Publicado

http://www.infoq.com/br/news/2019/05/guardian-mongodb-postgresql

A migração do The Guardian de MongoDB para o PostgreSQL no Amazon RDS

Original: https://www.infoq.com/news/2019/01/guardian-mongodb-postgresql
Autor: Edeilson Silva

Resumo:

O Guardian migrou o armazenamento de dados do CMS em 2018 de um cluster MongoDB autogerenciado para o PostgreSQL na Amazon RDS para obter uma solução totalmente gerenciada. A equipe fez uma migração baseada em API sem qualquer tempo de inatividade.

O Guardian migrou o armazenamento de dados do CMS em 2018 de um cluster MongoDB autogerenciado para o PostgreSQL na Amazon RDS para obter uma solução totalmente gerenciada. A equipe fez uma migração baseada em API sem qualquer tempo de inatividade.

O CMS interno da Guardian — chamado Composer — que armazena artigos, conteúdo de blog, galerias de fotos e vídeo foi originalmente construído em cima do MongoDB como um armazenamento de dados. Isso foi precedido por um software de fornecedor apoiado por um banco de dados Oracle. Essa configuração teve tempos de inatividade sempre que o esquema precisava ser migrado. Como alternativa, a equipe analisou vários db's NoSQL, e uma das principais razões para escolher MongoDB parece ter sido a flexibilidade. Originalmente hospedados em seu próprio datacenter, eles moveram seu MongoDB para seus servidores da AWS após uma interrupção. Os scripts de instalação e gerenciamento tiveram que ser escritos à mão pela equipe do Guardian. Eles optaram por um contrato de suporte e compraram a ferramenta OpsManager, uma aplicação frontend para gerenciar MongoDB. No entanto, a equipe não foi para a oferta Atlas do MongoDB, que é um "banco de dados totalmente gerenciado", por razões que não são claras. O OpsManager não gerencia implementações.

Depois de migrar para a AWS, a equipe enfrentou duas interrupções no MongoDB. Alguns dos motivos eram problemas básicos de administração do sistema, como não permitir que o NTP acesse horário dos servidores para manter os relógios sincronizados. Outros pertenciam à dificuldade de gerenciar o próprio OpsManager e obter suporte oportuno do

fornecedor, de acordo com o artigo. A equipe sentiu que mudar para uma solução que tivesse gerenciamento mínimo de banco de dados seria melhor para eles.

A equipe escolheu o PostgreSQL devido à sua maturidade e suporte para o tipo de dados jsonb, como um banco de dados hospedado na Amazon RDS. O tipo jsonb permite a indexação de campos dentro do objeto JSON. O plano de migração foi escrever uma nova API sobre Postgres e usar um proxy que enviaria tráfego para ambas as APIs para mantê-las sincronizadas para novos dados recebidos. Os dados existentes seriam migrados usando as APIs e, em seguida, o proxy mudaria para a nova API. Sua migração anterior da Oracle também foi feita usando uma abordagem semelhante. Os logs de script de migração foram enviados para o Elasticsearch para que a migração pudesse ser rastreada. No processo, eles também melhoraram seu registro estruturado.

O proxy direcionou todo o tráfego para a API MongoDB em tempo real e assíncrona para a API Postgres. Qualquer diferença nas respostas foram registradas e analisadas. Para garantir que a nova API e back-end pudessem suportar o tráfego de produção, os processos GoReplay foram executados para gerar tráfego. O GoReplay pode capturar o tráfego e reproduzi-lo em um ambiente diferente - neste caso, o de pré-produção. Uma migração completa foi feita no ambiente de pré-produção. A etapa final na migração de produção foi mudar o nome DNS do endpoint do proxy (um Amazon ELB) para a API Postgres (outro ELB). Isso permitiu que seus clientes funcionassem sem qualquer alteração. Após a migração, seus testes de integração falharam, pois eles não foram migrados para a nova API.

Outras organizações que migraram de MongoDB para PostgreSQL diversas razões.