Criando um Tópico PUB/sub

Para isso você usa o G Cloud. Assim, o G Cloud é uma das ferramentas que contam com o Cloud SDK. Você pode usar isso do Cloud Shell, por exemplo, ou você pode baixar o Cloud SDK em sua máquina local, e apenas execute isso na linha de comando.

Então você diz, G Cloud, crie um tópico, e o nome de um tópico aqui é San Diego.

Então, estou criando um tópico chamado San Diego, no pubsub.

Na verdade, esse tópico será criado no projeto, o namespace deste tópico vai incluir o nome do seu projeto. Se você quiser publicar uma mensagem neste tópico,

você pode usar o G Cloud novamente. Então, você diz G Cloud, os tópicos pubsub são publicados no tópico San Diego, publicar a mensagem "Olá".

As mensagens, no que diz respeito ao pubsub, são opacas. O Pubsub não vai analisar essas mensagens. É só pegar essa gota de comida que você dá e basicamente vai mantê-los e entregá-los aos assinantes. Então, tanto quanto pubsub está em causa, as mensagens em si, não há análise é apenas um fluxo opaco.

Agora estou mostrando os dois usando o G Cloud, mas a coisa a perceber é que o que o G Cloud está fazendo é isso, está realmente enviando uma chamada pela web,

uma chamada de API de descanso. Porque é apenas uma chamada de API de descanso

você pode fazer essa chamada de API de descanso a partir do seu próprio aplicativo.

Então, você pode codificar, você pode codificar o pedido HTTP e colocar os cabeçalhos,

e enviá-lo, ou você pode usar uma das bibliotecas do cliente pubsub de qualquer uma dessas 10 linguagens GRPC, ou você pode usar uma das bibliotecas codificadas manualmente.

Então, do Google.Cloud import pubsub, ao criar um cliente pubsub você o usa para uma das duas coisas, publicar no pubsub, ou para se inscrever no pubsub.

Então, aqui vou usar o cliente e eu estou basicamente criando um tópico San Diego,

e então eu estou publicando a mensagem deles, "Olá", que é apenas um conjunto de bytes, no tópico.

Portanto, existem outras opções de publicação. Uma delas, você poderia apenas publicar a mensagem e pronto, mas outra maneira de fazer isso é que você pode publicar uma mensagem, adicionando atributos extras.

Então, esses são pares de valores-chave. Lembre-se do que eu disse, no que diz respeito ao pubsub, as mensagens são opacas. Mas se você quiser passar metadados sobre a mensagem, então você pode passar a mensagem e passar metadados que são pares de valores-chave, e quando o assinante recebe a mensagem, eles recebem a mensagem, mas também obtêm os metadados. Assim, eles podem pesquisar os metadados dos pares de valores-chave nos quais estão interessados. Mas os atributos que você está fornecendo depende de você, e você decide quais são os atributos, e você acabou de colocá-los e usá-los no final do assinante.

Mas um atributo-chave que costumamos usar é um atributo que reflete o registro de data e hora da mensagem. Normalmente, o timestamp da mensagem é timestamp no qual você chama o método publicado. A hora em que você publica no pubsub, é o timestamp da mensagem.

Então, isso vai funcionar muito bem se você estiver trabalhando em tempo real.

Mas se você não está trabalhando em tempo real, se por exemplo você está simulando dados históricos, então você está publicando em algum momento de 2017, mas os dados realmente vêm de 2008, podemos querer definir metadados, que é o timestamp da mensagem, porque mais uma vez, o pubsub não vai analisar essa mensagem

e descobrir qual é o timestamp real dentro da mensagem.

A outra coisa que você quer estar ciente é que, você não precisa chamar a publicação uma de cada vez para cada mensagem.

Toda vez que você ligar, publique há uma chamada de rede do seu aplicativo para o pubsub, e as chamadas de rede introduzem um pouco de latência.

Então, se você tem um monte de mensagens que você quer publicar, uma boa ideia é fazer um lote.

Então, aqui está a maneira de fazer isso. Você diz qual tópico esse lote em lote, isso é em Python, mas você pode fazer a mesma coisa em qualquer idioma que seja suportado.

Você diz batch.publish primeira mensagem, batch.publish segunda mensagem,

batch.publish terceira mensagem e, em seguida, quando você sair da instrução with,

todo o lote é publicado de uma só vez.

Então, publicando um conjunto de mensagens como um lote, você ajuda a reduzir os custos da rede.

Então, existem dois tipos de fluxos de entrega, que um assinante pode escolher.

Então, dentro de um tópico você pode ter alguns assinantes que estão puxando,

e alguns assinantes que solicitam um push. Então, como isso funciona?

Na entrega push, pubsub, vai iniciar a solicitação para o aplicativo do assinante, para entregar mensagens. Então, sempre que houver uma nova mensagem no tópico,

essa mensagem é entregue ao assinante pelo pubsub.

Em uma assinatura pull, o assinante pede ao pubsub, "Existe uma nova mensagem?" Se houver, recebe a mensagem.

Então, quando o assinante está puxando mensagens, o que um assinante precisa?

O assinante precisa simplesmente ser capaz de fazer uma chamada pela web. Ele precisa ser capaz de fazer uma chamada de API, para puxar, o assinante só precisa ser capaz de fazer chamadas.

Que tal para empurrar?

Para push, o assinante precisa ser um aplicativo da Web HTTPS, ter um web hook, que é acessível via HTTPS. Porque o que o pubsub vai fazer é chamar essa URL, em outras palavras, em uma assinatura push, seu assinante precisa ser um aplicativo de servidor.

OK. Então, a assinatura do pull, haverá um atraso entre publicar no tópico e o assinante recebendo o tópico porque um assinante normalmente está verificando periodicamente. Se está verificando trinta vezes por segundo, então você basicamente tem uma latência de mais de 30 segundos aproximadamente. Como um assinante push, você não tem atrasos, o push é ideal se você quer baixa latência, processamento imediato de mensagens. A assinatura de pull é realmente boa se você tiver muitos assinantes e os assinantes são criados dinamicamente. Então, use push se você não quer quase nenhuma latência perto do desempenho em tempo real. Mas espere um segundo. O que acontece no caso de uma assinatura push se o assinante não estiver em execução? Assim, o assinante configurou uma assinatura. Suas mensagens estão indo para assinatura, mas o Pub / Sub não poderá alcançar o assinante porque, por qualquer motivo, essa URL da Web não está acessível.

Bem, nesse caso, o que o Pub / Sub faz é o backoff exponencial de redução.

Em outras palavras, ele tenta então aumenta esse tempo e depois demora e tenta novamente, e depois aumenta um pouco mais e tenta novamente, aumenta um pouco mais e tenta novamente, e assim por diante. Assim, você pode controlar essa nova tentativa, mas continuará tentando até sete dias. Em uma assinatura pull, o assinante solicita explicitamente entrega de uma mensagem que está na fila de assinatura, e o servidor Pub / Sub basicamente responde com a mensagem ou responde com um erro se a fila estiver vazia, e responde com um ID de confirmação. O assinante é responsável por chamar o método de reconhecimento usando esse ID de confirmação.

Se isso não acontecer, então o Pub / Sub acha que enviou a mensagem, mas entre o assinante pedindo a mensagem e o Cloud Pub / Sub enviando a mensagem, o assinante caiu porque acredita que tentará reenviar a mensagem.

Então, se você está fazendo uma assinatura é importante reconhecer a mensagem, é claro.

As bibliotecas do cliente que você está usando lidam com tudo isso para você. Isto é, se você estiver codificando as chamadas HTTP. Mas também é importante perceber que você pode ter mensagens duplicadas entregues, certo? Você terá mensagens duplicadas se você como assinante, invocou uma chamada de mensagem, você recebe a mensagem

você processa a mensagem ou faz algo com a mensagem, mas depois entre o processamento de uma mensagem e o reconhecimento se você falhar. Então você processou uma mensagem, você fez isso, e então você volta e porque Pub / Sub

não percebi que você realmente queria enviar um reconhecimento, você caiu no meio,

Ele enviará a você esta mensagem novamente porque essa mensagem não foi confirmada.

Mas esse é o tipo de coisa que o Dataflow vai manipular se o Dataflow for o próximo passo em seu pipeline.

Em uma assinatura do Push, o servidor Pub / Sub chamará esse endpoint HTTPS pré-configurado, e a resposta do assinante a essa chamada serve como um reconhecimento implícito.

Então, se você obtiver uma resposta de sucesso, uma resposta de 200, indica que a mensagem foi bem sucedida, processado e, portanto, o sistema Pub / Sub pode excluí-lo da assinatura. Na resposta sem sucesso, uma resposta 400, por exemplo, 404, 429, etc, a resposta indica que o servidor Pub / Sub deve reenviar a mensagem. Para garantir que os assinantes possam manipular o fluxo de mensagens, o que o Pub / Sub vai fazer é ajustar dinamicamente essa taxa na qual é enviando essas mensagens e vai usar um algoritmo para avaliar novas tentativas de limite, de modo que ele esteja tentando garantir que o ponto final do assinante não fique sobrecarregado.

Então, e quanto ao outro lado da equação?

Bem, se você estiver interessado em mensagens como assinante, você criará uma assinatura. Então, as assinaturas do gcloud pubsub criam, você está criando uma assinatura, e você está criando uma assinatura para mensagens que aparecem no tópico, San Diego.

Você está dando um nome à assinatura e, novamente, esta assinatura é criada em seu projeto e o projeto para a mensagem e o tópico podem ser diferentes do projeto para os assinantes.

Então você faz isso e sempre que você estiver interessado, você pode fazer um puxão.

Então, é uma entrega puxada aqui, você está dizendo, "Dê-me a próxima mensagem" e usando as assinaturas do gcloud pubsub, reconhecer automaticamente e você

puxar e a próxima nova mensagem fora dessa assinatura. Agora, quando as mensagens são publicadas no tópico, San Diego, eles também vão aparecer nesta assinatura.

Você processou apenas um deles. Todas as outras mensagens estão enfileiradas

essa assinatura para a próxima vez que você chamar o pull.

Então, o que você pode fazer no gcloud, uma chamada à API REST, você também pode fazer no seu código de biblioteca do cliente.

Então aqui está o Python. Você vai ao tópico.

A maneira como você cria um tópico é muito semelhante à maneira como você criou o tópico anteriormente. Lembre-se de que criamos um tópico criando um cliente Pub / Sub e nós configuramos o client.topic San Diego. Então você pega o seu tópico e uma vez que você tem o seu tópico, você cria uma assinatura com o nome da sua assinatura,

diga mySub1 e crie essa assinatura, sempre que você estiver interessado em uma nova mensagem, você diz subscription.pull e você pode pedir para apenas sentar e esperar,

bloco em outras palavras, para uma nova mensagem até chegar uma nova mensagem,

ou você pode dizer retornar imediatamente.

Então, aqui estou dizendo que retorne imediatamente, é verdade.

Então, quando meus resultados voltarem, eles não serão nenhum ou eles realmente terão uma mensagem neles. Então, se o resultado não for nenhum, eu reconheci

subscription.knowledge, e aqui está o reconhecimento para confirmar ID e mensagem nos resultados.

Então todas as mensagens que recebi nos resultados, eu vou reconhecer todos eles.

Mas então, meus resultados têm todas as mensagens. Eu então vou em frente e procuro-os.