Fafka

یکی از بزرگ ترین چالش های همراه با کلان داده، تحلیل داده هااست. اما پیش از تحلیل ابتدا باید بتوان اطلاعات مورد نیاز را برای سیستم جمع آوری نموده و پس از پردازش، اطلاعات را در دسترس کاربران قرار داد. این نقطه ای است که آپاچی کافکا (Apache Kafka)به کار می آید. کافکا برای مدیریت جریان بلادرنگ داده، جمع آوری داده های کلان یا تحلیل بلادرنگ داده و یا هردو به کار گرفته می شود.

عملکرد کلی Kafka شامل مفاهیمی از قبیل Kafka Broker، Kafka Producer و Kafka Consumer است که زیاد آن ها را می شنویم. Kafka Broker یک node در کلاستر Kafka است که برای ذخیره سازی و همچنین replicate کردن دیتا از آن استفاده می شود. Producer پیام ها را درون ظرفی به نام topic قرار می دهد. Consumer نیز پیام ها را از topic می خواند.

هنگامی که ما داده هایی را از یک اپلیکیشن به اپلیکیشن دیگری منتقل می کنیم در این جا از Messaging System استفاده می کنیم. در نتیجه استفاده از چنین سیستمی، بدون نگرانی از نحوه اشتراک داده ها، اپلیکیشن مان تنها بر روی خود داده ها تمرکز می کند. در کل دو نوع الگوی پیام رسانی وجود دارد Point-to-Point .و .Publish-Subscribe در ادامه با هر یک بیشتر آشنا خواهیم شد.

سیستم تبادل پیام نقطه به نقطه (Point-to-Point)

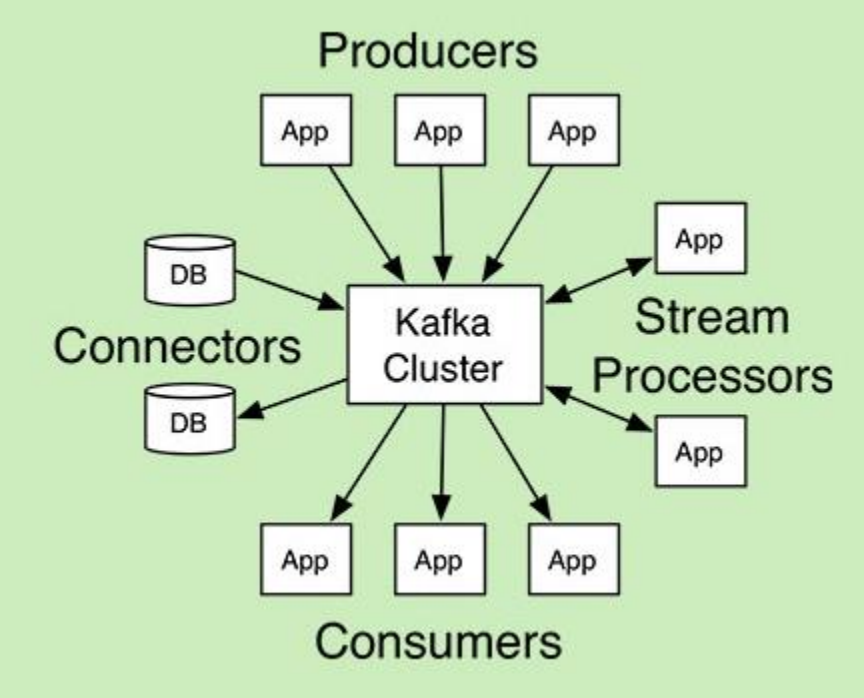
در این سیستم، پیام ها همگی در یک صف نگهداری می شوند. همچنین یک پیام خاص صرفا توسط یک consumer می تواند consume شود. این سیستم پیام رسانی این اطمینان را می دهد که هر زمان که یک consumer یک پیام را از صف می خواند، آن پیام دیگر از آن صف محو خواهد شد.

سیستم تبادل پیام Publish-Subscribe

در این سیستم، پیام ها داخل ظرفی به نام topic قرار می گیرند (publish می شوند). در واقع در این سیستم، consumer ها می توانند در یک یا چند topic عضو شوند (subscribe کنند) و تمامی پیام های آن تاپیک ها را consume کنند.

در مباحث big data معمولا با دو چالش اصلی مواجهیم. اولین چالش، جمع آوری حجم زیادی از داده ها بوده و چالش دوم تحلیل داده های جمع آوری شده می باشد. لازمه ی غلبه بر این دو چالش استفاده از یک messaging system است. Kafka این امکان را برایمان فراهم کرده است.

پاچی کافکا با ویژگی های گفته شده و به دلیل گستردگی استفاده به رویارویی با تکنولوژی های مشابه مانند ActiveMQ و RabbitMQ آمده است.



Rabbit mq

RabbitMQ یک نرم افزار برای انتقال پیام بین سیستم ها یا به عبارتی message-broker software که با استفاده از اون می تونیم بین سیستم های مختلف پیام ارسال کنیم و عملیات صف بندی به خوبی انجام بدیم.

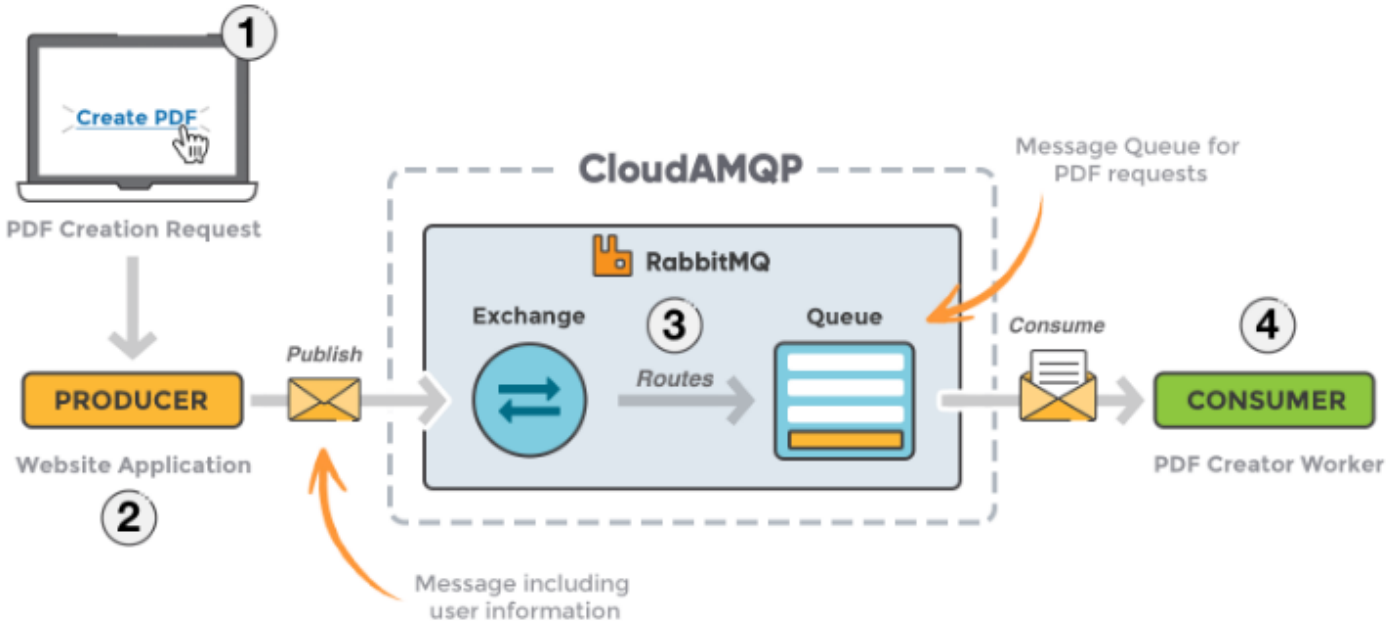
این سیستم به ما کمک می کنه که بین چندین برنامه مختلف که حتی با زبان های مختلفی هم نوشته شدن بتونیم ارتباط برقرار کنیم البته توی RabbitMQ چیزی وجود داره به نام صف یا همون queue که پیام هایی که ما برای برنامه های دیگه ارسال می کنیم درون یک صف قرار می ده و برنامه مقصد به ترتیب و طبق تنظیماتی که ما براش تعریف می کنیم اون پیام هارو میخونه و پردازش میکنه (البته اگه نیاز به پردازش داشته باشه).

RabbitMQ یکی از بهترین انتخاب ها برای ارتباط برقرار کردن بین microservice ها می باشد. و هم اکنون خیلی ها ازش استفاده میکنن.

جالبی RabbitMQ اینجاست که ما می تونیم با برنامه هایی ارتباط برقرار کنیم که حتی در یک سرور دیگه باشن. مثلا یک برنامه با زبان php در سرور هلند قرار داره و یک برنامه دیگه با زبان python در سرور ایران. این دوتا به راحتی می تونن از طریق RabbitMQ با هم ارتباط برقرار کنن و کنار هم کار کنن.

یکی دیگه از ویژگی های خیلی جالب و مهم RabbitMQ اینه که ما می تونیم چندین سرور برای یک کار تعریف کنیم و در صورتی که یک سرور از کار بی افته سرور دیگه جاشو بگیره و وظایفشو انجام بده. یا RabbitMQ میتونه کار هارو بین این سرور ها با تنظیمات ما به اشتراک بزاره تا فشار کمتری به سرور ها بیاد و سرعت پردازش بالا بره.

مثال: فکر کنید یک بخش از سیستم ما **pdf** بهش میدیم اون تمام صفحات اونو به صورت **jpg** بر میگردونه خب ما اگه این کارو با سرور اصلی انجام بدیم فشار زیادی به سرور میاد و اگه این کار در یک زمان واحد زیاد انجام بشه ممکنه سرور **down** بشه. برای همین این کارو به عهده یک سرویس دیگه میزاریم که فقط کارش همینه . با **RabbitMQ** این وظایف درون یک صف واحد بین این سرویس ها به اشتراک بذاریم و البته اگه یک روز یکی از این سرویس ها قطع شد بقیه به درستی کار خودشونو ادامه میدن. اینم در جریان باشید که ما بعدا می تونیم سرویس های دیگه هم به این چرخه اضافه کنیم.



producers کسی یا برنامه ای هست که پیام اولیه رو تولید میکنه و به صف پردازش ارسال میکنه و **consumers** کسی یا برنامه ای هست که این پیام هارو به ترتیب از صف میگیره و پردازش میکنه

بین **producers** و **consumers**۲تا مفهوم وجود داره به نام های **exchange** و **queue** که قبلا درباره **queue** توضیح دادم که یک صف که درخواست توی اون جمع میشه و **consumers** ها به ترتیب اونارو بر میدارن و پردازش می کنند.

Exchange پیام‌هایی را از تولیدکنندگان دریافت و آنها را بسته به قوانین تعریف شده توسط نوع تبادل، در صف‌ها قرار می‌دهد. برای دریافت پیام‌ها، یک صف نیاز است که حداقل به یک تبادل وابسته باشد.

چه زمان و چرا باید از RabbitMQ استفاده کنید؟

صف بندی پیام به سرورهای وب اجازه می‌دهد تا به سرعت به درخواست‌ها پاسخ دهند، به جای اینکه اجبار به اجرای روندهایی با منابع سنگین در محل شوند.

صف بندی پیام هنگامیکه بخواهید یک پیام را بین چندین گیرنده جهت مصرف/بکارگیری توزیع کنید یا برای تعدیل بارها بین کارکنان نیز مفید است.

مصرف کننده می‌توانند یک پیام را از صف برداشته و شروع به پردازش کند همزمان با اینکه تولیدکننده در حال صف بندی پیام‌های جدید در صف است.

مصرف کننده می‌تواند در یک سرور کاملا متفاوت از منتشرکننده باشد، یا می‌توانند در یک سرور یکسان قرار داشته باشند.

درخواست می‌تواند به یک زبان برنامه‌نویسی ایجاد شده و به زبان برنامه‌نویسی دیگری مدیریت شود.

دو برنامه تنها از طریق پیام‌هایی که به یکدیگر می‌فرستند در ارتباط خواهند بود.

بدین جهت، دو برنامه ارتباطی پایین (low coupling)بین فرستنده و گیرنده خواهند داشت.

آشنایی با بعضی مفاهیم در rabbit mq

تولیدکننده (:(Producerبرنامه‌ای که پیام‌ها را دریافت می‌کند.

صف (:(Queueبافری (حافظه‌ی موقت/میانی) که پیام‌ها را ذخیره می‌کند.

پیام (:(Messageاطلاعاتی که از طریق RabbitMQ تولیدکننده به یک مصرف‌کننده ارسال می‌شود.

پیوند (:(Connectionیک پیوند، پیوند TCP بین برنامه شما و واسطه‌ی RabbitMQ است.

کانال (:(Channelیک کانال، پیوندی مجازی درون یک پیوند است. هنگامیکه در حال انتشار یا مصرف پیام‌هایی از یک صف هستید- همگی بر روی یک کانال انجام می‌شود.

اتصال :(Binding) یک اتصال، لینکی بین یک صف و یک تبادل است.

کلید مسیریابی :(Routing Key) کلید مسیریابی، کلیدی است که تبادل جهت تصمیم‌گیری نحوه‌ی مسیردهی پیام به صف‌ها به آن نگاه می‌کند. کلید مسیریابی مانند یک آدرس برای پیام است.

کاربران :(Users) امکان اتصال به RabbitMQ با یک نام‌کاربری و رمزعبور معین وجود دارد.

می‌توان به همه‌ی کاربران مجوزهایی مانند حقوق خواندن، نوشتن و پیکربندی قوانین درون نمونه اختصاص دارد.

میزبان مجازی (Vhost, Virtualhost) یک میزبان مجازی روشی برای جداسازی برنامه‌ها با استفاده از نمونه‌ی RabbitMQ یکسانی فراهم می‌کند.

کاربران مختلف می‌توانند مزایای دسترسی متفاوتی به میزبان مجازی متفاوتی داشته باشند و صف‌ها و تبادلات می‌توانند ایجاد شوند، بنابراین تنها در یک میزبان مجازی وجود دارند.

RabbitMQ بطور پیش‌فرض به زبان یک پروتکل به نام AMQP صحبت می‌کند. برای اینکه قادر به برقراری ارتباط با RabbitMQ شوید، نیاز به کتابخانه‌ای دارید که همان پروتکل RabbitMQ را درک کند. نیاز است کتابخانه سرویس گیرنده برای زبان برنامه‌نویسی که قصد استفاده برای برنامه‌ی خود دارید را دانلود کنید. یک کتابخانه سرویس‌گیرنده یک رابط برنامه‌نویسی برنامه‌ها (APIبرای استفاده در نوشتن برنامه‌های سرویس‌گیرنده است. یک کتابخانه‌ی سرویس‌گیرنده چندین متد قابل استفاده، در این مورد، برای برقراری ارتباط با RabbitMQ دارد.

متدها باید در زمانیکه شما، برای مثال، به واسطه‌ی RabbitMQ متصل می‌شوید، استفاده شوند (با استفاده از پارامترها، نام میزبان، شماره درگاه معین و ...) یا هنگامیکه یک صف یا یک تبادل را اعلان می‌کنید.

تقریباً برای همه‌ی زبان‌های برنامه‌نویسی یک انتخاب کتابخانه وجود دارد.