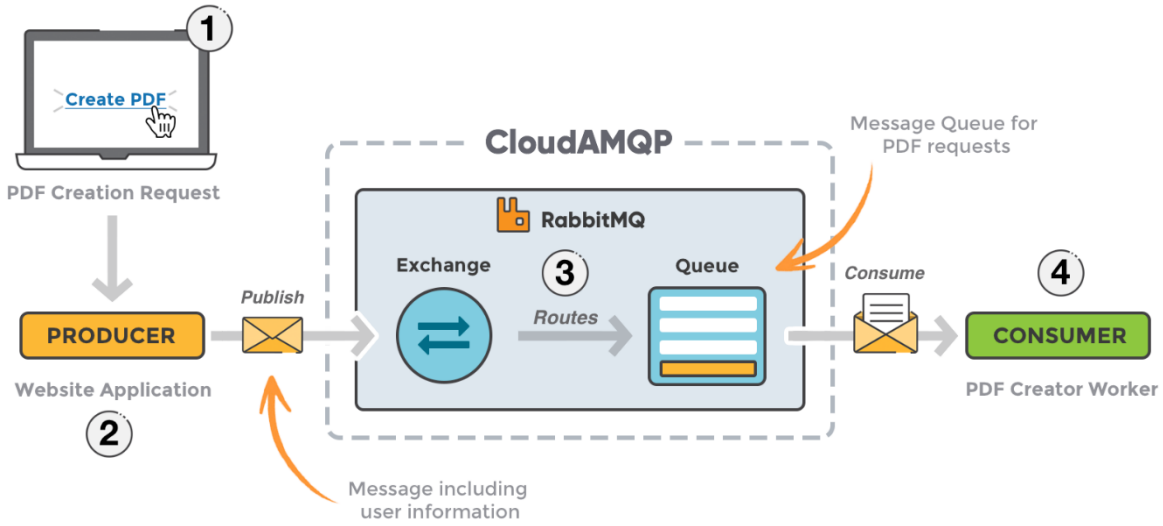
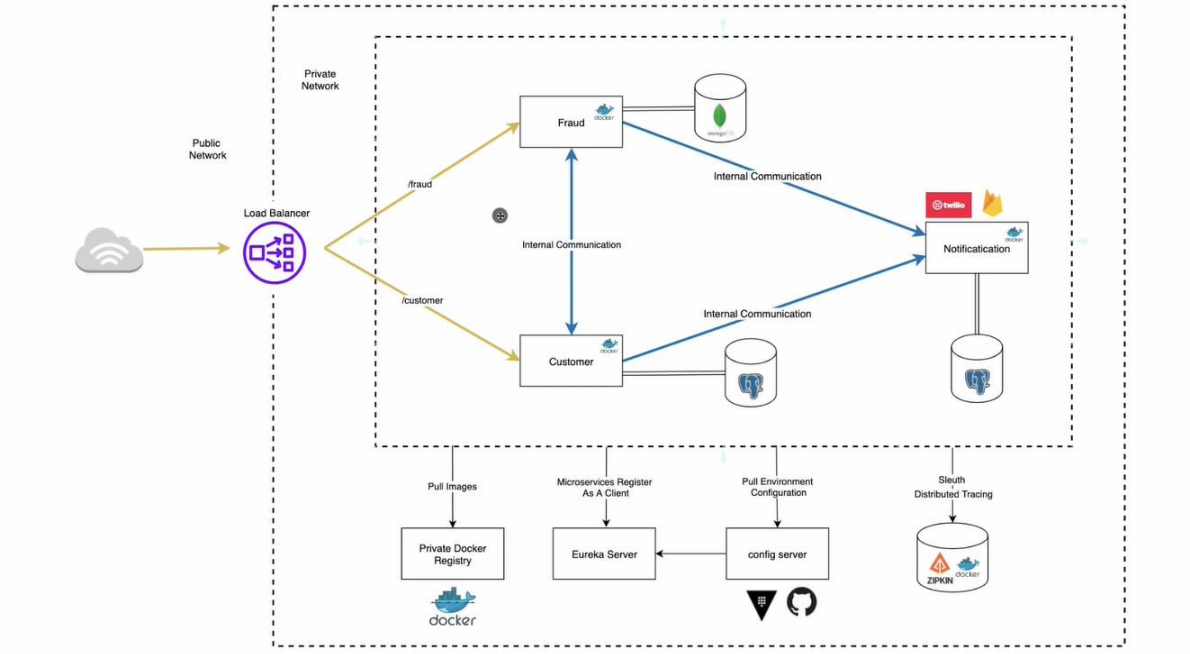
RabbitMQ a messaging

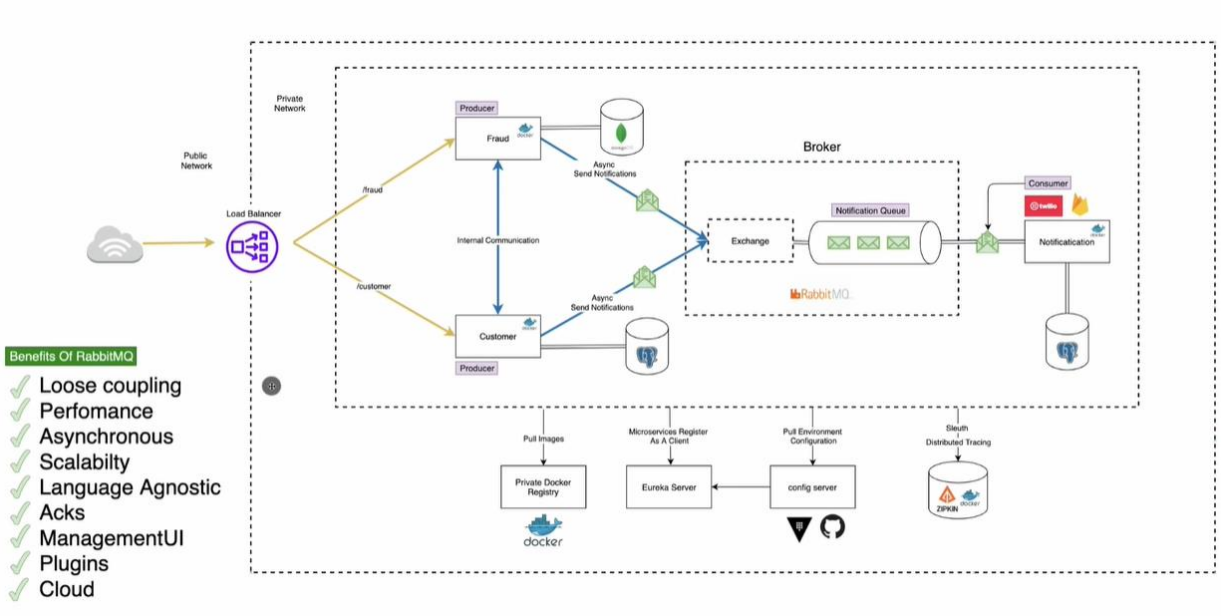
Na začiatok si dáme 2 príklady z praxe



Na začiatku sme mali jedného producera (webová aplikácia) a jedného consumera (aplikácia na spracovanie PDF). Ak aplikácia na spracovanie PDF zlyhá alebo ak súčasne prichádza veľa požiadaviek na spracovanie PDF, správy sa budú naďalej ukladať vo fronte, kým consumer znova nezačne. Potom by spracoval všetky správy, jednu po druhej. Keby sme nemali nastalo by zlyhanie, hromadili by sa nám správy a neprichádzala by odozva, nastal by timeout a chyba. Pre tento prípad bude super využiť message brokera ako RabbitMQ tak ako je vyššie na grafe.



Príklad 2



Predstavme si že „Notification“ odosiela používateľovi SMSku o potvrdení registrácie. V prípade vyššie bez message brokera by mohlo nastať hneď niekoľko problémov. Ak bude Notification zahltené veľa požiadavkami alebo len požiadavkami ktoré trvajú dlhšie alebo Firebase/Tvilio budú mať krátky výpadok tak nastane chyba že server nebude odpovedať a nastane problém, pretože nebude prijímať ďalšie požiadavky od „Customer“ ktorý posiela požiadavky synchrónne a nastane veľké zdržanie alebo dokonca 500ka z timeoutu. Rovnako ak napríklad nájdeme nejaký bug a chceme vykonať upgrade, za ten čas čo ho vykonávame (čas nemusí byť dlhý), nám môže prísť niekoľko requestov. V tomto stave sa stane to, že používateľ je zaregistrovaný ale request na „Notification“ zlyhá, vtedy dostane klient 500ku chybu a nedostane ani SMSku. Nám nevadí že klientovi príde SMSka za 1, 5, 20 alebo 30 sekúnd, dôležité je ale že mu príde a práve tu využijeme asynchrónnu komunikáciu a našeho message brokera ktorý si vie s podobnými prípadmi poradiť.

VÝHODY použitia RABBITMQ

Loose coupling = Fraud * Customer * Broker Notification môžu žiť na separate mašine, nemusia byť spolu

Performance = Notification je down, tak Fraud a Customer môžu stale posielať message a v Rabbite budu uložene v queue. Ak Notification naskočí tak môže prečitať messages z Rabbita a Producer nevie že sme boli down, nič sa nestratilo.

Asynchronously = každy naš producer posiela message acynchronne, čiže ak Consumerovi niečo zaberie 20 sekund prebrať ale stále môže producer posielať a dáta su uchovane tak je všetko alright

Scalability = ak chceme mať pusteneho brokera ako cluster 2 alebo viac zariadeni tak môžeme

Language agnostic = ak je producer java a notification je cečko tak je to okay

Acknowlegement = správa nezmizne z brokera ak notification nepotvrdi že prijalo správu. Ak napriklad notification malo problem prečitať message z queue tak message tam stále ostane.

ManagementUI – UI aplikácia

Plugin = poskytuje aj množstvo pluginov

Cloud = rabbitmq vie tiež bežať na cloude (AWS, Azure, Google Cloud, ..) . Môže bežať cez Docker

Niektoré dôležité koncepty je potrebné opísať skôr, ako sa hlbšie ponoríme do RabbitMQ. V príkladoch sa používa predvolený virtuálny hostiteľ, predvolený používateľ a predvolené povolenia, takže si prejdeme prvky a koncepty:

Producer: Aplikácia, ktorá odosiela správy.

Consumer: Aplikácia, ktorá prijíma správy.

Queue (fronta) : Buffer, ktorý ukladá správy.

Connection: TCP spojenie medzi vašou aplikáciou a brokerom RabbitMQ.

Channel: Virtuálne pripojenie v rámci pripojenia. Pri posielaní alebo prijímaní správ z fronty - všetko sa deje cez channel.

Exchange: Prijíma správy od producerov a posúva ich do front v závislosti od pravidiel definovaných typom exchange. Na prijímanie správ musí byť fronta viazaná aspoň na jednu exchange.

Binfing: Binding je prepojenie medzi frontom a exchangeov.

Routing key: Kľúč, na ktorý sa exchange pozerá, aby rozhodla, ako smerovať správu do frontov. Predstavte si routing key ako adresu správy.

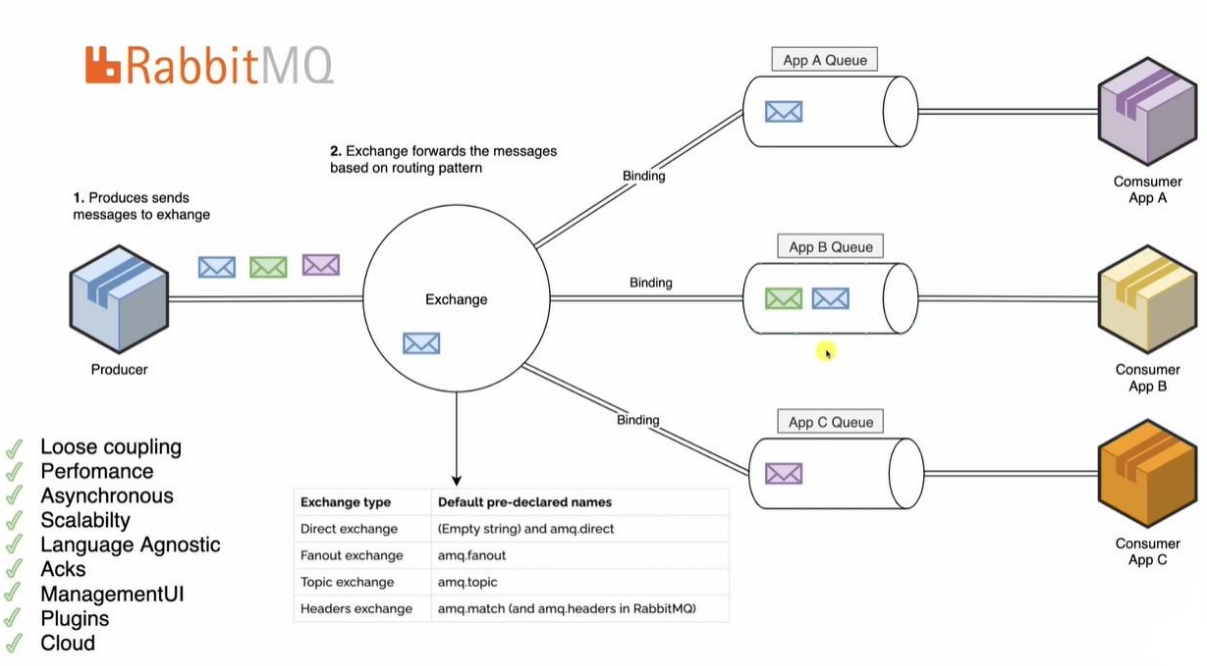
AMQP: Advanced Message Queuing Protocol je protokol, ktorý používa RabbitMQ na odosielanie správ.

Users : K RabbitMQ je možné sa pripojiť s daným užívateľským menom a heslom. Každému používateľovi možno prideliť oprávnenia, ako napríklad práva na čítanie, zápis a konfiguráciu privilégií v rámci inštancie. Používateľom možno tiež prideliť povolenia pre konkrétnych virtuálnych hostov.

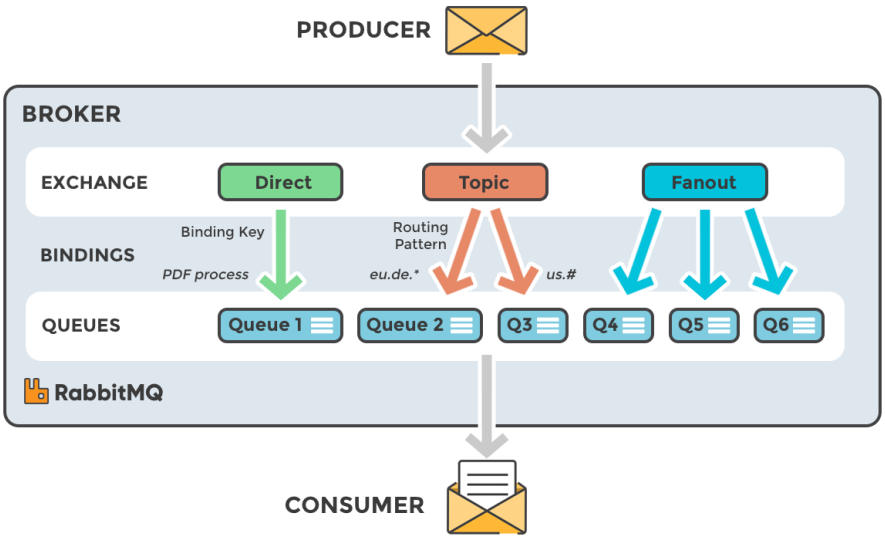
Vhost, virtuálny host: Poskytuje spôsob segregácie aplikácií pomocou rovnakej inštancie RabbitMQ. Rôzni používatelia môžu mať rôzne povolenia pre rôzne vhost a môžu byť vytvorené fronty a exchanges, takže existujú iba v jednom vhost

EXCHANGES

Správy sa neposielajú priamo do fronty; namiesto toho producer posiela správy do exchange. Exchange je zodpovedná za smerovanie správ do rôznych front pomocou väzieb a smerovacích kľúčov. Väzba je prepojenie medzi frontov a exchangeov.



EXCHANGE TYPES



Direct – routing key aj binding key musí byť rovnaky

Fanout = rozpošle správu všetkým queues, rozvetvenie

Topic = message je foo.bar a binding je foo.* tak sa to pošle aj takto kde je binding foo.*

Headers = využíva správu v headery namiesto routing key

Nameless = Špecificka pre Rabbit, ked routing key je rovnaka ako ako meno queue