده و پیام ها را ارسال و میخوانند ، در این بین المان های دیگری از جمله channel , exchange , queue نیز وجود دارد، در ادامه به توضیح هر یک از این المان ها میپردازیم



اینجا هنوز کلی کار داریم ، ولی مسئله اینجاس هنوز consumer رو نساختیم !

پس بریم سراغ اون و جنگو !